

Computer Science Project

**FOOD DELIVERY
MOBILE APPLICATION**

Contents

Research & Analysis	35
Problem Identification	35
Current Issue with Food Delivery Apps	35
Ways of Solving these Issues Computationally	36
Abstraction	36
Decomposition	37
Development Methodologies	38
Waterfall	38
Advantages	38
Disadvantages	38
Agile	40
Advantages	40
Disadvantages	41
Spiral	42
Advantages	42
Disadvantages	43
Choice of Methodology	43
Stakeholders	44
Group 1: Experienced Customer - Sam R	44
Stakeholder Justification	44

Q&A	44
Group 2: Customer with No Experience - Stevie T	46
Stakeholder Justification	46
Q&A	46
Group 3: Customer With Dietary Needs - Scott P	47
Stakeholder Justification	47
Q&A	48
Group 4: Restaurant Owner - Paddy R	49
Stakeholder Justification	49
Q&A	49
Researching the Problem	49
Existing Solutions	49
Examples of Existing Solutions	51
Natasha's Law	51
Just Eat	51
Frontend	51
Mobile App	51
Desktop Website	52
Backend	53
Summary	54
Design	54
Good Design	54

Bad Design	54
Features	55
Strengths	55
Limitations	56
Conclusion	56
Hello Fresh	57
Frontend	57
Summary	69
Design	70
Features	71
Strengths	72
Limitations	73
Conclusion	74
OpenTable	74
Frontend	74
Backend	77
Summary	78
Design	78
Features	79
Strengths	80
Limitations	80
Conclusion	81

Research Findings	81
Essential Features	82
Concept	82
Features & Reasoning	88
Limitations	90
Solution Requirements	91
Software	91
Language	91
Alternative Languages	91
Database	92
Alternative Databases	93
Intended Operating System(s)	93
Alternative Intended Operating Systems	94
Hardware	95
Storage	95
CPU	95
Memory	96
Connectivity	96
Success Criteria	96
Database	96
Login/Profile	97
Restaurants/Food	101

QR Code	106
Cart/Checkout	107
Appearance	109
Prototype 1A	110
Design & Development	110
App Design Analysis	110
Decomposition of the Problem	110
Database Development	112
Database Structure Design	112
Tables	113
Restaurant	113
Restaurant Times	114
Restaurant Exception Times	114
Menu Categories	114
Item	115
Food Categories	116
Customise Item	116
Customise Item Selection	117
Customise Item Selection Ingredients	117
Item Ingredients	118
Restaurant Rating	118
Item Rating	119

Profile	120
User Allergy	121
Ingredient	121
Ingredient Tags	121
Ingredient Category	122
Guest Profiles	122
Favourite Item	122
Favourite Restaurant	123
Order Information	123
Order Contents	124
Correction to Documentation (2)	124
Developing the Database	125
Ingredients Table	126
Restaurants Table	129
Updated Database	132
CustomiseItem	133
CustomiseItemOptions	134
DefaultItemIngredients	135
FoodCategories	136
Ingredient	137
IngredientTags	138
Item	139

MenuCategories	140
OptionsIngredientChange	141
Restaurant	142
RestaurantOpenTimes	143
App Development	144
Login & Profile Creation	144
Persistent Profiles	178
Navigation	182
Profiles Page	189
Login/Register from Profiles Page	201
Browse Page	203
Location Button	204
Restaurant Summaries	235
Favourites	242
Restaurant Page	254
Item Page	261
Post Development	262
Testing	262
Testing Summary	262
Testing Criteria (1.1)	267
Justification	267
Expected result	267

Actual result	267
Pass or Fail?	268
Testing Criteria (1.2)	268
Justification	269
Expected result	269
Actual result	269
Pass or Fail?	269
Testing Criteria (1.3)	269
Justification	269
Expected result	269
Actual result	269
Pass or Fail?	270
Testing Criteria (2.1)	270
Justification	270
Expected result	270
Actual result	271
Pass or Fail?	272
Fixing Issues	272
Recode Result	275
Pass or Fail?	275
Testing Criteria (2.4)	276
Justification	276

Expected result	276
Actual result	277
Pass or Fail?	277
Testing Criteria (2.5)	277
Justification	278
Expected result	278
Actual result	278
Pass or Fail?	279
Testing Criteria (2.6)	279
Justification	279
Expected result	279
Actual result	280
Pass or Fail?	280
Testing Criteria (2.7)	281
Justification	281
Expected result	281
Actual result	281
Pass or Fail?	282
Testing Criteria (2.8)	282
Justification	283
Expected result	283
Actual result	283

Pass or Fail?	284
Testing Criteria (2.21)	284
Justification	284
Expected result	284
Actual result	284
Pass or Fail?	284
Testing Criteria (2.22)	284
Justification	284
Expected result	285
Actual result	285
Pass or Fail?	285
Testing Criteria (2.23)	285
Justification	285
Expected result	285
Actual result	286
Pass or Fail?	286
Testing Criteria (6.1)	287
Justification	287
Expected result	287
Actual result	288
Pass or Fail?	288
Testing Criteria (6.2)	289

Justification	289
Expected result	289
Actual result	290
Pass or Fail?	290
Prototype 1A Final Code	291
File Structure	291
Files - Application	293
main.dart	293
wrapper.dart	294
sqlite_service.dart	297
fresh_profile.dart	308
fresh_newprofile_create.dart	310
fresh_login.dart	328
addprofile.dart	340
addprofile_newprofile_create.dart	343
addprofile_login.dart	359
homepage.dart	371
foryou.dart	372
browse.dart	373
profiles.dart	374
favourites.dart	384
apperror.dart	403

genericload.dart	405
locationbutton.dart	406
navigationbar.dart	435
restaurantlist.dart	440
restaurantitemcustomise.dart	460
restaurantmain.dart	466
Files - Server	488
favouriterestaurant.php	488
favouriterestaurantdata.php	489
favouriterestaurantlist.php	491
login.php	493
register.php	495
restaurantlist.php	497
restaurantmenucategories.php	498
restaurantmenuitems.php	499
Summary	501
Evaluation	501
Prototype 1B	502
Design	502
Concept Art	502
Concept 1	503
Concept 2	503

Concept 3	506
Concept 4 (Final)	507
Theme	507
Clean Install	508
Dark Mode	509
Light Mode	510
Home Page	512
For You Page	513
Browse Page	514
Search Page	518
Profile Page	519
Favourites Page	521
Allergies/Diet Page	522
Your Orders Page	523
Profile Settings Page	524
Settings Page	526
Share Profile Page	527
Restaurant Page	529
Item Page	533
Cart & Checkout	535
Blank Pages	542
Rating Page	544

Development	547
Adaptive App Icon	548
App Theme	548
New Splash Screen	554
Force Portrait mode	556
Redesigned Welcome Screens	557
Existing User Create Profile Page	561
Removal of Duplicate Users	562
Redesigned Top Bar	564
Profiles Page	571
Profile Selection	573
Browse Page	593
Search Bar	593
Categories	596
Restaurant Summaries	600
Restaurant Page	602
Prototype 1B Final Code	606
File Structure	606
Files - Application	613
Browse.dart	613
Foryou.dart	617
Homepage.dart	618

Profiles.dart	621
Profilesetup_create.dart	624
Profilesetup_existing.dart	658
Profilesetup_login.dart	664
Welcomescreen.dart	677
Restaurant_customise.dart	683
Restaurant_main.dart	689
Setupverification.dart	723
Dataencryption.dart	726
Localprofiles_service.dart	727
Queryserver.dart	735
Setselected.dart	736
Theme.dart	737
Browse_categories.dart	752
Elements.dart	761
Profiles_buttons.dart	762
Profiles_selection.dart	765
Search.dart	781
Genericloading.dart	783
Navigationbar.dart	784
Restaurants_list.dart	789
Topbar.dart	814

Main.dart	817
Files - Server	820
Favouriterestaurant.php	820
Favouriterestaurantdata.php	822
Favouriterestaurantlist.php	824
Login.php	825
Register.php	827
Restaurantlist.php	830
Restaurantmenucategories.php	831
Restaurantmenuitems.php	833
Summary	834
Evaluation	834
Prototype 2	834
Design	834
Success Criteria	835
Concept Art	835
Filters & Sorting	835
Item Customisation	837
Location	839
App Flowchart	840
Development	840
Location	840

Top Bar Update 1	895
Restaurant Distance	899
Sorting & Filtering	904
Sorting	906
Filtering	915
Favourites	915
Price Range	917
Max Delivery Fee	923
Min Order Price	925
Clear Filters	926
Saving Filters and Sorts	929
Recalling Filters and Sorts	929
Sending filters/Sort to Server	930
Processing Filters/Sorts	937
Displaying Delivery Fee on Restaurant List	943
Displaying Basic Data on Restaurant Page	945
Fix to Location Save Button	948
Fix to Filter & Sort	949
Customise Item Page	949
Getting Customise Data	949
Displaying Customise Data	956
Add to Cart Button	959

Determining Customise Type	964
Customise Select Widget	968
Customise Add Widget	1012
Customise Remove Widget	1023
Price Changes	1024
Required Fields	1031
Saving Customised Item	1035
Post-Development	1039
Testing	1039
Testing Summary	1039
Testing Criteria (1.3)	1041
Justification	1041
Expected result	1041
Actual result	1041
Pass or Fail?	1042
Testing Criteria (3.1)	1042
Justification	1042
Expected Result	1042
Actual Result	1042
Pass or Fail?	1043
Testing Criteria (3.2)	1044
Justification	1044

Expected Result	1044
Actual Result	1044
Pass or Fail?	1045
Testing Criteria (3.4)	1046
Justification	1046
Expected Result	1046
Actual Result	1046
Pass or Fail?	1047
Testing Criteria (3.5)	1048
Justification	1048
Expected Result	1048
Actual Result	1048
Pass or Fail?	1049
Testing Criteria (3.7)	1050
Justification	1050
Expected Result	1050
Actual Result	1050
Pass or Fail?	1051
Testing Criteria (3.8)	1051
Justification	1052
Expected Result	1052
Actual Result	1052

Pass or Fail?	1053
Testing Criteria (3.9)	1053
Justification	1054
Expected Result	1054
Actual Result	1054
Pass or Fail?	1054
Prototype 2 Final Code	1055
File Structure	1055
Files - Application	1060
browse.dart	1060
foryou.dart	1064
profiles.dart	1068
profilesetup_create.dart	1071
profilesetup_existing.dart	1105
profilesetup_login.dart	1111
welcomescreen.dart	1124
restaurant_customise.dart	1129
restaurant_main.dart	1204
filtersort.dart	1238
locationselection.dart	1258
setupverification.dart	1281
cartservice.dart	1284

dataencryption.dart	1286
localprofiles_service.dart	1288
queryserver.dart	1295
setselected.dart	1296
theme.dart	1298
browse_categories.dart	1313
elements.dart	1323
profiles_buttons.dart	1324
profiles_selection.dart	1326
search.dart	1341
genericloading.dart	1343
navigationbar.dart	1345
restaurants_list.dart	1350
topbar.dart	1382
main.dart	1386
Files - Server	1389
favouriterestaurant.php	1389
favouriterestaurantdata.php	1391
favouriterestaurantlist.php	1393
itemcustomise.dart	1394
login.php	1396
register.php	1398

restaurantlist.php	1401
restaurantmenucategories.php	1403
restaurantmenuitems.php	1405
Summary	1406
Evaluation	1406
Prototype 3	1407
Design	1407
Success Criteria	1407
Concept Art	1407
Cart	1407
Checkout	1408
App Pseudocode	1410
Cart	1410
Variable Info	1419
Checkout	1421
Saving cart data	1421
Data Validation	1426
Development	1426
Modifications to saving cart	1426
SQLite Modifications	1426
Creating the Cart	1429
Getting Cart Profiles	1432

Displaying Cart Profiles	1435
Getting Item Data	1440
Displaying Item Data	1456
Removing Items	1479
Displaying price	1482
Displaying item price	1483
Displaying total price	1487
Remaking the Cart	1494
Displaying the profile price	1530
Displaying subtotal	1534
Checkout verification	1536
Check for One Restaurant	1537
Minimum Order Price	1541
Fixing Price Quantity	1546
Delivery Distance	1550
Cancel Button	1551
Save Warning	1552
Clearing the Cart	1556
Sending Cart Data to Checkout	1557
Checkout	1560
Delivery Destination	1560
Restaurant Location	1566

Tipping	1582
Displaying the price	1601
Sending Cart Data to Server	1606
Processing Cart Data	1608
Fixing cartinfo Exports	1613
Processing New Cart Data in Server	1618
Responding to order creation	1625
Post-Development	1629
Testing	1629
Testing Summary	1629
Testing Criteria (1.3)	1631
Justification	1631
Expected result	1631
Actual result	1631
Pass or Fail?	1632
Testing Criteria (5.1)	1632
Justification	1632
Expected Result	1632
Actual Result	1632
Pass or Fail?	1633
Testing Criteria (5.2)	1633
Justification	1633

Expected result	1634
Actual result	1634
Pass or Fail?	1634
Testing Criteria (5.5)	1635
Justification	1635
Expected result	1635
Actual result	1635
Pass or Fail?	1636
Testing Criteria (5.10)	1636
Justification	1637
Expected Result	1637
Actual Result	1637
Pass or Fail?	1638
Testing Criteria (5.12)	1638
Justification	1638
Expected Result	1638
Actual Result	1638
Pass or Fail?	1639
Fixing Issues	1639
Pass or Fail?	1642
Prototype 3 Final Code	1643
File Structure	1643

Files - Application	1649
cart.dart	1649
cart_service.dart	1695
checkout.dart	1700
Files - Server	1732
orders.php	1732
Summary	1736
Evaluation	1736
Project Analysis	1736
Final Testing	1736
Incomplete Prototypes	1737
Prototype 4	1740
Prototype 5	1741
Prototype 6	1741
Prototype 7	1742
Success Criteria Test Summary	1742
Testing	1751
Test No. 1	1751
Test No. 2	1752
Test No. 3	1755
Test No. 4	1757
Test No. 5	1757

Test No. 6	1759
Test No. 7	1759
Test No. 8	1760
Test No. 9	1762
Test No. 10	1764
Test No. 11	1765
Test No. 12	1768
Test No. 13	1769
Test No. 14	1770
Test No. 15	1771
Test No. 16	1771
Test No. 17	1772
Test No. 18	1773
Test No. 19	1777
Test No. 20	1781
Test No. 21	1781
Test No. 22	1787
Test No. 23	1788
Test No. 24	1789
Test No. 25	1789
Test No. 26	1790
Test No. 27	1791

Test No. 28	1792
Test No. 29	1793
Test No. 30	1794
Test No. 31	1795
Test No. 32	1796
Test No. 33	1797
Test No. 34	1798
Test No. 35	1799
Test No. 36	1800
Test No. 37	1801
Test No. 38	1802
Test No. 39	1803
Test No. 40	1804
Test No. 41	1805
Test No. 42	1806
Test No. 46	1807
Test No. 47	1808
Test No. 48	1808
Test No. 49	1810
Test No. 50	1811
Test No. 51	1812
Test No. 52	1813

Test No. 53	1814
Test No. 54	1815
Test No. 55	1816
Test No. 56	1817
Test No. 57	1818
Test No. 58	1820
Test No. 59	1821
Test No. 60	1822
Test No. 61	1823
Test No. 62	1823
Test No. 63	1824
Test No. 64	1824
Test No. 65	1825
Test No. 66	1826
Test No. 67	1827
Test No. 68	1828
Test No. 69	1829
Usability testing	1829
UI Analysis	1829
Homepage	1830
Welcome Page	1831
Profile Creation	1832

Login Page	1833
Profiles Page	1834
Location Selection	1836
Browse Page	1838
Restaurant Page	1840
Item Page	1842
Cart	1847
Checkout	1850
Testing	1853
Black Box Testing	1853
Issue 1	1853
Issue 2	1856
White Box Testing	1858
Issue 1	1858
Issue 2	1859
Issue 3	1859
Stakeholder Judgement	1860
Group 1: Experienced Customer - Sam R	1860
Group 2: Customer with No Experience - Stevie T	1860
Group 3: Customer With Dietary Needs - Scott P	1860
Group 4: Restaurant Manager - Paddy R	1861
Evaluation of Maintenance	1861

Limitations	1861
Summary & Conclusion	1862
Prototype 1A	1863
Prototype 1B	1863
Prototype 2	1864
Prototype 3	1865
Final Code	1866
Client App File Library	1866
Structure	1866
Files	1869
main.dart	1869
cart.dart	1872
checkout.dart	1918
browse.dart	1951
foryou.dart	1955
homepage.dart	1956
profiles.dart	1959
profilesetup_create.dart	1962
profilesetup_existing.dart	1991
profilesetup_login.dart	1997
welcomescreen.dart	2009
restaurant_customise.dart	2015

restaurant_main.dart	2089
filtersort.dart	2123
locationselection.dart	2141
setupverification.dart	2163
cart_service.dart	2166
dataencryption.dart	2170
localprofiles_service.dart	2172
queryserver.dart	2179
setselected.dart	2180
theme.dart	2182
browse_categories.dart	2197
elements.dart	2212
profiles_buttons.dart	2213
profiles_selection.dart	2216
search.dart	2233
genericloading.dart	2235
navigationbar.dart	2236
restaurants_list.dart	2241
topbar.dart	2267
Server Database	2273
Structure	2273
Server File Directory	2281

Structure	2281
Files	2282
cartiteminfo.php	2282
favouriterestaurant.php	2286
favouriterestaurantdata.php	2290
favouriterestaurantlist.php	2293
itemcustomise.php	2296
login.php	2299
orders.php	2302
register.php	2308
restaurantmenucategories.php	2314
restaurantmenuitems.php	2316

Research & Analysis

Problem Identification

Current Issue with Food Delivery Apps

Through my use of many food delivery apps, I noticed one key factor that made it difficult to use the apps, the lack of a way to see what I could eat. Especially when I was vegan, these apps made it hard to find new places that were also vegan friendly without looking at each restaurant one at a time. By the time I would find what I was looking for, the usefulness of a food delivery app was eliminated. It wasn't quick nor was it easy.

For people with severe allergies, it makes it close to impossible to order food via a delivery app, with the fear that they are ordering in the blind and although new apps show if something is vegan or vegetarian, it still doesn't show if it contains gluten, nuts etc. As well, people with severe allergies still have to keep their guard up because they may misclick and order something that they are allergic to and therefore causing them to go to the hospital or die. To make things worse is that on most occasions, people order food when they want things quickly and are away from home, without an EpiPen, therefore, making them be in the most vulnerable state.

Some apps do something to help with this by creating a whole category for each dietary need but once again this stops the ease of use because you are left without a way to sort by categories like Italian or Chinese or find specific foods. I want to solve this by allowing users to hide any meals they can't eat, leaving them with only the ones they can eat. Along with the difficulty of finding vegan restaurants, I also ran into another issue when using these food delivery apps: there was no way to see what each person thought of the dish, only a way to rate the restaurant, making it difficult to know what was good and what was bad. To solve this issue, I intend to implement a system that allows each person to rate the dish they ate, and will then recommend it to other people with similar needs and who like similar dishes.

Ways of Solving these Issues Computationally

Abstraction

When it comes to creating a food delivery app, the hardest part is the recommendation algorithm used to help people quickly order foods and find new foods to eat. The biggest problem is that humans are complicated, with some people just wanting to have the same types of food day in and day out while others want to try new things and try different restaurants and cuisines. To counter this, I will make two different algorithms that will give two lists, one that will recommend new foods to try that people similar to them liked and the other will recommend foods similar to what they ate and people similar to them liked as well. When searching categories, these will be put together in one list to give a choice to the user.

My goal for the app is for it to use machine learning to compare the general groups of each dish the user likes. Because comparing specific ingredients will be computationally not possible due to the overall database size and the number of unneeded ingredients, I will reduce the size of the database by using key categories. Since using just categories will not be accurate to get a good recommendation I will use a rating system that will allow people to rate the food out of 5 stars and select the most relevant statement to describe their food. These may include statements like; It was spicy, I would eat this again, etc.

Reducing the size of the database is key to having an efficient program especially because of the number of ingredients that are in each dish and the total number of dishes with an ever-expanding number of restaurants. To reduce the size, I will allow the restaurant owner to select from a list of main ingredients and categories like the dish's country of origin, type of meat and the ingredients that people are allergic to. I will not be including all the ingredients like spices due to the size the database would need to have all ingredients including spices that do not influence the recommendation process. The use of only key ingredients and dietary needs ensures that the system can both work effectively and compare dishes more easily.

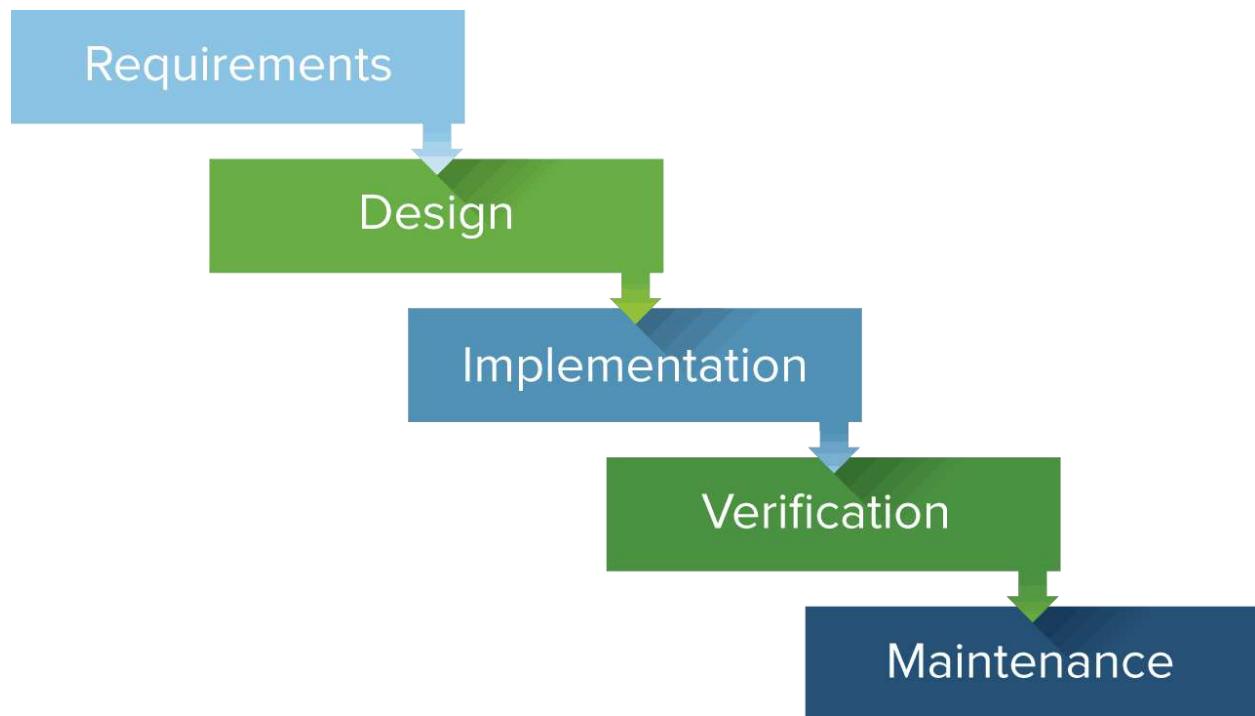
Decomposition

By use of decomposition, I can break the problem down into smaller tasks:

- I need to make a third normal form database that will hold the allergies/dietary needs, user login, food categories, restaurants and the food that will be ordered.
- The user should be able to create an account and select their allergies/dietary needs or log in with their preferences saved
- Then I need to make an algorithm to recommend food to the user dependent on foods they like, their dietary needs, what people similar to them like giving them the choice of either picking something new or something similar to what they ate before
- The user should also have the option to quickly share a scannable code to another person who is using the app to order that will input the dietary needs into the order and stop anyone from ordering anything they cannot eat.
- There should also be a way to order for a guest who doesn't have the app, giving them a quick way to input their dietary needs and allergies
- Before making the order, the app should be able to give an approximate time that the food will arrive
- After the order is made, the app should show the progress and a live map of the delivery driver and send updates when they collect the food from the restaurant and when they drop the food off at the destination
- After 3 hours after eating, the app should ask the user to rate the food and give feedback

Development Methodologies

Waterfall



<https://www.neigerdesign.com/insights/blog/blog-post/item/waterfall-vs-agile-in-a-professional-design-office>

The waterfall development methodology is a linear-sequential cycle model, where the developer gets the program requirements at the start from the stakeholders. From there the developer uses the requirements and ideas gathered and works through the steps (Analysis, Design, Implementation, Evaluation, Maintenance), and only at the end of one step, can the next to be started - if any changes must be made once past the step, the cycle has to start again.

Advantages

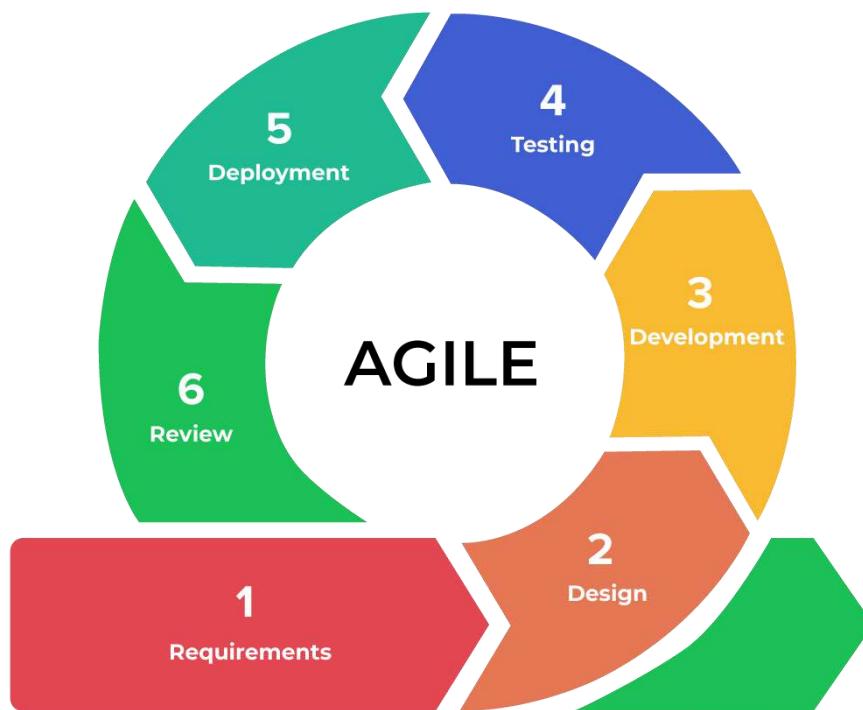
- Has a clear endpoint of where the software is complete
- Easy to understand, using a clear and intuitive structure
- Highly methodical, allowing information to be easily transferred.

Disadvantages

- Changes are difficult to make, the cycle has to restart if a change needs to be made

- There is no working software until later in the cycle
- Cannot accommodate changing customer requirements
- Difficult to judge when the stage is complete

Agile



<https://globalpricing.com/articles/gpis-agile-approach-to-project-management-helps-to-ensure-project-success/>

The agile development methodology is based on iterative development, where customer requirements change throughout the development cycle. Each cycle is called a sprint where a rough design gets refined after each sprint to result in the final product.

Advantages

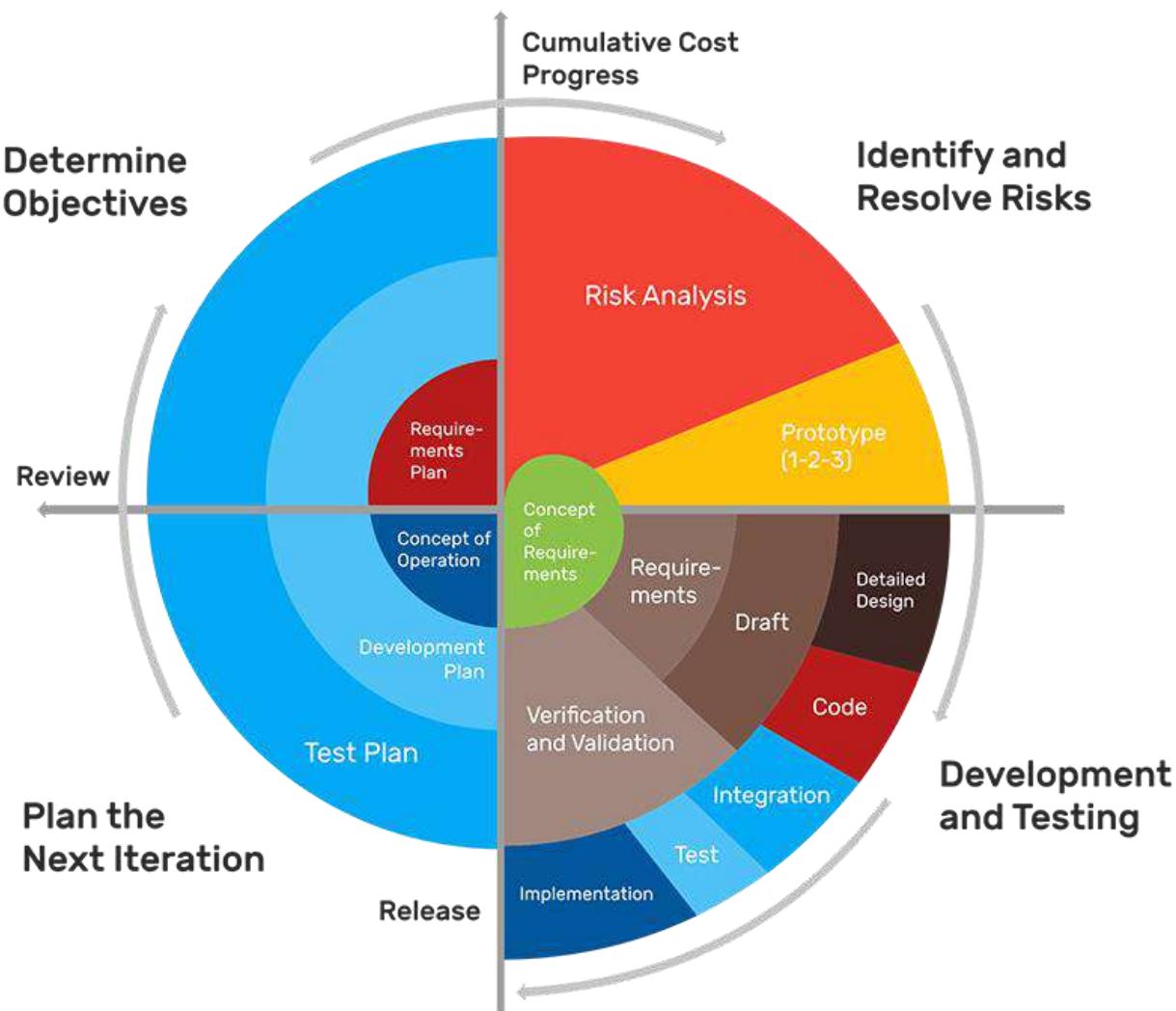
- Iterative development means that the design of the software can continually improve without needing to restart development
- With more improvement to the software, it means that the final quality will be high since there are continual discussions with the stakeholder
- Since the software is developed, it means that the stakeholder gets to see the product and make feedback quickly and the developer can make quick changes to show the customer

- With the quantity of feedback and quick iterations, any mistakes or errors can be fixed quickly

Disadvantages

- Because it uses cycles, there is no finite endpoint to the development project which makes it harder to determine when to end
- The documentation is often less detailed due to the number of changes that are made throughout development. With methods like the waterfall methodology, as development continues, documentation is made to show what each part does and how to maintain the program. With agile, because it runs in cycles, there are usually changes that make old documentation invalid.
- When using agile, updates are usually fragmented, with some cycles taking more time than others

Spiral



<https://techreceptives.com/development-methodology-spiral>

The spiral development methodology incorporates both the iterative development from agile and some aspects of the linear waterfall methodology. The spiral works similarly to agile, continuing the development and improving if an alteration needs to be made. The number of spirals depends on the size of the project, projects being able to be done gradually, being refined over time.

Advantages

- It is useful for projects on a large scale, where the software needs to be broken down into smaller tasks
- The software is made early in the process
- Since the software is made early, it allows the stakeholder to give feedback to the developers on what to change and improve, able to build on the initial customer requirements and add more features
- Since everything is being added to the initial software, the development is much more developed and has a higher standard compared to the agile methodology
- For projects that are high risk, spiral allows the initial software to be made, with the essential features made at the start and the rest built onto it

Disadvantages

- When it comes to small projects, the spiral methodology is one of the worst methods, requiring a high amount of resources compared to other methodologies, needing lots of loops where the process repeats
- Since things are built onto the project, there is no endpoint, making some projects be in development indefinitely
- On top of this, time management may be hard to determine since phases may make different amounts of time to complete and may need to loop again to get the features required

Choice of Methodology

When choosing my methodology of choice, there are a few things that must be true to my project, it can only take a maximum of 1 year to complete and I need to be able to make quick changes since this is made by one individual. For these reasons, I will be picking the agile development methodology. Its ease of making iterations of the code while contacting the stakeholder makes it great as I will need to talk to the customers and restaurants to ensure that the design and features of the program make it intuitive and has everything needed for restaurant owners to pick this app over other competition apps. On top of this, documentation is key for this program, requiring restaurant managers to be able to easily interact with the backend of the app without struggle and although there will be many iterations to the program, there should still be a good amount of documentation, paying attention throughout the development process to the ease of use.

Stakeholders

When it comes to my project, I preferably need to select people who would use the application including the front-end customer and the backend restaurant owners who will use the application as a link between their restaurant and the customer. To get the best out of the interview, I will select a variety of customers with a variety of experiences with food delivery apps and people with a variety of allergies.

I have grouped each stakeholder into a group, each having a completely different use in the project. The first group is the experienced customer; this customer has used food delivery apps and knows what to expect and so will be used to find out what the app is missing or not polished compared to other food delivery apps. They will also be useful for general testing since they know how to order food and don't need help around the app. The second group is a customer with little to no experience with food delivery apps; this customer will be useful for testing the intuitiveness of the app, helping me to find any features that don't have a good user interface. The third group is a customer with a dietary (or allergy). This will be useful to help find out what people with allergies need to know when getting food and understand how to make the dietary needs sections more accessible and easier to use for the customer. The final stakeholder group is restaurant owners; when setting up a food delivery app, the top priority is to convince a restaurant to use your app over customers and put in the effort of inputting the menu into the app. The restaurant owner stakeholder will help to determine key features needed by the restaurant to get an order and update the order status.

Group 1: Experienced Customer - Sam R

Stakeholder Justification

Sam is a college student and has lots of experience with using food delivery apps, and has tried out a variety of food delivery apps. I have chosen Sam to represent the group of people who will be coming from existing food delivery apps and will want a similar experience and not need to relearn anything that other apps have. Because he also has experience he can point out any key features that I have missed that other apps have and will be vital to ensure that they can be at the same level as other apps with their clean appearances and intuitive design.

Q&A

1a. What food delivery apps have you got experience with and what do you like and dislike about each one?

I have only tried out two delivery apps, JustEat and Deliveroo.

JustEat has a lot of local restaurants which other apps don't have but because of this, it can mean that more places are not good. I found that especially true for many places that serve cheap food, as usual, their standards are lower. One time they mixed up my order and other times they missed out on something although most of the time they were fine. As well the amount of restaurants makes the app more crowded and harder to find what I am looking for but overall the app works as intended even if it doesn't look the best. Sometimes though, I find that the app shows you places outside your area. Overall I have only ordered 8 times before so it might be because I haven't ordered enough and it is still recommending me bad restaurants

Deliveroo isn't as good as JustEat. It doesn't look as professional as JustEat with inconsistent designs and worse images although it does have a lot of deals and gives you better recommendations compared to JustEat. Some parts of the app are better, for example, the Deliveroo app allows you to easily switch from pickup and delivery so that you can choose depending on if you are going out or not. Even though I have only got food twice for Deliveroo, it still gives good recommendations.

1b. When picking your food on delivery apps, what factors do you consider?

To first find the food, I like to sort it by category. Most of the time, I go for pasta and pizza dishes. Once I find something I like, I check the rating of the restaurant so that I know it comes from a good restaurant, after which I check if it fits my price budget and distance from me because I hate waiting for over an hour for something like pizza otherwise it isn't worth it.

1c. What features do you think are missing in most food delivery apps?

In my opinion, the recommendations the apps give me are usually quite poor and the app has no idea the difference between the times of the day. As well, the number of options makes it hard to choose and sometimes I just give up and pick the first option.

2. How easy was it to order food on a delivery app for the first time?

It's hard to remember but I remember it being overwhelming, with so many buttons and things to do. I got the hang of it eventually but you don't get any way of how to use the app. Some apps work differently from others which makes them hard to use.

3. What do you like and dislike about ordering in person?

When you order in person, it is very easy to order food, requiring you to just say what food you want off of the menu although sometimes it can be harder to order in person if there are a lot of people and only one person can order for the table.

Another annoying thing about ordering in person is the fact that when you order food you have a long period when you are waiting for the food to be made. During this time, you have nothing to do but either wait, talk or check your phone which after 45 minutes, sometimes gets hard to wait. This is one of the biggest downsides of ordering in person, slowly getting more hungry. On the other hand, when you order online, you have an estimate of how much time you have until it arrives and can easily do something else while waiting or even just order ahead and not need to wait at all while you do an activity. That is why I now more commonly get food online rather than heading over to a restaurant and having to wait, especially when ordering for myself.

4. Would it be easier to have a website or an app for food delivery?

It honestly doesn't make a difference for me, if it is a website, I can just bookmark it and it means that I can order on the desktop. If it is an app, it would just add to the growing collection of apps

Group 2: Customer with No Experience - Stevie T

Stakeholder Justification

Stevie is a student and has very little experience with food delivery apps. I have chosen Stevie to represent the group of people with no experience as it will help get a better understanding of how to make the interface intuitive and easy to understand without much help. As a result, it will help me know which features are missing from when ordering in person compared to ordering on the app. Stevie has no allergies and likes Chinese and Indian foods

Q&A

1a. What food delivery apps have you got experience with and what do you like and dislike about each one?

I don't have any experience with food delivery apps. I usually get other people to order food if ever we get food. I understand the basics of food delivery apps though; you add the foods you want to a cart, pay, and just wait until the driver brings you your food from the restaurant. My friends use UberEats and JustEat and overall I like JustEat better because it usually arrives quicker.

1c. What features do you think are missing in most food delivery apps (that makes you not use them)?

There are a few features that are missing including no way to sort by spice level and ways of excluding some ingredients and meats dependent on dietary needs like if they are gluten-free or vegan. When you order in person, every person can look at the menu and then say what they want to the waiter but unlike in real life, when you are ordering you have to look at a single phone and order from one phone. It would be so much better if each person could choose what they want from their phones and have it so that one person could then pay and just pay for everything on one phone, especially since the delivery cost gets expensive fast if you have each person ordering by themselves. I think it is £3 for delivery.

2. What is the biggest hurdle that stops you from ordering food on a food delivery app?

There is no reason to have it. I have to order from one phone anyway and usually, I am with a group of people. If I want to check the menu I can just look at the website menu, it saves the hassle of having so many apps

3. What do you like and dislike about ordering in person?

The waiting times are usually the worst when ordering in person because it is usually time without any distractions so it makes you think about your hunger although that also gives time to talk to people. In such a busy world, it gives us time to chat and talk about our lives and interests and then once the food arrives we can just eat

4. Would it be easier to have a website or an app for food delivery?

Because of the amount of storage that some of these food delivery apps take, it is very annoying and the number of food delivery apps means that it just keeps filling up my storage. I only have 64GB of storage and compared to my friends who have 128GB and 256GB of storage, I always seem to run out of storage. I would much rather have a website so I can just get it online especially since I don't get food very often. With a website, it means I can also

Group 3: Customer With Dietary Needs - Scott P

Stakeholder Justification

Scott has some experience with food delivery apps but has a milk allergy. Unlike people who are pescetarian or vegetarian, having an allergy to milk makes it harder to select foods as lots of foods may have milk incorporated into an ingredient. This is why I chose Scott to represent this group, as he will be an extreme version of someone with a dietary need.

Q&A

1a. What food delivery apps have you got experience with and what do you like and dislike about each one?

So far to this date, I have used JustEat and Deliveroo because it is what other people have been using. There are a few things I like and dislike about each of them. Generally, they all look good and are easy to use, although finding what is in each dish is hard on both apps since it just says to either go to the restaurant site or call the restaurant which makes it a big hassle. As well, sometimes I find it hard to tell which food is mine after I get the food.

1b. When picking your food on delivery apps, what factors do you consider?

Usually there are a few factors that go into choosing my food. The first is if I can eat it which takes the longest to see if it is true, having to go through phone calls and waiting to see if the dish I want doesn't have milk (usually they say it does). The second is if it is something I like so long as it has good reviews and is the type of food I am looking for then I will try it.

1c. What features do you think are missing in most food delivery apps?

If there was a way to show the ingredients of each dish or even better if it just didn't let people choose foods that contain ingredients that you are allergic to.

3. What do you like and dislike about ordering in person?

For me at least, I find that ordering in person is so much easier since there is no need to go through getting a restaurant's phone number and calling them only for them to not have the food I am looking for. As well, in person, I can make special requests like leaving out an ingredient which food delivery apps are not able to do. There is one bad thing about ordering in person and that is if they don't have any foods that don't have milk it means we have to find a new restaurant which means travelling elsewhere.

4. Would it be easier to have a website or an app for food delivery?

I feel like a website could be good and an app. I mean if you can, make both because why not.

Group 4: Restaurant Owner - Paddy R

Stakeholder Justification

Paddy works at a small Italian restaurant, and although he is not the owner, he is the restaurant manager and works to expand the restaurant business. He is looking to expand the business to food delivery. He has tried UberEats but is willing to help out with this project.

Q&A

1. What features would help you to add your menu to the app?

When adding menu items, there should be a way to add an image, description, category, pricing, when it can be ordered, and VAT. Something that would make it better is having a user interface to move around dishes instead of just changing the category to change the order.

2. What features would help you to track and change the food status of the app?

If it was possible, there should be a way to have both a backend for managing the restaurant and one for a Point of Sale system for managing orders. Ideally, it should be compatible with an iPad so it will work seamlessly with the current system. It should allow you to see all the orders currently in progress with what needs to be made next to them so that it doesn't require touching the iPad.

3. Would it be easier to have a website or an app for a point of sale system?

It would be best if it was an app since there is no way for people to accidentally change pages.

Researching the Problem

Existing Solutions

When it comes to food delivery apps, the market is extremely saturated, with most doing the basics, allowing you to pick from a variety of restaurants and get it delivered within an hour.

With the number of apps on the market, it makes it hard to choose the best one for you with some giving perks like discounts while others give quick deliveries. Each has its positives and negatives to each other but what all of them do not have, is overall useability for people with allergies (nearly 11% of the US population) and the additional 14% of the UK population that are vegetarian and vegan.

Examining the current food delivery apps and talking with people with experience with food delivery apps, it was concluded that the current food recommendation systems are still not fully adequate for most people especially people with food allergies and dietary needs. For them, food delivery apps still give recommendations of foods they are allergic to and do not have options to exclude certain ingredients meaning that they need to use trial and error to find what they are looking for. Some recommendation systems also do not take into account times of day and the day of the week, instead just relying on food previously eaten.

At this current time, there are only a few advanced food delivery app recommendation systems that are good, including [Square's Caviar](#) app and [Uber Eats](#). While Uber Eats uses a slightly more conventional way of recommending foods, using a multi-factor algorithm graph to make recommendations to the user, Caviar uses machine learning with the experience of others similar to the customer to make recommendations. Although I will not be able to do machine learning due to the limited number of testers, I will attempt to make the algorithm easily expandable for when the number of users increases, making more accurate guesses of what you would want.

Examples of Existing Solutions

Natasha's Law

Natasha's Law took effect on October 1, 2021. On all pre-packaged foods, places will have to put a full-ingredients list of what is in the product. This change was put in place after a teenager by the name of Natasha Ednan-Laperouse died from an allergic reaction after eating sesame seeds that were inside a baguette. PPDS (Prepacked for Direct Sale) food must clearly show the following information: Name of the food & ingredients list, allergic ingredients shown in bold.

Just Eat

Frontend

Mobile App

The screenshot shows the Just Eat mobile application interface. At the top, there is a header with the location 'HP4 3RN' and a battery icon showing 17%. Below this is a search bar with the placeholder 'Search for a dish or restaurant'. To the right of the search bar are buttons for 'Choose cuisines' and 'Refine results'. The main content area displays a grid of restaurant cards under the heading 'Tasty offers'. Two cards are visible: 'Flamingos Sizzlers Grill' (Kebab, Burgers, Low Delivery Fee, 0.7 miles, 4.5 stars, 1584 reviews) and 'Papa Jc' (0.7 miles, 4.5 stars, 1.2 miles). Below this section, there is a heading '24 open restaurants' with a card for 'Quick Bite' showing three pizzas. To the right of the cards are two buttons: 'View 33 restaurants' and 'View 33 restaurants'. Further down the page, there is a 'Popular cuisines' section with cards for Kebab, Burgers, Sandwiches, Breakfast, Pizza, Indian, British, and Chinese. To the right of this is a 'Filters' sidebar with options like '5+ Stars', 'Open Now', 'Free Delivery', 'Special Offer', 'Delivery', 'Collection', 'New', and 'Hygiene Rating 3+/Pass'. Below the filters is a 'Sort by' section with options like 'Best match' (selected), 'Avg. review', 'Nearest first', 'Minimum order', and 'Delivery fee'. At the bottom right, there is a virtual keyboard with the word 'kebab' typed in, along with other letters and symbols.

The screenshots show the following sections:

- Top Left:** A grid of three burger and fries photos. Below is the restaurant's name, "The Meating Room", category "Burgers - British", and a 5-star rating with "View 1090 reviews".
- Top Middle:** Similar to the first, but with a "Do you have a food allergy?" pop-up overlay.
- Top Right:** Shows the time "16:54", battery level "20%", and a review summary: "1090 reviews" with a 5.25/6 average based on 1090 reviews.
- Middle Left:** Delivery information ("Delivers in 30-45 mins"), collection option ("I want to collect"), delivery fee ("£1 delivery fee"), and minimum order ("£20 min. order").
- Middle Middle:** A "Do you have a food allergy?" pop-up with instructions to call the restaurant if ordering for someone with allergies.
- Middle Right:** A "Meating Room" burger listing with price "from £10.50" and description "House burger". It includes allergen information: "Beef patty, gem lettuce, grilled beet tomato, fried onion, applewood smoked cheddar, turkey bacon & BBQ chilli sauce. May contain traces of allergens."
- Bottom Left:** A search bar and a grid of burger options from "Hand Crafted Burgers St.Peter" and "Hand Crafted Burgers Meating Room".
- Bottom Middle:** A "Add Extra" dropdown menu showing "Medium" selected, with other options: "Medium rare", "Medium Well Done", "Rare", and "Well done".
- Bottom Right:** A "More" button followed by a list of reviews:

 - Julian:** 5 stars, "Isabella", 18/03/2022 19:47
 - Ryan:** 5 stars, "Todd", 16/03/2022 18:25
 - Laura:** 5 stars, "Laura", 13/03/2022 18:53
 - Emma:** 5 stars, "Emma", 13/03/2022 17:23

Desktop Website

The screenshots show:

- Left Screenshot:** The main search interface with various food categories like "Dinner", "Burgers", "Sandwiches", etc. Below is a list of restaurants including "The Meating Room".
- Right Screenshot:** A detailed view of "The Meating Room" burger listing. It shows the burger with fries, the "House burger" description, price "£10.50", and a "Get it delivered" button.

Backend

Order history

Search by order number: 1234567

Find order

Choose the start date and how long a period to view orders
You can only view orders placed in the last 30 days.

From: 01/07/2020 Period: 1 Day

View orders

Today

Orders: 0 Sales (inc. Delivery Fee): £0.00 Status: OPEN

Close for the day

Edit your menu

+ Add

- Beverages: 6 items
- Appetisers: 19 items
- Soups: 6 items
- Aromatic Duck Dishes: 1 item
- Szechuan Dishes: 6 items
- Kung Po Dishes: 7 items
- Braised Dishes: 4 items

Reviews and ratings summary

★★★★★ 4.9 / 6

Average Based on 541 ratings

All reviews %: 96

Click here to see ratings breakdown

Customer reviews

- John from London: ★★★★★ 05/09/2019 Order #: 353264596
- John from London: ★★★★★ 05/09/2019 Order #: 353264597
- John from London: ★★★★★ 05/09/2019 Order #: 353264598

15:19

☰ Menu

3 Preparing

4 Delivering

2 Done

21 mins BS13 8RX
134 Stuart Avenue, Bristol
Paid #288 371 222 On its way >

24 mins BS7 2LP
22 Claredon Street, Hengrove, Withywo...
Paid #288 371 222 On its way >

27 mins BS1 4DJ
16 Queen Street, Corn Town, Bristol
Paid #288 371 222 On its way >

Summary

Just Eat is a food delivery application with the primary goal of ordering food quickly from restaurants. The ease of use and the simplicity means that many local restaurants can easily become a partner and show their restaurant on the app. I have chosen JustEat as one of my selected applications because it was mentioned by multiple of my stakeholders and was complimented for its speed but people found that the number of restaurants made it hard to choose.

Design

Just Eat uses a basic colour scheme, black and white with an accent colour of orange (#ff8000). Its minimalist colours help to make it more intuitive for people who sometimes find it hard to understand the difference between interactive buttons and a basic block. The overall app is fairly easy to understand, with extremely similar looks to other apps like Deliveroo, Uber Eats and other food delivery apps. This works both in favour and against Just Eat. Because it is so similar it helps the user seamlessly transfer from one app to the other without needing to relearn the layout although, on the other hand, it means it doesn't stand out and makes it hard for people to understand which app is which and "wow" the user.

Good Design

- Consistent colour scheme
- Easy access to categories and filtering
- Large restaurant images to show the type of food they serve
- Rounded corners to soften edges
- Overlapping blocks to add depth
- Info about the restaurant including wait time and reviews are front and centre
- Offers shown in a bold colour (orange on white background)
- Switch on checkout to select a collection
- Most popular dishes shown first
- Info icon shows hygiene rating, location and opening time

Bad Design

- Selecting your location opens a new window with just a search bar with many icons
- To get to your account, you need to go to the location page and press the hamburger menu and can then see your orders and can log in
- Discounts are found under the submenu with the name 'For You' which is not intuitive

- Info about the restaurant doesn't include a phone number or a website to contact the restaurant directly
- Very few categories only some have images
- Filtering buttons are small and are cramped on the left side of the screen
- Because of the number of restaurants and the size of each restaurant summary block and image, hard to find what you are looking for and is hard to compare restaurants because of the distance you need to scroll between them

Features

The main feature of Just Eat is the speed at which you can order food, needing to only click a few times to get a full order. This is done by including the needed information summarised and having buttons on the side to give you more information if you need it. For example, each category only gets 2 lines, one for the type of dish and the other for giving you some of the dishes. Once you select it, it gives you a choice of dishes which are then added to your cart.

Another big feature of Just Eat is the ability to sort by Best Match, which bases recommendations dependent on user feedback, ratings and frequency of ordering from a restaurant. This feature is useful for finding things similar to what the user has done before but not for finding new restaurants that are not the same as what they have tried before. This safe algorithm means that people often end up being stuck having the same type of food and not getting recommended anything different until they sort by a different filter.

Just Eat's offers are big and bold on the app, being one of the first things you see when selecting a restaurant. If a restaurant has an offer, it shows it under the summary in orange and allows you to easily compare different offers.

The delivery tracking is another feature by Just Eat, giving you a timeline of key events of where your food is and an estimate of the time of arrival.

Strengths

Thanks to the simple colour scheme and large images with quick summaries, it makes using Just Eat a breeze, with few buttons to click and intuitive buttons, its simplicity means that it can run on a variety of devices while still being quick. On top of this, those with less experience with food delivery apps can still easily move around the app, using the orange as a key to which are buttons and which are basic blocks. The app also has the "best match" filter which helps users to find new restaurants dependent on previous orders made. Under the info tab, you can also get information about the hygiene rating and location of the restaurant so you know if the food is safe.

Limitations

In contrast to the strengths the app brings, there are also a lot of big limitations of the app including the little thought put into people with allergies and although not as large as people without allergies, it is still a large audience who has to look elsewhere. The only thing that has been added is the option for restaurants to add a little box giving them a number to call (which makes the whole app pointless if there is no ease of use), and a place to add documents about their food nutrition which is not very intuitive. The lack of care towards people with food allergies makes it a big turn-away since the whole app is useless if you have an allergy, needing to look at outside sources to know what they can eat.

There has also been little attention to the small parts of the app like the account section where if you want to change your details or look at previous orders you need to go to a special section where there is a side menu that has the details about your account. This is not very intuitive and makes it hard for people who are not tech-savvy or haven't tried a food delivery app, to find this important information.

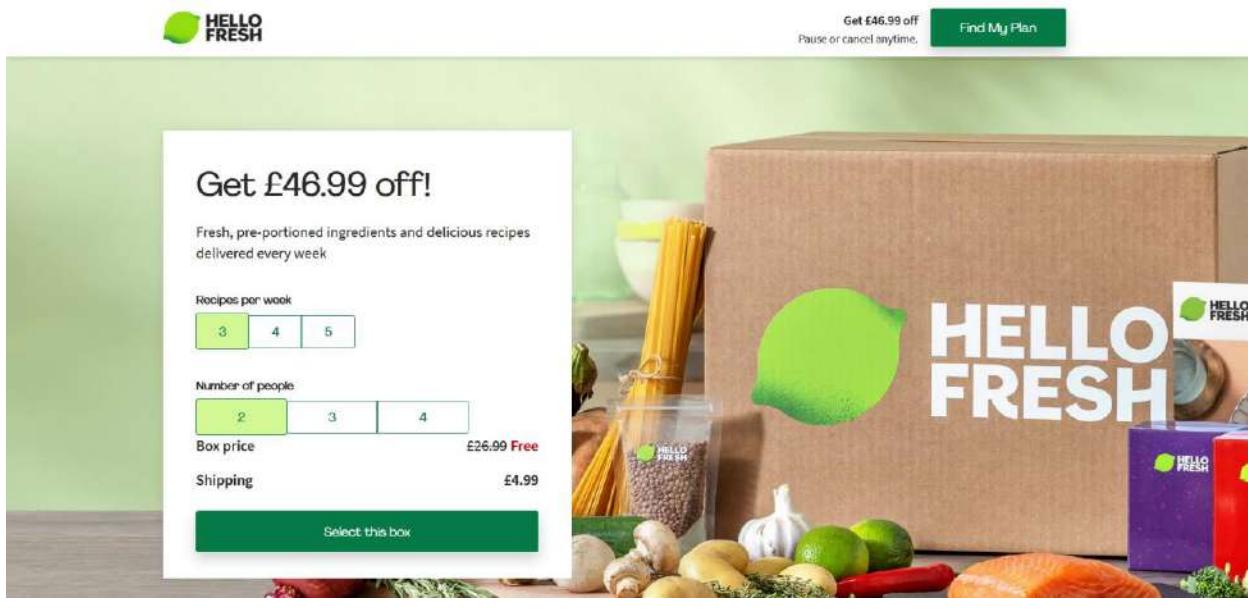
Conclusion

In summary, I need to ensure that all the visible sections are friendly for all including the parts that may not be used often. As seen in Just Eat's app, where there is little care and attention made to the front-end, it will stick out like a sore thumb and make the overall user experience worse so I will take this on board when creating my program to ensure that it is intuitive for all users.

Another take from Just Eat is how simplicity can help the user experience dramatically, with bold colours on a basic background creating a good contrast and catching the eye to indicate interactable objects. To better keep the user's attention, the depth should be created by either using outlines on rounded boxes or overlapping elements.

Hello Fresh

Frontend



This screenshot shows the "Personalize your plan" section of the Hello Fresh website. At the top, there are navigation links: "Select plan", "Register", "Address", "Payment Details", and "Select meals". Below that, the title "Personalize your plan" is centered. The page is divided into two main sections: "1. Choose your preferences" on the left and "2. Customize your plan size" on the right.

1. Choose your preferences

Your preferences will help us show you the most relevant recipes first. You will still have access to all recipes each week.

Checklist items include:

- Meat & Veggies
- Veggie
- Family Friendly
- Fit & Wholesome
- Quick and Easy
- Pescatarian

Text below the checklist: Swap proteins and sides if they aren't a perfect match. Learn more below.

2. Customize your plan size

Number of people: 2

Recipes per week: 3

Meat & Veggies (Most Popular)

3 meals for 2 people per week
6 total servings

Box price: \$59.94

Price per serving: \$9.99

Shipping: + \$9.99

Total: \$89.93

A "Feedback" link is located on the far right edge of the page.

Get started

Salmon in a Creamy Dijon Chive Sauce
with Roasted Potato Wedges and Lemony Zucchini

★★★★★ 4.5/5
6,351 Fresh Recipe Reviews

[Continue with Facebook](#)

[Continue with Google](#)

[Continue with Apple](#)

or

[Continue](#)

By submitting, you agree to receive marketing emails.
Unsubscribe at any time.

Save Time

Let us do the planning, shopping, and delivering, so you can do the fun part: cooking! And eating, of course.

Get Inspired

Each week, our chefs curate 25+ deliciously simple recipes featuring a variety of ingredients and flavors.

Be Unstoppable

Easy, step-by-step instructions to help you chop, zest, roast, and serve like a pro.

You'll save £58.99 in total [Learn more](#)

Order summary	
	3 meals for 3 people per week 9 servings at £4.99 £0.00 each
Box price	£38.99
Shipping	£4.99
Discount	£-38.99
First box total	£43.98 £4.99

Shipping

Delivery details	
First delivery:	Tue, Apr 05 08:00 - 19:00
Delivery instructions:	Front porch/front door
Subsequent deliveries:	

Select Plan Register Address Checkout Select Meals

You'll save £58.99 in total Learn more (X)

Add a credit or debit card VISA

Add a PayPal account PayPal

I accept the [terms and conditions](#) and have read the [privacy policy](#).

Yes, I'd like to receive sample gifts (including alcohol) and other offers, competitions, and news via email. By ticking this box, I confirm I'm over 18 years old.

Save money
Spend less on dinner every week.

Flexible plans
Easily cancel your subscription up to 5 days before delivery.

Place order

Order summary

3 meals for 3 people per week
9 servings at £4.99 each

Box price	£38.99
Shipping	£4.99
Discount	£-38.99
First box total	£43.98 £4.99

Shipping

Delivery details Edit

First delivery: Tue, Apr 05 08:00 - 19:00
 Delivery instructions: Front porch/front door.
 Subsequent deliveries:

HelloFriends Ilya ▾

£4.99 Save

X Select min 3 meals

Menu for Tue, Apr 5
In your box

ULTIMATE
Ultimate Bacon Cheeseburger and Chips
with French Dressed Baby Gem & Tomato Salad
40 min

Add GU Hot Chocolate Brownies £2.20 / serving

£0.99 / serving Add

CUSTOMER FAVOURITES
Mango Chutney Curried Chicken Wraps
with Wedges, Pepper and Baby Gem
40 min

Add Handmade Ciabatta Loaf £0.56 / serving

- **1 in your box** (3 servings) +

Serrano Ham and Butternut Linguine
with Tenderstem® Broccoli and Crème Fraîche
30 min

Add Handmade Ciabatta Loaf £0.56 / serving

- **1 in your box** (3 servings) +

Feedback

p59

X Select min 3 meals

£4.99 ⓘ

Save

Feedback

Other delicious meals this week



Premium
Serrano Ham Wrapped Chicken Breast
with Cheesy Truffled Roast Potatoes, Tenderstem® and C...
45 min

Add GU Hot Chocolate Brownies £2.20 / serving
£2.99 / serving



Premium
Sticky Pistachio Crusted Salmon
with Herby Bulgur Wheat, Charred Courgette and Rocket
40 min Cook together

Add GU Hot Chocolate Brownies £2.20 / serving
£2.99 / serving



STREET FOOD
Tex-Mex Style Beef and Potato Tacos
with Cheese, Refried Beans and Baby Gem Salad
45 min

Add GU Hot Chocolate Brownies £2.20 / serving
£2.99 / serving



X Select min 3 meals

£4.99 ⓘ

Save

Feedback

Add Handmade Ciabatta Loaf £0.56 / serving

Add your extras
No extras

Add Handmade Ciabatta Loaf £0.56 / serving



Beef Rogan Josh Style Curry
with Pepper, Ginger Rice and Toasted Almonds
30 min Family Friendly

Add Handmade Ciabatta Loaf £0.56 / serving



Five Spice Beef Fried Rice
with Green Beans
30 min

Add Handmade Ciabatta Loaf £0.56 / serving



SOLD OUT
Prawn Red Thai Style Curry
with Courgette and Basmati Rice
20 min Rapid • Under 650 calories

Add Handmade Ciabatta Loaf £0.56 / serving

[← Make your week even tastier. Add Extras!](#)

£4.99 ⓘ

[Skip extras](#)Yum! Your selection is saved! 

Lunch

Delicious quick Lunch recipes, ready to prep in 5 mins or less.



Ham and Cheese Pretzel Roll
with Honey Mustard Mayo | 5 min prep | Serves 1

£3.49

[Add](#)[Feedback](#)[← Make your week even tastier. Add Extras!](#)

£4.99 ⓘ

[Skip extras](#)

Breakfast

Start your day deliciously with a Breakfast recipe, prepared in less than 10 minutes.



Smoked Salmon Bagel
with Dill and Black Pepper Cream Cheese | Serves 1

£3.99

[Add](#)[Feedback](#)

Sides

Tasty side dishes to complement or add a twist to your meals.

← Make your week even tastier. Add Extras!

£4.99 ⓘ

Skip extras

Sides

Tasty side dishes to complement or add a twist to your meals.



Artisan Garlic Bread
250g | Serves 4 - 5 | Best served warm

£2.60

Add



Handmade Ciabatta Loaf
300g | Serves 4 - 6 | Best served warm

£1.69

Add



Onion Focaccia
300g | Serves 4 - 5 | Best served warm

£3.49

Add



← Make your week even tastier. Add Extras!

£4.99 ⓘ

Skip extras

Satisfy your sweet tooth with these delicious desserts.



Gü Hot Sticky Toffee Puddings Bundle
Save 10% on Gü Bundles | 4 portions

£6.40
£5.94

Add



Gü Hot Chocolate Brownies
2 portions x 76g

£3.30

Add



Gü Hot Sticky Toffee Puddings
2 portions x 85g

£3.30

Add



Feedback

Feedback

UPCOMING Change by Thu, Mar 22
You selected 3 meals

Mango Chutney Curried Chicken Wraps

CUSTOMER FAVOURITES

Preparation Time: 40 min | Calories: 925 kcal | Difficulty: Medium

1 in your box (3 servings)

Add Handmade Ciabatta Loaf \$0.56 / serving

Show details

Ingredients

- Potatoes
- Echalote Shallot
- Garlic Clove
- Green Pepper
- Baby Gem Lettuce
- Diced Chicken Thigh
- Tomato Puree
- North Indian Style Spice Mix
- Mango Chutney
- Plain Taco Tortilla (Contains Cereals containing Gluten)

Not included in your delivery

Mayonnaise

Learn more

Add Handmade Ciabatta Loaf

Feedback



Mango Chutney Curried Chicken Wraps

with Wedges, Pepper and Baby Gem

CUSTOMER FAVOURITES

A customer favourite, these Mango Chutney Curried Chicken Wraps are a tried-and-tested recipe that always wins with a crowd.

Preparation Time: 40 minutes

Difficulty level: Medium

Allergens: Cereals containing Gluten

Boxes and ingredients are packed in facilities that handles Peanut, Nuts, Sesame, Fish, Crustaceans, Milk, Egg, Mustard,

Feedback

Ingredients



450 grams
Potatoes



1 unit(s)
Garlic Clove



1 unit(s)
Baby Gem Lettuce



1 sachet
Tomato Puree



2 sachet
Mango Chutney

serving amount

2 3 4



1 unit(s)
Echalлон Shallot



1 unit(s)
Green Pepper



280 grams
Diced Chicken Thigh



1 sachet
North Indian Style Spice Mix



6 unit(s)
Plain Taco Tortilla
(Contains Cereals containing Gluten)

Nutritional information

Per serving Per 100g

Energy (kJ) 3871 kJ

Energy (kcal) 925 kcal

Fat 31.0 g

of which saturates 8.0 g

Carbohydrate 121 g

of which sugars 29.0 g

Protein 41 g

Salt 1.5 g

Always refer to the product label for the most accurate ingredient and allergen information.

Feedback

Not included in your delivery



2 tbsp
Mayonnaise

Utensils

- Baking Tray
- Garlic Press
- Frying Pan
- Bowl

Instructions

PDF



1

Preheat your oven to 200°C. Chop the potatoes into 2cm wide wedges (no need to peel). Pop the wedges onto a large baking tray in a single layer. Drizzle with oil, season with salt and pepper then toss to coat. Spread out in a single layer. TIP: Use two baking trays if necessary. When the oven is hot, roast on the top shelf until golden, 25-35 mins. Turn halfway through.

Feedback



2

Halve, peel and thinly slice the shallot. Peel and grate the garlic (or use a garlic press). Halve the pepper, discard the core and seeds, then thinly slice. Trim the baby gem, halve lengthways, then thinly slice widthways.

You might also like...



Curried Chicken Wraps
with Mango Chutney, Yoghurt and Baby Gem Salad

627 kcal · 40 minutes



North Indian Style Cauliflower & Lentil Dal
with Caramelised Onion and Spinach

475 kcal · 35 minutes



Chicken and Spinach Curry
with Basmati Rice and Mango Chutney

778 kcal · 20 minutes



Tar
wi

661

Feedback

HELLOFRESH

HelloFresh[®]
Gift cards
Unidays
Student Beans
TOM
Recipps
Blog
Affiliate
Key Workers Discount

OUR COMPANY

HelloFresh Group
Careers
Press
New Look

HELP

Find an answer
Help Centre
Corporate Sales
Partnerships/Inquiries

Download our app

Manage your deliveries from anywhere, anytime.



Win big with our Golden Lime Giveaway!



REWARD



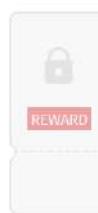
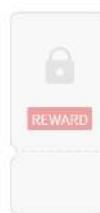
REWARD



REWARD

Yes! You've ordered your first box towards our Fresh Rewards.

Keep going! There's only 3 more orders to your first reward.



[Find out more](#)



My Menu Getting started Gift Cards

HelloFriends

Illya

HelloFriends

Invite them to HelloFresh with one of these delicious deals.

Give £47 off

Send friends £47 off, and you'll get £20 credit when they sign up using your personal code.

[send £47 off](#)



My credit: 0 credit received

discounts sent by email

free boxes sent

HELLOFRESH

HelloPerkX
Gift cards
Unidays

OUR COMPANY

HelloFresh Group
Careers
Press

HELP

Find an answer
Help Centre
Corporate Sales

Download our app

Manage your deliveries from anywhere, anytime.

[Download on the App Store](#) [GET IT ON Google Play](#)

HelloFriends

Illya

Account Settings

Account Setti

Favourite Recipes

Cookbook

Fresh Achievements

Help

Contact

Sustainability

CO2 Neutral

Log Out

Feedback



My Menu Getting started Gift Cards

HelloFriends

Ilya

Payment Methods

Plan Settings

Account Info

Payment Methods

Notification Settings

Order History

Gifts and discount codes

Your saved payment methods

Edit or add backup payment methods at any time. Additional payment methods will be used if your preferred payment method cannot be charged.

i****1@gmail.com

Classic Plan [Change Payment method](#)

[Remove payment method](#)

Feedback

Did you know?



489



My Menu Getting started Gift Cards

HelloFriends

Ilya

Order History

Plan Settings

Account Info

Payment Methods

Notification Settings

Order History

Gifts and discount codes

Date	Delivery	Price	Status
4/5/2022	Order number - 2717982842 Classic Plan Delivery Fees Total	0 £4.99 £4.99	Pending Processing

Feedback

Did you know?



489

10:11 84% 10:11 84%

Thanks - we've received your order

HelloFresh 09:44 to me

YOUR ORDER

Hi Ilya,

We've received your order and we're looking forward to delivering your first box. You'll find all the details of your order

FULL PRICE/WEEKLY ONGOING PRICE:	
Classic - 3 meals per week for £43.98 3 persons	
Discount	-38.99
Shipping	£4.99
Order price (with discount): £4.99	
	Delivery Address 63 Granville Road HP43RN
	Delivery Date Tuesday, 5 April 2022 08:00 - 19:00

HELLOFRESH APP

Summary

HelloFresh is a meal kit delivery service that ships ingredients to you weekly that contain all the ingredients needed to make the meals that you specify. First, you set your meal plan preferences with options for carnivores, vegetarians, calorie-counters and more. You can then choose from 27+ weekly recipes carefully put together by chefs. You will then receive those recipes with instructions on the day of your choosing.

Because I do not currently use HelloFresh I will be referencing sources of the experience that I am using to make my judgments.

I have chosen HelloFresh because they are similar to a food delivery app, with a variety of different foods that are categorised and that allow you to see the ingredients of each meal. Because it is a meal kit delivery service, they have spent more time on the individual meals and

how they are made, not like a food delivery service which mainly focuses on the delivery portion and restaurant sorting. I will mainly be analysing the design and how they layout the data, not the recipe portion of HelloFresh since a food delivery service will not contain that.

Design

HelloFresh is another company that also uses a clearly defined colour scheme, using only three colours but unlike JustEat, HelloFresh sticks to it and consistently only uses these colours without exception, the main colours are green (#056835), with white and black, the rest of the site uses lighter and darker shades of this green to add variation to the site and make it more clear when a button is pressed and a part is interactable. HelloFresh tries to stick to this one shade though as it is the main colour of the company logo.

The overall design of HelloFresh relies on vibrant images to portray what each dish looks like complete, with warm hues and placing the dish in different environments to better convey what they look like and the type of place that it would best be served. For example, a Hero sandwich is placed on a bench with salt and pepper and a warm hue to indicate a summer atmosphere.

Unlike many other companies, HelloFresh pays attention to detail, having a great design from first look to the end when you are cooking the food from the kit. One place where they stand out from others is their packaging, when you get your meals, it is presented in recycled materials and the correct portions. The outside box has a unique look, with large and vibrant packaging, fitting with the rest of their brand. Upon opening the box, you are presented with detailed pictured instructions set for each of the meals with the meal number on the top right. Corresponding with this, you are given 4 different bags with 3 having the number of the meal it goes with. The final bag contains the food that should be refrigerated, having eco-friendly packaging that keeps it cool. These bags make it easy to store the food as there is almost no reason to open them until you need them and can be kept in the box (except the refrigerated bag)



Features

The main feature of HelloFresh is its selection, only giving you the best, most high-quality meals. The team at HelloFresh constantly refine the menu to people's pallets, using the foods that people most pick again to make new recipes. As well, thanks to user feedback, they can account for people with allergies by giving multiple options and giving a list of ingredients showing exactly what is in it, including their nutrition. This nutrition label helps those who are diabetic, want more protein or just generally want a low-calorie diet to get what they need. When cooking at home, it is hard to know exactly what you are getting, only getting a general idea but the team of HelloFresh, because they can deconstruct the food, can give nutrition descriptions.

Another feature is the flexibility of the menu, allowing the user to change the number of portions you want as well as with some meals, being able to change the type of meat in the dish. This not only helps out if you can't eat a certain meat but also allows variety if you want to try having it again but want a slight change. Not only can a specific dish be flexible but also the entire menu can be flexible, with the ability to change from, "meat and veggies" to "veggie" or even "pescetarian". They also have a family option which makes it easy to make food that tastes good for all pallets and can be made on a larger scale.

Strengths

The main strength of HelloFresh is the overall speed of the service, its target audience is people with busy lives meaning that it has to be quick and easy to use (which it is). This is done in a variety of ways; The first way is that it gives you a limited menu, usually about 20 different dishes, helping both logically and eliminating option fatigue. The second way is by making it automatically select next week's food even if you forget so you don't end up hungry. With experience, it has worked well even when I forgot to select what I wanted, it picked some delicious food and delivered it at the same time as usual. The third way is with the actual dishes themselves; Sometimes with other recipes other than HelloFresh, I find that they don't think about the time taken to prep and move around and sometimes found myself having to do two things at the same time whereas, with this, I feel more relaxed since I can keep under control and everything is spaced out properly. The fourth way is if I was a family, it would mean that children would be able to help out since the instructions are very clear, with pictures of every step. When I first tried to cook it felt hard since the instructions were not clear, for example, it would say something like "dice" the ingredient whereas with HelloFresh they would say "cut lengthways and cut into 2cm cubes". In summary, they perform better than the competition and their simplicity, clean design and overall ease of use make it a good brand.

Before you start

Our fruit and veggies need a little wash before you use them! Wash your hands before and after prep.

Cooking tools

Baking tray, garlic press, frying pan and bowl.

Ingredients

	2P	3P	4P
Potatoes**	450g	700g	900g
Eschalot Shallot**	1	1	2
Garlic Clove**	1	2	2
Green Pepper**	1	2	2
Baby Gem Lettuce**	1	2	2
Diced Chicken Thigh**	280g	420g	560g
Tomato Puree	1 sachet	1 sachet	2 sachets
North Indian Style Spice Mix	1 pot	1 pot	2 pots
Mango Chutney	2 sachets	3 sachets	4 sachets
Plain Taco Tortilla	6	9	12
Mayonnaise*	2 tbsp	3 tbsp	4 tbsp

*Not included **Store in the Fridge

Nutrition

Per serving	Per 100g	
for uncooked ingredient:	696g	
Energy (kJ/kcal)	3871/925	
Fat (g)	31	5
Salt (g)	8	1
Carbohydrate (g)	121	17
Sugars (g)	39	4
Protein (g)	41	6
Salt (g)	1.50	0.22

Nutrition for uncooked ingredients based on 2 person recipe

Allergens

1.1) Cereals containing Gluten

Always remember to check your ingredient packaging for the most up to date information on allergens and traces of allergens. Boxes are packed in facilities that handle peanut, nut, sesame, fish, crustaceans, milk, egg, mustard, celery, soya, gluten and sulphites.

Use separate equipment to handle raw and cooked meat (or wash between uses). Missing or replaced ingredients, as well as any recipe step changes, will be communicated where possible via email.

Contact

Let us know what you think!
Share your creations with #HelloFreshSnares
Head to hellofresh.co.uk or use our app to rate this recipe

HelloFresh UK
Packed in the UK
The Fresh Farm
10 Worship St, London EC2A 2EZ
You can recycle me!

Roast the Wedges
Preheat your oven to 200°C. Chop the **potatoes** into 2cm wide wedges (no need to peel). Pop the **wedges** onto a large baking tray. Drizzle with oil, season with **salt** and **pepper** then toss to coat. Spread out in a single layer. **TIP:** Use two baking trays if necessary. When the oven is hot, roast on the top shelf until golden, 25-35 mins. Turn halfway through.

Finish the Prep
Halve, peel and thinly slice the **shallot**. Peel and grate the **garlic** (or use a garlic press). Halve the **pepper**, discard the core and seeds, then thinly slice. Trim the **baby gem**, halve lengthways, then thinly slice widthways.

Start Cooking
Heat a drizzle of oil in a large frying pan on medium-high heat. Once hot, add the **diced chicken** and **sliced pepper** to the pan. **IMPORTANT:** Wash your hands and equipment after handling raw chicken and its packaging. Season with **salt** and **pepper**. Stir-fry until the **peppers** have softened and the **chicken** is golden all over, 6-8 mins.

Bring on the Flavour
Add the **shallot** to the pan and continue to stir-fry until the onion has softened and the **chicken** is cooked, 3-4 mins. **IMPORTANT:** The **chicken** is cooked when no longer pink in the middle. Meanwhile, mix the **garlic**, **tomato puree**, **North Indian style spice mix** and half the **mango chutney** together in a small bowl. Stir the mixture into the pan until well combined. Cook until everything begins to caramelise, 1 more min. Remove from the heat.

Warm the Tortillas
Pop the **tortillas** onto a baking tray and onto the middle shelf of your oven to warm through, 2-3 mins.

Assemble the Wraps
To assemble, lay out the **tortillas** (3 per person) and share out **half the mayonnaise** (see ingredients for amount) in the centre of each one. Add the **baby gem lettuce** and **mango chutney chicken**, then drizzle over the remaining **mayo** and **mango chutney**. Fold over one end to encase the **filling** and roll up. Serve with the **potato wedges** on the side.

Enjoy!

Limitations

HelloFresh has very few limitations although they can easily be fixed. The first limitation is that for people with allergies, it does not work. To start with, if you have an allergy to something, there is no way to say to stop recommending foods with that type of food meaning that if you forget to change your order, it may give you foods with that in. As well, some ingredients are hidden within the sauces and spices, for example, I had a mango chutney and it didn't say anywhere what it had in it so if someone had an allergy to a spice within it, it could be dangerous. Another limitation is the number of recipes you get. HelloFresh has a maximum of 5 recipes which means if you can't cook all week and you want to have food for both lunch and dinner, it is not possible.

Conclusion

From HelloFresh, I can take away that it is important to ensure that the total user experience throughout the use of the service should be good, from when users look at the foods to when they get their order and reorder.

Another take from HelloFresh is how speed and efficiency should be a top priority. Similar to HelloFresh, a food delivery app is designed for people who are busy and are unable to cook so should therefore be easy to use without a hassle. The way that HelloFresh does it is by automatically selecting dishes and giving a limited choice. For my food delivery app, I will do a similar thing and have the main page only contain 20-30 options which if the user wants, can switch to looking at restaurants instead or by category. In this way, they will not be overwhelmed by the number of options and can easily find something they like by having a recommendation system that gives them options they may like.

OpenTable

Frontend

For Businesses Mobile Help EN

[Sign up](#) [Sign in](#) [Search icon](#)

Find your table for any occasion

29 Mar 2022 19:00 2 people Location, Restaurant, or Cuisine Let's go

It looks like you're in South West London. Not correct? [Get current location](#)

New to OpenTable

[View all](#)


Amici Lounge Bar & Resta...
★★★ 3 reviews
Lebanese • EEE • Soho



Jeru
★★★★ 1 review
Middle Eastern • EEEE • Piccadilly



temper Covent Garden
★★★★ 1261 reviews
Steak • EEE • Covent Garden



Alohabrazilian
★★★ 0 reviews
Brazilian • EEE • Charing Cross



Faros Restaurant
★★★★ 65 reviews
Italian • EEE • Holborn

For Businesses Mobile Help EN

[Sign up](#) [Sign in](#) [Search icon](#)

Home United Kingdom London

29 Mar 2022 19:00 2 people Italian Find a table

You searched for "italian" in London

251 restaurants match "italian"

Dining Options: All dining options (selected), Delivery only, Takeaway only

Experiences: Set menu, Discounted offer

Seating options: Bar, Counter, Standard, High Top

Featured

How are Featured results ranked?

Negroni's Promoted
★★★★ 5 reviews Awesome (136)
EEE • Italian • Soho

(*) You're in luck! We still have 4 timeslots left!
 17:00 17:15 20:15 20:30
 Experiences Experiences Experiences Experiences

Top experience
Soft Launch 50% discount on food

Rossopomodoro - Covent Garden
★★★★ 5 reviews Awesome (952)
EEE • Pizzeria • Covent Garden

OpenTable London > West End London Covent Garden

Home United Kingdom London West End London Covent Garden

Sign up Sign in



Overview Photos Popular dishes Menu Special Offers Reviews

Rossopomodoro - Covent Garden

★★★★ 4.4 952 reviews £25 and under Pizzeria

Top tags: Lunch Pre/post Theatre Casual

Welcome to Rossopomodoro. We were born in Naples – the home of pizza – and we've

Make a booking

Party Size For 2

Date Today Time 19:00

Select a time:

18:30	18:45	19:00
+ 1,000 pts	+ 1,000 pts	+ 1,000 pts
19:15	19:45	
+ 1,000 pts	+ 1,000 pts	

Menu

Drinks Menu Main Menu

Starters

Olive Miste	£3.95	Polpette di Maiale	£7.95
Prime green and black Gaeta olives marinated with garlic and chilli		Home-made pork meatballs with padron peppers and spicy mayonnaise	
Pane	£2.95	Fresella al Tonno	£8.95
Basket of mixed home-made bread		Fresella bread with tuna, San Marzano tomatoes, french beans, carrots, spring onions and extra virgin olive oil	
Focaccia all'Aglio	£5.45	Burrata Rucola e Pomodorini Secchi	£10.95
Freshly baked tomato and garlic pizza bread		125g fresh creamy burrata from Puglia served with rocket leaves, semi blushed datterini tomatoes and extra virgin olive oil	
Add Fior di Latte Mozzarella	+£1.95	Add Parma Ham	+£2.95
Add Parma Ham	+£2.95	Add Home-Made Bread	+£2.95
Bruschetta Caprese	£6.95	Tagliere Aperitivo	£14.95
Chargrilled homemade bread, fresh San Marzano tomatoes, garlic, fresh basil, oregano, buffalo bocconcini mozzarella and superior Sorrento extra virgin olive oil DOP		Neapolitan street food aperitivo with mixed Burrata and Parma ham and friarielli, Provolone and mozzarella + 1,000 pts	
Parmigiana	£7.95		

95 photos

All Food Drink Exterior



What 952 People Are Saying

Overall ratings and reviews

Reviews can only be made by diners who have eaten at this restaurant

4.4 based on recent ratings

4.5 | 4.3 | 4.3 | 4.1
Food | Service | Ambience | Value



Noise · Energetic

Sort by

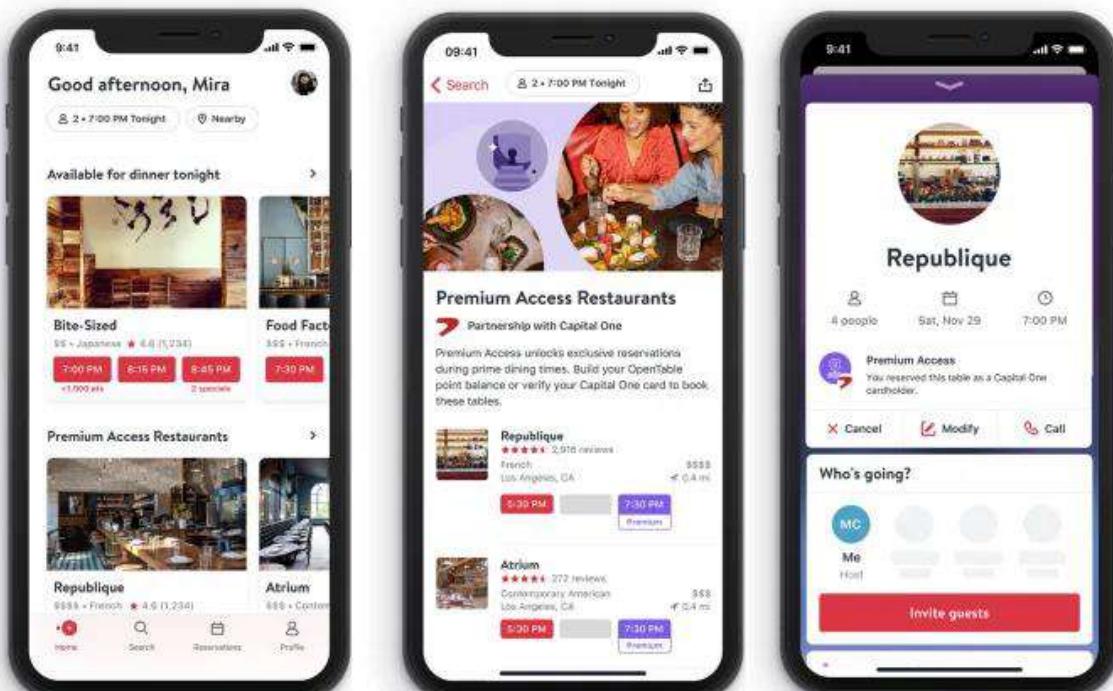
Newest

Filters

- Good for groups
- Gluten Free Pizza (22)
- Gnocchi (8)
- Neapolitan Pizza (5)
- Tagliatelle (5)



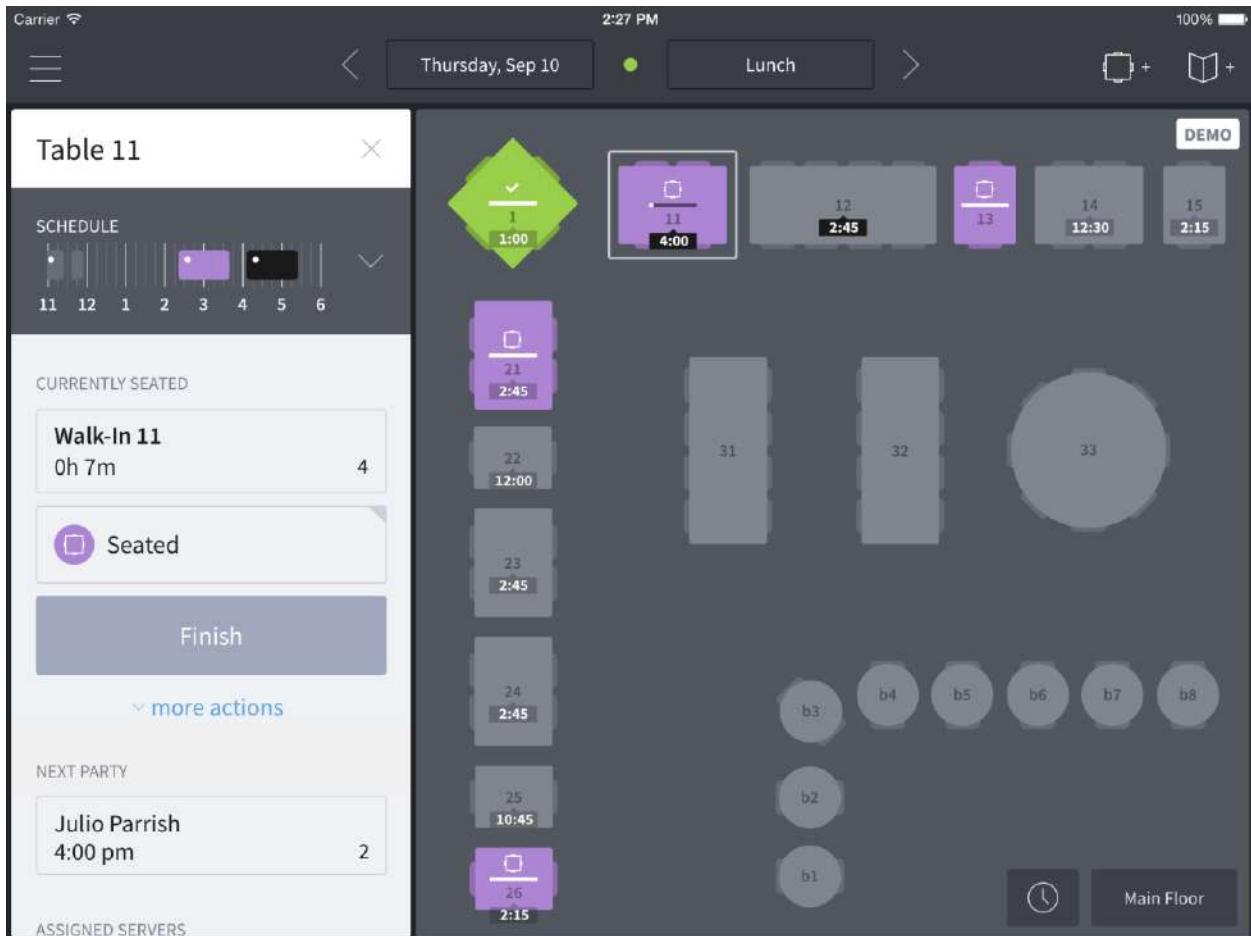
· Dined 3 days ago
Overall 5 · Food 5 · Service 5 · Ambience 5



Backend

Popular dishes

Pizza Fritta	Marinara	Bruschetta Caprese
Fried calzone with buffalo ricotta cheese, Neapolitan... 135 reviews	The most classic pizza found throughout the streets of... 7 reviews	Chargrilled homemade bread, fresh San Marzano... 4 reviews



Summary

OpenTable is a platform for finding and booking restaurant reservations online, collecting useful information about each restaurant and compiling it into one place. I have chosen OpenTable because according to many of the positive reviews of the app, its search and review system is good, with the ability to search by dish and giving the ability to rate restaurants in multiple categories.

Design

OpenTable uses a 4 tone colour scheme, using a white background with black text, red primary colour to stand out and blue as secondary for accent colours like lower priority buttons and interactives. This contrast between blue and red allows two buttons to be next to each other and present to the user which one should be pressed. Using red as the primary colour represents

urgency and importance whereas blue is the complete opposite, representing relaxation and calmness.

Similar to Just Eat, OpenTable uses overlapping blocks to add depth and bring attention to the main content. The restaurant image is behind the title and menu of the restaurant so the user knows that the image is an addition not part of the content. When scrolling down when in a restaurant, the booking and delivery options stick to the screen, giving the option to book at all times no matter how far down the page you are. This is useful as the user might find it hard where to book if they have to look through the whole page to find the book button.

Features

The main feature of OpenTable is its reservation system, being one of the only places allowing you to do it. Its reservation system also syncs with the restaurant and knows who is reserved for the day with a visual interface of the restaurant which can be easily customised.

Another feature is its grouping of restaurant data; with the help of many restaurants, OpenTable allows them to input lots of data together including full menus with descriptions, transport to the restaurant, location, reviews and many others. The menu and location are compiled using TripAdvisor's SinglePlatform which allows restaurants to create online versions of their menus. The reviews are done through the OpenTable site and are usually accurate because it allows you to select categories it goes best in. If multiple people select the same category, it will have a tag at the top of the restaurant page saying what it is best in.

The final main feature is the searching feature. Unlike many other search bars, it allows you to search under multiple variables including date, time, number of people and anything else, using a search bar if you want to search by category, specific name, dish, or nothing. The result means that it gives you a more defined result and therefore don't get option fatigue.

• • • •

What would you recommend this restaurant for?

Select all that apply

Great for Lunch Special Occasion Local Ingredients
 Hot Spot Notable Wine List Great for Brunch
 Great for Outdoor Dining Neighborhood Gem
 Scenic View Live Music Sunday Lunch
 People Watching Good for Birthdays Fun
 Seasonal Worth the Drive Afternoon Tea

Loved for ⓘ

Vibrant Bar Scene Hertfordshire	Best Overall St. Albans	Best Food St. Albans	Best Service St. Albans
Best Ambience St. Albans	Best Value St. Albans	Fit for Foodies St. Albans	Lunch St. Albans
Special Occasions St. Albans	Most Booked St. Albans		

Back
Next

Skip this step

Strengths

The best part of OpenTable is how unique it is. Unlike many other apps and websites, it was one of the first in the game, starting in 1998. Because of its age, it has had time to develop and understand the needs of people which makes it such a good application. With all this time, it has made lots of brand awareness and therefore has made it a first choice for checking restaurants.

The first strength is its overall clean and minimal design, showing the key information and compiling all the information about each restaurant. In the summary, it shows the name, rating, category and price bracket with quick booking buttons if you already know you want to pick it. When you select a restaurant, it gives you more details and uses external sources as well as internal sources to show information which means even if a restaurant hasn't used OpenTable, it still can show you information about it

The second strength is that OpenTable is aware that it cannot do everything so if a restaurant doesn't have reservation bookings, it links to a food delivery app as a secondary.

Another strength is its search functionality. Unlike many other search bars, it allows you to search under multiple variables including date, time, number of people and anything else, using a search bar if you want to search by category, specific name, dish, or nothing. The result means that it gives you a more defined result and therefore don't get option fatigue.

Limitations

There are very few limitations for OpenTable, one of which being its lack of restaurant chains. After looking at places like Dominos, Pizza Express and Papa John's, I found that they didn't have a restaurant page on them. Although I am not sure what it could be, my best understanding is that OpenTable doesn't support restaurant chains. Other than that, it doesn't have any big flaws or limitations.

Conclusion

From OpenTable, there are a few things I can take away. To start with, since I know that I will not be able to get all restaurants to use my food delivery app, I will try to make a way for it to work together or at a minimum link to where you can order it so it means there are no missing restaurants. I found with experience that some apps would not show the restaurant if it wasn't on the app so I had no idea that it was an option until I switched to another food delivery app.

Like with other existing solutions, I found that colour schemes and a good layout are essential to having a good user experience and therefore will try to stick with a few colours and not give excess information unless the user wants it. Summaries should be short and give essential information like category, rating and price bracket.

Research Findings

From researching these large brands, I will take away a variety of things and although they were not directly correlated to food delivery apps, they still provide a variety of features that can be incorporated into my project.

From JustEat, I took that simplicity is important to make an effective food delivery application, ensuring that it is easy to use. The colour scheme from JustEat makes it good as it sticks with orange and uses it effectively to show the user what can be interacted with and is a great accent colour to the black text and white background. Using rounded boxes and overlapping elements creates depth and helps further draw the eye to the content and away from the images. From what JustEat has not done, I found that you should never forget to design part of an application since if something doesn't fit with the overall theme, it will stick out like a sore thumb and break the immersion and seamlessness of the application.

From HelloFresh, I found that speed and efficiency should be a top priority, and should be the first thing to think of since similarly to HelloFresh, my food delivery app will be tailored to people who want to quickly find something to eat and order it quickly. Like what HelloFresh has done, instead of giving a person a full menu to read through and putting them into the blind, the application will give them 20-30 options to choose from that are recommended by previous

times ordered and dietary needs. If a person wants to not do that, there will then be an option to search by category or restaurant.

From OpenTable, I will incorporate the feature that means that if a restaurant is currently not on the food delivery app I am making, it will show where you can order it and will still try to get information about the restaurant from other sources.

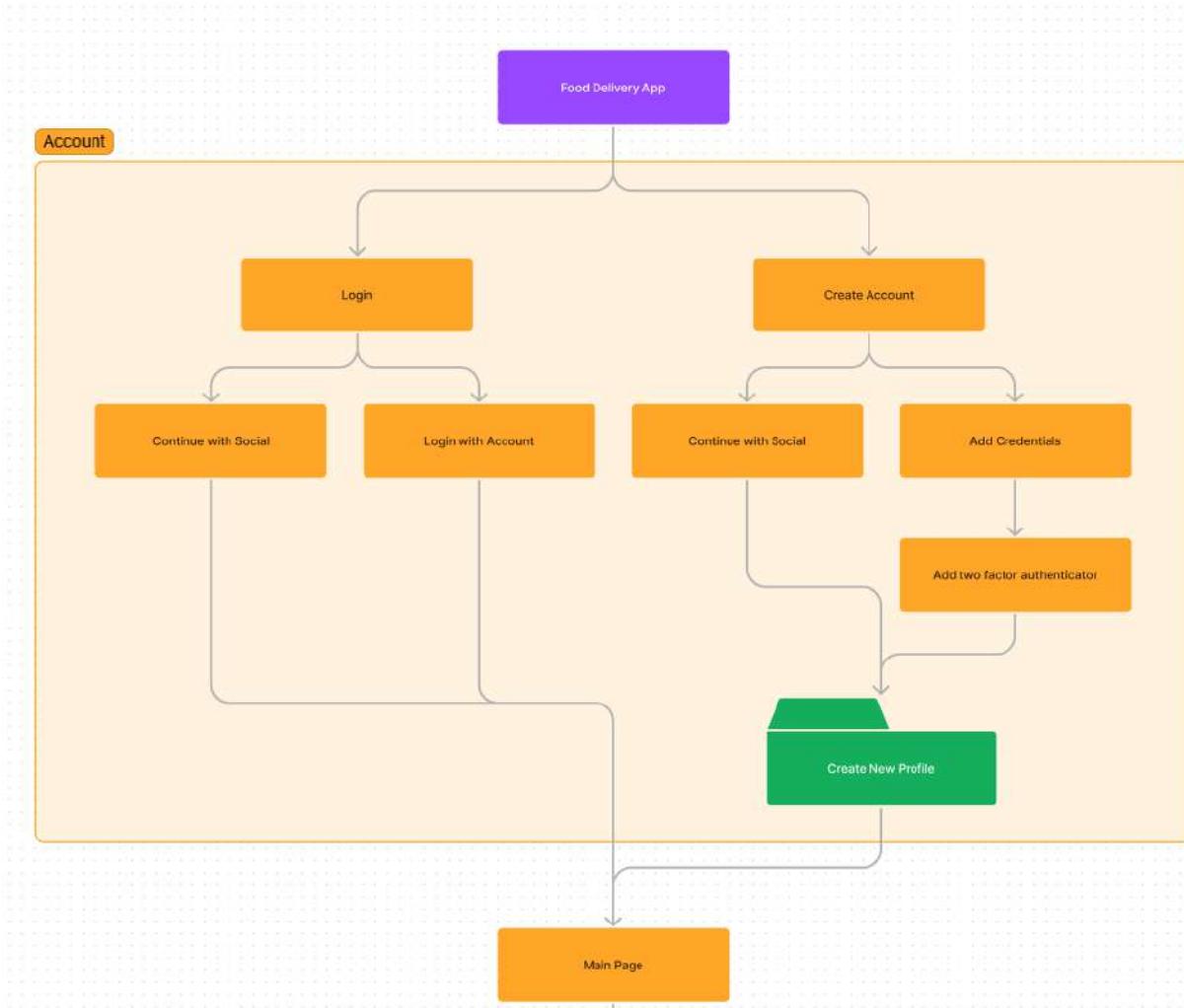
Essential Features

Concept

My problem to solve was "How do I help people with food allergies order food without the risk of cross-contamination?"

To solve this, my solution will be a food delivery app based on existing apps, using a similar design. This will be done to ensure that people who are both unfamiliar and familiar with existing apps have a seamless and easy experience. Unlike these apps, my application will be for an audience with food allergies and dietary needs, although if anyone else wants to use the app, it will also work for them.

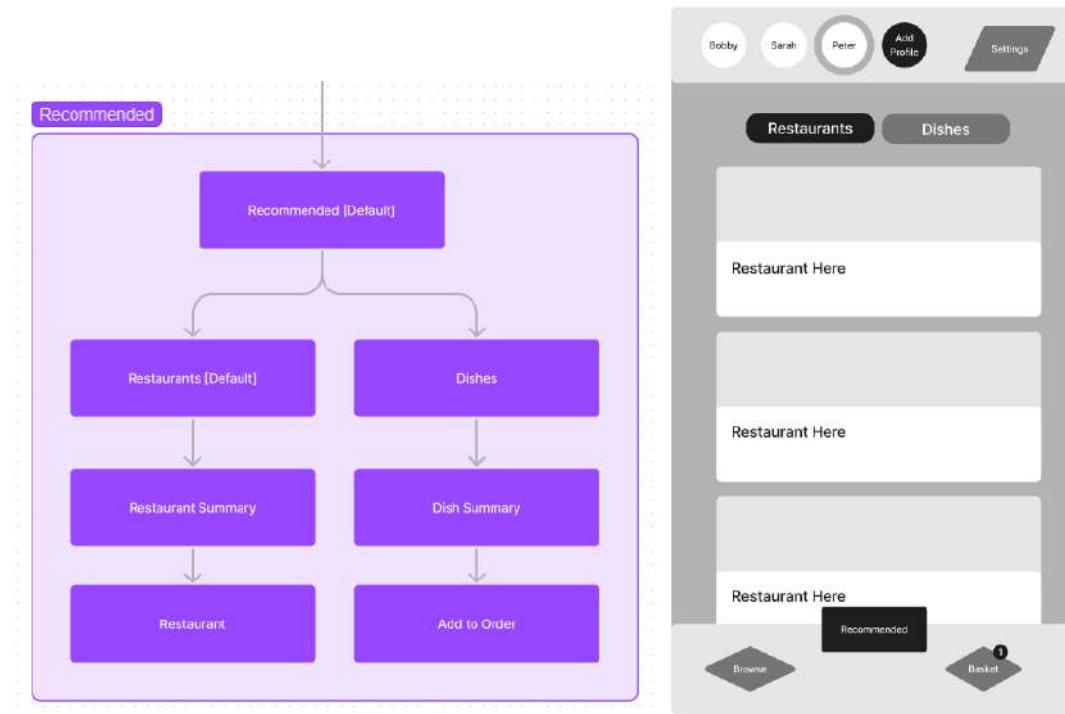
When a user first gets the app, they will be greeted with a **login/signup** page where they can sign in with their social media or sign up/sign in to an account straight through the app. If they are signing in through the app, it will send a **confirmation email/message** for extra security since the food delivery app has access to spending money. Upon making their account, it will ask them to create a profile, by adding their name and **selecting which foods they are allergic to** and then it will ask about their **dietary needs** (vegetarian, pescetarian etc.). Then it will then allow you to select any **foods they like** from a list.



After completing creating a profile, it will bring them to the main page, where they will by default, be put on the recommendation page. At the bottom of the screen will be a menu bar which will have the recommendation page, browse page, account page and profile page. At the top of the page will be the profiles made and added, with the ability to add more profiles or create a new one. As well, when you create a new account, it will ask you if you would like to use the tutorial to understand how the app works.

On the recommendation page, you can select between either looking at restaurants or looking at individual recommended dishes. By default, it will look at restaurants where the recommended restaurants will be shown using an algorithm, looking at ordering habits, allergies of the current profile selected, and ratings made. Together, it will sort by the most recommended, putting a badge next to the restaurants with the highest chance you would like it. This badge will be put on the restaurants that are recommended throughout the entire app as well as on dishes that are recommended. The recommended page will have 20 different

restaurants that are recommended and 30 different dishes that are recommended to ensure that it doesn't recommend places you will not like as well reducing the chance of choice fatigue.



On the browse page there will be two sections similar to the recommendation page, where you can switch between looking at restaurants and looking at categories of food. On the restaurant part you can filter and sort them by a variety of different variables:

Sort: Most recommended, rating, distance

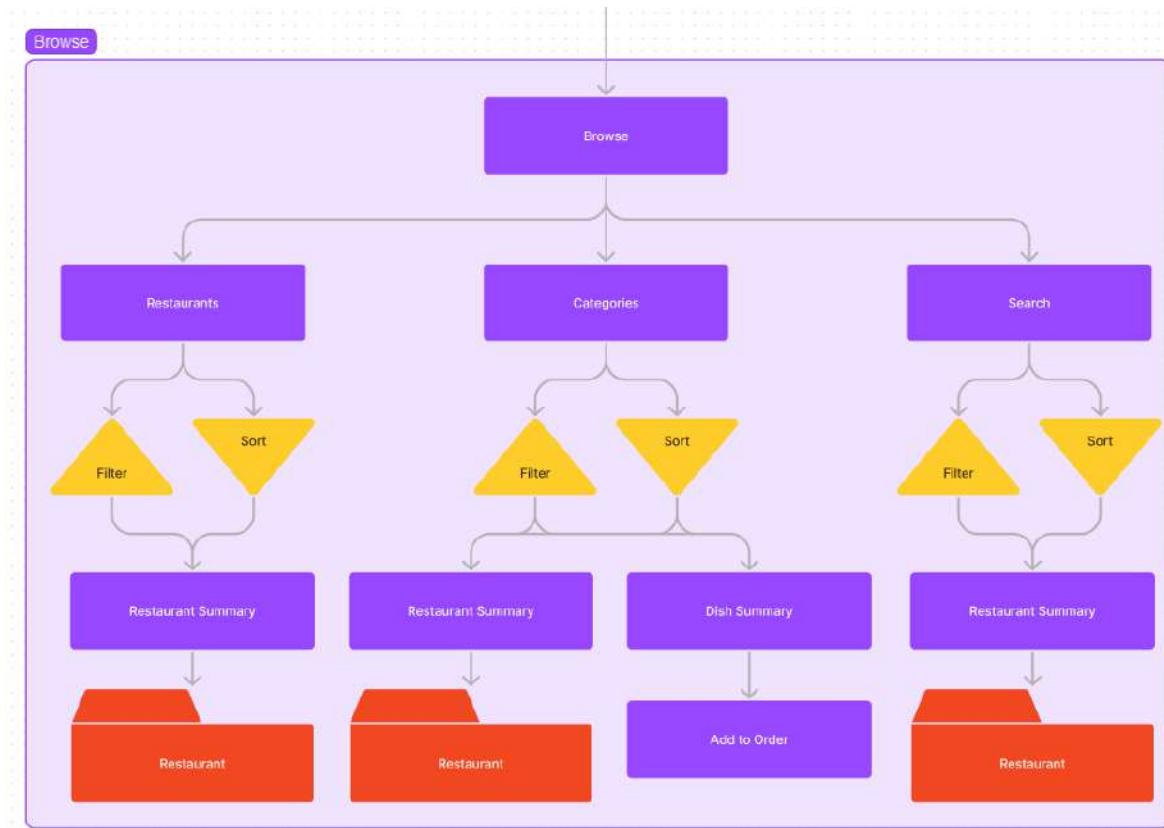
Filter: Price range, delivery, pickup, delivery price, New

On each restaurant summary, it will show distance from destination, rating, delivery fee, category, name of the restaurant and an image of some of the food at the restaurant. On the categories part, it will show types of food including Italian, Indian, Chinese, sandwiches, pizza, seafood etc. having both dishes and types of dishes. When you select one, at the top will be recommended dishes from restaurants which fit under that category, with an image of the dish, name, restaurant it comes from, rating and price. There will be around 4-8 dishes which will show up at the top. Under these dishes will be a list of restaurants which also fall under these category and will be by default be sorted by most recommended but can be changed using filters and sorters:

Sort: Most recommended, rating, distance

Filter: Price range, delivery cost, pickup, delivery price, New

At the top of both pages will be a search bar where you can search for specifics, giving you a list of restaurants which are similar to what you searched. Like the other pages, it will allow you to sort and filter using the same parameters.



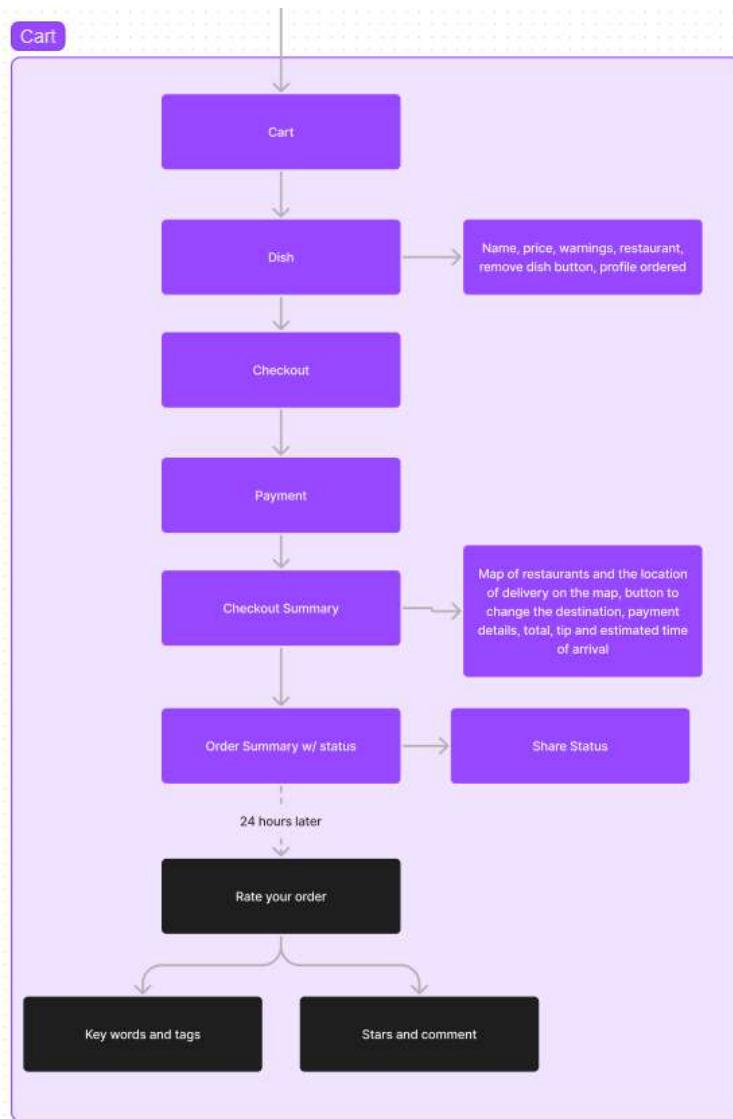
For each restaurant, there will be an image of some of the foods that the restaurant serves that is the same as the one on the restaurant summary. The restaurant page will also contain the name of the restaurant and key information inclusion ratings, tags and price bracket. If a person wants to pick up, there should also be a way to see the location and opening hours. At the bottom will be the menu which contains foods that are recommended for you having a badge so you know what you should get as well having the main menu with categories and foods with the name and customisation options.



For any dish that you are unable to eat, there will be a badge next to the dish and when you try to add that dish, it will give you a popup that you are unable to eat a food in the dish and tell you to change your dietary needs in order to get this food as well as saying what is in it that you cannot eat. This ensures that you are not able to eat that food unless you go out of your way to change your profile.

When you are done with the order, you can go to the cart page where it will show a summary of the dishes with the name of the dish, price, restaurant, profile ordered and a remove dish button. At the bottom it will say the total price and have a button to split the total to each person or pay on current profile. If you select split, each profile will be charged for whatever they chose. If they select for the current profile to pay, only that profile will be charged for the food. After clicking continue, it will show you a map of the restaurants and the place where it will be delivered with a button to go to the status page where it will show a progress bar of which restaurants have been collected. If a person wants to track their order from their phone, they can sign in with their phone and go to their profile and it will show up as a shared order as long as their profile ordered something for the order

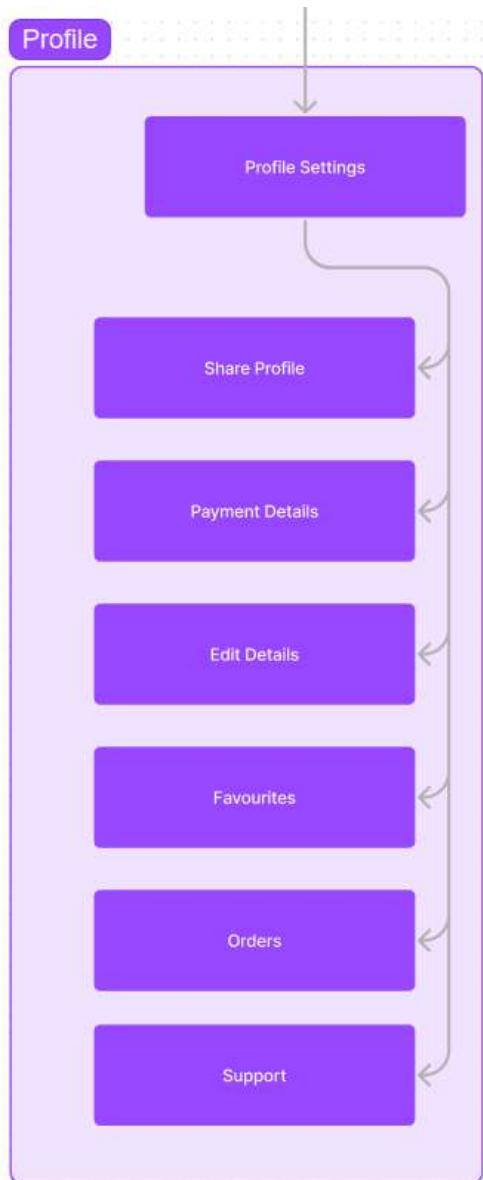
After 24 hours, you can go back to the app and rate each food, giving it a star rating and adding a comment and adding tags which will show up on the restaurant if it is a popular tag for a restaurant.



The QR function allows you to import a profile to a phone or share a dish with another phone. This is done adaptively within the QR code scanner in the app, able to know the difference between the two functions. When you share a dish with another phone, it will automatically add the dish to the cart with the same customised options. The QR code shows up next to the add to cart button.

When you select a profile from the top bar, you can click on the profile settings cog to see your payment details, share your profile, edit your name, look at your favourites and orders and get

to the support page where you can find out how to use the app. When you click share profile, you will be able to see your QR code to share with other people with the app or see where you are currently signed in and remove yourself from people's devices. Profiles are cloud-based in order to be able to not be connected by a single device and easily have your recommendations and details accessible from anywhere.



Features & Reasoning

Feature	Reason
Social Media Login	To be able to have a quick login and link the

	account to the cloud
Two Factor Authentication	In order to have high security, two factor authentication will mean that there is a lower chance of the account being hacked.
Profile Maker	To be able to understand what allergies each person has and get a good recommendation system without a person needing to order lots.
QR Scanner	<p>In order to order quickly while also having all the details of a person while ordering, the QR Scanner is able to send the details of a person to another phone with a single click. This ensures that even if you have a variety of allergies, it can easily warn you if you are ordering the wrong food.</p> <p>It can also be used to quickly add food to an order if multiple people are ordering since each person can find what they are looking for on their phone and just send it to a single phone for ordering</p>
Recommendation Page	The recommendation page is useful for quickly finding new restaurants and foods, giving a compilation of both new dishes but also foods that are similar to what they had previously. This is good if you have an allergy as it shows the dishes that you wouldn't have known you could eat. It also makes it quick and easy to find new things to eat without needing to know exactly what you want.
Filters & Sorting	This is useful for narrowing down searches from hundreds down to a few. In a place like a city, there can be a lot of restaurants so it is useful to have a way to get exactly what you are looking for.
Profiles	Profiles are a way of having multiple people be able to pick their foods. Having it be only one profile would not work when working

	with allergies since some people want their own types of foods. For example if someone is vegan, you would not want a whole group of people forced to only see vegan options or in another situation, have the vegan get food which is not vegan and especially with food delivery, they usually do not have the ingredients list on the food so you would never know.
Unable to Eat Popup	Having a popup blocking people who are allergic from getting food with the thing they are allergic to ensures that there is almost no way of getting bad food and means that people will not get injured or die because of it.
Searching	A searching function means that if you know what you are looking for and need to get to it quickly, you can just search it. If there is something that is not in a category, it will also show up. For example, let's say you are looking for soup, you can just search it and it will show up, giving you restaurants that serve soups.

Limitations

1. I will be working on the main program only since making a backend for both the restaurant management and order management would take a lot of time with little algorithm to it or advanced programming.
2. Due to time constraints, the final front-end will not be final since I am not making a company to run it. It will only have fake restaurants and will not be for consumers.
3. The server running this will be temporary since this is not funded but has the potential for running as a real application with only few additions
4. Because there will not be many people using the app, some parts like the recommendation system will not be very accurate since there is not much user data to link users. As well, because there is only a few users, I will not be able to simulate the overall efficiency of the app when thousands of people are using it simultaneously

5. I will be using Flutter for the programming language which uses Dart but I do not have any knowledge of this language so will need to learn the language while making it so there may be a lot of mistakes during the development and will likely not be exactly what the original plan is currently since I do not know its limits. I have chosen Flutter because of its customizability and its ability to work on a variety of devices without much change of code.

Solution Requirements

Software	Advantages	Disadvantages
Language		
Flutter (Dart)	Cross-device support for Android, Windows, MacOS and iOS	Very new programming language with less documentation compared to larger languages
	Fast Development, using the same code for both iOS and Android	Apps made with flutter are larger than most in-order to have the customizability it has
	Hot-reload allows you to easily make changes and see what the changes are in real-time.	
	Very customisable with quick rendering	
Alternative Languages		
Web (HTML, CSS, JavaScript)	Compatible with all devices that are able to connect to the internet and have a browser	Unable to utilise the sensors and the full capabilities of the device
	Easy to code, with a lot of documentation to help with development	Limited formatting and customizability
	No need to install any apps when using it (Doesn't use any storage)	Not very stable, with some devices not working with it

	Updates roll out immediately, with the version stored centrally - no need to care about app stores	
	Reduced software piracy since it only shows the user however much they are seeing	
Kotlin	Similar to Java so quick to learn if you know Java	Not much documentation so harder to debug and learn by itself, with not many people using it
	Lower chance of buggy code because it compiles better	Compilation speed slower than other programming languages
	Easier to maintain, being able to be put in many IDEs	Distinctly different from Java, having its own quirks
Swift	Easy to read and write, using concise code compared to C.	Requires a computer running MacOS to write
	Good performance and safety, able to translate code and optimise code quickly. It is able to handle errors well and prevents crashes and errors in production.	Language is still new so it still doesn't have all the essential features and customizability of bigger programming languages like C
	Reduced memory usage, efficiently able to reuse code in order to be more efficient and work on older devices	Doesn't support apps for devices on Android and Windows
	Cross-device support, with little edits to code in order to get it onto another platform	Not compatible for IOS 7 or earlier
Database		
SQL	Queries are completed quicker even with large database sizes	Databases in SQL are under threat since they hold large

		amounts of sensitive data
	Lots of documentation and is an established language	SQL doesn't grant complete control over databases to its users due to hidden business rules.
	Can be used on a variety of platforms, being lightweight	
	Queries can be made in a few lines, using keywords to specify what is being queried	

Alternative Databases

Access	Easy to use, with a fully functional database	Finite amount of data that can be stored on it, with a maximum file size limit
	Anything based in Windows can integrate Access, being easier to maintain than other systems	All the data is saved in a single file and performance is hit hard with large databases
	Quick and easy to import and export data for backups	Not good at time-critical transactions, taking time to update
		Access has a poor relational design which makes it difficult to learn or use the database

Intended Operating System(s)

Android	Flutter was originally based on Android so has more documentation and is better suited for mobile rather than desktop solutions like Windows and MacOS	Because of the variety of devices, it can make it harder to develop for it since each device runs different hardware and software
	Android is open-source so has better documentation and easier to get the most from the	People with android are less likely to spend as much money as on iOS

	OS with less restrictions	
	There are more devices running Android than iPhone and can be run easily on tablets and phones without remaking the app	
IOS	Unlike many languages, Flutter can run on both Android and iOS with the same code so can be tested on both, benefiting on the advantages of each	Apps on the Apple App Store have more rigorous checks before an update can be released on it with a higher chance of rejection
	IOS is able to run programs with better performance than on Android	Third-party software within the app can sometimes not work on iOS
	There are less devices to test for so is more reliable and is less likely to be buggy	
	iOS users are more likely to spend more money within apps than people on Android	
	iOS apps have more consistent and sleek designs over their Android counterparts	

Alternative Intended Operating Systems

Windows	Apps made in Windows are easy to debug, being able to access the files and the console	Because I will be making a food delivery app, it is intended for people who are on the go which is usually not people who are on laptops or on Desktop
	There is a large number of users which use Windows	Application speed heavily relies on computer hardware which many people have

		slow computers People run a variety of versions of Windows and hardware so hard to test for them
	People usually have enough space to install desktop apps and are fine having another one	Application speed heavily relies on computer hardware which many people have slow computers People run a variety of versions of Windows and hardware so hard to test for them

Hardware	Advantages	Disadvantages
Storage		
	On average, apps made with flutter take around 150MB which is the average size. Just Eat has a size of 90MB, and UberEats a size of 220MB.	Because there are a lot of types of apps, files and general things on a phone, it can easily be filled up and therefore people will be unable to install the app or update it
CPU		
	The processor in any relatively new phone can easily make apps with little latency. Even phones from 6 years ago can still run apps although may buffer and with the battery being a limiting factor for people to upgrade,	The processor in a phone can range drastically, people having phones from 6 years ago to people with brand new phones so the speed of the app will vary depending on the phone but will likely have a minimum requirement

	the chance of a person having a slow phone is unlikely so there will be little worry about that.	of the Apple A8 or Qualcomm Snapdragon 810 or Exynos 8890 because processors before this time were very limited and slow with little graphics power.
Memory		
	Since there will not be any 3D graphics, memory is not a big issue and there is little that can change a person's memory so accounting for people with a variety of phones is essential so that people are able to use the app.	Scalable images are essential for having a good experience since people's phones can't be changed. A default image for when a phone is unable to show the image so people know that something is meant to be there is also good since it means that the design will stay uniform and clean
Connectivity		
	The main server needs to have a connection to each device from the user all the way to the restaurant to ensure there are no hitches.	Devices must be constantly connected to a network in order to see the status and make an order which means that the amount of data sent should be minimal to ensure that on a slow 3G/4G connection, there is no issues

Success Criteria

No.	Success Criteria	Justification	How will it be tested?
Database			
1.1	All inputs should be checked	This ensures that the	A screenshot of SQL

	for SQL injection, checking for key terms that use SQL	database cannot be edited by the user	injection code being inputted into an input and it being rejected
1.2	Passwords used to make and log into a profile should be kept secure by using a password hash	Passwords should use a hash to ensure if a database is hacked, the passwords are not readable	A screenshot of the database storing the hash of a password
1.3	The database should be in third-normal form and normalised	It reduces the amount of duplicate data, avoids data anomalies, ensures referential integrity, and simplifies data management.	I will ask 2 people to check if the database is in third normal form

Login/Profile

2.1	When there is no profiles on the app, the user should be presented with the login/profile creation page	This means that the app will not break when it tries to recommend foods or restaurants, having some data to make recommendations and know any existing allergies.	When the app is reset or profiles are logged out, it should automatically bring the user to the login/profile creation page. A screenshot of the login/profile creation screen will be evidence of this working.
2.2	On the login/profile creation page, there should be buttons that allow you to continue with Google, Facebook or Apple.	This allows the user to easily create an account quickly and not need to input any passwords or credentials, having it autofill	There should be buttons clearly indicated what they do on the login/profile creation page which redirects you to sign in with Google, Facebook or Apple depending on the button you click
2.3	If the user cancels the continue with Google, Facebook or Apple, it should not create an account and	This will mean that the database does not have any half made profiles without the correct	A screenshot of the database not having half made profiles with missing information

	instead bring them back to the profile creation page where the user can put in their email or login with it again	information	
2.4	If the user closes the app or goes back during profile creation, the app should not process the profile		
2.5	When creating a profile using an email, it should only allow emails that contain an @ symbol	To ensure that emails are sent, they should validate that it isn't a plain string	A screenshot of the app not accepting an email without an @ symbol
2.6	First names and last names should only be accepted if they are under 50 characters each	This ensures that a name does not go over to another line.	A screenshot of the app not letting you type over 50 characters for both the first name and last name.
2.7	Passwords made should be a minimum of 7 characters and a max of 99	This ensures that the password is secure since it cannot be quickly guessed	A screenshot of the app rejecting a password under 7 characters and over 99 characters
2.8	Passwords should contain a letter and a number		A screenshot of the app rejecting a password that doesn't have either a letter, a number, a capital letter and/or a symbol
2.9	Phone numbers should be 11 digits long and a minimum of 10 digits	This ensures that there is a smaller chance of inputting an invalid number	A screenshot of the app not letting the user input a phone number longer than 11 digits and less than 10
2.10	When there is no profiles on the app, the user should be presented with the login/profile creation page	This means that the app will not break when it tries to recommend foods or restaurants, having	When the app is reset or profiles are logged out, it should automatically bring the user to the

		some data to make recommendations and know any existing allergies.	login/profile creation page. A screenshot of the login/profile creation screen will be evidence of this working.
2.11	A card number should not be fully shown, only showing the last 4 digits unless it is being changed	This ensures that if you take a screenshot or give people your phone to use, people will not be able to see your card details and save them	A screenshot of a saved card number only showing the last 4 numbers
2.12	When logging into a device and creating a new profile, the account should either be able to be authenticated using a 2FA app or a 2FA message that is sent to the phone set up with the profile.	This ensures that the account is safe and that the correct user has logged in	A screenshot of the app asking the user to input the 2FA code
2.13	When creating a profile, there should be a search bar that allows you to search foods you are allergic to	This will mean that the profile will never get recommended foods and not be allowed to order foods containing this food	A screenshot of the search bar that allows you to search by ingredient
2.14	The allergic foods search function should allow you to select foods to and show them in a list below the search bar	This will reduce the time to select items and will mean that the user will not miss anything	A screenshot of some foods being selected from the allergy list
2.15	When searching for allergic foods, the user should be able to select food categories like vegan or gluten-free	This will allow the recommendation system to choose foods that best suit them and will mean that the user doesn't have	A screenshot of a food category being able to be selected and it including a variety of foods
2.16	Upon making a profile, the user should be able to select up to 10 dishes they like from a list of different dishes.	This will allow the recommendation system to choose foods that best suit them and will mean that the user doesn't have	When a user has finished inputting their details it should show them a list of foods randomly picked and allow them

		to order to get a good recommendation	to select the foods that they like. If the user wants to keep searching for foods they like they can continue. A screenshot of the selection page of the foods is evidence of this.
2.17	When on the recommendation page, browse page, restaurant page and checkout, the user should be able to change profiles after which it should refresh the page and change depending on the user and the page content like the card information and the recommended restaurants.	This allows a user to switch profiles quickly when they are selecting foods so that they can get better recommendations for their taste. On checkout they can switch profiles if they would like to pay as another user	A screenshot of one profile and another from when it is switched on a page
2.18	There should be a button that the user can click that allows them to see information about the current profile that is selected	Being able to see information about the profile means that you can change preferences and have the ability to see orders. As well, having a different one for each profile means that it can be unique and personalised for each profile	A screenshot of the profile information page
2.19	The user should be able to edit their profile details including their name and password	This means that if something changes, they are able to fix any mistakes or make their password more secure	A screenshot of the edit profile page
2.20	The user should be able to add and remove payment cards from their profile	This will allow the user to pay under different cards for when they want to pay for a business or	A screenshot of the add card page with a screenshot of multiple cards saved

		personally or another reason	
2.21	A user should be able to sign in with their profile using multiple devices	This means that if you have multiple devices or are signed in on a family/friends phone, you can still use it on another device	A screenshot of the same profile being signed in on multiple devices
2.22	Information of the profile should be synced to the cloud	If you are using multiple devices and change some details or get an order, you could see them on another device without needing to sign out	A screenshot of changes made on one device updating on another
2.23	Each profile should be able to access their favourites	This allows the user to access all of their favourites	A screenshot of all the dishes and restaurants that have been favourited
2.24	A user should be able to control how long their profile will stay logged in for when on a guest device. This will be done before generating a QR code for the profile to share.	This will ensure that if they are trying to login onto a device that is not fully trusted, you can make it so that it only stays on for a day	A screenshot of the profile share QR code creation page with buttons to select the amount of time it will be logged in
2.25	Profiles on a guest device will have a timer stating how long until it is logged out	So that the user understands that it is a guest profile, it will show a timer, being at peace knowing that they do not need to log out and will stop confusion when a profile randomly gets logged out	A screenshot of the timer next to a guest profile saying how long is remaining
Restaurants/Food			
3.1	The user should be able to sort restaurants using a	This will allow the user to change how the many	A screenshot of the sort button when looking at

	button	restaurants are shown so that they can see the best option first that they are looking for	a list of restaurants
3.2	When sorting restaurants, it should allow the user to sort by recommended, rating or distance		A screenshot of all the ways that the user can sort the restaurants by
3.3	The user should be able to flip the sort from highest to lowest and vice versa		A screenshot of the flip sort button reversing the order the restaurants are shown
3.4	The user should be able to filter restaurants using a button	This will allow the user to reduce the amount of options and only see the relevant restaurants they are looking for	A screenshot of the filter button when looking at a list of restaurants
3.5	When filtering restaurants, it should allow the user to filter by price range, delivery, pickup, delivery price, new		A screenshot of all the ways the user can filter the restaurants by and a screenshot of it reducing the number of options when one is selected
3.6	When filtering restaurants, it should show the user how many restaurants there are under each filter and how many will be shown when all the filters are used that are selected.	This ensures that the user knows how many options it is reducing	A screenshot of the filter menu with the number of restaurants next to each filter
3.7	Before filtering, there should be a button to confirm that they would like to filter the restaurant.	This ensures that the user would like to filter the options and shows how many restaurants will be shown after the filter so the user is not surprised if there are a lot or only a few restaurants remaining	A screenshot of the button that says how many restaurants will be remaining after the filter and asking them to click it to confirm the filter
3.8	When the user has selected some restaurant filters, the user should have a button they can click to clear the	This decreases the amount of time it takes to unselect the filters, with a single button to undo any	A screenshot of a button that resets the filters selected and it showing all the restaurants

	filters and have it show all the restaurants	changes	
3.9	When the user has selected a restaurant filter, they should be able to unselect a filter and change the settings	This means if the user wants to remove a filter because it is too refined, they can do it	A screenshot of the user being able to unselect a filter
3.10	The user should be able to sort food items using a button	This will allow the user to change how many food items are shown so that they can see the best option first that they are looking for	A screenshot of the sort button when looking at a list of items
3.11	When sorting food items, it should allow the user to sort by recommended, rating or price		A screenshot of all the ways that the user can sort the food by
3.12	The user should be able to flip the sort from highest to lowest and vice versa		A screenshot of the flip sort button reversing the order the foods are shown
3.13	The user should be able to filter food using a button	This will allow the user to reduce the amount of options and only see the relevant foods they are looking for	A screenshot of the filter button when looking at a list of foods
3.14	When filtering food, it should allow the user to filter by price range, delivery, pickup, delivery price, food category, and spice level		A screenshot of all the ways the user can filter the food by and a screenshot of it reducing the number of options when one is selected
3.15	When filtering food, it should show the user how many dishes there are under each filter and how many will be shown when all the filters are used that are selected.	This ensures that the user knows how many options it is reducing.	A screenshot of the filter menu with the number of dishes next to each filter
3.16	Before filtering food, there should be a button to	This ensures that the user would like to filter the	A screenshot of the button that says how

	confirm that they would like to filter the dishes.	options and shows how many items will be shown after the filter so the user is not surprised if there are a lot or only a few dishes remaining	many items will be remaining after the filter and asking them to click it to confirm the filter
3.17	When the user has selected some food filters, the user should have a button they can click to clear the filters and have it show all the dishes	This decreases the amount of time it takes to unselect the filters, with a single button to undo any changes	A screenshot of a button that resets the filters selected and it showing all the dishes
3.18	If there are no dishes under a category filter, it should not show the filter	Because there are lots of different categories, it ensures that it only shows the number that can be filtered.	A screenshot of the filter not showing any category filters that have 0 dishes under it
3.19	When an item is recognised to be in the list of foods that a profile is allergic to, it should have an icon to indicate that it contains a food that cannot be eaten	This means that if you are browsing, you will know that you cannot add it to your order	A screenshot of both an item you can and cannot eat, with an indicator showing that it contains a food that the profile cannot eat
3.20	If a profile that has an allergy to a food in a dish or cannot eat a food within a dish tries to add a dish to their order, a popup will tell them that they are unable to add the food	This means that if the user does not look at the ingredients or the indicator icon, the popup will stop them from adding it	A screenshot of the allergy popup when a user tries to add a dish that contains a food they are allergic to
3.21	On the allergy popup that occurs when a user tries to add a dish that they are allergic to, it should include the ingredient(s) that it contains that caused the app to block them from ordering it and the quantity it contains	This allows the user to get a better understanding of why the app blocked them	

3.22	The user should be able to bypass the popup by clicking a small secondary button and having to confirm the bypass	This will mean that if a person is only slightly allergic or the dish contains food which they think they can eat, they should be able to bypass it. For example, some vegans think that honey is vegan while others don't. If it blocked honey because you were vegan but you could bypass it, it would be good for both types of vegans.	A screenshot of the bypass button on the popup
3.23	When a user is looking at a restaurant, there should be a button to see the location and its opening hours	This is useful to know if you want to see where it is and so you can visit it in person	A screenshot of a restaurant page with the location and opening hours
3.24	When a user is checking the location of a restaurant, it should show a pin where the user is currently, where the restaurant is and the destination of where the food will be delivered, with a way to get directions to the restaurant.	This is useful for if you are picking up the food and want to know where to go or would like to know where the food is coming from	A screenshot of the map of where the device is and where the restaurant is
3.25	If there is no location of where the device is, it should only show the location of where the restaurant is and the destination, hiding the marker of where the device is	This means that if you don't have location services enabled, it still looks good and will not annoy the user since it is not required for any part of the app except this	A screenshot of the map where the user's current location is not on the map
3.26	When a user is on the recommendation page, it should never include foods that the profile is allergic to	This is to stop them from having to sort through foods they cannot eat and so they do not order any foods they cannot eat	A screenshot of a person with an allergy's recommendation page not containing any dishes that they are

			allergic to
3.27	A user should be able to favourite dishes and restaurants using a button	This allows the user to get back to foods/restaurants that they like and indicate to the app that it is a very liked thing	A screenshot of the favourites button for both each restaurant and each dish

QR Code

4.1	There should be a button from the recommendation page and browse page to scan QR codes	This means that the user can quickly and effectively get to the QR code scanner. It should be quick to access because it can be used to quickly add profiles and go to items and add items to the cart	A screenshot of the QR code scanner button on a main page
4.2	When the QR code scanner is activated, it should automatically look for QR Codes	This means that the user doesn't need to take a picture of the QR code and instead the user can constantly move around until the camera notices the QR code and tries to analyse it	A screenshot of the scanner detecting a QR code and it showing what it recognised
4.3	If the QR code scanner recognises a QR code, it should align the image by the reference points on the code and read it		A screenshot of the camera recognising a QR which can be used
4.4	When the correct QR code is scanned, a popup should show, indicating what it has recognised		A screenshot of the camera recognising an incorrect QR code which cannot be used in the app
4.5	When an incorrect QR code that is either not for the app or is corrupted, it should indicate that it is an invalid QR code		A screenshot of a dish
4.6	For each dish, there should	This means that if you are	

	be a button to generate a QR code which can be scanned by another phone to add the dish to the cart under the profile that is currently selected	ordering from one phone, the other people can use another device like a desktop or another phone to look what they are looking for and generate a QR code so that the other phone can add it to the cart without looking for it	having a QR code generator button which makes a QR code
4.7	When a user selects to make a QR code, it should be encrypted and made	This means that if the data is taken off of the QR code, the password is never revealed	A screenshot of the QR code recognising the data in the QR code as being cypher code
4.8	QR codes should have data loss packets where if some of the QR code is hidden or does not scan properly, it can rely on the rest of the other data to put together the missing data	This ensures that if the QR code is on a bad device or if the conditions are not perfect, it can still read the code and doesn't need to re-read the code if some code is missing	A screenshot of the QR code containing parity bits and extra code
4.9	When a QR code is scanned, it should be decrypted	Because the QR code will be encrypted, in order to read the data, it must be decrypted	A screenshot of the encrypted data and decrypted data being in plain text.
4.10	If a QR code contains profile data, the hash should be checked if it is the same as the one in the database	This means that the original password that needs to be inputted into the login page is never revealed and ensures that the hash has not been faked	A screenshot of the code checking the QR code with the password hash in the database
Cart/Checkout			
5.1	There should be a button to remove an item from the order before it has been checked out.	This means that if you make a mistake or want to change it, you can	A screenshot of the remove button on the current order page

5.2	For each item, it should show which profile the item is for	This ensures that you know which item is for which person and removes the confusion	A screenshot of the current order page
5.3	If a user has bypassed the allergy warning that stops them from ordering foods they have been allergic to, an indicator will show that they have done this at the bottom of the page	This ensures that if you accidentally add something you are allergic to, to your order, you still have time to remove it and see that you cannot eat it	A screenshot of an item the profile is allergic to being in the order and there being an indicator next to the item
5.4	When completing the order, the app should ask the type of payment where you can either pay with one profile or split the order by the price of the food individually that each person will get (the equivalent of ordering for one but the delivery and tip split over the different profiles paying)	This reduces the price of delivery if everyone pays individually since it will all come at the same time and the delivery price and tip is split	A screenshot of the options to pay and a screenshot of the same food being split in payment for one and the other all being charged to one profile
5.5	When checking out, there should be a way to input the amount you would like to tip	This helps out delivery drivers and encourages them to continue	A screenshot of a part of the checkout page showing a way to select how much to tip the delivery driver
5.6	After the food has been delivered, the app should ask the user to rate their food.	Ratings are key to having a good recommendation system as it shows the system how much you like or dislike a food and give you recommendations around it	A screenshot of the app asking the user to rate their order
5.7	When the app asks the user to rate the food, it should only ask them to rate the food that is associated to	Having them only rate dishes they ate ensures that they get the right recommendations and do	A screenshot of the food ordered and the dishes that has been asked to rate for a profile

	their profile	not get recommendations for foods they didn't eat	
5.8	When a user switches to their profile, it should ask them to rate any foods they ate previously	Having them only rate dishes they ate ensures that they get the right recommendations and do not get recommendations for foods they didn't eat	A screenshot of different profiles being asked to rate different dishes for the same order
5.9	If a user is logged into their profile from another device, it should sync ratings for foods and if they haven't rated a previous dish, ask them to rate it	Having them only rate dishes they ate ensures that they get the right recommendations and do not get recommendations for foods they didn't eat	A screenshot of the app asking the user to rate foods on different devices for the same profile
5.10	The status of the order should sync with all profiles on all devices they are signed in with	This will mean if they order on one phone, they can see the status on another phone	A screenshot of the order status on two phones on the same profile
5.11	Recommendations should be based on number of times ordered, favourites, rating of each dish and ratings of each restaurant	This gives a better indicator of what a person likes	A screenshot of the code taking the variables into account when making a recommendation

Appearance

6.1	The user should be able to open the application using an app icon with the correct logo	This is useful to make sure that the user goes to the correct application on the phone	Testing if the app is on the device and has an icon which is unique and is able to be clicked on. A screenshot may be used for evidence
6.2	By default, the user should be brought to the home page when the app is opened	This will allow for the user to quickly make an order with something they may like	When the app is opened or a new profile is made, it will put the user on the recommendation page. A screenshot of the recommendation page will be evidence of

			this
--	--	--	------

Prototype 1A

Design & Development

Name of application: Allivery (*Allergy + Delivery / All (for everyone) + delivery*)

App Design Analysis

Decomposition of the Problem

Allivery - A food delivery service application			
User Interface (UI)			
Login Page	Profile Creation Page	QR Code Scanner	Recommendation Page
Browse by restaurant page	Browse by dish page	Restaurant template	Cart page
Profile settings Page	Order Status Page	Checkout Page	Rate Order Template

Algorithms			
Sort restaurants	Filter restaurants	Restaurant recommendation algorithm	Dish recommendation algorithm
Searching algorithm	QR Code encryption and decryption	Estimate time of arrival algorithm	Profile password hashing algorithm

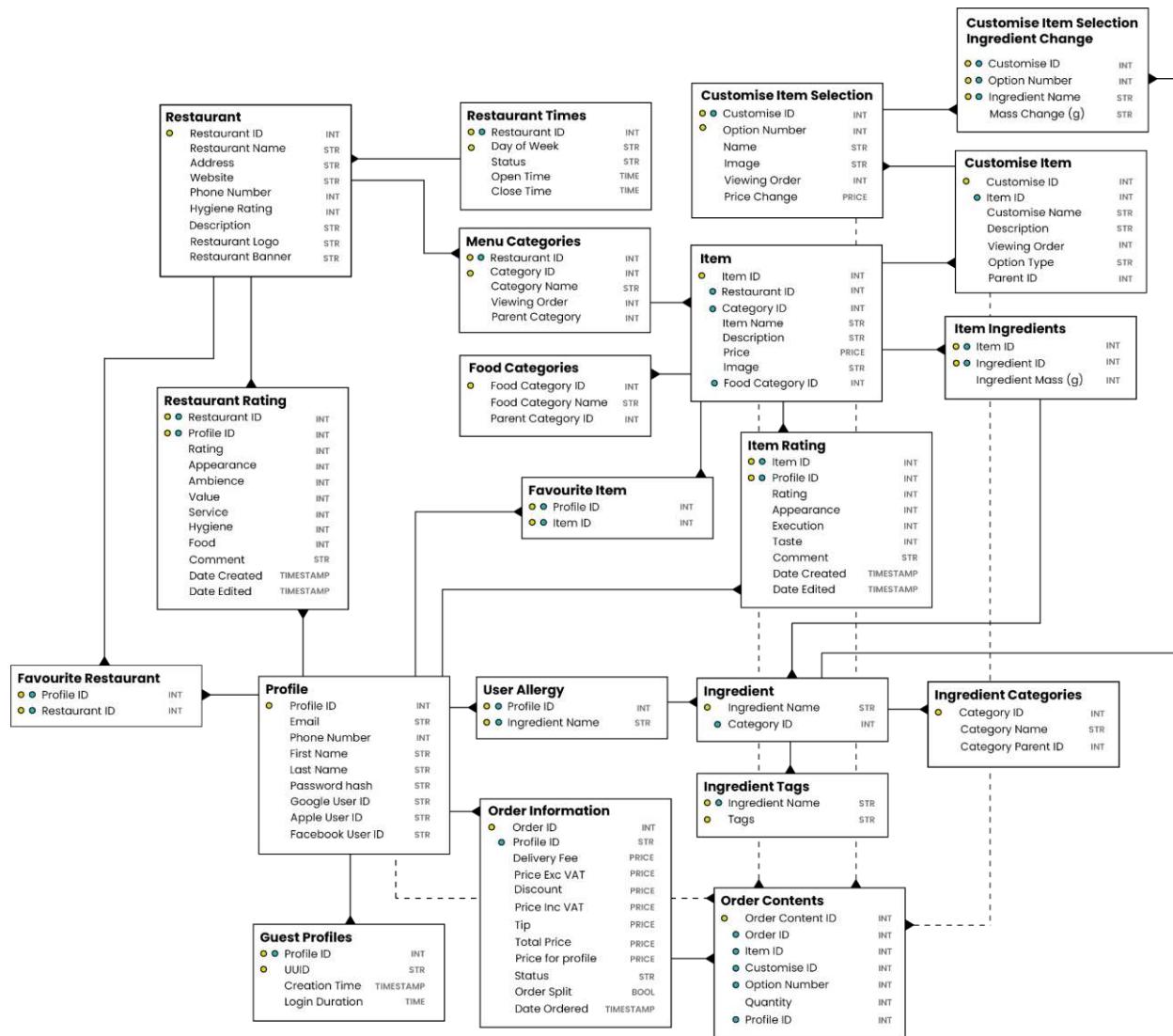
Profile Section

Share Profile with QR Code.	Ability to sign out of other accounts remotely	Change Payment Details	Edit profile information & password
Add dish to favourites and view favourites	View current order with status	View previous orders	Sign out of profile

The features above provide a basic overflow of the app for each section. Looking at this, it was obvious to see that there will be a lot of data that will move around on the client-side and server-side, including the profile details (first name, last name, email, password and payment details), restaurant information, menu information, dish information, allergy information, ingredient information and more. In order to better organise this, I will use a database to structure this information as it can provide data validation with set data types as well as being able to use foreign keys to link the tables so that less information needs to be stored about each other.

Database Development

Database Structure Design



The yellow dots represent a primary key. If there are multiple primary keys on a single table, it is a composite key and uses multiple fields to create a unique identifier.

The blue dots represent foreign keys, fields from other tables that are used within the table

After checking with other people, the database has verified that it is in third-normal form, meaning that there is no unnecessary data duplicates and that the relations between the tables are correct

Tables

Table Name	Field	Type	Description
Restaurant	Restaurant ID	<i>Unique Pri. Key Number</i>	Stores an incremental number for each restaurant. You cannot use the restaurant name because there are chains and restaurants with the same name
	Restaurant Name	<i>Short Text</i>	Used to identify the restaurant that is human recognisable
	Address	<i>Short Text</i>	Used to get the location of where the restaurant is for tracking and showing the user where it is
	Website	<i>Hyperlink</i>	If there is more information that cannot be shown on the app, a website can be linked where the user can learn more
	Phone Number	<i>Number</i>	If there are any issues or would like to talk to a person for any reason, the user can use the phone number to contact the restaurant
	Hygiene Rating	<i>Number</i>	This is useful for the user to get information about the cleanliness of the restaurant
	Description	<i>Long Text</i>	Gives more information about the restaurant to the user
	Restaurant Logo	<i>Hyperlink</i>	The logo associated with the logo used to help the user recognise the restaurant
	Restaurant Banner	<i>Hyperlink</i>	The banner is designed to show the user a variety of foods that the restaurant serves as well as how it looks

Restaurant Times	Restaurant ID	<i>Comp. Key Fore. Key Number</i>	Used to link the restaurant to the day
	Day of Week	<i>Comp. Key Short Text</i>	For each restaurant there is 7 different entries for each day of the week
	Status	<i>Short Text</i>	If the restaurant is open or closed on the day
	Open Time	<i>Date/Time</i>	The time that the restaurant should be able to be ordered from
	Close Time	<i>Date/Time</i>	The time that the restaurant should not be able to get orders from
Restaurant Exception Times	Restaurant ID	<i>Comp. Key Fore. Key Number</i>	The restaurant associated with the change of time
	Exception Date	<i>Comp. Key Short Text</i>	The date where there is a change of times when the restaurant is open
	Status	<i>Short Text</i>	If the restaurant is open on that date
	Open Time	<i>Date/Time</i>	The updated time for the date when the restaurant can start to get orders from
	Close Time	<i>Date/Time</i>	The updated time for the date when the restaurant should not get any orders from
Menu Categories	Restaurant ID	<i>Comp. Key Fore. key Number</i>	For each restaurant, there are multiple categories, they can make categories
	Category ID	<i>Comp. Key Unique Number</i>	Each category should have an ID associated with it because they might have 2 categories with the

			same name
Item	Category Name	<i>Short Text</i>	The category name that will be visible for the restaurant
	Viewing Order	<i>Number</i>	The order in which the categories will be shown
	Parent Category ID	<i>Number</i>	If there is a subcategory, then the parent will be the one that it is nested in
	Item ID	<i>Pri. Key Unique Number</i>	For each restaurant and within a category, there are items, including food and drinks. The ID is used to identify which item it is since there may be multiple items with the same name
	Restaurant ID	<i>Fore. Key Number</i>	For each item, it is attached to a restaurant for which it can be ordered from
	Item Name	<i>Short Text</i>	Each item should have a name (eg. Cheeseburger) so the user knows what they are ordering
	Description	<i>Long Text</i>	The user should be able to know more information about each item to help their choice
Food Category	Price	<i>Currency</i>	Each item has its own price which should be charged.
	Image	<i>Hyperlink</i>	For each item, there should be an image of the food that they are ordering, so they can understand what they are ordering
Category	Food Category ID	<i>Fore. Key Number</i>	For each item, it should be put in a category so that the user can find it easily

Food Categories	Food Category ID	<i>Pri. Key Unique Number</i>	For each category of each food, there should be an ID associated with it so that if there are several under the same name, there is no issue
	Food Category Name	<i>Short Text</i>	The name of each food category
	Parent Category ID	<i>Number</i>	Each category may be a subcategory of a bigger category so the parent ID means that it can be associated with the main category
Customise Item	Customise ID	<i>Pri. Key Unique Number</i>	Since there is likely that the title of the customise page for each item will be the same, an ID will be needed to ensure that they are unique
	Item ID	<i>Fore. Key Number</i>	Not all the items need a customisation page so for each customisation, an item ID needs to be associated with it
	Customise Name	<i>Short Text</i>	There needs to be a title and name for each item since there may be multiple customisation options and helps the user know what to do
	Description	<i>Long Text</i>	To help the user understand what they are changing, a description can inform the user about what is changing
	Viewing Order	<i>Number</i>	Since there may be many different customisations to each item, the order in which they are shown should be important
	Option Type	<i>Short Text</i>	There are a few different types for customisation including; text entry, multiple selection,

			dropdown or single section
	Parent ID	Number	If you want to customise within a customisation option, the parent ID means that it will associate with the one it is nested within
Customise Item Selection	Option Number	<i>Comp. Key Unique Number</i>	When customising, if the option type is either multiple selection, dropdown or single selection, there will be a set number of options the user can pick. The option number means the order in which they show up since there can only be one for each number in the order that they show up and can be composited with the customisation ID to make a unique key
	Customise ID	<i>Comp. Key Fore. Key Number</i>	
	Name	<i>Short Text</i>	There will be a name for each option that can be selected so the user knows what the item will be modified with
	Image	<i>Hyperlink</i>	For multiple and single selection (not dropdown or text entry) an image can be added next to it to get a better understanding of the changes
	Viewing Order	Number	The option that the option shows up in the list
	Price Change	Number	The change in price from the original price
Customise Item Selection Ingredients	Option Number	<i>Comp. Key Fore. Key Number</i>	Because there will be a change in ingredients for each option for each customization part and the table has a composite key, this table needs to use a triple composite key in order for it to be unique.
	Customise ID	<i>Comp. Key Fore. Key Number</i>	

	Ingredient Change (Ingredient Name)	<i>Comp. Key Short Text</i>	For every option, multiple ingredients can change, either being added or removed
	Mass Change (g)	<i>Number</i>	If the change is negative for an ingredient that is already in the dish then it will decrease the total mass for that ingredient in the dish and vice versa. If the total negative makes the total for the ingredient 0 then there should be a check for when the app is checking if the person can eat it since some places may allow you to remove an ingredient. If an ingredient is added, it should also be checked.
Item Ingredients	Item ID	<i>Comp. Key Fore. Key Number</i>	For each item, including drinks and food dishes, there are ingredients which can be selected so that the app can understand what is within the food and help the app show a warning to people who cannot eat the food
	Ingredient ID	<i>Comp. Key Fore. Key Number</i>	
	Ingredient mass (g)	<i>Number</i>	This is added so that the nutrition can be compiled to the whole dish/drink and also inform the user if they only have a light allergy when in a large quantity so they can determine if there is enough that they can bypass the warning
Restaurant Rating	Restaurant ID	<i>Comp. Key Fore. Key Number</i>	For each rating made by a profile, it is associated with a restaurant ID
	Profile ID	<i>Comp. Key Fore. Key Number</i>	So that there are no duplicates, it uses a composite key so a profile can only make a single rating per restaurant.

	Rating	<i>Number</i>	For each category, it can be rated out of 5 stars, allowing from 1 star to 5 and can have half stars
	Appearance	<i>Number</i>	
	Ambience	<i>Number</i>	
	Value	<i>Number</i>	
	Service	<i>Number</i>	
	Hygiene	<i>Number</i>	
	Food	<i>Number</i>	
	Comment	<i>Long Text</i>	If the user has anything else to say, a comment allows them to fit a full review with details about their experience about the restaurant
	Date Created	<i>Date/Time Extended</i>	A timestamp of when the comment allows the user to sort by date made in the app
	Date Edited	<i>Date/Time Extended</i>	A timestamp of when it was edited ensures that people understand that they have changed their opinion
Item Rating	Item ID	<i>Comp. Key Fore. Key Number</i>	For each rating by a profile, it is associated to a item ID (food/drink item)
	Profile ID	<i>Comp. Key Fore. Key Number</i>	So that there are no duplicates, it uses a composite key so a profile can only make a single rating per item.
	Rating	<i>Number</i>	For each category, it can be rated out of 5 stars, allowing from 1 star to 5 and can have half stars
	Appearance	<i>Number</i>	

	Execution	<i>Number</i>	
	Taste	<i>Number</i>	
	Comment	<i>Long Text</i>	If the user has anything else to say, a comment allows them to fit a full review with details about their experience about the restaurant
	Date Created	<i>Date/Time Extended</i>	A timestamp of when the comment allows the user to sort by date made in the app
	Date Edited	<i>Date/Time Extended</i>	A timestamp of when it was edited ensures that people understand that they have changed their opinion
	Profile ID	<i>Pri. Key Unique Number</i>	Each profile has an ID associated with it since people can have the same name and it is shorter to use an ID since @ can cause the issues for an email
Profile	Email	<i>Short Text Unique</i>	An email is used to ensure that the account is secure, with password resets being sent to it and using it to check if it is a unique profile
	Phone Number	<i>Number</i>	Another way of contacting the profile is via their phone number
	First Name	<i>Short Text</i>	Each user has a profile with a first name and last name that will be used to identify them. Shortened, it will just include the first name but when looking in detail, it will show the last name if there are two people with the same first name
	Last Name	<i>Short Text</i>	

	Password Hash	<i>Short Text</i>	For extra security, the password hash is stored of the password so that if the database is hacked, the passwords will still not be leaked
	Google User ID	<i>Short Text</i>	The user ID for social media mean that the system still knows the profile from an external account
	Apple User ID	<i>Short Text</i>	
	Facebook User ID	<i>Short Text</i>	
User Allergy	Profile ID	<i>Comp. Key Fore. Key Number</i>	For each profile, people can have multiple allergies or none at all. If there are entries in this table for a profile then it means the person has no allergies.
	Ingredient Name	<i>Comp. Key Fore. Key Number</i>	Each allergy is associated to a specific ingredient that is in the database
Ingredient	Ingredient Name	<i>Pri. Key Unique Short Text</i>	Each ingredient must have a unique name so users know the difference
	Category ID	<i>Fore. Key Number</i>	Each ingredient can be categorised to make it easier to find
Ingredient Tags	Ingredient Name	<i>Comp. Key Fore. Key Short Text</i>	As there will be multiple tags per ingredient, it needs to be put in a new table
	Tags	<i>Comp. Key Short Text</i>	A tag can either be "vegetarian", "vegan", "peskitarian" or "Gluten Intolerance". These are the biggest categories of people and can be selected when selecting allergies. This will select ingredients with the tag

Ingredient Category	Category ID	<i>Pri. Key Unique Number</i>	Each category because a category name could include a space which does not work well when referencing
	Category Name	<i>Short Text</i>	Each category has a name that the user can see when searching ingredients
Guest Profiles	Profile ID	<i>Comp. Key Fore. Key Number</i>	When using a QR code to share a profile, they will be made as a guest where they will be a secondary login. In order to know which profile is logged in, the profile ID is needed
	Device UUID	<i>Comp. Key Short Text</i>	Each app has a unique UUID which gives details about the device
	Creation Time	<i>Date/Time Extended</i>	The creation time helps know when the device should log out from a profile
	Login Duration (Hours)	<i>Number</i>	The login duration can be customised and can be set to 0 if a user decides they want that device to be logged out after creating it. This will mean that the next time the app is loaded, it will be logged out.
Favourite Item	Profile ID	<i>Comp. Key Fore. Key Number</i>	Each profile is able to favourite an item in order to get back to it if they like it
	Item ID	<i>Comp. Key Fore. Key Number</i>	

Favourite Restaurant	Profile ID	<i>Comp. Key Fore. Key Number</i>	Each profile is able to favourite a restaurant in order to get back to it if they like it
	Restaurant ID	<i>Comp. Key Fore. Key Number</i>	
Order Information	Order ID	<i>Comp. Key Unique Number</i>	Each order can have multiple profiles which are connected to the order if they order together
	Delivery Fee	<i>Currency</i>	Depending on the distance and the restaurant, there may be different delivery fees
	Price Exc VAT	<i>Currency</i>	The price of the items combined for the order excluding VAT
	Discount	<i>Currency</i>	If you have a deal, the price will be discounted for the whole order
	Price Inc VAT	<i>Currency</i>	The price of the items combined for the order including VAT
	Tip	<i>Currency</i>	To be nice, a tip can be added to help out the delivery driver
	Total Price	<i>Currency</i>	The final price for the items with the discount
	Price for Profile	<i>Currency</i>	If there is a split, this will be different
	Status	<i>Short Text</i>	As the order gets delivered, it will change status and can be tracked from multiple location
	Order Split	<i>Number</i>	The number of people connected to the order
	Date Ordered	<i>Date/Time Extended</i>	A timestamp is added so that they can appear on the order history in the correct order

Order Contents	Order Content ID	<i>Pri. Key Number</i>	Since there are lots of options that is possible which is possible that there can multiple of, the order content has an entry for each item
	Order ID	<i>Fore. Key Number</i>	For each order there are multiple items (dishes and drinks) that are in it, each with a profile that it is associated with.
	Item ID	<i>Fore. Key Number</i>	Each item has multiple variables which need to be recorded in order for the restaurant to know what you picked as well as how many you want
	Customise ID	<i>Fore. Key Number</i>	
	Option Number	<i>Fore. Key Number</i>	
	Quantity	<i>Number</i>	
	Profile ID	<i>Fore. Key Number</i>	Each profile is associated with an item to ensure that people don't order foods that they are allergic to. The profile ID is also useful for the app to ask each profile how they rate each item that they had and not ask them to review food they didn't eat.

Correction to Documentation (2)

I changed the categorisation method for the ingredients. It originally had ingredients in single categories where they were grouped in large chunks of ingredients but will now be changed to have categories and sub-categories, with the ability to both select a category of ingredients and also select a specific ingredient. For each category the user should be able to either check a checkbox and click on the category name to get into the category and select the sub-categories/ingredients within it.

To do this, I will remove the existing category called "Categorised Ingredients" which had the fields of "Ingredient Name" and "Category Name" to make a new table called "Ingredient

Categories" which will have the fields of "Category ID", "Category Name" and "Parent Category ID". The Category ID will be used to have a unique primary key, with the name being associated with it while the new "Parent Category ID" is used to create sub-categories.

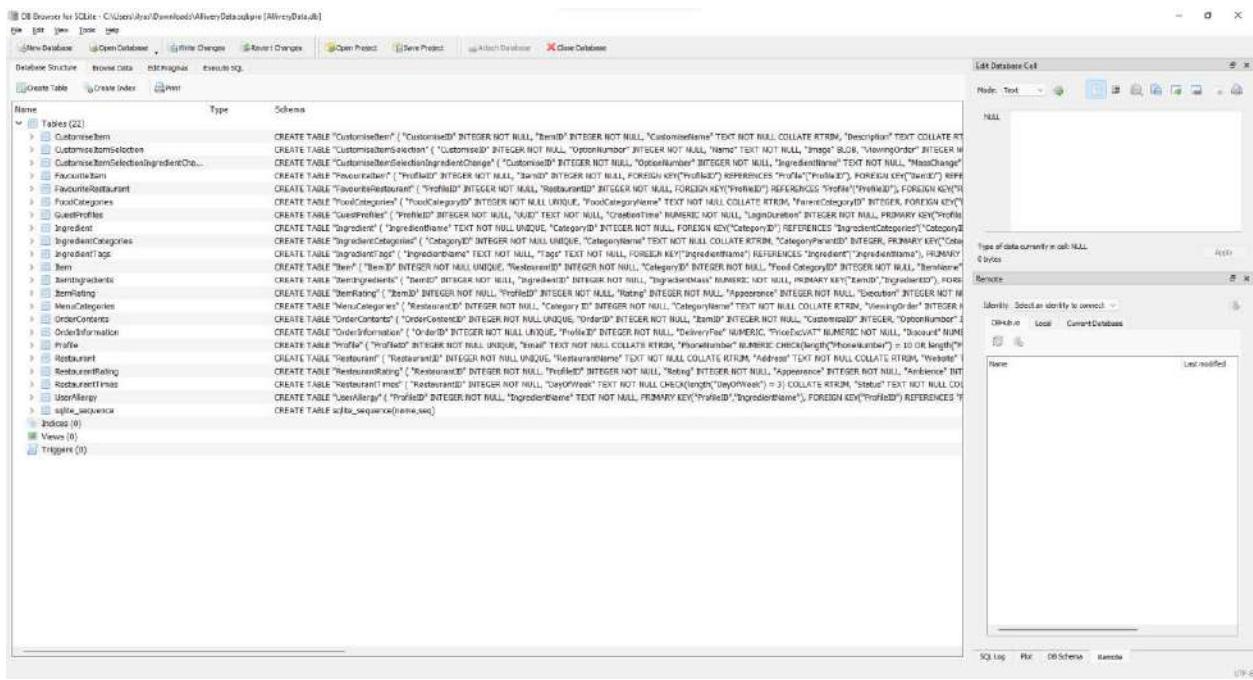
After taking a look at the logistics behind having the nutrition calculator for each dish, I have found that it will not work. With the varying portion size since some dishes are made in bulk and portioned out, customisation options and the difficulty for a restaurant to give exact measurements for all things including spices. The fix to this is to remove the "Calories (Per 100g)", "Protein (Per 100g)", "Sodium (Per 100g)", "Sugar (Per 100g)", "Saturated Fat (Per 100g)" and "Trans Fat (Per 100g)". For each item, the restaurant will have the option to put a range of mass. For example "<1g", "1-10g", "10-50g" etc. This will allow the feature that says how much of the ingredient is in the food without needing exact values, aiding the user.

Something that was forgotten was how customisation options can add or remove items from the item. To fix this, for each option, there will be new fields that will allow new ingredients to be added and the mass added as well as remove items with specific masses removed. They will have the field names of "Ingredient Changed" and "Mass Changed (g)". On top of this, options can now change the total price of the item, either increasing or decreasing the price depending on if it is positive or negative.

As well, a big mistake which I had forgotten to add was what was in each order, with their customisations to the order so these were added to a new table which had a foreign primary key of "Order ID" since there are multiple items per order.

Developing the Database

Since the application is for mobile, the database also needs to be designed with this in mind so with guidance from Flutter, I have found that SQLite will work well due to how lightweight it is and its efficiency. Since this is not directly SQL, it has a few quirks like the fact that it doesn't have many data types, having only INTEGER, TEXT, BLOB, REAL & NUMERIC. For something like price and timestamps, a plugin will be used in order to have a similar look and feel to SQL. Since this is only the type of file I am making, I used "DB Browser for SQLite" to create the database, with the GUI allowing me to easily create the database with the correct coding for SQLite



IngredientCategories	CREATE TABLE IngredientCategories ("CategoryID" INTEGER NOT NULL, "CategoryName" TEXT NOT NULL)		
IngredientTags	CREATE TABLE "IngredientTags" ("IngredientName" TEXT NOT NULL, "Tag" TEXT NOT NULL)		
Item	CREATE TABLE "Item" ("ItemID" INTEGER NOT NULL UNIQUE, "RestaurantID" INTEGER NOT NULL, "CategoryID" INTEGER NOT NULL, "Food CategoryID" INTEGER NOT NULL, "ItemName" TEXT NOT NULL COLLATE RTRIM, "Description" TEXT COLLATE RTRIM, "Price" NUMERIC NOT NULL, "Image" BLOB)		
ItemID	INTEGER	"ItemID"	INTEGER NOT NULL UNIQUE
RestaurantID	INTEGER	"RestaurantID"	INTEGER NOT NULL
CategoryID	INTEGER	"CategoryID"	INTEGER NOT NULL
Food CategoryID	INTEGER	"Food CategoryID"	INTEGER NOT NULL
ItemName	TEXT	"ItemName"	TEXT NOT NULL COLLATE RTRIM
Description	TEXT	"Description"	TEXT COLLATE RTRIM
Price	NUMERIC	"Price"	NUMERIC NOT NULL
Image	BLOB	"Image"	BLOB
ItemIngredients	CREATE TABLE "ItemIngredients" ("ItemID" INTEGER NOT NULL, "IngredientID" INTEGER NOT NULL, "Quantity" REAL NOT NULL)		
ItemRating	CREATE TABLE "ItemRating" ("ItemID" INTEGER NOT NULL, "Rating" REAL NOT NULL, "Count" INTEGER NOT NULL)		

With these new tables on the database, I will add some example data that will be used to test out the program when I am complete with prototype 1A.

Ingredients Table

Tables: *Ingredient*, *IngredientCategories*

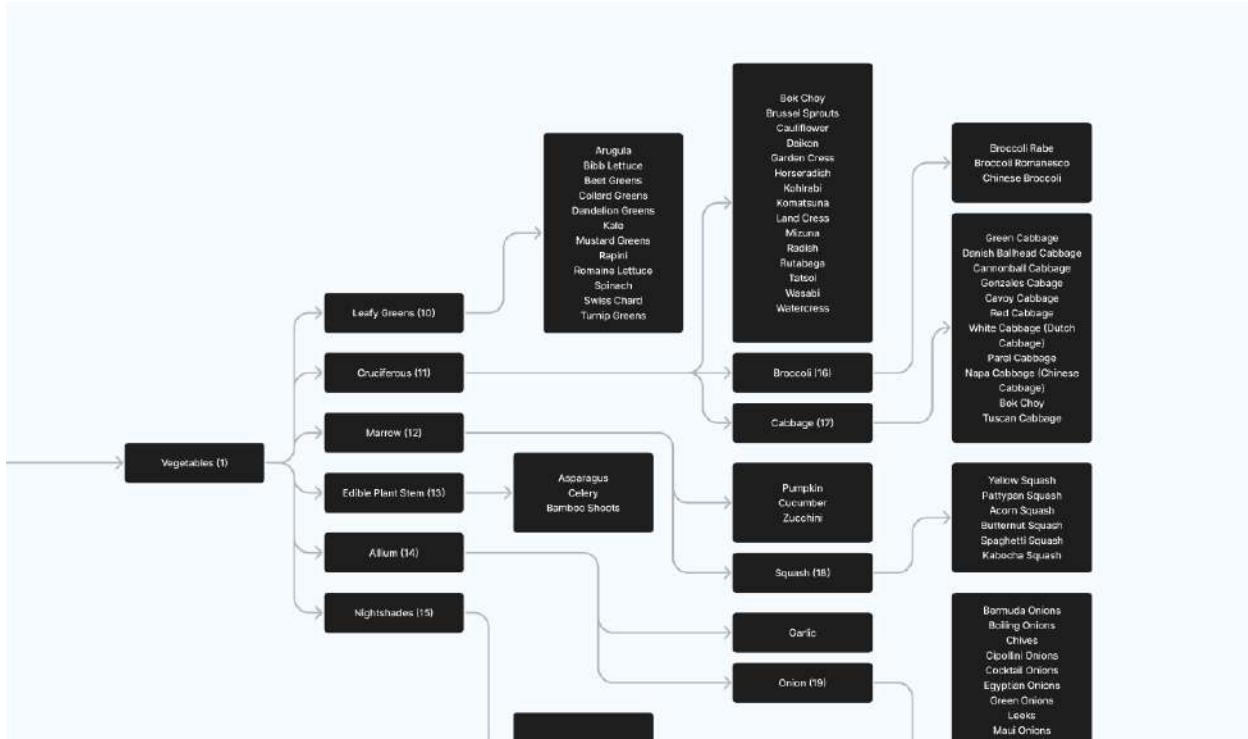
The ingredients table is intended for restaurants to select what is in each of their dishes and drinks. This data for each item can then be used to recommend dishes by analysing similarities between each order and giving a better recommendation of what to eat, connecting the dots and correlating the similarities of different users. The ingredients table also has the other use of allowing the user to select which ingredients they are allergic to and have the app then prevent the user from getting food that has those ingredients in.

With the number of ingredients that are in the world and the possibility for most of them to be served in a restaurant, I will not be doing every ingredient but if it ever needs to be maintained, new ingredients and ingredient categories can be added easily.

After lots of research, I found that there was no suitable database which would help with the correct ingredients and categorisation. Some databases contained too much extensive data while others contained lots of useless information which was hard to sort through. In the end, I decided to categorise everything manually, using a diagram to visualise categories and subcategories. In the end, it resulted in 475 ingredients and 46 categories & subcategories.

Table: IngredientCategories Filter in any column

	CategoryID	CategoryName	CategoryParentID
1	1	Vegetables	NULL
2	2	Fruits	NULL
3	3	Spices & Herbs	NULL
4	4	Cereals	NULL
5	5	Meats	NULL
6	6	Dairy Products / Dairy Alternatives	NULL
7	7	Seafood	NULL
8	8	Nuts & Seeds	NULL
9	9	Syrups	NULL
10	10	Leafy Greens	1
11	11	Cruciferous	1
12	12	Marrow	1
13	13	Edible Plant Stem	1
14	14	Allium	1
15	15	Nightshades	1
16	16	Broccoli	11
17	17	Cabbage	11
18	18	Squash	12
19	19	Onion	14
20	20	Tomatoes	15
21	21	Potatoes	15
22	22	Peppers	15
23	23	Drupes	2
24	24	Berries	2
25	25	Pomes	2
26	26	Pepos	2
27	27	Citrus	2
28	28	Dairy	6
29	29	Dairy Alternatives	6
30	30	Milk	28
31	31	Butter	28
32	32	Cheese	28
33	33	Yogurt	28
34	34	Cream	28



Restaurants Table

Tables: *Restaurant*, *RestaurantTimes*

When adding restaurants to the database, I was not able to get in contact to make arrangements with all the restaurants nearby so instead, I took the data from existing food delivery apps and the restaurants' websites to fill in the details. For the test data, I will import 8 restaurants since each one has a lot of data within it that needs to be filled in.

Table: Restaurant							
RestaurantID	RestaurantName	Address	Website	PhoneNumber	CuisineRating	Description	Link
1	Papa John's Pizza	211b High Street, Berkhamste...	https://www.papajohns.co.uk/	1442862900	3	The iconic, internationally-...	http
2	PizzaExpress	300 High Street, Berkhamsted...	https://www.pizzaexpress.com/	1442750510	4	NULL	http
3	The Fat Buddha	378 High Street, Berkhamsted...	https://www.thefatbuddha.co.uk	1442879995	5	NULL	http
4	The Meating Room	307 High Street, Berkhamsted...	https://www.meating-room.co.uk	1442879994	5	Our menu, though simple, is ...	http
5	Starbucks	200 High Street, Berkhamsted...	https://www.starbucks.co.uk/	1422777800	5	NULL	http
6	Zero Sushi	8-12 Lower Kings Road, ...	https://zerosushi.co.uk/	1442877982	NULL	Zero aims to delivery ...	http
7	Dojo Asian Kitchen	43 Lower Kings Road, ...	https://www.tasty-af.co.uk	NULL	4	NULL	http
8	Riverside Fish and Chips	14 Bridge Street, Hemel ...	https://www.riversidefishnchips.co.uk/	1442244488	5	NULL	http

Table: RestaurantTimes

RestaurantID	DayOfWeek	Status	OpenTime	CloseTime
Filter	Filter	Filter	Filter	Filter
29	5 Mon	Open	700	1830
30	5 Tue	Open	700	1830
31	5 Wed	Open	700	1830
32	5 Thu	Open	700	1830
33	5 Fri	Open	700	1830
34	5 Sat	Open	700	1830
35	5 Sun	Open	800	1700
36	6 Mon	Closed	0	0
37	6 Tue	Open	1100	1700
38	6 Wed	Open	1100	1700
39	6 Thu	Open	1100	1700
40	6 Fri	Open	1100	1900
41	6 Sat	Open	1100	1900
42	6 Sun	Closed	0	0
43	7 Mon	Closed	0	0
44	7 Tue	Open	1700	2100
45	7 Wed	Open	1700	2100
46	7 Thu	Open	1700	2100
47	7 Fri	Open	1700	2100
48	7 Sat	Open	1700	2100
49	7 Sun	Open	1700	2000
50	8 Mon	Open	1200	2300
51	8 Tue	Open	1200	2300

29 - 52 of 56

Go to: 1

Table: Item

ItemID	CategoryID	Item Category	ItemName	Description	Price	Image
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	15 All the Meats	Our tomato sauce with ...	17.99	https://drive.google.com/file/d/1Umee0KEsOo261IdV9u7scUfD3u541Q/view?usp=sharing
2	2	1	15 American Hot	Our tomato sauce with ...	15.99	https://drive.google.com/file/d/1Uvew3qJUodqjEW0AYExd167XWnzhzvC/view?usp=sharing
3	3	1	15 BBQ Meet Feast	BBQ sauce with mozzarella...	17.99	https://drive.google.com/file/d/1V1kVUsyJnULSG-6u9A8mNNVdSkzJZ/view?usp=sharing
4	4	1	15 BBQ Chicken Classic	Our tomato sauce with ...	16.99	https://drive.google.com/file/d/1V6vZF_jwWseajM-l6hizszOladxv_W0f/view?usp=sharing
5	5	1	15 Cheese & Tomato	The classic. Full of flavour, ...	14.99	https://drive.google.com/file/d/1VkgkW9yCrlg6WF14D7PMflurWWavY/view?usp=sharing
6	6	2	15 BBQ Chicken & Bacon	Our new Italian flat-bread ...	6.49	https://drive.google.com/file/d/1VEpjM5Bsc73RCKE17az08fhfrVADope/view?usp=sharing
7	7	2	15 Italian Sausage & Pepperoni	Our new Italian flat-bread ...	5.99	https://drive.google.com/file/d/1V17DzV2XjBM8FCUq5tdc0AEV7mv/view?usp=sharing
8	8	2	15 Vegan Philly Mushroom	Our new Italian flat-bread ...	5.99	https://drive.google.com/file/d/1V3Lz60pp712_LnPVlxcmd8Y_-W0bS/view?usp=sharing
9	9	3	3 4 Cheese Garlic Sticks	Mozzarella topped with ree...	6.99	https://drive.google.com/file/d/1V5KMrkrh1lFHv4MFznMu29s8k0dT/view?usp=sharing
10	10	3	3 10 Chicken Poppers	Ten crispy coated chicken ...	6.49	https://drive.google.com/file/d/1VkfR0th2kkQyqeqnjGdnLbzotizzlvg/view?usp=sharing
11	11	4	NULL BBQ Dip	Choose from a range of di...	0.49	https://drive.google.com/file/d/1V1tgZPRll29_UvAlp0TluJg7g1_Tigby/view?usp=sharing
12	12	4	NULL Special Garlic Dip	Choose from a range of di...	0.49	https://drive.google.com/file/d/1V1Dz5564kmMdeAvW0XA2fPfpllkI/view?usp=sharing
13	13	4	NULL Garlic and Herb Dip	Choose from a range of di...	0.49	https://drive.google.com/file/d/1W4P1eba7Ha5hMOjjFFOEjNh5xU7baray/view?usp=sharing
14	14	4	NULL Hot Buffalo Dip	Choose from a range of di...	0.49	https://drive.google.com/file/d/1W5KMsbeMcqjVMd6gNLREeMA3qjUw9C/view?usp=sharing
15	15	4	NULL Tomato and Herb Dip	Choose from a range of di...	0.49	https://drive.google.com/file/d/1W7qqqvw_hPYEB9Wak3pXV15Ebzo2/view?usp=sharing
16	16	5	18 Pepsi Max (Small)	NULL	1.49	https://drive.google.com/file/d/1WFVm3_jYJmRkB10Age5PCMhInEjbqRj/view?usp=sharing
17	17	5	18 Pepsi Max (Large)	NULL	2.49	https://drive.google.com/file/d/1WF17ml0_0pehFK2keYsj6aEjhQayRj/view?usp=sharing
18	18	5	18 7UP Free	NULL	2.49	https://drive.google.com/file/d/1W3l17ml0_0pehFK2keYsj6aEjhQayRj/view?usp=sharing
19	19	6	6 Chocolate Fudge Brownie - Ben & Jerry's	Chocolate ice cream with ...	5.99	https://drive.google.com/file/d/1WLc6Dm7d92DQkeRuAxAXDD-0DjYtY/view?usp=sharing
20	20	6	6 Cookie Dough - Ben & Jerry's	Vanilla ice cream with ...	5.99	https://drive.google.com/file/d/1W75swM0beMcqjV632ccyjBrjY2EpHpkX3/view?usp=sharing
21	21	6	6 On Cookie Dough - Non-dairy - Ben & Jerry's	Caramel ice cream with ...	5.99	https://drive.google.com/file/d/1WR-ObpGP1TbKG1Sb7cuLcz2h21Tg6fN/view?usp=sharing
22	22	7	15 Piccolo To Go	A Piccolo Pizza, 4 dough ...	7.25	https://drive.google.com/file/d/1W1Gd-67xDAd0ukZDXDFqtvnMN_deDU/view?usp=sharing
23	23	7	15 Family Sharer	2 Pizzas and 2 sides. Build ...	28.95	https://drive.google.com/file/d/1W1wl0f8djQv632ccyjBrjY2EpHpkX3/view?usp=sharing
24	24	8	15 Margherita	The hero here is our ...	6.95	https://drive.google.com/file/d/1W1ZWwJTkzkdZ45in80lNGdulJx0x0/view?usp=sharing
25	25	8	15 American	On the menu since day one...	9.95	https://drive.google.com/file/d/1Wu0j52kzWshGMj4BhwScdAohNw8iTYKu/view?usp=sharing
26	26	8	15 Vegan Giardiniere	A veggie lover's delight. ...	12.45	https://drive.google.com/file/d/1WvjwWvSGMGh5aMxtunFadDucSWZIBTr/view?usp=sharing
27	27	9	22 Pollo Pesto	Our signature pesto dish: ...	12.95	https://drive.google.com/file/d/1Wzt5JleV5ge0Or0Udr1el_5W7_7j1Rn/view?usp=sharing
28	28	9	22 Bolognese	Beef Bolognese ragù with ...	12.95	https://drive.google.com/file/d/1X6wvQjQjWWezebhlmX4vP667ic5FZ2r/view?usp=sharing
29	29	10	15 Calzone Nduja	Spicy 'nduja sausage, ...	14.95	https://drive.google.com/file/d/125o1zlufRtnJzMF5kd51zU2022XAl9/view?usp=sharing
30	30	10	15 Calzone Verdure	Roasted sweet peppers an...	14.45	https://drive.google.com/file/d/126onyyyrsttpJ3cyKKY6YR0LH10P91-/view?usp=sharing

After multiple attempts to fix the database, it had caused many issues due to the program so I have moved to SQLiteStudio

DB Browser for SQLite

X

! Error checking foreign keys after table modification. The changes will be reverted.

OK

```

CREATE TABLE Ingredient (
    ItemID INTEGER NOT NULL PRIMARY KEY,
    IngredientName TEXT,
    CategoryID INTEGER NOT NULL FOREIGN KEY REFERENCES Categories (CategoryID)
);

CREATE TABLE Categories (
    CategoryID INTEGER NOT NULL PRIMARY KEY,
    CategoryName TEXT NOT NULL
);

CREATE TABLE Tags (
    TagID INTEGER NOT NULL PRIMARY KEY,
    TagName TEXT NOT NULL
);

CREATE TABLE Ingredients (
    ItemID INTEGER NOT NULL PRIMARY KEY,
    IngredientName TEXT NOT NULL,
    CategoryID INTEGER NOT NULL FOREIGN KEY REFERENCES Categories (CategoryID),
    TagID INTEGER NOT NULL FOREIGN KEY REFERENCES Tags (TagID)
);

CREATE TABLE Items (
    ItemID INTEGER NOT NULL PRIMARY KEY,
    ItemName TEXT NOT NULL,
    Description TEXT NOT NULL,
    Price REAL NOT NULL,
    Rating REAL NOT NULL,
    Ambience TEXT NOT NULL,
    Value TEXT NOT NULL,
    Service TEXT NOT NULL,
    Hygiene TEXT NOT NULL,
    Food TEXT NOT NULL
);

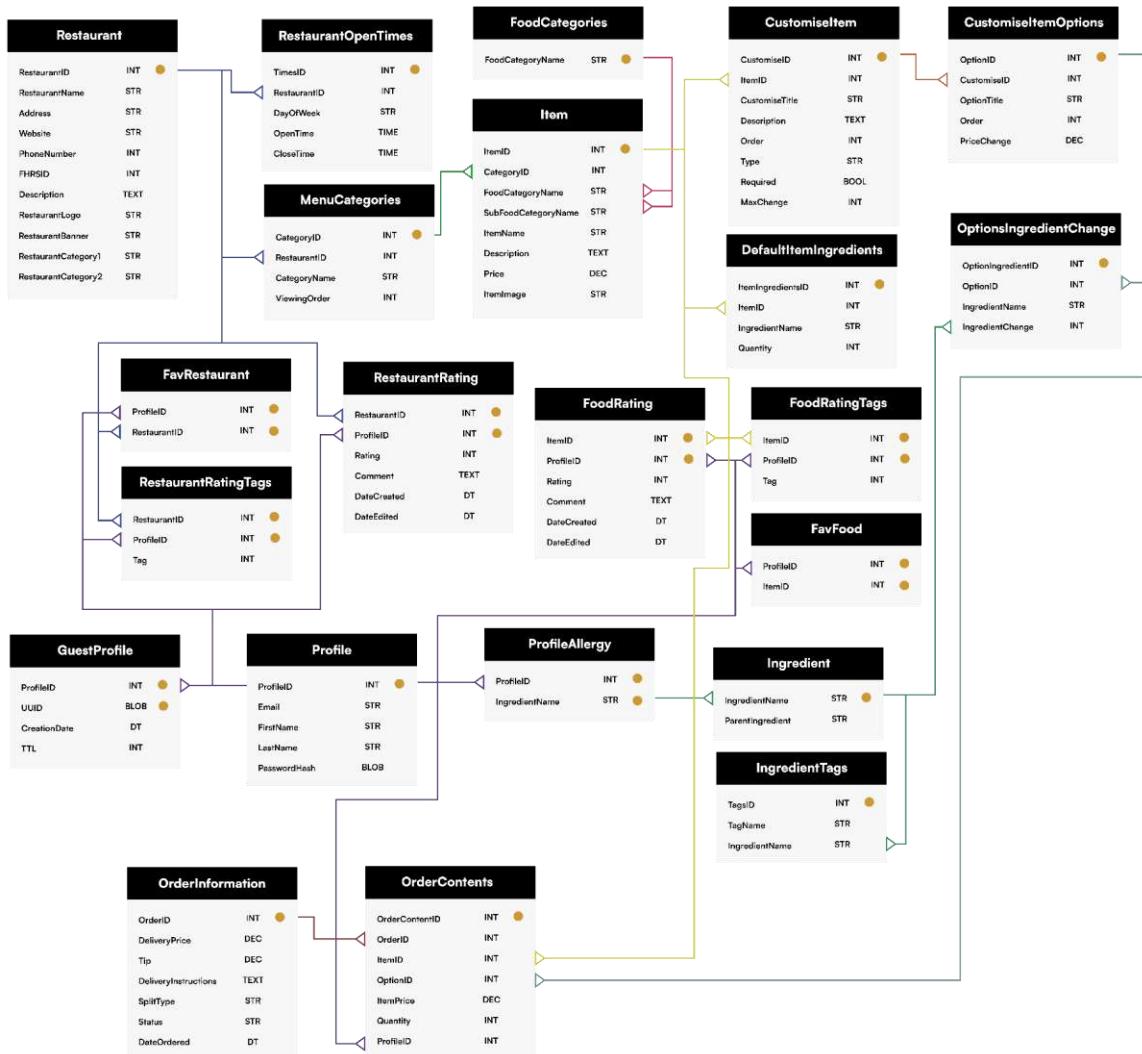
CREATE TABLE Orders (
    OrderID INTEGER NOT NULL PRIMARY KEY,
    ProfileID INTEGER NOT NULL,
    DeliveryMethod TEXT NOT NULL,
    Email TEXT NOT NULL,
    Phone TEXT NOT NULL
);

```

When remaking the database in the new software, I noticed I could have made some changes in order to improve the efficiency and ease of use. Instead of including ingredients that people will not have, I will only include ingredients that people with allergies have. As well, instead of having a separate database for category names, I will have the ingredient name and the parent ingredient as two columns in the same table so that people can put the name instead of a combination of names and numbers. I have removed specifics like specific types of ingredients like cheeses. There were many more changes but in the end, a lot changed so I decided to include the new database below.

- Restaurant category and sub-category added to ease the process of filtering by category.
- Restaurant open times remade to not have status built in. If a restaurant has an entry for the day, it means they are open that day. If they don't have an entry, they are closed
- Restaurant ratings appearance, ambience, value, service, hygiene and food have been removed from the restaurant ratings table since it was a not very useful metric.
- Food categories have been cut down to just category name since it can be used as a primary key and since the relationship between each category is a web not a tree, it cannot be put into the table.
- The items table has had a sub food category added so that if it falls under 2, they can be marked down and will show up under 2 categories
- On the customise item table, the required field has been added which will allow for the app to know if the customise section is required or not. Max change has also been added to ensure that if an option is customised there is a maximum limit to the number of things added or removed.
- For the options ingredient change table, the ingredient change has been added to ensure that if there is more than one ingredient within the dish and one is removed but not the other, it still flags the user that the ingredient is still in the dish

Updated Database



CustomiseItem

CustomiseI	ItemID	CustomiseTitle	Description	Order	Type	Required	MaxChangi
1	1	1 Select your Crust	NULL	1	SELECT	TRUE	NULL
2	2	1 Select the Size	NULL	2	SELECT	TRUE	NULL
3	3	1 Defaults	NULL	3	REMOVE	FALSE	NULL
4	4	1 Optionals	NULL	4	ADD	FALSE	20
5	5	2 Select your Crust	NULL	1	SELECT	TRUE	NULL
6	6	2 Select the Size	NULL	2	SELECT	TRUE	NULL
7	7	2 Defaults	NULL	3	REMOVE	FALSE	NULL
8	8	2 Optionals	NULL	4	ADD	FALSE	20
9	9	3 Select your Crust	NULL	1	SELECT	TRUE	NULL
10	10	3 Select the Size	NULL	2	SELECT	TRUE	NULL
11	11	3 Defaults	NULL	3	REMOVE	FALSE	NULL
12	12	3 Optionals	NULL	4	ADD	FALSE	20
13	13	4 Select your Crust	NULL	1	SELECT	TRUE	NULL
14	14	4 Select the Size	NULL	2	SELECT	TRUE	NULL
15	15	4 Defaults	NULL	3	REMOVE	FALSE	NULL
16	16	4 Optionals	NULL	4	ADD	FALSE	20
17	17	5 Select your Crust	NULL	1	SELECT	TRUE	NULL
18	18	5 Select the Size	NULL	2	SELECT	TRUE	NULL
19	19	5 Defaults	NULL	3	REMOVE	FALSE	NULL
20	20	5 Optionals	NULL	4	ADD	FALSE	20
21	21	9 Defaults	NULL	1	REMOVE	FALSE	NULL
22	22	9 Optionals	NULL	2	ADD	FALSE	20
23	23	10 Pick One	NULL	1	SELECT	TRUE	NULL
24	24	10 Optionals	NULL	2	ADD	FALSE	20
25	25	22 Pick your Dough Balls	NULL	1	SELECT	TRUE	NULL
26	26	22 Pick your Piccolo to Go Pizza	NULL	2	SELECT	TRUE	NULL
27	27	22 Pick your Cawston Press	NULL	3	SELECT	TRUE	NULL
28	28	23 Pick your Dough Balls / Garlic Bread	NULL	1	ADD	TRUE	2
29	29	23 Pick your 2 Pizzas	NULL	2	ADD	TRUE	2
30	30	24 Which Base?	NULL	1	SELECT	TRUE	NULL
31	31	24 Would you like to add any extra toppings?	NULL	2	ADD	FALSE	15
32	32	25 Which Base?	NULL	1	SELECT	TRUE	NULL
33	33	25 Would you like to add any extra toppings?	NULL	2	ADD	FALSE	15
34	34	26 Which Base?	NULL	1	SELECT	TRUE	NULL

CustomiseItemOptions

	OptionID	CustomiseID	OptionTitle	Order	PriceChange
1	1		1 Original	1	0
2	2		1 Authentic Thin Crust	2	0
3	3		1 Stuffed Crust	3	2.99
4	4		2 Small	1	0
5	5		2 Medium	2	2
6	6		2 Large	3	4
7	7		2 XXL	4	6
8	8		3 Bacon	1	0
9	9		3 Base Pizza Sauce	2	0
10	10		3 Ham	3	0
11	11		3 Mozzarella Cheese	4	0
12	12		3 Pepperoni	5	0
13	13		3 Pork Sausage	6	0
14	14		3 Spicy Beef	7	0
15	15		3 Dip Special Garlic	8	0
16	16		4 Anchovies	1	1.5
17	17		4 Bacon	2	1.5
18	18		4 Base Pizza Sauce	3	1.5
19	19		4 Barbecue Drizzle	4	1.5
20	20		4 Black Olives	5	1.5
21	21		4 Chargrilled Chicken	6	1.5
22	22		4 Fresh Tomatoes	7	1.5
23	23		4 Green Peppers	8	1.5
24	24		4 Ham	9	1.5
25	25		4 Italian Sausage	10	1.5
26	26		4 Jackfruit "Pepperoni"	11	1.5
27	27		4 Jalapeno Peppers	12	1.5
28	28		4 Mozzarella Cheese	13	1.5
29	29		4 Mushrooms	14	1.5
30	30		4 Onions	15	1.5
31	31		4 Pepperoni	16	1.5
32	32		4 Pineapple	17	1.5
33	33		4 Pork Sausage	18	1.5
34	34		4 Red Chillies Peppers	19	1.5

DefaultItemIngredients

	ItemIngredientsID	ItemID	IngredientName	Quantity
1	1	1	Pork	4
2	2	1	Beef	2
3	3	1	Milk	2
4	4	1	Mustard	1
5	5	1	Wheat / Flour	1
6	6	2	Milk	2
7	7	2	Pork	1
8	8	2	Beef	1
9	9	2	Wheat / Flour	1
10	11	3	Milk	2
11	12	3	Pork	4
12	13	3	Beef	2
13	14	3	Wheat / Flour	1
14	15	3	Mustard	1
15	16	4	Milk	2
16	17	4	Chicken	1
17	18	4	Pork	1
18	19	4	Wheat / Flour	1
19	20	4	Soy	1
20	21	4	Mustard	2
21	22	5	Milk	2
22	23	5	Wheat / Flour	1
23	24	6	Chicken	1
24	25	6	Pork	1
25	26	6	Milk	1
26	27	6	Wheat / Flour	1
27	28	6	Soy	1
28	29	6	Mustard	1
29	30	7	Wheat / Flour	1
30	31	7	Mustard	1
31	33	7	Milk	1
32	34	7	Pork	1
33	35	7	Beef	1
34	36	8	Wheat / Flour	1

FoodCategories

	FoodCategoryName
1	Seafood
2	Burgers
3	American
4	Breakfast
5	Chinese
6	Dessert
7	Greek
8	Halal
9	Indian
10	Italian
11	Japanese
12	Lebanese
13	Mexican
14	Catering
15	Pizza
16	Sushi
17	Thai
18	Soft Drink
19	Wine
20	Beer
21	Iced Drink
22	Pasta
23	Curry
24	British
25	Coffee
26	Asian

Ingredient

	IngredientName	ParentIngredient
1	Barley	Cereals
2	Brown Rice	Cereals
3	Farro	Cereals
4	Quinoa	Cereals
5	Millet	Cereals
6	Oats	Cereals
7	White Rice	Cereals
8	Wild Rice	Cereals
9	Buckwheat	Cereals
10	Wheat / Flour	Cereals
11	Corn / Maize	Cereals
12	Rye	Cereals
13	Cereals	NULL
14	Celery	NULL
15	Celery Seeds	Celery
16	Celeriac	Celery
17	Egg	NULL
18	Seafood	NULL
19	Abalone	Fish
20	Anchovies	Fish
21	Barramundi	Fish
22	Black Cod	Fish
23	Bombay Duck	Fish
24	Breams	Fish
25	Brill	Fish
26	Carp	Fish
27	Clams	Shellfish
28	Cod	Fish
29	Coley	Fish
30	Crab	Shellfish
31	Crayfish	Shellfish
32	Cuttlefish	Shellfish
33	Dover Sole	Fish
34	Flounder	Fish

IngredientTags

	TagsID	TagName	IngredientName
1	1	Vegan	Egg
2	2	Vegan	Fish
3	3	Vegan	Shellfish
4	4	Vegan	Milk
5	5	Vegan	Egg
6	6	Vegan	Meat
7	7	Vegan	Honey
8	8	Vegan	Gelatin
9	9	Vegetarian	Meat
10	10	Vegetarian	Gelatin
11	11	Vegetarian	Fish
12	12	Vegetarian	Shellfish
13	13	Pescetarian	Meat
14	14	Pescetarian	Gelatin
15	15	Dairy-free	Milk
16	16	Gluten-free	Wheat / Flour
17	17	Gluten-free	Barley
18	18	Gluten-free	Rye

Item

ItemID	Category	FoodCode	SubFoodCode	DesName	Description	Price	Barcode
1	1	1	Pizza	NULL	All the Meats	17.99	https://drive.google.com/file/d/1UmeDkTzOe626Hid9yuYvLIQ3u54QI/view?usp=sharing
2	2	1	Pizza	NULL	American Hot	17.99	https://drive.google.com/file/d/1UwaeTeUoqfEW0AY3x16720WjHrC5iwm9gsharing
3	3	1	Pizza	NULL	BBQ Meat Feast	17.99	https://drive.google.com/file/d/1Vt1ByrdJLS-9e44BmNIV4skz23view?usp=sharing
4	4	1	Pizza	NULL	BBQ Chicken Classic	16.99	https://drive.google.com/file/d/1V02z_jwW5eM-10-62oBav8y_WfRview?usp=sharing
5	5	1	Pizza	NULL	Cheese & Tomato	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
6	6	2	Pizza	NULL	BBQ Chicken and Bacon	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
7	7	2	Pizza	NULL	Italian Sausage & Pepperoni	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
8	8	2	Pizza	NULL	Vegan Philly Mushroom	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
9	9	3	American	NULL	4 Cheese Garlic Sticks	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
10	10	3	American	NULL	10 Chicken Poppers	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
11	11	4	NULL	NULL	BBQ Dip	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
12	12	4	NULL	NULL	Special Garlic Dip	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
13	13	4	NULL	NULL	Garlic and Herb Dip	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
14	14	4	NULL	NULL	Hot Buffalo Dip	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
15	15	4	NULL	NULL	Tomato and Herb Dip	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
16	16	5	Soft Drink	NULL	Pepsi Max (Small)	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
17	17	5	Soft Drink	NULL	Pepsi Max (Large)	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
18	18	5	Soft Drink	NULL	7UP Fresh	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
19	19	6	Dessert	NULL	Chocolate Fudge Brownie - Ben & Jerry's	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
20	20	6	Dessert	NULL	Cookie Dough - Ben & Jerry's	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
21	21	6	Dessert	NULL	Cookies on Cenice Dough - Non-dairy - Ben & Jerry's	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
22	22	7	Pizza	NULL	Piccolo to Go	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
23	23	7	Pizza	NULL	Family Share	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
24	24	8	Pizza	NULL	Margherita	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
25	25	8	Pizza	NULL	American	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
26	26	8	Pizza	NULL	Vegan Gardenia	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
27	27	9	Pasta	NULL	Pesto	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
28	28	9	Pasta	NULL	Bolognese	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
29	29	10	Pizza	NULL	Cabone 'N' Up	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
30	30	10	Pizza	NULL	Cabone Verdura	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
31	31	11	NULL	NULL	House Dressing Dip	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
32	32	12	Italian	NULL	Nicose	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
33	33	14	Dessert	NULL	Tiramisu	14.99	https://drive.google.com/file/d/1V1Bkg0972e9gjWf1407PMlLurbmwVview?usp=sharing
					On the menu since '85. The big flavours of white anchovies, capers, olives and tuna, with baby spinach and feta cheese.	11.95	https://drive.google.com/file/d/1XTPAfYhysghmifcB9k2_9bedObvView?usp=sharing
					Coffee and Mesihi wine give this iconic dessert its kick.	6.25	https://drive.google.com/file/d/1XeK9lkLlkLlkCksOsMjQjctD7View?usp=sharing

MenuCategories

	CategoryID	RestaurantID	CategoryName	ViewingOrder
1	1		Pizza	1
2	2		Papadias	2
3	3		Sides	3
4	4		Dips	4
5	5		Drinks	5
6	6		Desserts	6
7	7		Pizzeria Bundles	1
8	8		Pizzas	2
9	9		Al Forno	3
10	10		Calzone	4
11	11		Dips	5
12	12		Salads	6
13	14		Desserts	8
14	15		Soft Drinks	9
15	16		Starters & Appetisers	1
16	17		Vegetable Sides	2
17	18		Breads and Rice	3
18	19		Chef's Specials	4
19	20		All Time Favourites	5
20	21		Seafood	6
21	22		Biryani Dishes	7
22	23		Tandoori Dishes	8
23	24		Drinks	1
24	25		Specials	2
25	26		Hand Crafted Burgers	3
26	27		Shakes	4
27	28		Sides	5
28	29		Sauces	6
29	30		Children's Meals	7
30	31		Desserts	8
31	32		Extras	9
32	33		What's New	1
33	34		Beverages	1
34	35		Food	2

OptionsIngredientChange

	OptionIngredientID	OptionID	IngredientName	IngredientChange
1	1	8	Pork	-1
2	2	10	Pork	-1
3	3	11	Milk	-1
4	4	12	Pork	-1
5	5	12	Beef	-1
6	6	13	Pork	-1
7	7	14	Beef	-1
8	8	15	Mustard	-1
9	9	16	Anchovies	1
10	10	17	Pork	1
11	11	19	Mustard	1
12	12	19	Barley	1
13	13	21	Chicken	1
14	14	24	Pork	1
15	15	25	Pork	1
16	16	28	Milk	1
17	17	31	Pork	1
18	18	31	Beef	1
19	19	33	Pork	1
20	20	35	Beef	1
21	21	37	Tuna	1
22	22	38	Oats	1
23	23	39	Wheat / Flour	1
24	24	39	Barley	1
25	25	41	Mustard	1
26	26	54	Milk	-1
27	27	55	Beef	-1
28	28	55	Pork	-1
29	29	56	Mustard	-1
30	30	57	Anchovies	1
31	31	58	Pork	1
32	32	60	Mustard	1
33	33	60	Barley	1
34	34	62	Chicken	1

Restaurant

Restaurant	RestaurantName	Address	Website	PhoneNumbr	FHRSID	Description	RestaurantLogo	RestaurantBanner	Restaurant
1	Papa John's Pizza	211b High Street, Berkhamsted, HP4 1AD	https://www.papajohns.co.uk/	1442862900	202320	The iconic, ...	https://...	https://drive.google.com/...	Pizza American
2	PizzaExpress	300 High Street, Berkhamsted, HP4 1ZZ	https://www.pizzaexpress.com/	1442750510	1285623	NULL	https://...	https://drive.google.com/...	Italian Pizza
3	The Fat Buddha	378 High Street, Berkhamsted, HP4 1HU	https://www.thefatbuddha.co.uk	1442879995	1252835	NULL	https://...	https://drive.google.com/...	Indian Curry
4	The Meating Room	307 High Street, Berkhamsted, HP4 1AL	https://www.meating-room.co.uk	1442879994	1150863	Our menu, ...	https://...	https://drive.google.com/...	Burgers British
5	Starbucks	200 High Street, Berkhamsted, HP4 3AP	https://www.starbucks.co.uk/	1422777800	1452823	NULL	https://...	https://drive.google.com/...	Breakfast Coffee
6	Zero Sushi	8-12 Lower Kings Road, Berkhamsted, HP4 2AE	https://zerosushi.co.uk/	1442877982	720477	Zero aims to ...	https://...	https://drive.google.com/...	Japanese Sushi
7	Dojo Asian Kitchen	43 Lower Kings Road, Berkhamsted, HP4 2AB	https://www.tasty-af.co.uk	NULL	918510	NULL	https://...	https://drive.google.com/...	Asian Chinese
8	Riverside Fish and Chips	14 Bridge Street, Hemel Hempstead, HP1 1EF	https://www.riversidefishnchips.co.uk/	1442244488	1181644	NULL	https://...	https://drive.google.com/...	British Burgers

RestaurantOpenTimes

	TimesID	RestaurantID	DayOfWeek	OpenTime	CloseTime
1	1	1	Mon	1100	2300
2	2	1	Tue	1100	2300
3	3	1	Wed	1100	2300
4	4	1	Thu	1100	2300
5	5	1	Fri	1100	2300
6	6	1	Sat	1100	2300
7	7	1	Sun	1100	2300
8	8	2	Mon	1130	2230
9	9	2	Tue	1130	2230
10	10	2	Wed	1130	2230
11	11	2	Thu	1130	2230
12	12	2	Fri	1130	2230
13	13	2	Sat	1130	2230
14	14	2	Sun	1130	2230
15	15	3	Mon	1700	2230
16	16	3	Tue	1700	2230
17	17	3	Wed	1700	2230
18	18	3	Thu	1700	2230
19	19	3	Fri	1700	2230
20	20	3	Sat	1700	2230
21	21	3	Sun	1200	2200
22	22	4	Mon	1700	2200
23	23	4	Tue	1700	2200
24	24	4	Wed	1700	2200
25	25	4	Thu	1700	2200
26	26	4	Fri	1700	2200
27	27	4	Sat	1200	2200
28	28	4	Sun	1200	2100
29	29	5	Mon	700	1830
30	30	5	Tue	700	1830
31	31	5	Wed	700	1830
32	32	5	Thu	700	1830
33	33	5	Fri	700	1830
34	34	5	Sat	700	1830

App Development

For app development for prototype 1A, I will be implementing the following features:

- Login & Profile Creation
- Favourites
- Profile Switching

With the number of features of this app, in order to get a final result, I will be implementing them in at a later time, with this prototype being a restaurant viewer.

I will be creating the app on Visual Studio Code and testing them on PC due to its hot reload which is able to remake the app in seconds.

Login & Profile Creation

As I had no experience with Flutter and Dart, I used the following to get a grip of the basics of forms and verification

<https://docs.flutter.dev/cookbook/forms/validation>

During this time, I had many issues installing Flutter onto the system.

```
import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:email_validator/email_validator.dart';

void main() {
    runApp(const MyApp());
}

class MyApp extends StatelessWidget {
```

```
    @override

    Widget build(BuildContext context) {
        const appTitle = 'Register Profile';

        return MaterialApp(
            title: appTitle,
            home: Scaffold(
                appBar: AppBar(
                    title: const Text(appTitle),
                ),
                body: const MyCustomForm(),
            ),
        );
    }
}

// Create a Form widget.

class MyCustomForm extends StatefulWidget {
    const MyCustomForm({super.key});

    @override
    MyCustomFormState createState() {

```

```
        return MyCustomFormState();  
    }  
}  
  
// Create a corresponding State class.  
  
// This class holds data related to the form.  
  
class MyCustomFormState extends State<MyCustomForm> {  
  
    // Create a global key that uniquely identifies the Form widget  
    // and allows validation of the form.  
  
    //  
  
    // Note: This is a GlobalKey<FormState>,  
    // not a GlobalKey<MyCustomFormState>.  
  
    final _formKey = GlobalKey<FormState>();  
  
    static final validCharacters = RegExp(r'^[a-zA-Z0-9_\\-=@,\\.\\;]+$');  
  
    final passwordController = TextEditingController();  
  
  
    @override  
  
    Widget build(BuildContext context) {  
  
        // Build a Form widget using the _formKey created above.  
  
        return Form(  
            key: _formKey,  
            child: Column(  
                crossAxisAlignment: CrossAxisAlignment.start,  
                children:  
                    [  
                        // Input fields and validation logic...  
                    ]  
            ),  
        );  
    }  
}
```



```
        }

        return null;

    } ,

) ) ,

) ,

Expanded(

    child: ListTile(

        title: TextFormField(

            decoration: (const InputDecoration(


                labelText: 'Surname',


            ) ,


            inputFormatters: [


                FilteringTextInputFormatter.allow(RegExp('^[a-zA-Z.-]'))


            ] ,


            validator: (surname) {


                if (surname == null || surname.isEmpty) {


                    return "Required";


                }


                if (surname.length > 50) {


                    return "Surname too long";


                }


                if (surname.length < 2) {


                    return "Surname must be a minimum of 2 characters";

```

```
        }

        return null;

    } ,

) ) )

] ) ,  
ListTile(  
    title: TextFormField(  
        decoration: (const InputDecoration(  
            labelText: 'Email',  
            icon: Padding(  
                padding: EdgeInsets.only(top: 15.0),  
                child: Icon(Icons.email))),  
            inputFormatters: [  
                FilteringTextInputFormatter.allow(RegExp('[a-zA-Z0-9@,. -]'))  
            ] ,  
            validator: (email) {  
                if (email == null || email.isEmpty) {  
                    return "required";  
                }  
                if (EmailValidator.validate(email) == false) {  
                    return "Please enter valid email";  
                }  
                return null;  
            }  
        )  
    )  
);
```



```
    if (password.length > 99) {

        return "Password must be under 99 characters";

    }

    if (!password.contains(RegExp(r'[0-9]')) ||

        !password.contains(RegExp(r'[a-z]')))) {

        return "Password must contain at least 1 number and a
letter";

    }

    return null;

} ,

) ,

) ,

ListTile(
    title: TextFormField(
        keyboardType: TextInputType.visiblePassword,
        decoration: (const InputDecoration(
            labelText: 'Confirm Password',
            icon: Padding(
                padding: EdgeInsets.only(top: 15.0),
                child: Icon(Icons.lock)))),
        inputFormatters: [
            FilteringTextInputFormatter.allow(
                RegExp('[a-zA-Z0-9!@#%^&*(),.?":{}|<>]'))
        ]
);
```

```
],  
  
    obscureText: true,  
  
    // ignore: body_might_complete_normally_nullable  
    validator: (confirmPassword) {  
  
      if (confirmPassword == null || confirmPassword.isEmpty) {  
  
        return "Required";  
  
      }  
  
      if (confirmPassword != passwordController.text) {  
  
        return "Invalid Password";  
  
      }  
  
      if (confirmPassword.length < 8) {  
  
        return "Invalid Password";  
  
      }  
  
      if (confirmPassword.length > 99) {  
  
        return "Invalid Password";  
  
      }  
  
      if (!confirmPassword.contains(RegExp(r'[0-9]'))) ||  
  
          !confirmPassword.contains(RegExp(r'[a-z]'))) {  
  
        return "Invalid Password";  
  
      }  
  
      return null;  
  
    },  
  
  ),
```

```
),
const SizedBox(height: 50),
Center(
  child: Container(
    height: 70,
    width: 200,
    padding: const EdgeInsets.symmetric(vertical: 16.0),
    child: ElevatedButton(
      style: ButtonStyle(
        shape:
MaterialStateProperty.all<RoundedRectangleBorder>(
        RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10.0),
      ))),
      onPressed: () {
        // Validate returns true if the form is valid, or false
otherwise.
        if (_formKey.currentState!.validate()) {
          // If the form is valid, display a snackbar. In the real
world,
          // you'd often call a server or save the information in
a database.
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Processing Data'))),
        }
      },
    ),
  ),
);
```

```
        ) ;  
  
    }  
  
    ,  
  
    child: const Text('Submit'),  
  
    ),  
  
))  
  
],  
  
) ,  
  
) ;  
  
}  
}
```

After spending some time making the register page with a password checker, I did some research and found that to save the password into a database I would need a randomised salt that would encrypt the password and store the database. To then authenticate the password when logging in, I would need to use an algorithm to compare the hash to the password. During some research I found a package called FireBase which connects to Google servers and has the authentication built into it and has the ability to work with both iOS and Android which is very useful for this.

Problem 1

While trying to add the Firebase core package, it consistently kept giving me errors where Firebase was not opening. After 6 days of trying to find the answer, I gave up on Firebase and resorted back to the original plan, where the entire login and registration would be coded in, including encryption and hashing.

```
C:\All Eat\v2\alleat\android\app\src\debug\AndroidManifest.xml: Error:  
    uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library [com.google.firebaseio.firebaseio-analytics:21.1.0] C:\Users\iliyas\.gradle\caches\transforms-3\411025991a78f616ca1d1f292f6aff84\transformed\jetified-firebase-analytics-21.1.0\AndroidManifest.xml as the library might be using APIs not available in 16  
        Suggestion: use a compatible library with a minSdk of at most 16,  
        or increase this project's minSdk version to at least 19,  
        or use tools:overrideLibrary="com.google.firebaseio.firebaseio_analytics" to force usage (may lead to runtime failures)  
  
FAILURE: Build failed with an exception.  
  
* What went wrong:  
Execution failed for task ':app:processDebugMainManifest'.  
> Manifest merger failed : uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library [com.google.firebaseio.firebaseio-analytics:21.1.0] C:\Users\iliyas\.gradle\caches\transforms-3\411025991a78f616ca1d1f292f6aff84\transformed\jetified-firebase-analytics-21.1.0\AndroidManifest.xml as the library might be using APIs not available in 16  
        Suggestion: use a compatible library with a minSdk of at most 16,  
        or increase this project's minSdk version to at least 19,  
        or use tools:overrideLibrary="com.google.firebaseio.firebaseio_analytics" to force usage (may lead to runtime failures)  
  
* Try:  
> Run with --stacktrace option to get the stack trace.  
> Run with --info or --debug option to get more log output.  
> Run with --scan to get full insights.
```

I ended up using the code that I originally made to make the app. During the time when I was attempting to make the Firebase system, I looked over many videos where they had multiple pages for the login and registration pages, and used it to get a better grip on how Flutter works. When I went back to my app, I knew exactly what to do, learning about the different encryption methods and hashing methods used and ensuring data is kept secure and private.

The following progress was smooth with very smooth, having few issues. One issue I did end up finding which I didn't have before was overflowing elements.

After finishing making the register system, I realised that in order to have persistent users, the user email and hash needs to be stored in a local database. This will also allow for profile switching.

Register Profile

First Name Surname
Required Required

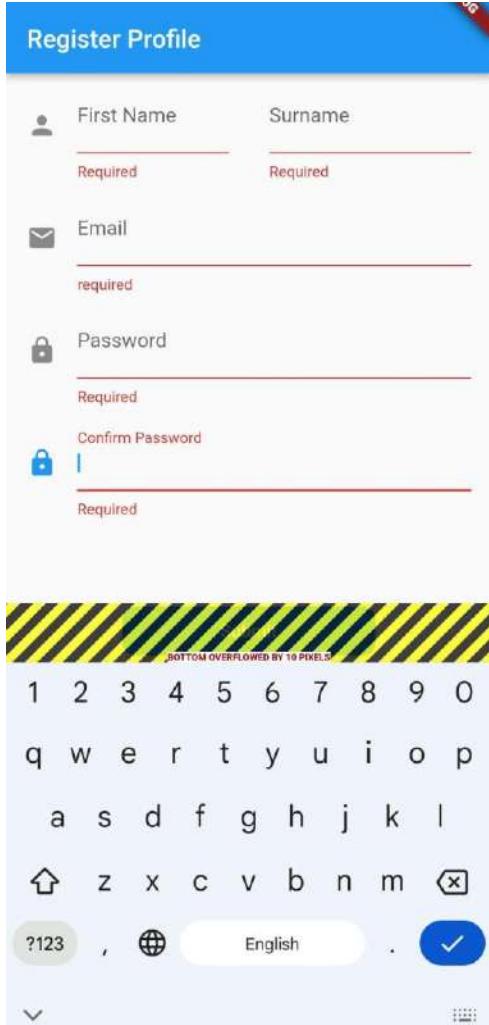
Email
required

Password
Required

Confirm Password
Required

BOTTOM OVERFLOWED BY 10 PIXELS

1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123 , English .



Problem 2

As it turned out, you needed to add a line of code to specifically enable scrolling on the element block. This as it turned out, to also fix another issue where if you had the app horizontal, it would allow you to scroll down.

Problem 3

Another issue I faced was with the server. While making the app, I was testing the app on my phone, I had the register function in place to send the data over to the local server on the computer. I forgot to set the url to the IP address instead of localhost which took me a while to figure out since it kept crashing the app and not reporting it to the IDE. After changing it I stumbled upon another issue where the server connection was timing out. To debug this, I used print statements between the lines to find out where the issue was starting but to no avail.

```
I/MSHandlerLifeCycle(27219): isMultiSplitHandlerRequested: windowingMode=1 isFullscreen=true isPopover=false isHidden=false skipActivityType=false isHandlerType=true this: DecorView@3018489[MainActivity]
I/flutter (27219): Got to freshregister
I/flutter (27219): 1
I/flutter (27219): validation complete
I/ViewRootImpl@9e7dcf[MainActivity](27219): ViewPostIme pointer 0
I/MSHandlerLifeCycle(27219): isMultiSplitHandlerRequested: windowingMode=1 isFullscreen=true isPopover=false
```

I finally decided to test it out on the computer with the server and it successfully went through.

```
Connecting to VM Service at ws://127.0.0.1:54209/p3ID0SWpnss=/ws
flutter: Got to freshregister
flutter: 1
flutter: validation complete
flutter: 2
flutter: "noexist""addedprofile"
```

Problem 4

Although this looked like it added the data to the database, in fact it had not since after checking the database, there were still no entries but this did mean that it was the php document that needed to be edited since the data was being received to the server, just not added.

As it turned out, the data being sent was not correct, and was sending the raw data

```
<tr><th align='left' bgcolor='#f57900' colspan="5"><span style='background-color: #cc0000; color: #fce94f; font-size: x-large;'>( ! )</span> Warning: Cannot modify header information - headers already sent by (output started at C:\wamp64\www\register.php:64) in C:\wamp64\www\register.php on line <i>90</i></th></tr>
<tr><th align='left' bgcolor='#e9b96e' colspan='5'>Call Stack</th></tr>
<tr><th align='center' bgcolor='#eeeeec'>#</th><th align='left' bgcolor='#eeeeec'>Time</th><th align='left' bgcolor='#eeeeec'>Memory</th><th align='left' bgcolor='#eeeeec'>Function</th><th align='left' bgcolor='#eeeeec'>Location</th></tr>
<tr><td bgcolor='#eeeeec' align='center'>1</td><td bgcolor='#eeeeec' align='center'>0.0030</td><td bgcolor='#eeeeec' align='right'>360456</td><td bgcolor='#eeeeec'>{main}(</td><td title='C:\wamp64\www\register.php' bgcolor='#eeeeec'>...\\register.php<b>:</b>0</td></tr>
<tr><td bgcolor='#eeeeec' align='center'>2</td><td bgcolor='#eeeeec' align='center'>0.0199</td><td bgcolor='#eeeeec' align='right'>392952</td><td bgcolor='#eeeeec'><a href='http://www.php.net/function.header' target='_new'>header</a>(<span>$header = </span><span>Content-Type: application/json</span></span> )</td><td title='C:\\wamp64\\www\\register.php' bgcolor='#eeeeec'>...\\register.php<b>:</b>90</td></tr>
</table></font>
```

To correct this, I rewrote the code for sending the data to the server.

Previously:

```
//Future freshRegisterProfile(BuildContext cont) async {
//  print("Got to freshregister");
//  var url = "http://192.168.0.22/register.php";
//  var response = await http.post(Uri.parse(url), body: {
//    "firstname": firstname.text,
//    "lastname": lastname.text,
//    "email": email.text,
//    "password": password.text,
//  });
//  print("Complete sending");
//  var data = json.decode(response.body);
//  if (data == "addedprofile") {
//    print("done");
//    Navigator.push(
//      cont, MaterialPageRoute(builder: (context) => FreshAllergies())
//    ) else {
//      Fluttertoast.showToast(
//        msg: "Failed to connect to server.",
//        toastLength: Toast.LENGTH_SHORT,
//        gravity: ToastGravity.BOTTOM,
//      );
//
//      var map = new Map<String, dynamic>();
//      map['firstname'] = 'firstname';
//      map['lastname'] = 'lastname';
//      map['email'] = 'email';
//      map['password'] = 'password';
//      print('1');
//      final response = await http.post(
//        Uri.parse('http://192.168.0.22/register.php'),
//        body: map,
//      );
//      print('2');
//      print(response.body);
//    }
//  }
//}
```

New:

```
late bool error, sending, success;
late String msg;

String phpurl = "http://192.168.0.22/register.php";
@Override
void initState() {
    error = false;
    sending = false;
    success = false;
    msg = "";
    super.initState();
}

Future<void> sendData() async { //On submit, start
    var res = await http.post(Uri.parse(phpurl), body: {
        "firstname": firstname.text,
        "lastname": lastname.text,
        "email": email.text,
        "password": password.text,
    });
    //Sending post request with data

    if (res.statusCode == 200) {
        var data = json.decode(res.body); //Decode to array
        if (data["error"]) {
            setState(() {
                //refresh the UI when error is received from server
                sending = false;
                error = true;
                msg = data["message"]; //error message from server
            });
        } else {
            firstname.text = "";
            lastname.text = "";
            email.text = "";
        }
    }
}
```

```

password.text = "";
//clear form

setState(() {
  sending = false;
  success = true; //mark success and refresh UI with setState
});
}
} else {
//there is error
setState(() {
  error = true;
  msg = "Connection to database failed.";
  sending = false;
//mark error and refresh UI with setState
});
}
}

```

Although this is a method of sending data over the internet, the password should be encrypted to ensure that it is not readable if it is intercepted. To do this, I used AES encryption. After finding that another package used the term "Key", I had to import the package as "encrypty." since it guaranteed that it would not be used by another package and still be understandable what package it is referencing.

Problem 5

While coding the encryption, I got this error:

```

269 final iv = encrypty.IV.fromLength(16);
270 final encrypter = encrypty.Encrypter(encrypty.AES(key));
D 271 encryptedPassword = encrypter.encrypt(plainText, iv: iv);
Exception has occurred. ×
_TypeError (type 'TextEditingController' is not a subtype of type 'String')
272 setState(() {

```

After doing some more research, I was able to convert it to a string but it just caused another error where it was type 'Encrypt' and was not able to find a solution to fix this so that I could send this data over. After trying to find other methods of encrypting the data like using TLS/SSL

or HTTPS, it seems that these have yet to be added to Flutter as mainstream packages. Luckily I found a different package which directly converts the string to an encrypted string.

Problem 6

Attempting to import it, I got a message indicating that this package was using a deprecated version of Android which meant I couldn't use it

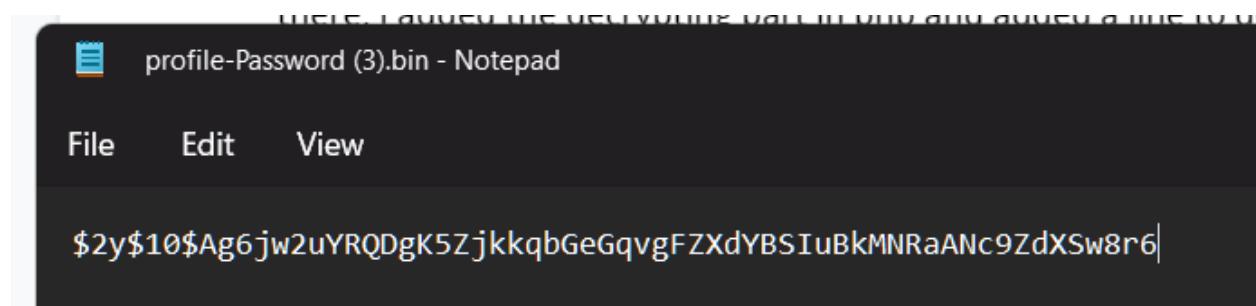
```
Changelog dependency!
The plugin "flutter_string_encryption" uses a deprecated version of the Android embedding.
To avoid unexpected runtime failures, or future build failures, try to see if this plugin supports the Android v2 embedding. Otherwise, consider removing it since a future release of flutter will remove these deprecated APIs.
If you are plugin author, take a look at the docs for migrating the plugin to the v2 embedding: https://flutter.dev/go/android-plugin-migration.
See CHANGELOG.md for details.
```

To get it to a string, as it turned out, it just required a single line of code ".base64". From there, I added the decrypting part in php and added a line to do hashing where it gets converted into a hash so that if there is ever a database leak, the data is kept secure.

```
$key = 'IlpqCiDpIr4WN1sh6Rzc4Q=='; //combination of 16 character
$iv = ''; //combination of 16 character
$method = 'aes-128-cbc';
$base64 = base64_decode($password);
$decryptedString = openssl_decrypt($base64, $method,
|   $key, OPENSSL_RAW_DATA, $iv);
$hashpassword = password_hash($password, PASSWORD_DEFAULT);
```

Problem 7

After forgetting that I needed to change the variables the other variables to \$hashpassword, it started working and it gave me a hashed 64 character password



This password was stored as a BLOB in the database to ensure it is not readable when viewing it and means that each password has to be checked individually meaning it is more secure.

After creating the server's database, I created the local database. This database would store the profiles that have been logged in. To store the user data, I used SQLite. I had originally not planned for this database but realised that it would be small, only containing things that should only include things that should not be synced with the main server database.

```
import 'package:flutter/foundation.dart';

import 'package:flutter/rendering.dart';

import 'package:sqflite/sqflite.dart' as sql;

class SQLiteLocalDB {

    static Future<void> createTables(sql.Database database) async {

        await database.execute("""CREATE TABLE localprofiles(
            id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
            firstname TEXT,
            lastname TEXT,
            email TEXT,
            password TEXT,
            selected TEXT,
            createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
        )
        """);

    }

    static Future<sql.Database> db() async {

```

```
return sql.openDatabase(
    'alleatlocal.db',
    version: 1,
    onCreate: (sql.Database database, int version) async {
        await createTables(database);
    },
);

}

// Create new profile

static Future<int> createProfile(
    String firstname, String lastname, String email, String password)
async {
    final db = await SQLiteLocalDB.db();

    final data = {
        'firstname': firstname,
        'lastname': lastname,
        'email': email,
        'password': password
    };

    final id = await db.insert('localprofiles', data,
        conflictAlgorithm: sql.ConflictAlgorithm.replace);
}
```

```
        return id;  
    }  
  
    // Get profiles  
  
    static Future<List<Map<String, dynamic>>> getProfiles() async {  
  
        final db = await SQLiteLocalDB.db();  
  
        return db.query('localprofiles', orderBy: "id");  
    }  
  
    // Set profile to be selected  
  
    static Future<List<Map<String, dynamic>>> setProfileSelected(int id)  
async {  
  
    final db = await SQLiteLocalDB.db();  
  
    db.execute('UPDATE localprofiles SET selected = "FALSE"');  
  
    db.execute('UPDATE localprofiles SET selected = "TRUE" WHERE id =  
"$id"');  
  
    return db.rawQuery('SELECT * FROM localprofiles WHERE selected =  
"TRUE" LIMIT 1');  
}  
  
    // Get selected profile  
  
    static Future<List<Map<String, dynamic>>> getProfileSelected() async {  
        final db = await SQLiteLocalDB.db();  
    }
```

```

        return db.rawQuery('SELECT * FROM localprofiles WHERE selected =
"TRUE" LIMIT 1');

    }

}

// Get profile info from id

static Future<List<Map<String, dynamic>>> getProfileFromID(int id) async {
    final db = await SQLiteLocalDB.db();

    return db.query('localprofiles',
        where: "id = ?", whereArgs: [id], limit: 1);
}

}

// Get profile info from email

static Future<List<Map<String, dynamic>>> getProfileFromEmail(String
email) async {

    final db = await SQLiteLocalDB.db();

    return db.rawQuery('SELECT * FROM localprofiles WHERE email = "$email"
LIMIT 1');
}

}

// Update an profile by id

static Future<int> updateProfile(int id, String firstname, String
lastname,
    String email, String password) async {

    final db = await SQLiteLocalDB.db();

```

```
final data = {  
    'firstname': firstname,  
    'lastname': lastname,  
    'email': email,  
    'password': password,  
};  
  
final result = await db  
    .update('localprofiles', data, where: "id = ?", whereArgs: [id]);  
return result;  
}  
  
// Delete profile  
  
static Future<void> deleteProfile(int id) async {  
    final db = await SQLiteLocalDB.db();  
    try {  
        await db.delete("localprofiles", where: "id = ?", whereArgs: [id]);  
    } catch (err) {  
        debugPrint("Something went wrong when deleting an item: $err");  
    }  
}  
}
```

To then have this part of the main program, I added a few lines of code

```
    await SQLiteLocalDB.createProfile(  
        firstname.text, lastname.text, email.text, data["hash"]);  
  
    // ignore: unused_local_variable  
    List<Map> profileInfo =  
        await SQLiteLocalDB.getProfileFromEmail(email.text);
```

What this does is creates the profile and then grabs the whole profile info from the email so that it can be used in the future including the product ID.

Problem 8

While testing the app, the server was not online one time which caused the whole app to crash since the connection was not available. I then spent the next day figuring out how to address all errors, redesigning the app to use try statements. During this, what happened was I was using Future statements to do the verification but when the if statement was checking if something was working, it would say it was false because as it turned out Future statements are completed after the main Build. This meant that you would have to press create profile multiple times before it would create it. To fix this, I did the if statements within another Future so they would happen in parallel.

```

Future<void> submitVerification() async {
    _checkConnectivityState();
    if (_connectivityResult == ConnectivityResult.wifi ||
        _connectivityResult == ConnectivityResult.mobile) {
        setState(() {
            //change status to sending to verify
            sending = true;
        });
        sendData(); //run
    } else {
        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
                content: Text('Internet connection not found. Please try again')), // Snackbar
        );
    }
}

Future<void> _checkConnectivityState() async {
    final ConnectivityResult result = await Connectivity().checkConnectivity();

    setState(() {
        _connectivityResult = result;
    });
}

```

```

setstate() {
    sending = false;
    success = true; //mark success and refresh UI with setState
    ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Profile Successfully Created.')),
    );
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => const FreshAllergies()), // MaterialPageRoute
    );
}

return success = true;

```

This fixed the error messages so next, I tackled the login system. Since I already did the register system, a login system was a lot easier, using much of the same code. The email and password was sent forward to the server where it was decrypted and converted to the cash. The account with the same email is checked if it exists and the hashes compared to see if they are the same.

Problem 9

While developing the login system, I stumbled upon another issue. Since I implemented the error catch system, I was not able to get all the info about which part of the connection between the server and the client was failing, only receiving "Failed to connect to server" as the error response since that was the name of the error I put in place.

In order to find out the error, I used Wireshark, a network inspection tool, to search network traffic. This also helped me find exactly what data was being sent over the network while also testing the security of the data being sent.

Frame	Source IP	Destination IP	Protocol	Details
7029	171.751214	192.168.0.95	TCP	74 44508 → 80 [SYN] Seq=1
7030	171.751363	192.168.0.22	TCP	74 80 → 44508 [SYN, ACK]
7031	171.764966	192.168.0.95	TCP	66 44508 → 80 [ACK] Seq=2
7032	171.823215	192.168.0.95	TCP	254 44508 → 80 [PSH, ACK]
7033	171.827081	192.168.0.95	HTTP	131 POST /login.php HTTP/1.1
7034	171.827158	192.168.0.22	TCP	66 80 → 44508 [ACK] Seq=3

```

> Frame 7024: 352 bytes on wire (2816 bits), 352 bytes captured (2816 bits) on interface \Device\NPF_{1FB
> Ethernet II, Src: ARRISGro_17:ac:08 (c0:05:c2:17:ac:08), Dst: 5e:7b:fd:23:3c:9c (5e:7b:fd:23:3c:9c)
> Internet Protocol Version 4, Src: 162.159.130.234, Dst: 192.168.0.22
> Transmission Control Protocol, Src Port: 443, Dst Port: 56988, Seq: 180255, Ack: 217, Len: 298
> Transport Layer Security

```

Frame	Source IP	Destination IP	Protocol	Details
0000	5e 7b fd 23 3c 9c c0 05	c2 17 ac 08 08 00 45 00		^{ .#<.....E-
0010	01 52 f9 e9 40 00 37 06	62 74 a2 9f 82 ea c0 a8		.R..@.7..bt.....
0020	00 16 01 bb de 9c 1c 89	52 03 5e 30 04 89 50 18	 R.^0..P-
0030	00 57 f0 4d 00 00 17 03	03 01 25 22 09 58 34 d6		.W.M.... .%"X4-
0040	95 c1 3a 24 58 0f b2 33	5a dc 65 da 19 b7 dd 66		..:\$X..3 Z.e....f
0050	b9 7e db 5a 91 5a 9c 83	17 5a c2 c7 a3 cd 6b e7		..~Z.Z... .Z....k.
0060	c3 b5 19 cd e9 02 54 53	3a 52 53 6c 69 46 95 34	TS :RSliF-4
0070	45 ce 58 09 01 5e aa b5	db 5e 8b 2e 96 af 6c ba		E.X..^. .^.1.
0080	ac 0c 6e 87 b2 74 8b e0	4f 0f 9f e6 80 38 44 57		..n..t.. 0....8DW
0090	1a 88 d4 08 2f f4 62 d9	fd 38 86 62 18 78 95 98	/.b. .8.b.x..
00a0	4b c6 8d 0d 66 d2 97 fc	59 ab cb ba 66 06 44 59		K....f... Y...f.DY
00b0	7a 59 42 f8 fe b4 71 af	0e 16 f8 95 6a a5 f4 12		zYB...q.j...
00c0	97 5a 7c 03 62 2d 93 62	7e ef 9c 88 ca 29 bb 81		.Z .b..b ~....).
00d0	66 d2 cc c4 ea a9 19 ec	7f 87 fd 53 51 af 65 4e		f..... SQ.eN

wireshark_WiFi0W9VQ1.pcapng || Packets: 7070 · Displayed: 7070 (100.0%) · Dropped: 0 (0.0%) || Profile: Default

```

[truncated]<tr><td bgcolor="#eeeeec" align="center">1</td><td bgcolor="#eeeeec" align="center">0.
[truncated]<tr><td bgcolor="#eeeeec" align="center">2</td><td bgcolor="#eeeeec" align="center">0.
</table></font>\n
<br />\n
<font size='1'><table class='xdebug-error xe-notice' dir='ltr' border='1' cellspacing='0' cellpadding='0'>
<tr><th align='left' bgcolor='#f57900' colspan="5"><span style='background-color: #cc0000; color: white; font-weight: bold; padding: 2px 5px;'>Call Stack</span></th></tr>\n
[truncated]<tr><th align='center' bgcolor="#eeeeec">#</th><th align='left' bgcolor="#eeeeec">Time</th>
[truncated]<tr><td bgcolor="#eeeeec" align="center">1</td><td bgcolor="#eeeeec" align="center">0.
</table></font>\n
{"error":true,"message":"Database error","hash":"1ZDZVJZ30i0kxcsGQE1awZ+jEYbw5Euvwawl+B3kMYMDFxVs1"

```

0de0	3e 0a 3c 2f 74 61 62 6c	65 3e 3c 2f 66 6f 6e 74	7b 22 65 72 72 6f	72 22 3a 74 72 75 65 2c
0df0	3e 0a	22 6d 65 73 73 61 67 65	22 3a 22 44 61 74 61 62	61 73 65 20 65 72 72 6f
0e00	7b 22	65 73 61 67 65	22 3a 22 44 61 74 61 62	72 22 2c 22 68 61 73 68
0e10	65 73	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	22 3a 22 31 5a 44 5a 56
0e20	65 73	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	4a 5a 33 30 69 30 6b 78
0e30	73 47	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	63 73 47 51 45 31 61 77
0e40	51 45	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	5a 2b 6a 45 59 62 77 35
0e50	31 45	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	45 75 76 77 61 77 6c 2b
0e60	77 61	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	42 33 6b 4d 59 4d 44 46
0e70	77 61	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	78 56 73 6c 4b 49 30 31
0e80	61 77	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	51 62 64 2b 54 5a 32 57
0e90	77 61	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	34 4e 6a 43 32 79 32 7a
0ea0	61 77	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	36 5c 2f 48 77 4a 51 35
0eb0	77 61	65 20 65 72 72 6f	72 22 2c 22 68 61 73 68	45 64 62 55 79 38 70 50
				32 4b 44 75 36 6e 55 55
				38 62 70 55 56 4b 38 78
				51 76 33 77 46 4c 4f 66
				42 4e 50 6c 66 30 41 53
				70 6a 6e 4e 64 49 35 74
				6a 54 30 62 22 2c 22 65
				78 69 73 74 73 22 3a 66
				61 6c 73 65 7d

Text item (text), 195 bytes || Packets: 7070 · Displayed: 7070 (100.0%) · Dropped: 0 (0.0%) || Profile: Default

It seems it was a database error and by referencing the login file we find that the \$res was not met.

```

if($res){
    //write success
}else{
    $return["error"] = true;
    $return["message"] = "Database error";
}

$res = mysqli_query($link, $sql);}

```

This means that the data sent to the database was not received successfully by the database. After doing some checks, it seems that even though the password provided are the same when registering and logging in, they have different hashes

Registering

```
Form item: "firstname" = "water"
  Key: firstname
  Value: water
Form item: "lastname" = "water"
  Key: lastname
  Value: water
Form item: "email" = "water@cpur.net"
  Key: email
  Value: water@cpur.net
Form item: "password" = "NC/wH2rsV8B8GXvoD9pk+A=="
  Key: password
  Value: NC/wH2rsV8B8GXvoD9pk+A==
```

```
File Data: 171 bytes
JavaScript Object Notation: application/json
  Object
    > Member: error
    > Member: message
    > Member: hash
      [Path with value: /hash:rTbs08+Ywwq0mNAMcKek4I5o/IZ1UN1WrJnx1NbNk/NEahME0JcBtyjz9+Z]
      [Member with value: hash:rTbs08+Ywwq0mNAMcKek4I5o/IZ1UN1WrJnx1NbNk/NEahME0JcBtyjz9+Z]
      String value: rTbs08+Ywwq0mNAMcKek4I5o/IZ1UN1WrJnx1NbNk/NEahME0JcBtyjz9+ZjRtVYGFo0Mk
      Key: hash
      [Path: /hash]
```

Logging in

```
File Data: 64 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "email" = "water@cpur.net"
    Key: email
    Value: water@cpur.net
  Form item: "password" = "NC/wH2rsV8B8GXvoD9pk+A=="
    Key: password
    Value: NC/wH2rsV8B8GXvoD9pk+A==
```

```
:correct","hash":"NFAYBpsakzemJI117UBBqqkvRGnG61H0zCWYPHR++r6LCjlJpGfU2bJ57dLap
```

After doing testing, I finally found the issue:

That is the way bcrypt works. Bcrypt is a stronger password hashing algorithm that will generate different hashes for the same value depending on the current system entropy, but that is able to compare if the original string can be hashed to an already hashed password.

To solve your problem use the `check()` function instead of the `hash()` function:

```
→add('current_password', 'custom', [
    'rule' => function($value, $context){
        $user = $this->get($context['data']['id']);
        if ($user) {
            if ((new DefaultPasswordHasher)->check($value, $user->password)) {
                return true;
            }
        }
        return false;
},
'message' => 'Você não confirmou a sua senha atual corretamente'
```

<https://stackoverflow.com/questions/29499287/defaultpasswordhasher-generating-different-hash-for-the-same-value>

According to this answer I found online, php's built-in hash creates a different hash each time but the hashes can be compared.

This means that I will need to do the login system differently. Previously, it was using an SQL select function to find the account associated with the email and password hash but what I will have to do instead, is get the account associated with an email and compare the hashes with each other.

3

The `password_hash` is used to hash a password, generating a new salt if necessary (you may pass your own salt, but this is not necessary), while the `password_verify` function is used to compare them (this function uses the salt stored alongside the hash in order to perform the proper comparison).



When creating the passwords, the user provides it twice on plaintext, so all you need to check is `$password1 == $password2`.

If you wish to verify the password obtained from the database, you need to do:

```
$password_hash = <some database query>
if (password_verify($password, $password_hash)) {
    login_success();
} else {
    login_failure();
}
```

You could compare the passwords in the register page the same way, but there's no need.

<https://stackoverflow.com/questions/35531880/compare-passwords-on-register-using-password-hash>

Before:

```

$email = mysqli_real_escape_string($link, $email);
$hashpassword = mysqli_real_escape_string($link, $hashpassword);
//escape inverted comma query conflict from string

$sql = "SELECT * FROM profile WHERE email = '$email' AND '$hashpassword'";
$userexist = mysqli_num_rows($result);
if($userexist == 1){
$return["exists"] = true;
$return["profile"] = $sql;
}
else{
$return["error"] = true;
$return["message"] = "Email or password incorrect";
}

```

While trying to do testing with passwords and hashes, I found that the program was never decrypting the password, and instead, getting a false result meaning that the key and iv were incorrect

```
;""DecryptedString = "false{
```

To fix this issue required a total redesign of the encryption system and decryption and after 7 hours, it was finally fixed. Thanks to Wireshark, it was easy, being able to pass back values and get the output, seeing if the data was even being sent and if things were encrypted.

As it turned out, the whole time, no data was being sent to the database, with the hashing algorithm creating a hash from null. This meant that it was impossible to tell that no data had been put into it.

Profile Creation

```

Future<bool> sendData() async {
    //On submit, start
    plainText = password.text;
    String strkey = 'ZC8cegCGG45d1IjIACTrfypDXtkgJ1rA+4JABPncUE=';
    String striv = 'yvhsTTh739b2yUW9NNWrcKmHTtLTZNbjbiV3F/cSRzM=';
    var iv = crypto.sha256
        .convert(convert.utf8.encode(striv))
        .toString()
        .substring(0, 16);
    var key = crypto.sha256
        .convert(convert.utf8.encode(strkey))
        .toString()
        .substring(0, 16);
    encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);
    encrypty.Key keyObj = encrypty.Key.fromUtf8(key);
    final encrypter =
        encrypty.Encrypter(encrypty.AES(keyObj, mode: encrypty.AESMode.cbc));
    encryptPassword = encrypter.encrypt(plainText, iv: ivObj);
    encryptPassword = encryptPassword.base64;
}

```

Login

```

Future<bool> loginUser() async {
    plainText = password.text;
    String strkey = 'ZC8cegCGG45d1IjIACTrfypDXtkgJ1rA+4JABPncUE=';
    String striv = 'yvhsTTh739b2yUW9NNWrcKmHTtLTZNbjbiV3F/cSRzM=';
    var iv = crypto.sha256
        .convert(convert.utf8.encode(striv))
        .toString()
        .substring(0, 16);
    var key = crypto.sha256
        .convert(convert.utf8.encode(strkey))
        .toString()
        .substring(0, 16);
    encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);
    encrypty.Key keyObj = encrypty.Key.fromUtf8(key);
    final encrypter =
        encrypty.Encrypter(encrypty.AES(keyObj, mode: encrypty.AESMode.cbc));
    encryptPassword = encrypter.encrypt(plainText, iv: ivObj);
    encryptPassword = encryptPassword.base64;
}

```

Register (Server)

```

if($val){
    //checking if there is POST data
    $firstname = $_POST["firstname"]; //grabing the data from headers
    $lastname = $_POST["lastname"];
    $email = $_POST["email"];
    $password = $_POST["password"];

    $secret_iv = '5fd0b31f582beb1f';
    $secret_key = 'e16a3f356b2366c1';
    $cipher = "AES-128-CBC";
    $decryptedString = openssl_decrypt ($password, $cipher, $secret_key, 0, $secret_iv);
    $hashpassword = password_hash($decryptedString, PASSWORD_DEFAULT);
    $base64hash = base64_encode($hashpassword);
    $encryptedhashedpassword = openssl_encrypt ($base64hash, $cipher, $secret_key, 0, $secret_iv);
    $return["hash"] = $encryptedhashedpassword;
}

```

Login (Server)

```

$email = $_POST["email"];
$password = $_POST["password"];

$secret_iv = '5fd0b31f582beb1f';
$secret_key = 'e16a3f356b2366c1';
$cipher = "AES-128-CBC";
$decryptedString = openssl_decrypt ($password, $cipher, $secret_key, 0, $secret_iv);

```

```

$sql = "SELECT FirstName, LastName, Email, Password FROM Profile WHERE Email = '$email'";
if($result = mysqli_query($link, $sql)){
    if(mysqli_num_rows($result) == 1){
        while($row = mysqli_fetch_array($result)){
            $FirstNameR = $row['FirstName'];
            $LastNameR = $row['LastName'];
            $EmailR = $row['Email'];
            $PasswordR = $row['Password'];
        }
        // Free result set
        mysqli_free_result($result);
        $ispasswordsame = password_verify($decryptedString, $PasswordR);
    } else{
        $return["error"] = true;
        $return["message"] = "No matching profiles";
    }
} else{
    $return["error"] = true;
    $return["message"] = "Failed profile search";
}

```

Persistent Profiles

In order for a user to not need to sign in every time they close the app, a persistent profile is needed to be kept in a database. As well, a check is needed to check if the user is still signed in and bring them to the starting page instead of the home screen.

To do this, I used a wrapper which does a check every time a user opens the app, signs out and in the future, when a guest profile's TTL runs out.

Problem 10

During the process of making the wrapper, I had a few issues where the whole app would cease to function and instead get an error. To fix this, I added a wait statement so that it wait for the future before continuing which fixed the issue

```
setState() or markNeedsBuild()
called during build.
This Overlay widget cannot be
marked as needing to build
because the framework is
already in the process of
building widgets. A widget can
be marked as needing to be
built during the build phase
only if one of its ancestors
is currently building. This
exception is allowed because
the framework builds parent
widgets before children, which
means a dirty descendant will
always be built. Otherwise,
the framework might not visit
this widget during this build
phase.
The widget on which setState()
or markNeedsBuild() was called
was:
  Overlay-[LabeledGlobalKey<Ov
erlayState>#679ef]
The widget which was currently
being built when the offending
call was made was:
  SetupWrapper
See also: https://flutter.dev/docs/testing/errors
```

```
import 'package:alleat/screens/Homepage/home.dart';

import 'package:alleat/screens/fresh_app/fresh_profile.dart';

import 'package:alleat/services/sqlite_service.dart';

import 'package:flutter/material.dart';

class SetupWrapper extends StatefulWidget {

  const SetupWrapper({Key? key}) : super(key: key);
```

```

@Override

State<SetupWrapper> createState() => _SetupWrapperState();

}

class _SetupWrapperState extends State<SetupWrapper> {

bool isComplete =

false; //Default setup not complete if something wrong happens

Future<void> isSetupComplete() async {

List<Map> profileInfo = await SQLiteLocalDB

.getFirstProfile(); //Call Database for the first entry

if (profileInfo.isEmpty) {

//If the first entry is empty

isComplete = false; //Then setup is not complete (pass to build)

} else if (profileInfo.isNotEmpty) {

//If the first entry exists

isComplete = true; //Setup is complete (pass to build)

}

}

@Override

Widget build(BuildContext context) {

```

```

isSetupComplete(); //Run check with Future

Future.delayed(Duration.zero, () {
    //Wait for Future to complete

    if (isComplete = false) {

        //If setup not complete

        Navigator.push(
            context,
            MaterialPageRoute(
                builder: (context) =>
                    const FreshProfile())); //Go to register/sign in page
    } else if (isComplete = true) {

        //If setup is complete

        Navigator.push(
            context,
            MaterialPageRoute(
                builder: (context) => const Homepage())); //Go to homepage
    }
});;

return MaterialApp(
    //If check fails, go to page with error code

    home: Scaffold(
        appBar: AppBar(
            title: const Text('Error (-2): Local database error.'),

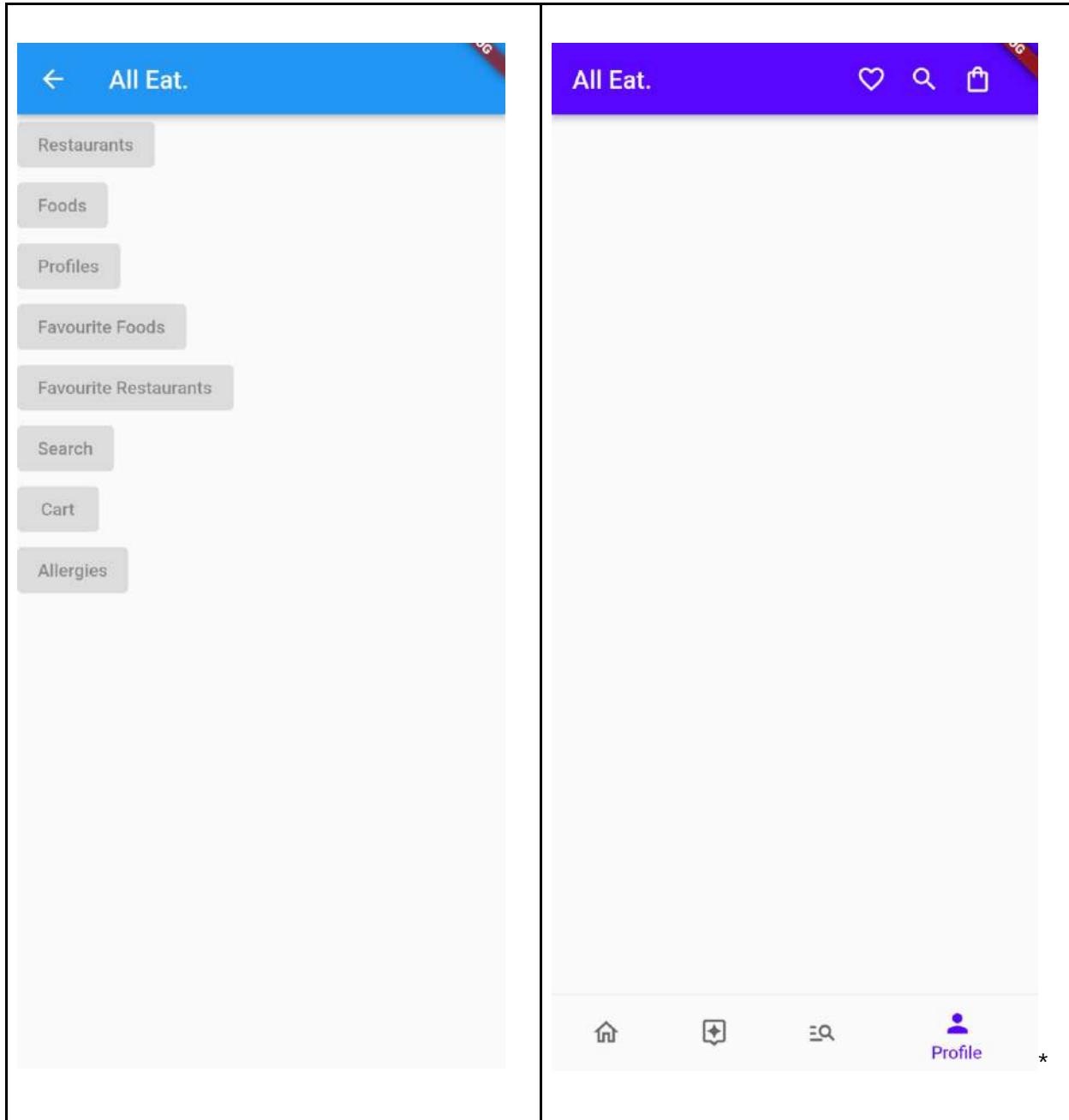
```

```
) ,  
)  
) ;  
}  
}
```

Navigation

Originally, I was going to do buttons to get to each part of the app but after realising that making a navigation bar is not as hard, I decided to remake the page to have it.

Before	After



```
import 'package:flutter/material.dart';

class HomePage extends StatefulWidget {

  const HomePage({Key? key}) : super(key: key);
```

```
    @override

    State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {

    int _selectedIndex = 0; //Start at Home page

    static const TextStyle optionStyle =
        TextStyle(fontSize: 30, fontWeight: FontWeight.bold);

    static const List<Widget> _widgetOptions = <Widget>[
        Text(
            'Index 0: Home',
            style: optionStyle,
        ),
        Text(
            'Index 1: For You',
            style: optionStyle,
        ),
        Text(
            'Index 2: Browse',
            style: optionStyle,
        ),
    ];
}
```

```
Text(  
    'Index 3: Profile',  
    style: optionStyle,  
)  
];  
  
void _onItemTapped(int index) {  
    //When user click button on bottom navigation bar, change to that  
index  
    setState(() {  
        _selectedIndex = index;  
    });  
}  
  
@override  
Widget build(BuildContext context) {  
    return MaterialApp(  
        title: "All Eat.",  
        theme: ThemeData(  
            scaffoldBackgroundColor: const Color(0xffFAFAFA),  
            primaryColor: const Color(0xff5806FF)),  
        home: Scaffold(  
            appBar: AppBar(
```

```
title: const Text("All Eat."),

backgroundColor: const Color(0xff5806FF),

actions: const [

    Icon(Icons.favorite_outline), //Top navigation bar icons

    Padding (

        padding: EdgeInsets.symmetric(horizontal: 18),

        child: Icon(Icons.search)),

    Icon(Icons.shopping_bag_outlined),

    Padding (

        padding: EdgeInsets.symmetric(horizontal: 18),

    )

],


),


body: Stack (

    children: <Widget>[

        Positioned (

            left: 0,

            right: 0,

            bottom: 0,

            child:

                bottomNavBar(), //Add navigation bar to the bottom of

the screen

        ),


    ],


)
```

```
        ] ,  
        ),  
    ) );  
  
}  
  
Widget bottomNavBar() {  
  
    return Container(  
  
        child: BottomNavigationBar(  
  
            selectedItemColor: const Color(0xff5806FF),  
            unselectedItemColor: const Color(0xff666666),  
            items: <BottomNavigationBarItem>[  
  
                //Bottom navigation bar items  
  
                BottomNavigationBarItem(  
  
                    icon: Container(  
  
                        padding: const EdgeInsets.only(  
  
                            bottom: 3), //Each have padding to separate from the  
text  
  
                        child: const Icon(  
  
                            Icons.home_outlined)), //Unselected icon is outlined  
  
                    label: 'Home',  
  
                    activeIcon: Container(  
  
                        padding: const EdgeInsets.only(bottom: 3),  
  
                        child: const Icon(Icons.home)), // Selected icon is filled
```

```
) ,  
  
    BottomNavigationBarItem(  
        icon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.assistant_outlined)),  
        label: 'For You',  
        activeIcon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.assistant)),  
    ),  
  
    BottomNavigationBarItem(  
        icon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.manage_search_rounded)),  
        label: 'Browse',  
        activeIcon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.manage_search)),  
    ),  
  
    BottomNavigationBarItem(  
        icon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.person_outlined)),  
    ),
```

```

        label: 'Profile',

        activeIcon: Container(
            padding: const EdgeInsets.only(bottom: 3),
            child: const Icon(Icons.person)),
        ),
    ],
onTap: _onItemTapped,
currentIndex: _selectedIndex,
) );
}

}

```

* After looking at some time constraints, the search function will be removed temporarily but will be revisited later.

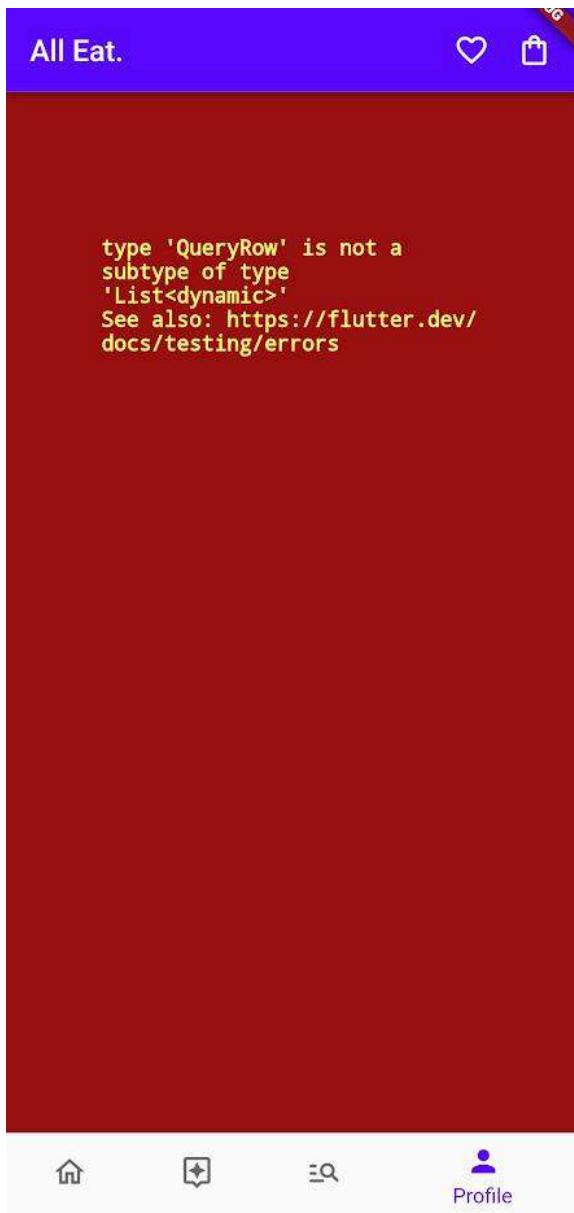
While starting work, I realised that in order to have separate files for each page, some of the code will need to be modified. The BottomNavBar was moved directly into the main page and a list of the page files was made and replaced the body so that depending on the index selected, a different body will be shown.

Profiles Page

The profiles page will currently have vertical containers with each containing the name of each profile and a button to remove it. Each profile will have an auto generated profile picture with the first two letters of their first name and a randomly generated colour. As well, there will be a button to add a new profile.

Problem 11

While developing the profiles page, I had an issue when taking the query response and making each profile have their own container



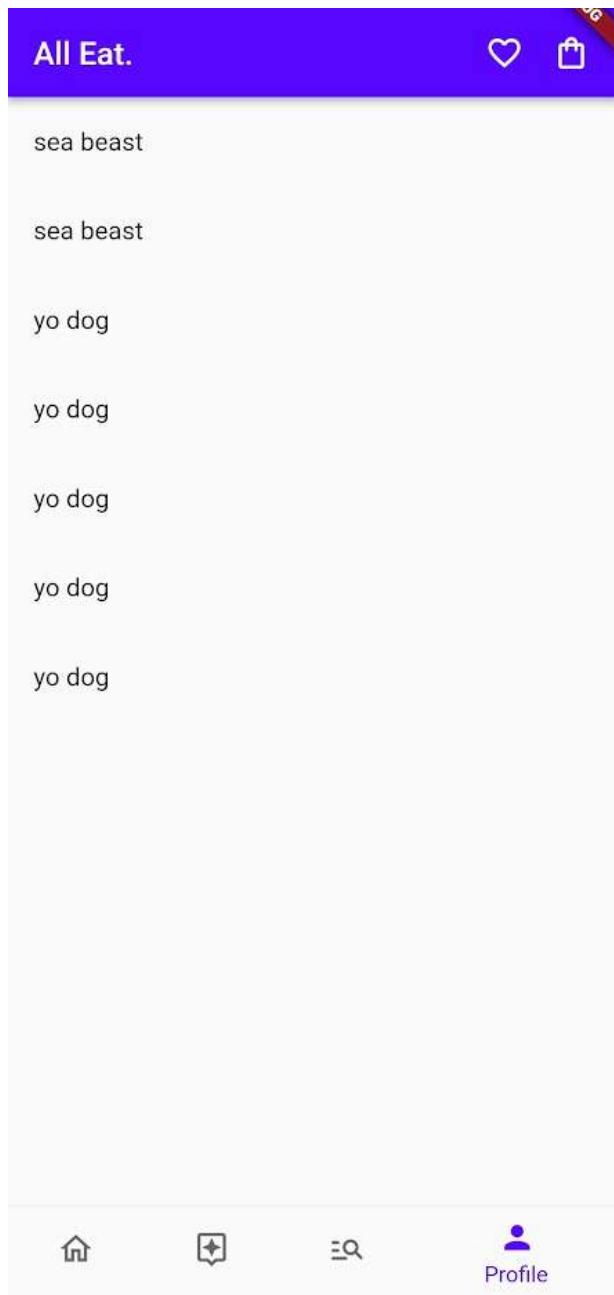
```
12  @override
13  Widget build(BuildContext context) {
14      getProfileInfo();
15      return FutureBuilder<List>(
16          future: getProfileInfo(),
17          builder: (context, snapshot) {
18              List profiles = snapshot.data ?? [];
19              return ListView.builder(
20                  itemCount: profiles.length,
21                  itemBuilder: (context, index) {
22                      Map<String, dynamic> profile = profiles[index];
23                      print(profile);
24                      return ListTile(title: profiles[index]);
```

Exception has occurred. ×
_TypeError (type 'QueryRow' is not a subtype of type 'Widget?')

```
25                  });
26              },
27          ); // FutureBuilder
```

In order to fix this, it just required me to change the type to a text widget

```
@override
Widget build(BuildContext context) {
    getProfileInfo();
    return FutureBuilder<List>(
        future: getProfileInfo(),
        builder: (context, snapshot) {
            List profiles = snapshot.data ?? [];
            return ListView.builder(
                itemCount: profiles.length,
                itemBuilder: (context, index) {
                    Map<String, dynamic> profile = profiles[index];
                    print(profile);
                    return ListTile(
                        title:
                            Text(profile["firstname"] + " " + profile["lastname"])); // ListTile
                });
        });
    ); // FutureBuilder
}
```

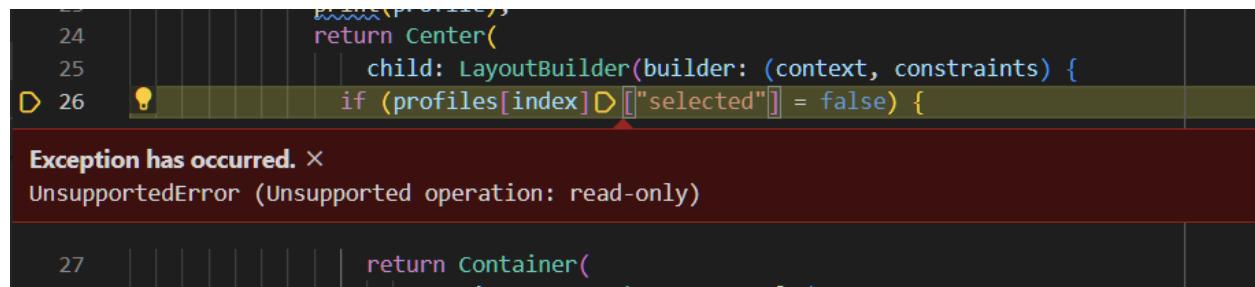


Problem 12

Although this did fix the issue, it resulted in me finding a new issue, a user is able to login to their profile multiple times. This happened when during development, it never redirected to the main page so it allowed you to keep logging in but should be fixed in the future to prevent errors.

Problem 13

During the making, I came across another issue while doing a check which profile is selected.



```
23 |     child: LayoutBuilder(builder: (context, constraints) {
24 |       return Center(
25 |         child: LayoutBuilder(builder: (context, constraints) {
D 26 |           if (profiles[index]["selected"] = false) {
27 |             return Container(
```

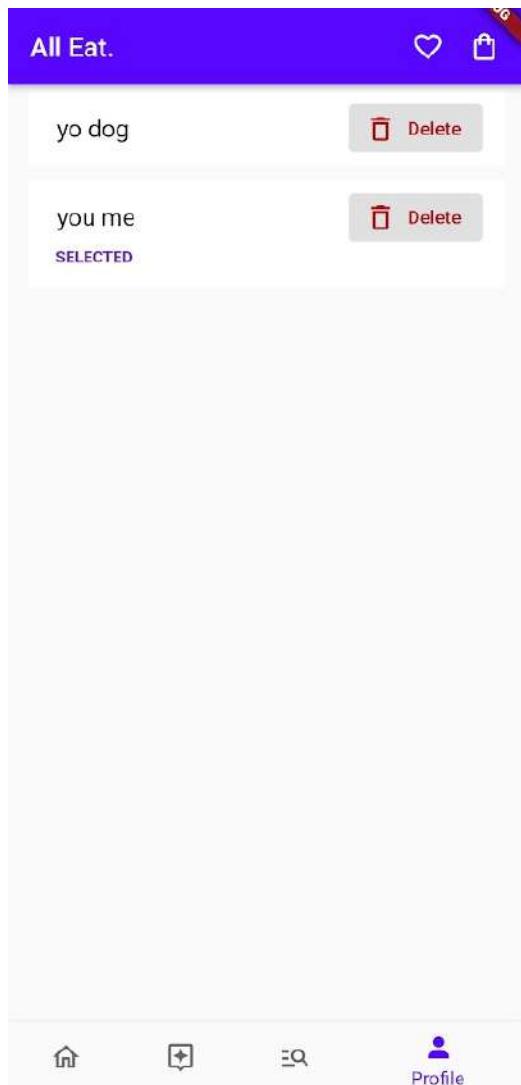
Exception has occurred. ×
UnsupportedError (Unsupported operation: read-only)

This error was a bit puzzling since I was reading not writing to the profile. After checking the profile's selected state, they were null. In order to fix this, I decided to delete the local profiles manually but once I deleted all of them, I found that it didn't bring me to the sign in/ register page so once again, I had to fix the wrapper so that it would do a check.

After fixing the wrapper, I did some more debugging and found that the issue was that there was a missed '=' sign for the if statement. It was automatically assigning the value to true during the if statement meaning it would always be true

```
itemBuilder: (context, index) {
  Map<String, dynamic> profile = profiles[index];
  var profileSelected = profile["selected"].toString();
  return Center(
    child: LayoutBuilder(builder: (context, constraints) {
      try {
        if (profileSelected == "0") {
          return Container(
```

After this was fixed, I then went to work making the profile page where when you click on the name of the profile, it will select that and you can delete the profile. Since there were some issues while making this, I decided to add a catch, so that if it can't receive the profiles, it will just say "Failed to receive profile" and if there are no profiles, "No profiles".



```
try {
  if (profileSelected == "0") {
    return Container(
      margin: const EdgeInsets.only(
        top: 5, bottom: 5, left: 15, right: 15), // EdgeInsets.only
      color: const Color(0xffffffff),
      padding: const EdgeInsets.only(left: 5),
      child: ListTile(
        title: Text(
          profile["firstname"] +
            " " +
            profile["lastname"],
          style: const TextStyle(
            color: Colors.black,
            fontWeight: FontWeight.w400,
            fontSize: 18,
          ), // TextStyle // Text
        trailing: ElevatedButton.icon(
          onPressed: null,
          icon: const Icon(
            Icons.delete_outline,
            color: Color(0xffAF0E17),
          ), // Icon
          label: const Text(
            "Delete",
            style: TextStyle(color: Color(0xffAF0E17)),
          ), // Text
        )), // ElevatedButton.icon // ListTile // Container
  }
}
```

```
    } else if (profileSelected == "1") {
      return Container(
        margin: const EdgeInsets.only(
          top: 5, bottom: 5, left: 15, right: 15), // EdgeInsets.only
        color: ■const Color(0xffffffff),
        padding: const EdgeInsets.only(left: 5),
        width: 500,
        height: 80,
        child: Column(
          children: [
            Expanded(
              child: SizedBox(
                child: ListTile(
                  title: Text(
                    profile["firstname"] +
                    " " +
                    profile["lastname"],
                    style: const TextStyle(
                      color: □Colors.black,
                      fontWeight: FontWeight.w400,
                      fontSize: 18,
                    ), // TextStyle // Text
                  trailing: ElevatedButton.icon(
                    onPressed: null,
                    icon: const Icon(
                      Icons.delete_outline,
                      color: ■Color(0xffAF0E17),
                    ), // Icon
                    label: const Text(
                      "Delete",
                      style: TextStyle(
                        color: ■Color(0xffAF0E17)), // TextStyle
                    ), // Text
                  ) // ElevatedButton.icon // ListTile // SizedBox
                )
              )
            )
          ]
        )
      )
    }
  }
}
```

```

        )), // ElevatedButton.icon // ListTile // SizedBox
    ), // Expanded
    Expanded(
        child: Container(
            alignment: Alignment.bottomLeft,
            padding: const EdgeInsets.all(5),
            child: const Padding(
                padding:
                    EdgeInsets.only(bottom: 10, left: 10),
                child: Text(
                    "SELECTED",
                    style: TextStyle(
                        color: Colors.deepPurple,
                        fontWeight: FontWeight.w800,
                        fontSize: 12,
                    ), // TextStyle
                    )), // Text // Padding
            ), // Container
        ), // Expanded
    ],
),
); // Column // Container
),
// Column // Container
} else {
    return Container(
        margin: const EdgeInsets.only(
            top: 5, bottom: 5, left: 15, right: 15), // EdgeInsets.only
        color: const Color(0xffffffff),
        child: const ListTile(title: Text("No Profiles"))); // Container
}
} catch (e) {
    return Container(
        margin: const EdgeInsets.only(
            top: 5, bottom: 5, left: 15, right: 15), // EdgeInsets.only
        color: const Color(0xffffffff),
        child: const ListTile(
            title: Text("Failed to grab profile."))); // ListTile // Container
}
);
}); // LayoutBuilder // Center

```

The next step was to allow the user to change the selected profile. This was pretty easy, since the ListTile can be assigned an onTap function which can run a Future. This future then runs a premade function which I made earlier which changes the selected to be whichever id you input.

```
padding: const EdgeInsets.all(10),  
child: ListTile(  
    onTap: () => selectProfile(profiles[index]["id"])),  
    title: Text(  
        ...  
    ),  
);
```

```
Future<void> selectProfile(index) async {  
    await SQLiteLocalDB.setSelected(index);  
}
```

Problem 14

Since an object is different when it is selected, there was no need to put any safety mechanisms to prevent selecting it twice. The only issue was that once it was selected, the user would need to change the page and reopen the profiles to update the containers.

To fix this, I used a setstate function, which rebuilds the build widget when the value within it is updated

```
Future<void> selectProfile(index) async {  
    await SQLiteLocalDB.setSelected(index);  
    List profileInfo = await SQLiteLocalDB.getDisplayProfiles();  
    setState(() {  
        profileInfo = profileInfo;  
    });  
}
```

```
child: ListTile(  
    onTap: () {  
        selectProfile(profiles[index]["id"]);  
    },  
);
```

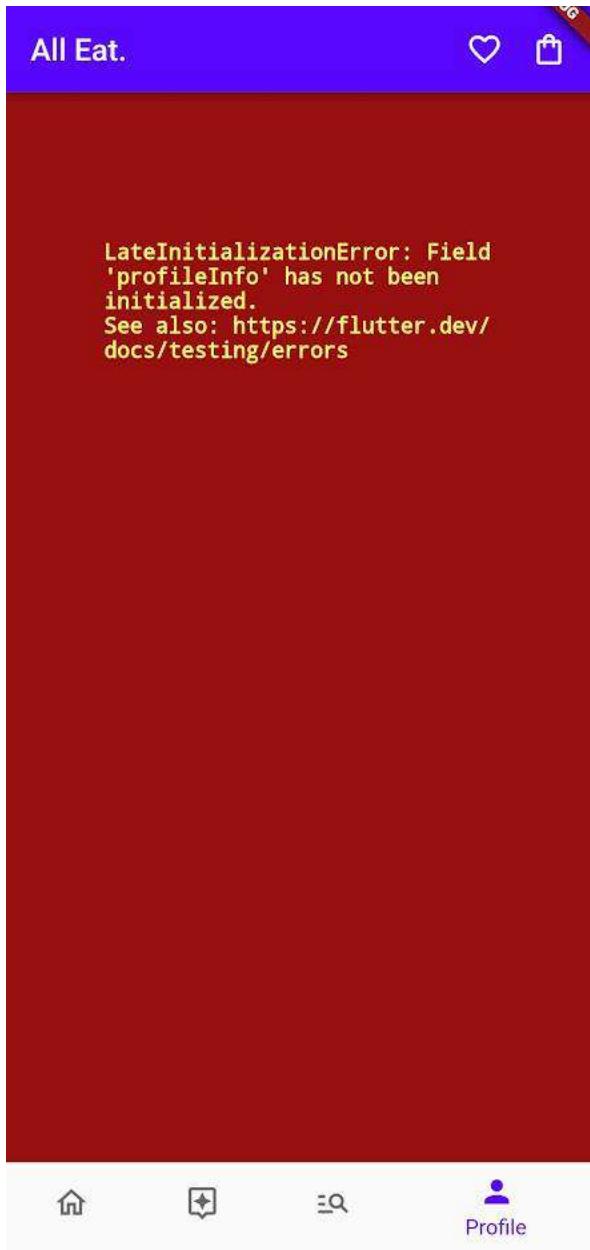
I also did the same to the delete button.

```
Future<void> deleteProfile(index) async {
    await SQLiteLocalDB.deleteProfile(index);
    List profileInfo = await SQLiteLocalDB.getDisplayProfiles();
    setState(() {
        profileInfo = profileInfo;
    });
}
```

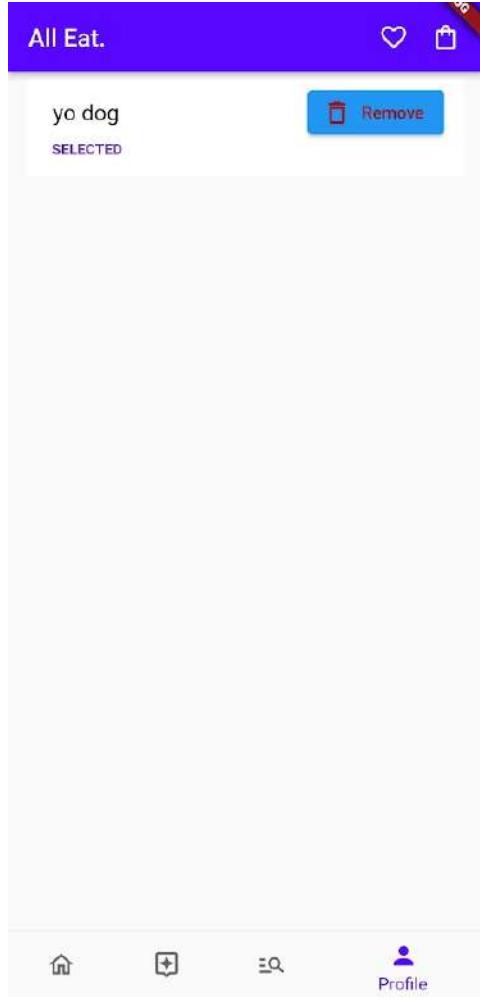
```
    onPressed: () {
        deleteProfile(profiles[index]["id"]);
    },
    icon: const Icon(
```

Problem 15

While testing the functionality, I deleted all the profiles and logged in but when I came back to the profiles page, it gave me an error



It seems that `profileInfo` was called before the profile was added. Even when the app was closed and reopened, the error persisted. This made no sense since `profileInfo` should be initiated before it is called so I did a rebuild and it started working again.



Login/Register from Profiles Page

In order to do the register and login page, I just had to copy and paste the code from the setup section and add a back button to the appbar. Underneath the profiles page is a button that allows you to add a new profile.

The screenshot shows a mobile application interface. At the top, there is a purple header bar with the text "All Eat." on the left and two icons (a heart and a shopping bag) on the right. Below the header is a section titled "Saved Profiles". This section contains a list of profiles, each with a name and a red trash can icon. The profiles listed are: Ilya Sullivan (SELECTED), CPUR Admin, Max Gething, Robbie Sucho, Sam Ripley, and Stevie Turner. At the bottom of the list is a blue button labeled "Add Profile". Below the list is a navigation bar with icons for home, search, and profile, with "Profile" being the active tab. A sidebar on the left is open, showing a hierarchical menu structure under the "add_profile" category.

- ✓ add_profile
 - ⌚ addprofile_create.dart
 - ⌚ addprofile_login.dart
 - ⌚ addprofile.dart

```

    ), // FutureBuilder // Expanded
Container(
  padding: const EdgeInsets.all(10),
  child: ElevatedButton(
    style: ElevatedButton.styleFrom(
      minimumSize: const Size.fromHeight(50),
      primary: Colors.color(0xff5806FF)),
    onPressed: () {
      Navigator.push(context,
        MaterialPageRoute(builder: (context) => const AddProfile()));
    },
    child: const Text("Add Profile"),
  ), // ElevatedButton
) // Container
)); // <Widget>[] // Column // SizedBox // scaffold

```

Browse Page

The Browse page will be a single section, only having a list of restaurants and a way to sort them by recommended, distance and most popular. You can also favourite restaurants.

To start off with, I had to remake the database made previously on phpmyadmin. Since it was made on SQLStudio, it ended up not being compatible. With phpmyadmin, it did allow me to do more data types which helped with development since I could sort a lot better and allowed for a better recommendation algorithm.



Since the first thing I needed to make was the front browse page where you can see the name of the restaurant and the images, I made only the restaurant and restaurant open times tables initially. I then used PHP on the server side to do the sorting and sending data so that the phone would not need to process all the data.

The first sorting method was with distance since it is the easiest, plugging in the user's location and the restaurant's location and using the longitude and latitude numbers and getting the shortest direct path from the user. Although this was not accurate since a street a couple of metres away could take a while to get to you, it did decrease coding time and did mean that it could be improved in the future. Before coding the php file for sorting, I first had to make a page that would allow the user to switch locations and have a location detection system.

Location Button

The location system allows you to manually add the address or to automatically set the address by using the location of the device.

All Eat.    Address Details.

63 Granville Rd 

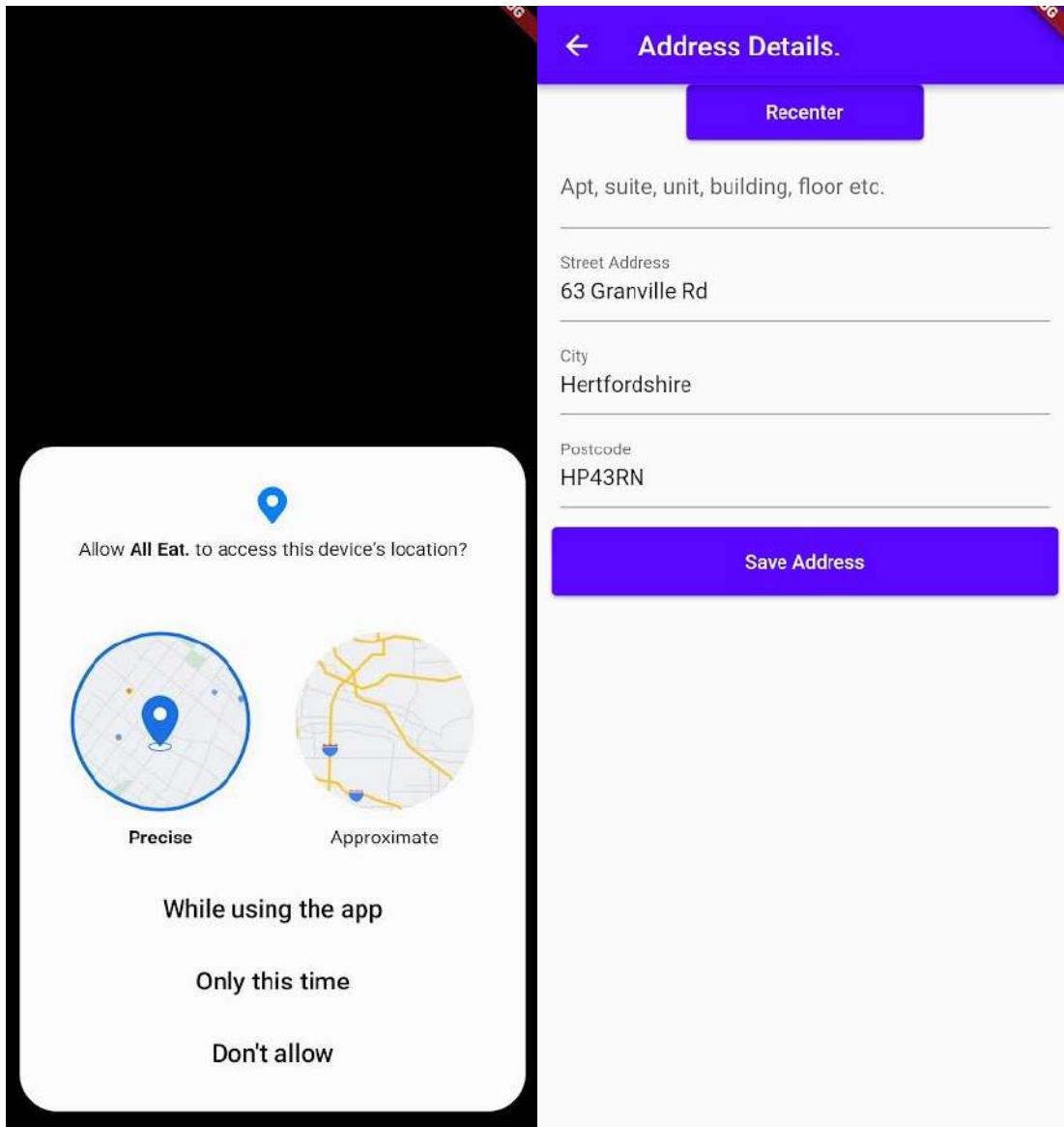
Apt, suite, unit, building, floor etc.

Street Address
63 Granville Road

City
Berko

Postcode
HP43RN



```
import 'package:alleat/screens/main_app/browse.dart';

import 'package:alleat/services/sqlite_service.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:geocoding/geocoding.dart';

import 'package:geolocator/geolocator.dart' as geo;
```

```
class CurrentLocation extends StatefulWidget {

  const CurrentLocation({Key? key}) : super(key: key);

  @override

  State<CurrentLocation> createState() => _CurrentLocationState();
}

class _CurrentLocationState extends State<CurrentLocation> {

  Future<String> getSavedStreet() async {

    try {

      // Try to see if there is an address saved

      List address = await SQLiteLocalDB.getAddress(); // Get address

      setState(() {

        // Update the widget if there is a change

        address = address;

      });

      return address[0]["savedaddressstreet"]; // Send back the street
      address
    } catch (e) {

      return "Set Location."; // Send back that no location is saved
    }
  }
}
```

```
@override

Widget build(BuildContext context) {

    return FutureBuilder<String>(
        future: getSavedStreet(), //Get the saved street address

        builder: (context, snapshot) {
            var location = snapshot.data ?? [];

            if (!snapshot.hasData) {
                //While getting the street address, add button with loading
                bar
            }

            return Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                    Container(
                        padding: const EdgeInsets.only(top: 20, bottom: 10),
                        child: Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            crossAxisAlignment: CrossAxisAlignment.center,
                            children: [
                                Container(
                                    padding: const EdgeInsets.only(
                                        top: 5, bottom: 7, left: 35, right: 20),

```

```
decoration: const BoxDecoration(
```

```
    color: Color(0xffEBE0FF),
```

```
    borderRadius:
```

```
        BorderRadius.all(Radius.circular(30))),
```

```
    child: Row(
```

```
        children: [
```

```
            const Padding(
```

```
                padding: EdgeInsets.only(right: 12),
```

```
                child: SizedBox(
```

```
                    width: 80,
```

```
                    height: 3,
```

```
                    child: LinearProgressIndicator(
```

```
                        color: Color(0xff4100C4),
```

```
                        backgroundColor:
```

```
                        Color(0xffEBE0FF),
```

```
                    ),
```

```
                ),
```

```
            ),
```

```
            IconButton(
```

```
                onPressed: () {
```

```
                    Navigator.push(
```

```
                        context,
```

```
                        MaterialPageRoute(
```

```
builder: (context) =>

    const
FromSavedManualLocation()))); //Go to the manual location page

    } ,

    icon: const Icon(
        Icons.location_on,
        color: Color(0xff4100C4),
    ) )

] ,
) )

] ,
) )

] ;
} else {

    // Once the street address is received, output the street
address instead of the loading bar

    return Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
            Container(
                padding: const EdgeInsets.only(top: 20, bottom: 10),
                child: Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
```

```
children: [  
  InkWell(  
    onTap: () {  
      Navigator.push(  
        context,  
        MaterialPageRoute(  
          builder: (context) =>  
            const  
FromSavedManualLocation()));  
  
    },  
    child: Container(  
      padding: const EdgeInsets.only(  
        top: 5, bottom: 7, left: 35, right:  
20),  
      decoration: const BoxDecoration(  
        color: Color(0xffEBE0FF),  
        borderRadius: BorderRadius.all(  
          Radius.circular(30))),  
      child: Row(  
        children: [  
          Text(  
            location.toString(),  
            style: const TextStyle(  
              fontSize: 14,
```

```
        color: Color(0xff4100C4),  
  
        fontWeight: FontWeight.w600),  
  
) ,  
  
IconButton(  
  
onPressed: () {  
  
    Navigator.push(  
  
        context,  
  
        MaterialPageRoute(  
  
            builder: (context) =>  
  
            const  
FromSavedManualLocation())); //Go to the manual location page  
  
    } ,  
  
    icon: const Icon(  
  
Icons.location_on,  
  
color: Color(0xff4100C4),  
  
) )  
  
] ,  
  
) )  
  
] ,  
  
) )  
  
] );  
  
}  
  
} );
```

```
        }

    }

}

class FromSavedManualLocation extends StatefulWidget {

    const FromSavedManualLocation({Key? key}) : super(key: key);

    @override

    State<FromSavedManualLocation> createState() =>

        _FromSavedManualLocationState();
}

class _FromSavedManualLocationState extends State<FromSavedManualLocation> {

    final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

    final RegExp verifyPostcode = RegExp(
        r"^(Gg|Ii|Rr) ([Aa]{2}) | (([A-Za-z][0-9]{1,2}) | ([A-Za-z][A-Ha-hJ-J-Yj-y][0-9]{1,2}) | ([A-Za-z][0-9][A-Za-z]) | ([A-Za-z][A-Ha-hJ-J-Yj-y][0-9]?[A-Za-z])) [0-9] [A-Za-z]{2})$"; //Postcode formatting

    Future<List> getSavedAddress() async {

        return await SQLiteLocalDB.getAddress(); // get the saved address
    }
}
```

```
Future<void> saveFilledAddress(String extraAddress, String streetAddress, String cityAddress, String postcodeAddress) async {
try {
SQLiteLocalDB.setAddress(extraAddress, streetAddress, cityAddress, postcodeAddress); // Try to save address
ScaffoldMessenger.of(context).showSnackBar(
const SnackBar(content: Text('Successfully saved address.')));
} catch (e) {
ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
content: Text(
'Failed to save address'))); // If failed to saved, output failed to save
}
}

@Override
Widget build(BuildContext context) {
return FutureBuilder<List>(
future: getSavedAddress(), //Get the saved address to fill the fields
builder: (context, snapshot) {
if (snapshot.connectionState != ConnectionState.done) {
return const Center(

```

```

        child:

            CircularProgressIndicator()); //While loading, display
loading page

    }

    if (snapshot.hasError) {

        return const Center(
            child: Text("Failed")); //If there is a failure, output
Failed

    }

    List savedAddress = snapshot.data ?? []; //Get address in list
form

    if (savedAddress.isEmpty) {

        //If it is an empty list (nothing is saved for profile) output
empty fields in the right format

        savedAddress = [
            {
                "savedaddressextra": "",
                "savedaddressstreet": "",
                "savedaddresscity": "",
                "savedaddresspostcode": ""
            }
        ];
    }

    final TextEditingController extraAddress =
TextEditingController(

```

```
        text: savedAddress[0]

                ["savedaddressextra"]); //For each field, fill with
        saved data

        final TextEditingController streetAddress =
TextEditingController(
        text: savedAddress[0]["savedaddressstreet"]);

        final TextEditingController cityAddress =
TextEditingController(text:
        savedAddress[0]["savedaddresscity"]);

        final TextEditingController postcodeAddress =
TextEditingController(
        text: savedAddress[0]["savedaddresspostcode"]);

        return Scaffold(
            resizeToAvoidBottomInset: false,
            appBar: AppBar(
                title: const Text(
                    'Address Details.'), //Create page for manual fill
                leading: IconButton(
                    icon: const Icon(Icons.arrow_back),
                    onPressed: () =>
                        Navigator.of(context).pop()), //Go back button
                backgroundColor: Theme.of(context).primaryColor,
            ),
            body: Form(
```

```
key: _formKey,  
  
child: SingleChildScrollView(  
  
    child: Column(  
  
        mainAxisAlignment: MainAxisAlignment.start,  
  
        children: [  
  
            Align(  
  
                alignment: Alignment.center,  
  
                child: SizedBox(  
  
                    width: 170,  
  
                    height: 40,  
  
                    child: ElevatedButton(  
  
                        // Button for getting current location  
  
                        style: ElevatedButton.styleFrom(  
  
                            minimumSize: const  
Size.fromHeight(50),  
  
                            primary: const Color(0xff5806FF)),  
  
                            onPressed: () {  
  
                                Navigator.pop(context);  
  
                                Navigator.push(  
  
                                    context,  
  
                                    MaterialPageRoute(  
  
                                        builder: (context) =>  
  
                                            const  
FromCurrentManualLocation()));
```

```
        } ,  
  
        child: const Text("Use current Location") ,  
  
    ) ,  
  
) ) ,  
  
ListTile(  
  
    title: TextFormField(  
  
        keyboardType: TextInputType.name,  
  
        controller: extraAddress,  
  
        inputFormatters: [  
  
FilteringTextInputFormatter.allow(RegExp(  
  
    "[a-zA-Z0-  
9|\\".|\\\'|\\#\\|@\\|%\\|&\\|/|\\ ]") // Must only contain a-z, A-Z, 0-9,  
. , ' , # , @ , % , & , / , (space)  
  
] ,  
  
decoration: (const InputDecoration(  
  
    labelText:  
  
    "Apt, suite, unit, building, floor  
etc."))),  
  
ListTile(  
  
    title: TextFormField(  
  
        keyboardType: TextInputType.name,  
  
        controller: streetAddress,  
  
        validator: (streetAddress) {
```

```

        if (streetAddress == null ||

            streetAddress.isEmpty) {

                // Must be filled

                return "Required";

            }

        } ,

        inputFormatters: [

            FilteringTextInputFormatter.allow(RegExp(
                "[a-zA-Z0-9|\\.|\\'|\\#|@|%|&|/|\\| ]") // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)

        ] ,

        decoration: (const InputDecoration(
            labelText: "Street Address")),

        ) ,

        ListTile(
            title: TextFormField(
                keyboardType: TextInputType.name,
                validator: (cityAddress) {

                    if (cityAddress == null ||

cityAddress.isEmpty) {

                        //Must be filled

                        return "Required";

                    }

                }

        } ,

```

```
controller: cityAddress,  
  
decoration:  
  
    (const InputDecoration(labelText: "City")) ,  
  
    inputFormatters: [  
  
        FilteringTextInputFormatter.allow(RegExp(  
  
            "[a-zA-Z0-9|\\.|\\'|\\#\\||@\\||%\\||&\\||/|\\|\\|") // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)  
        ] ,  
  
    ) ,  
  
    ListTile(  
  
        title: TextFormField(  
  
            controller: postcodeAddress,  
  
            keyboardType: TextInputType.name,  
  
            onChanged: (value) {  
  
                postcodeAddress.value = TextEditingValue(  
  
                    //Force capital letters  
                    text: value.toUpperCase() ,  
  
                    selection: postcodeAddress.selection);  
  
            } ,  
  
            inputFormatters: [  
  
                FilteringTextInputFormatter.allow(RegExp(  
  
                    '[a-zA-Z0-9]')) // Must only contain a-z, A-Z and 0-9  
            ] ,
```

```

decoration:

  (const InputDecoration(labelText:
"Postcode")),

  validator: (postcodeAddress) {

    if (postcodeAddress == null ||
        postcodeAddress.isEmpty ||
        postcodeAddress.length != 6) {

      // Must be filled and be exactly 6
      characters

      return "Enter valid postcode";
    }
  },
),
Container(
  padding: const EdgeInsets.all(10),
  child: ElevatedButton(
    style: ElevatedButton.styleFrom(
      minimumSize: const Size.fromHeight(50),
      primary: const Color(0xff5806FF)),
    onPressed: () {
      if (_formKey.currentState!.validate()) {
        //If passed validation continue
        saveFilledAddress(
          extraAddress.text,

```

```

        streetAddress.text,
        cityAddress.text,
        postcodeAddress.text);
    Navigator.pop(context); // Go back to main
page

} else {

ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
        content: Text('Invalid
Address')));
}

},
child: const Text("Save Address"),
),
)
]
)))));

} );
}

}

class FromCurrentManualLocation extends StatefulWidget {
const FromCurrentManualLocation({Key? key}) : super(key: key);
}

```

```

@Override

State<FromCurrentManualLocation> createState() =>

    _FromCurrentManualLocationState();
}

class _FromCurrentManualLocationState extends
State<FromCurrentManualLocation> {

    final _formKey = GlobalKey<FormState>();

    final verifyPostcode = RegExp(
        r"^[Gg][Ii][Rr] [Aa]{2})|(([A-Za-z][0-9]{1,2})|([A-Za-z][A-Ha-
hJ-Yj-y][0-9]{1,2})|(([A-Za-z][0-9][A-Za-z])|([A-Za-z][A-Ha-hJ-Yj-y][0-
9]?[A-Za-z])))[0-9][A-Za-z]{2})\$"; // Postcode format

    Future<void> saveFilledAddress(String extraAddress, String
streetAddress,

        String cityAddress, String postcodeAddress) async {

    try {
        SQLiteLocalDB.setAddress(extraAddress, streetAddress, cityAddress,
            postcodeAddress); // Try to save

        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Successfully saved address.')));
    } catch (e) {
        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Failed to save address')));
    }
}

```

```
        }

    }

Future<List> determinePosition() async {

    bool serviceEnabled;

    geo.LocationPermission permission;

    // Test if location services are enabled.

    serviceEnabled = await geo.Geolocator.isLocationServiceEnabled();

    if (!serviceEnabled) {

        return Future.error('Location services are disabled.');
    }

    // Check if the location is denied

    permission = await geo.Geolocator.checkPermission();

    if (permission == geo.LocationPermission.denied) {

        permission = await geo.Geolocator.requestPermission();

        if (permission == geo.LocationPermission.denied) {

            return Future.error('Location permissions are denied');
        }
    }

    if (permission == geo.LocationPermission.deniedForever) {
```

```
// Permissions are denied forever, handle appropriately.

return Future.error('Location permissions are permanently denied.');
}

// Access the current position of the device

geo.Position locationDevice = await
geo.Geolocator.getCurrentPosition();

List locationDeviceLatLong = [
    locationDevice.latitude,
    locationDevice.longitude
];

List<Placemark> placemarks = await placemarkFromCoordinates(
    locationDeviceLatLong[0],
    locationDeviceLatLong[1]); //Get the lat and long

Placemark place = placemarks[0];

List address = [
    //Convert to address
    {
        "savedaddressextra": "",
        "savedaddressstreet": place.street,
        "savedaddresscity": place.subAdministrativeArea,
        "savedaddresspostcode": place.postalCode?.replaceAll(' ',
```

```
        '') //Remove the space between the each 3 characters to fit
with the constraint

    }

];

return address;

}

Future<List> getCurrentAddress() async {

try {

List location = await determinePosition(); //Get current location

return location;

} catch (e) {

//If failed to get location, replace with nothing

return [
{
"savedaddressextra": "",

"savedaddressstreet": "",

"savedaddresscity": "",

"savedaddresspostcode": ""

}
];

}
}
```

```
@override

Widget build(BuildContext context) {

  return FutureBuilder<List>(

    future: getCurrentAddress(),

    builder: (context, snapshot) {

      if (snapshot.connectionState != ConnectionState.done) {

        //While waiting to get location, show loading circle

        return const Center(child: CircularProgressIndicator());

      }

      if (snapshot.hasError) {

        //On future failure, show failed

        return const Center(child: Text("Failed"));

      }

      List savedAddress =

        snapshot.data ?? [] ; //Get the location from future

      if (savedAddress.isEmpty) {

        //If it is empty, replace with blanks

        savedAddress = [

          {

            "savedaddressextra": "",

            "savedaddressstreet": "",

            "savedaddresscity": "",
```

```
        "savedaddresspostcode": ""

    }

];

}

final TextEditingController extraAddress =
TextEditingController(
    text: savedAddress[0][
        "savedaddressextra"]); //For each field, fill with the
current address

final TextEditingController streetAddress =
TextEditingController(
    text: savedAddress[0]["savedaddressstreet"]);

final TextEditingController cityAddress =
TextEditingController(text:
savedAddress[0]["savedaddresscity"]);

final TextEditingController postcodeAddress =
TextEditingController(
    text: savedAddress[0]["savedaddresspostcode"]);

return Scaffold(
    resizeToAvoidBottomInset: false,
    appBar: AppBar(
        title: const Text('Address Details.),
        leading: IconButton(
            icon: const Icon(Icons.arrow_back),
            onPressed: () =>
```

```
        Navigator.of(context).pop(), //Go back button

        backgroundColor: Theme.of(context).primaryColor,
    ) ,
body: Form(
    key: _formKey,
    child: SingleChildScrollView(
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Align(
                    alignment: Alignment.center,
                    child: SizedBox(
                        width: 170,
                        height: 40,
                        child: ElevatedButton(
                            style: ElevatedButton.styleFrom(
                                minimumSize: const
Size.fromHeight(50),
                                primary: const Color(0xff5806FF)),
                            onPressed: () {
                                //Recenter button to get current
                                location again
                                Navigator.pop(

```

```
        context); //Reopen current location

class

    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) =>
            const
FromCurrentManualLocation())));
}

),
child: const Text("Recenter"),
),
),
ListTile(
    title: TextFormField(
        keyboardType: TextInputType.name,
        controller: extraAddress,
        inputFormatters: [
FilteringTextInputFormatter.allow(RegExp(
    "[a-zA-Z0-9|\\.|\\'|\\#|@|%|&|/| ]") // Must only contain a-z, A-Z, 0-9,
., ', #, @, %, &, /, (space)
],
decoration: (const InputDecoration(
    labelText:
```

```
        "Apt, suite, unit, building, floor
etc."))),

ListTile(
    title: TextFormField(
        keyboardType: TextInputType.name,
        controller: streetAddress,
        validator: (streetAddress) {
            if (streetAddress == null ||
                streetAddress.isEmpty) {
                return "Required";
            }
        },
        inputFormatters: [
            FilteringTextInputFormatter.allow(RegExp(
                "[a-zA-Z0-9|\\.|\\'|\\#|@|%|&|/|\\| ]")) // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)
        ],
        decoration: (const InputDecoration(
            labelText: "Street Address")),
    ),
    ListTile(
        title: TextFormField(
            keyboardType: TextInputType.name,
            validator: (cityAddress) {
```

```

        if (cityAddress == null ||

cityAddress.isEmpty) {

            return "Required";

        }

    } ,

    controller: cityAddress,

    decoration:

        (const InputDecoration(labelText: "City")),

    inputFormatters: [

        FilteringTextInputFormatter.allow(RegExp(
            "[a-zA-Z0-9|\\.|\\'|\\#|@|%|&|/|\\| ]") // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)

    ] ,

) ) ,

ListTile(

    title: TextFormField(

        controller: postcodeAddress,

        keyboardType: TextInputType.name,

        onChanged: (value) {

            postcodeAddress.value = TextEditingValue(
                text: value.toUpperCase(),
                selection: postcodeAddress.selection);

        }

    ) ,

    inputFormatters: [

```

```
FilteringTextInputFormatter.allow(RegExp(
    '[a-zA-Z0-9]')) // Must only contain a-z,
A-Z, 0-9
],
decoration:
(const InputDecoration(labelText:
"Postcode")),

validator: (postcodeAddress) {
    if (postcodeAddress == null ||
        postcodeAddress.isEmpty ||
        postcodeAddress.length != 6) {
        return "Enter valid postcode";
    }
},
),
Container(
padding: const EdgeInsets.all(10),
child: ElevatedButton(
style: ElevatedButton.styleFrom(
minimumSize: const Size.fromHeight(50),
primary: const Color(0xff5806FF)),
onPressed: () {
if (_formKey.currentState!.validate()) {

```

```

        // If all validation correct, send to be
        saved and close the page

        saveFilledAddress (

            extraAddress.text,
            streetAddress.text,
            cityAddress.text,
            postcodeAddress.text);

        Navigator.pop(context);

    } else {

ScaffoldMessenger.of(context).showSnackBar (
    const SnackBar(
        content: Text('Invalid
Address')));
}

},
child: const Text("Save Address"),
),
)
]
))));
```

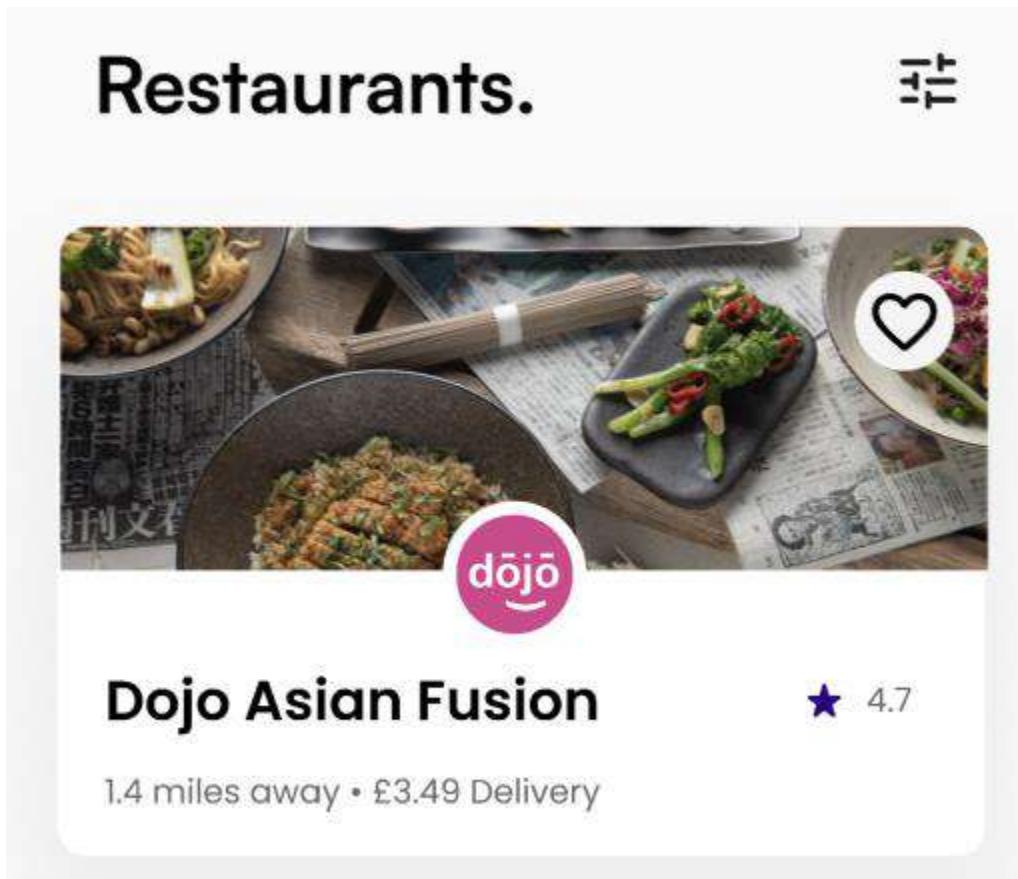
) ;

}
}

Restaurant Summaries

To do the restaurant summaries, I need to grab the data from the server and have it display it in a readable form with a place where the user can add it to favourites and get to the main restaurant page.

For the final prototype, I want it to look like concept 4



To simplify things down, I will first try to get both the banner and restaurant logo and have it display if it is open so that in the future, I can make it not appear if it is closed. The following is how I would like it to show for prototype 1A.



Dojo Asian Fusion

OPEN - Closes 21:00



Instead of using a filter at the beginning, I will just import them by id ascending. This simplifies debugging since, I will just receive them in an array.

While trying to get the open times, I found that with my current knowledge, I was not able to get all the open times within a 2D array since each restaurant has a week of open times. Instead, it will be grabbed when a user looks at a restaurant.

To get the restaurants, I first developed the restaurant php file which gets the sorting method and sends the data associated with restaurants. What I plan to do in the future is have it load restaurants in groups of 20 and as a user scrolls, it gets another 20.

```
<?php  
  
$db = "alleat"; //database name  
  
$dbuser = "root"; //database username  
  
$dbpassword = "XDWQw7CCKRQ4ZjvMzVyN"; //database password  
  
$dbhost = "localhost"; //database host
```

```

$return["error"] = false;

$return["message"] = "";

$return["restaurants"] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["sort"]);

if($val) {

    $sort = $_POST["sort"];

    if ($sort == "id"){

        $sql = "SELECT res.restaurantid, res.restaurantname,
res.restaurantlogo, res.restaurantbanner FROM restaurant res LIMIT 20";

        if($result = mysqli_query($link, $sql)) {

            while($row = mysqli_fetch_assoc($result)) {

                array_push($return["restaurants"],
[$row["restaurantid"], $row["restaurantname"], $row["restaurantlogo"],
$row["restaurantbanner"]]);

            }

            mysqli_free_result($result);
        }
    }
}

```

```

else{

    $return["error"] = true;

    $return["message"] = "sort method not specified";

}

}

else{

    $return["error"] = true;

    $return["message"] = "data not specified";

}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>

```

By using a single file instead of multiple for the sorting, it reduces the amount of files and allows for a reduced number of lines of code, sending the same data in different Futures.

The next thing I did was create the restaurant list file for the app, this will be a widget that can be added to other parts of the app since there are many places where a restaurant list can be added.

I started by getting the list of restaurant names and just displaying them using text widgets and a future which used the same code from the profiles page.

Problem 16

After some issues where the names would not display, I found that it was importing the data differently, with php processing an array as {} whereas dart used [] so I had to convert the data to be selecting item 0 for all elements and do the same for the rest.

```
shrinkWrap: true,  
itemCount: restaurantsdata[0]["restaurants"].length,  
itemBuilder: (context, index) {  
    return Container(...);  
}
```

On top of this, the data being returned contained error message data so to get the first restaurant, I would need to do restaurantdata[0]["restaurants"][0]

The next thing I did was make the data look better and not just in plain text, putting it in a container with padding to ensure that there is a difference between each item

While importing the restaurants, I also included the urls to the saved banner and restaurant logos.

Problem 17

To import them, I used the network image widget which allowed for the direct import of images without needing them to be saved on the device but when I used it, nothing happened.

Debugging this, I found that the URL I was specifying was to the share page not the main image page



To fix this, I used inspect element to find the source so that instead of getting

<https://drive.google.com/file/d/1TR-NTkLJdnHEkNnjaDRZBPkk4kyV729W/view>

I had

<https://lh3.googleusercontent.com/drive-viewer/AJc5JmRUuaExolqkLiUOQSmUsWybB07eAShLMHtX3SP1wBfP8ffdXnsjINvRemcXGrEU961vjJfXSWk=w1920-h932>

This fixed the issue but when I loaded the page, it would overflow off the edges so I used a container with the box decoration widget of the image to fit it.

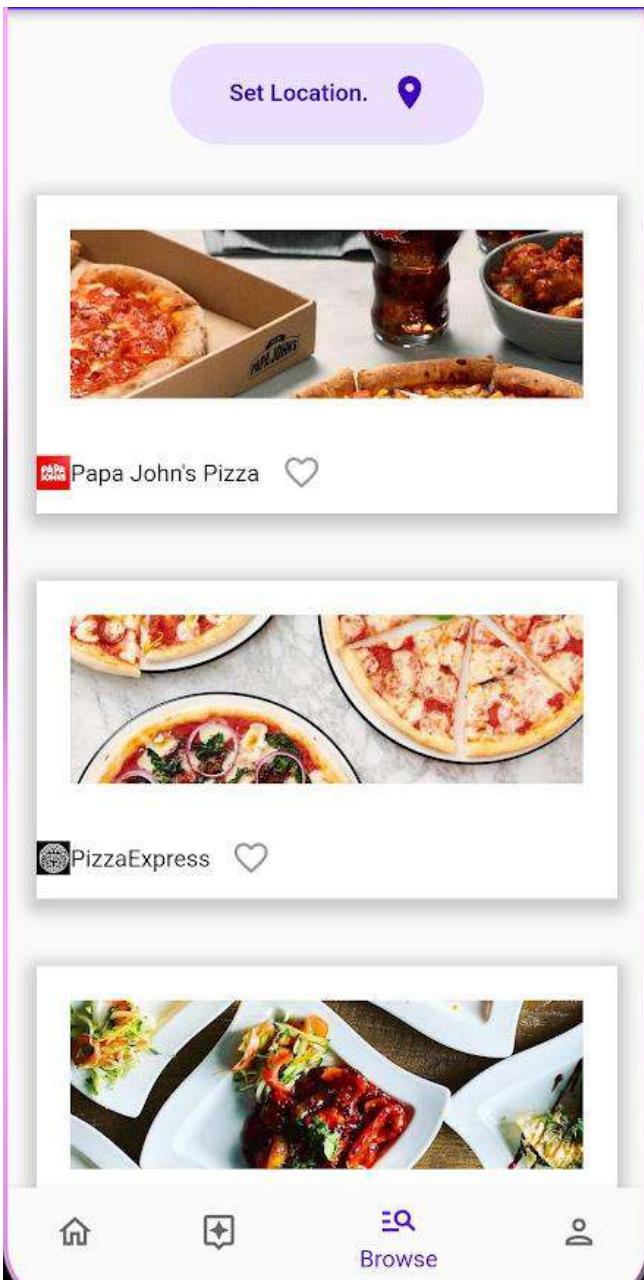
```
child: Column(children: [
  Padding(
    padding: const EdgeInsets.all(10),
    child: Container(
      width: MediaQuery.of(context).size.width,
      height: 150,
      decoration: BoxDecoration(
        borderRadius:
          const BorderRadius.all(
            Radius.circular(10)), // Border
        image: DecorationImage(
          fit: BoxFit.fitWidth,
          image: NetworkImage(
            restaurants[index][3]
              .toString()), // NetworkImage
        ), // DecorationImage
      ), // BoxDecoration
    )), // Container // Padding
```

Problem 18

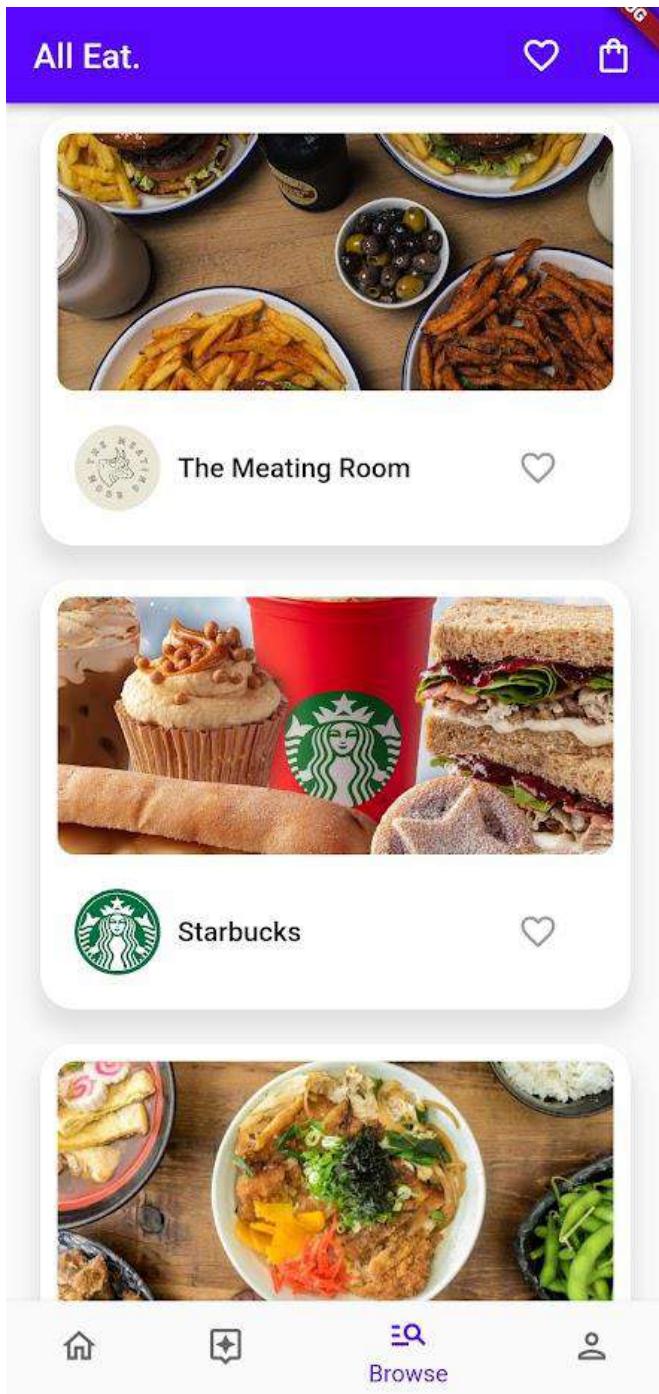
The next issue I faced was scrolling and although scrolling was enabled, I could only scroll on the part that had the location button on it. To fix this, I disabled the default scrolling on the ListView which meant that the whole page scrolled together.

```
list< RestaurantFavourites> snapshotData : [ ],  
return ListView.builder(  
  physics: const NeverScrollableScrollPhysics(),  
  scrollDirection: Axis.vertical,  
  shrinkWrap: true,  
  itemCount: restaurantsData[0][ "restaurants" ].length,
```

After all of this, it got me up to the part where it was visually looking similar to how I wanted it to look but the favourites section was not working



After some further modifications, it got even closer to the final product

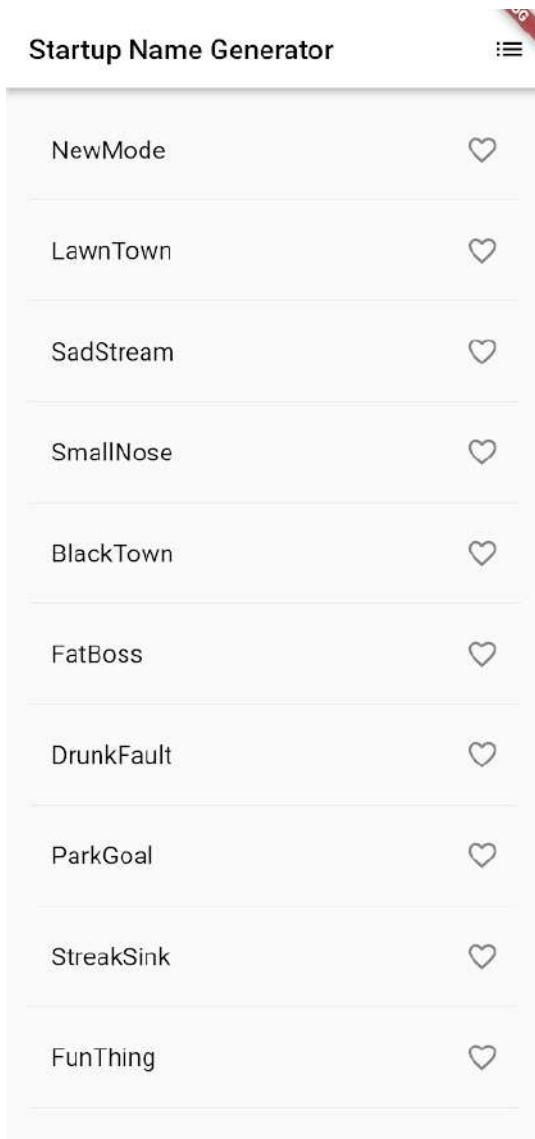


Favourites

This section was interesting since I made a similar function while learning how to use Flutter.

With Flutter, they have a cookbook where it gives you the code and explains each part step by step, allowing you to understand how it works.

<https://codelabs.developers.google.com/codelabs/first-flutter-app-pt2#0>



Although this contained the code for favoriting items, it was saved locally and used built-in functions for a List tile which had on select and on deselect which meant that I could only use the basic fundamentals of it in order to make mine.

The favourite button works by using a Future builder. The Future gets a list of restaurants favoured by the profile from the server. For each restaurant, it does a check if it is in the list and if so, it shows a filled heart and if not, an outline. There are then two actions which can

occur if the user clicks on the favourite button. If the restaurant is in the list, then it sends the restaurant id and the string unfavourite to the Future favouriteRestaurant, else it sends the restaurant id and the string favourite to the Future. The future then passes on the email, restaurant id and the action to the server for it to either insert a new value or remove it.

```
return FutureBuilder<List>(
    future: getFavourites(),
    builder: ((context, snapshot) {
        if (!snapshot.hasData) {
            return const LinearProgressIndicator(
                color: Color(0xff4100C4),
                backgroundColor: Color(0xffEBE0FF)); // LinearProgressIndicator
        } else {
            List restaurantFavourites = snapshot.data ?? [];
            return ListView.builder(
                physics: const NeverScrollableScrollPhysics(),
                scrollDirection: Axis.vertical,
                shrinkWrap: true,
                itemCount: restaurantsdata[0]["restaurants"].length,
                itemBuilder: (context, index) {
```

```
    ), // Row
    IconButton(
      onPressed: () {
        if (restaurantFavourites[0]
            ["restaurantids"]
            .contains(
                restaurants[index][0]
                .toString())))
        {
          favouriteRestaurant(
              restaurants[index][0]
              .toString(),
              "unfavourite");
          setState(() {
            getFavourites();
          });
        } else {
          favouriteRestaurant(
              restaurants[index][0]
              .toString(),
              "favourite");
          setState(() {
            getFavourites();
          });
        }
      },
    ),
  ],
)
```

```

Future<String> favouriteRestaurant(restaurantID, action) async {
  String phpurl = "http://192.168.0.22/favouriterestaurant.php";
  var profile = await SQLiteLocalDB.getProfileSelected();
  String email = profile[0]["email"].toString();

  try {
    var res = await http.post(Uri.parse(phpurl), body: {
      "action": action.toString(),
      "profileemail": email.toString(),
      "restaurantid": restaurantID,
    });
    if (res.statusCode == 200) {
      var data = convert.json.decode(res.body); //Decode to array
      if (data["error"]) {
        print("had error");

        return "failed";
      } else {
        getFavourites();
        return "success";
      }
    } else {
      return "failed";
    }
  } catch (e) {
    return "failed";
  }
}

```

```

<?php

$db = "alleat"; //database name

$dbuser = "root"; //database username

$dbpassword = "XDWQw7CCKRQ4ZjvMzVyN"; //database password

$dbhost = "localhost"; //database host

```

```
$return["error"] = false;

$return["message"] = "";

$return["success"] = false;

$return["temp"] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["action"]) && ($_POST["profileemail"]) &&
isset($_POST["restaurantid"]);

if($val){

$email = $_POST["profileemail"]; //grabing the data from headers

$restaurantid = $_POST["restaurantid"];

$action = $_POST["action"];

$sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT
1";

if($result = mysqli_query($link, $sql)) {

    if(mysqli_num_rows($result) == 1) {

        while($row = mysqli_fetch_array($result)) {

            $ProfileID = $row['ProfileID'];

        }

        // Free result set

        mysqli_free_result($result);
    }
}
}
```

```

    if ($action == "unfavourite") {

        $return["temp"] = $ProfileID;

        $sqlunfavourite = "DELETE FROM favrestaurant WHERE profileid
= $ProfileID AND restaurantid = $restaurantid";

        if ($link->query($sqlunfavourite) === TRUE) {

            $return["success"] = true;

        } else{

            $return["error"] = true;

        }

    }

    else if ($action == "favourite") {

        $return["temp"] = "favourite";

        $sqlfavourite = "INSERT INTO favrestaurant (profileid,
restaurantid) VALUES ($ProfileID, $restaurantid)";

        if ($link->query($sqlfavourite) === TRUE) {

            $return["success"] = true;

        } else{

            $return["error"] = true;

        }

    }

    else{

```

```

        $return["error"] = true;

        $return["message"] = "action unspecified";

    }

}

else{

    $return["error"] = true;

    $return["message"] = "duplicate found";

}

}

else{

    $return["error"] = true;

    $return["message"] = "failed";

}

}

else{$return["error"] = true;

    $return["message"] = "data not specified";}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

```

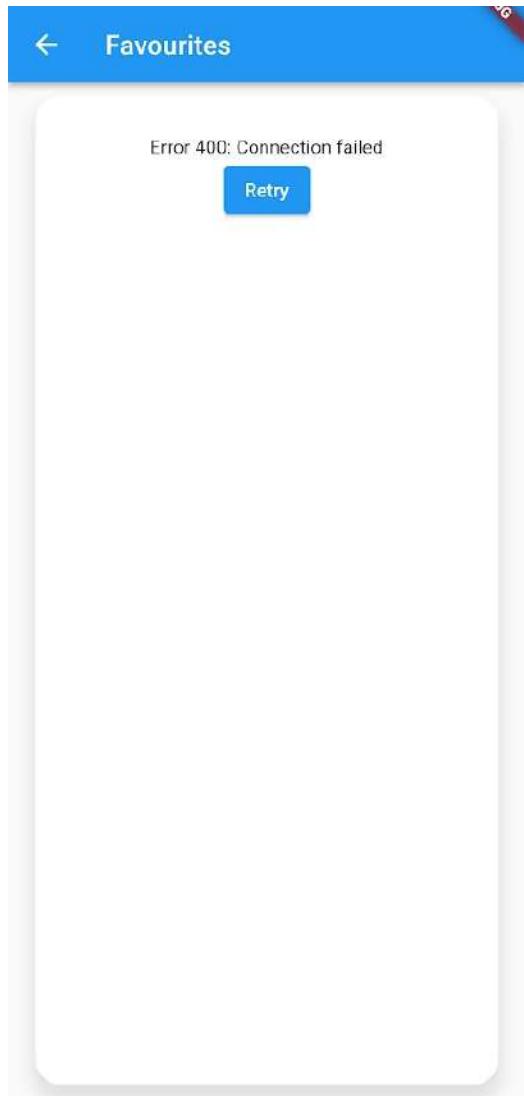
```
//converting array to JSON string  
?>
```

The next thing I did was make the favourites page. This used very similar code to the browse page but only got data favourite restaurants instead of all restaurants initially.

Problem 19

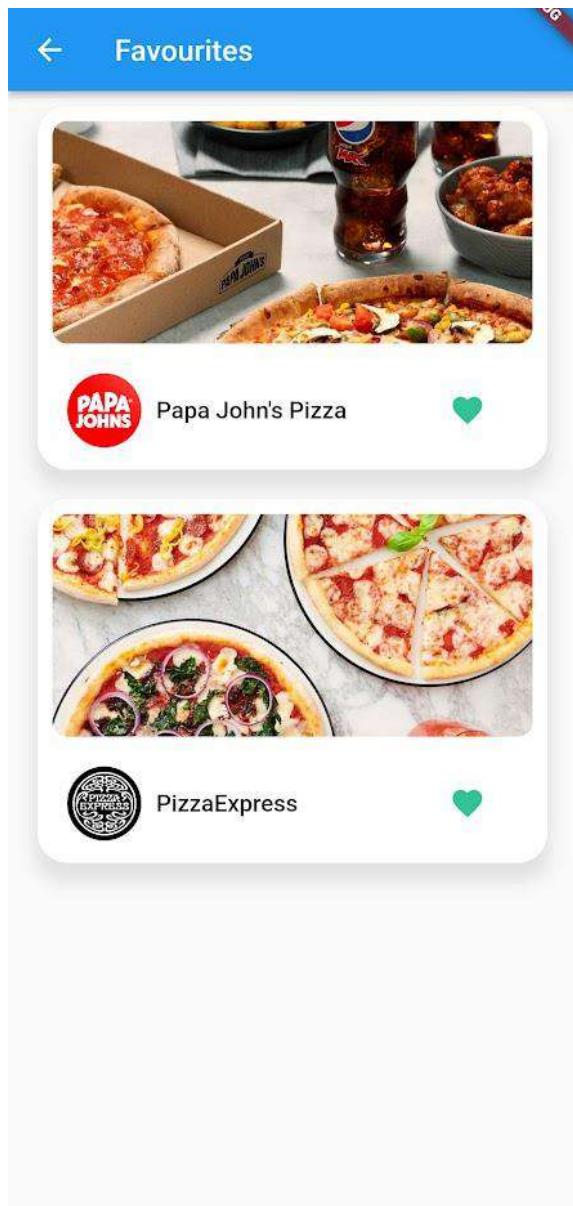
While developing the favourites page, I made a few mistakes with the biggest but most comedic being that while trying to get the list of favoured restaurants information, it would not be able to assign values to it causing the error page to show. This was found in WireShark

Time	Source IP	Destination IP	Protocol	Port	Description
2750... 1525.427092	192.168.0.95	192.168.0.22	TCP	66 58752 → 80 [ACK] Seq:	
2750... 1525.429310	192.168.0.95	192.168.0.22	TCP	66 80 → 38752 [ACK] Seq:	
2750... 1525.429348	192.168.0.22	192.168.0.95	TCP	66 80 → 38752 [ACK] Seq:	
2750... 1525.429414	192.168.0.22	192.168.0.95	TCP	66 80 → 38752 [ACK] Seq:	
2750... 1525.434549	192.168.0.95	192.168.0.22	TCP	66 38752 → 80 [ACK] Seq:	
2750... 1525.445875	192.168.0.22	192.168.0.95	TCP	9935 80 → 38754 [PSH, ACK]	
2750... 1525.446319	192.168.0.22	192.168.0.95	HTTP	323 HTTP/1.1 200 OK (text)	
2750... 1525.453525	192.168.0.95	192.168.0.22	TCP	66 38754 → 80 [ACK] Seq:	
2750... 1525.453525	192.168.0.95	192.168.0.22	TCP	66 38754 → 80 [ACK] Seq:	
2750... 1525.453525	192.168.0.95	192.168.0.22	TCP	66 38754 → 80 [FIN, ACK]	
2750... 1525.453598	192.168.0.22	192.168.0.95	TCP	66 80 → 38754 [ACK] Seq:	
<hr/>					
> Hypertext Transfer Protocol					
▼ Line-based text data: text/html (67 lines)					
 \n<table class='xdebug-error xe-notice' dir='ltr' border='1' cellspacing='0' cellpadding='0'>\n[truncated]<tr><th align='left' bgcolor='#f57900' colspan='5'>Call Stack</th></tr>\n<tr><th align='center' bgcolor='#eeeeec' colspan='5'>Time</th></tr>\n<tr><td bgcolor='#eeeeec' align='center'>1</td><td bgcolor='#eeeeec' align='center'>0.000000</td><td></td><td></td><td></td></tr>\n</table>\n \n<table class='xdebug-error xe-notice' dir='ltr' border='1' cellspacing='0' cellpadding='0'>\n[truncated]<tr><th align='left' bgcolor='#f57900' colspan='5'>Notice: Trying to access array offset on value of type null in C:\xampp\php\PEAR\Net\SSH2.php on line 100</th></tr>\n<tr><th align='center' bgcolor='#eeeeec' colspan='5'>Time</th></tr>\n<tr><td bgcolor='#eeeeec' align='center'>1</td><td bgcolor='#eeeeec' align='center'>0.000000</td><td></td><td></td><td></td></tr>\n</table>					
0070	3e 0a 3c 74 72 3e 3c 74 68 20 61 6c 69 67 6e 3d	><tr><th align='left' bgcolor='#f57900' colspan='5'>Notice: Trying to access array offset on value of type null in C:\xampp\php\PEAR\Net\SSH2.php on line 100</th></tr>\n<tr><th align='center' bgcolor='#eeeeec' colspan='5'>Time</th></tr>\n<tr><td bgcolor='#eeeeec' align='center'>1</td><td bgcolor='#eeeeec' align='center'>0.000000</td><td></td><td></td><td></td></tr>\n</table>			
0080	27 6c 65 66 74 27 20 62 67 63 6f 6c 6f 72 3d 27				
0090	23 66 35 37 39 30 30 27 20 63 6f 6c 73 70 61 6e				
00a0	3d 22 35 22 3e 3c 73 70 61 6e 20 73 74 79 6c 65				
00b0	3d 27 62 61 63 6b 67 72 6f 75 6e 64 2d 63 6f 6c				
00c0	6f 72 3a 20 23 63 63 30 30 30 30 3b 20 63 6f 6c				
00d0	6f 72 3a 20 23 66 63 65 39 34 66 3b 20 66 6f 6e				
00e0	74 2d 73 69 7a 65 3a 20 78 2d 6c 61 72 67 65 3b				
00f0	27 3e 28 20 21 20 29 3c 2f 73 70 61 6e 3e 20 20 4e				
0100	6f 74 69 63 65 3a 20 54 72 79 69 6e 67 20 74 6f				
0110	20 61 63 63 65 73 73 20 61 72 72 61 79 20 6f 66				
0120	66 73 65 74 20 6f 6e 20 76 61 6c 75 65 20 6f 66				
0130	20 74 79 70 65 20 6e 75 6c 6c 20 69 6e 20 43 3a				



The solution was that while copying the previous code, I had forgotten to change \$row2 to \$row3. After fixing this, the page finally loaded with the correct data.

```
sult3)) {  
|, [$row3["restaurantid"], $r
```



Problem 20

While working in multiple locations in real life, an issue occurred which caused me to redo the databases. Having a local database meant that when I changed wifi networks, I would have to recode the app. To fix this issue, I piggybacked off my own website and created a subdomain (alleat.cpur.net) which I then installed phpmyadmin on and edited the code to go to the new domain instead of the IP address

```

Future<List> getFavourites() async {
    String phurl = "https://alleat.cpur.net/query/favouriterestaurantlist.php";
    var profile = await SQLiteLocalDB.getProfileSelected();
    String email = profile[0]["email"].toString();
    try {

```

As well, to reduce the page loading times, I decided to have the images hosted on the site instead of from Google.

ress	website	phonenumber	fhrsid	description	restaurantlogo	restaurantbanner	restaurantcategory
b High sel. khamsted, 4 1AD	https://www.papajohns.co.uk/	1442862900	202320	The iconic, internationally- adored Papa John's nee...	https://alleat.cpur.net/assets/images/restaurant_l... https://alleat.cpur.net/assets/images/restaurant_b... Pizza		
i High sel. khamsted, 4 1ZZ	https://www.pizzaexpress.com/	1442750510	1285623		https://alleat.cpur.net/assets/images/restaurant_l... https://alleat.cpur.net/assets/images/restaurant_b... Italian		
i High sel. khamsted, 4 1HU	https://www.thefatbuddha.co.uk	1442879995	1252835		https://alleat.cpur.net/assets/images/restaurant_l... https://alleat.cpur.net/assets/images/restaurant_b... Indian		
' High sel. khamsted, 4 1AL	https://www.meeting-room.co.uk	1442879994	1150863	Our menu, though simple, is quality focused as we	https://alleat.cpur.net/assets/images/restaurant_l... https://alleat.cpur.net/assets/images/restaurant_b... Burgers		
i High sel. khamsted, 4 3AP	https://www.starbucks.co.uk/	1422777800	1452823		https://alleat.cpur.net/assets/images/restaurant_l... https://alleat.cpur.net/assets/images/restaurant_b... Breakfast		
				Zero aims to...			

Restaurant Page

To start with, I first passed the data from the restaurant list to get the initial banner, logo, name and id so that it does not need to query the server

```

MaterialPageRoute(
    builder: (context) =>
        RestaurantMainPage(
            resid: restaurantinfodata[0],
            resname: restaurantinfodata[1],
            reslogo: restaurantinfodata[2],
            resbanner:
                restaurantinfodata[3],
        )));
// RestaurantMainPage // MaterialPageRoute

```

```
class RestaurantMainPage extends StatelessWidget {
  var resid;
  var resbanner;
  var reslogo;
  var resname;

  RestaurantMainPage(
    {Key? key, this.resid, this.resname, this.reslogo, this.resbanner})
    : super(key: key);

  @override
  Widget build(BuildContext context) {
```

The name of the restaurant is then put as the appBar title

```
  resizeToAvoidBottomInset: false,
  appBar: AppBar(
    title: Text(resname),
    leading: IconButton(
```

As I will be needing the dish information and menu categories, I will need to create the tables on the SQL server.

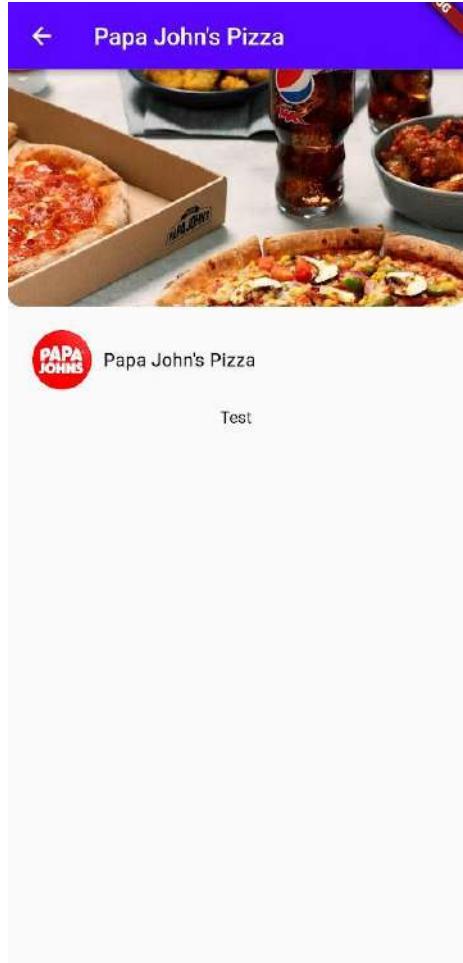
```

INSERT INTO `menucategories` (`categoryid`, `restaurantid`, `categoryname`, `viewingorder`)
('2', '1', 'Papadias', '2'),
('3', '1', 'Sides', '3'),
('4', '1', 'Dips', '4'),
('5', '1', 'Drinks', '5'),
('6', '1', 'Desserts', '6'),
('7', '2', 'Pizzeria Bundles', '1'),
('8', '2', 'Pizzas', '2'),
('9', '2', 'Al Forno', '3'),
('10', '2', 'Calzone', '4'),
('11', '2', 'Dips', '5'),
('12', '2', 'Salads', '6'),
('14', '2', 'Desserts', '7'),
('15', '2', 'Soft Drinks', '8'),
('16', '3', 'Starters & Appetisers', '1'),
('17', '3', 'Vegetable Sides', '2'),
('18', '3', 'Breads and Rice', '3'),
('19', '3', 'Chefs Specials', '4'),
('20', '3', 'All Time Favourites', '5'),
('21', '3', 'Seafood', '6'),
('22', '3', 'Biryani Dishes', '7'),
('23', '3', 'Tandoori Dishes', '8'),
('24', '4', 'Drinks', '1'),
('25', '4', 'Specials', '2'),
('26', '4', 'Hand Crafted Burgers', '3'),
('27', '4', 'Shakes', '4'),
('28', '4', 'Sides', '5'),

```

NOTE: During the making of the restaurant page there was an update for Flutter so some code previously made had to be adapted to fit the new coding. Overall, there was not many changes to fit with this new update

The next thing I did was display the restaurant details and display them to the user so they can see which restaurant they have selected



This just used a slight iteration from the restaurant list file, using an import from the restaurant list page instead of querying the database.

The next thing I did was import restaurant categories from the database. Since this table was not made on the new server, it had to be coded in.

	<input type="button" value="←"/> <input type="button" value="→"/>	categoryid	restaurantid	categoryname	viewingorder
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	1	Pizza	1
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	1	Papadias	2
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	1	Sides	3
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	4	1	Dips	4
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	5	1	Drinks	5
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	6	1	Desserts	6
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	7	2	Pizzeria Bundles	1
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	8	2	Pizzas	2
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	9	2	Al Forno	3
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	10	2	Calzone	4
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	11	2	Dips	5
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	12	2	Salads	6
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	14	2	Desserts	7
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	15	2	Soft Drinks	8
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	16	3	Starters & Appetisers	1
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	17	3	Vegetable Sides	2

I then created a PHP file which would send the data associated with the restaurant id within a single array to display.

Papa John's Pizza

PAPA JOHN'S

Pizza

Papadias

Sides

Dips

Drinks

Desserts

At this point, I realised that the restaurant page will certainly take more than a single screen to show so implemented the scroll mechanic in order for the user to be able to scroll down.

Since an empty restaurant page was pointless, items were implemented, with the summary showing an image preview, name, short description and the price. While developing this, there were a variety of issues;

Problem 21

The first issue was that for longer named items, it would overflow the container and go past the screen. The solution was to add the overflow function which broke the line.

Problem 22

The next issue was that the price would show up next to the description instead of on the right side of the container. In order to fix this, I used the Flexible which allowed to specify the ratio that the items showed up. I then made the price text aligned to the right.

Problem 23

The final issue I faced was with the description, since it is a summary, it should be short but the whole description was showing which meant some items were over 10 lines long. To fix this, I used the overflow class, resulting in only one line of the description to show.

The image shows a screenshot of a mobile application for Papa John's Pizza. At the top, there is a purple header bar with a back arrow icon and the text "Papa John's Pizza". Below the header is a photograph of a pizza in a box next to a bottle of Pepsi and some side dishes. Underneath the photo is the Papa John's logo. The main content area is titled "Pizza" and lists four pizza options:

Pizza Type	Description	Price
All the Meats	Our tomato sauce with mozzarella...	£18
American Hot	Our tomato sauce with mozzarella...	£16
BBQ Meat Feast	BBQ sauce with mozzarella, spicy ...	£18
BBQ Chicken Classic	Our tomato sauce with mozzarella...	£17

Item Page

To do the item page, I forwarded the data from the item summary to the page, allowing for a page which showed information about each item.



Nicoise £12
Salad · Italian

On the menu since 85. The big flavours of white anchovies, capers, olives and tuna, with baby spinach, rocket, cucumber, free-range egg, parsley and our updated house dressing

Post Development

Testing

Testing Summary

Test Number	Success Criteria	What it is	Expected	Actual	Pass/Fail
-------------	------------------	------------	----------	--------	-----------

1	1.1	All inputs should be checked for SQL injection, checking for key terms that use SQL	When the user inputs SQL, it should prevent the user from saving the data and force them to retype the form field	My program currently does not allow the user to input ' or " in order to prevent the closing of the form field and therefore prevents the use of SQL injection. In the future, I would like to allow the user to user to input ' & " but for now this prevents the use of SQL injection. The result means that the typed key ' or " does not get registered by the app.	PASS
2	1.2	Passwords used to make and log into a profile should be kept secure by using a password hash	The database should have hashed passwords stored not plaintext passwords	The passwords are hashed but the hash uses a generic key and iv not a random one	PASS
3	1.3	The database should be in third-normal form	The data is third-normal form	The database uses third-normal, using	PASS

		form and normalised		foreign keys to create relationships between the tables	
4	2.1	When there is no profiles on the app, the user should be presented with the login/profile creation page	Both on setup and when the user removes the profiles, the user should be brought to the setup page	Upon removing all the profiles, the app does not automatically go to the setup page. You need to reopen the app for it to check if there is no profiles logged in	FAIL
5	2.1	When there is no profiles on the app, the user should be presented with the login/profile creation page	Both on setup and when the user removes the profiles, the user should be brought to the setup page	When there is no profiles logged in, the app automatically goes to the setup page	PASS
6	2.4	If the user closes the app or goes back during profile creation, the app should not process the profile	If the profile login/creation page is not complete, it should not continue	The required form fields have to be completed before the app allows for the button performs any actions. As well, all the	PASS

				data is sent at once over the internet.	
7	2.5	When creating a profile using an email, it should only allow emails that contain an @ symbol	Invalid emails should not be allowed and force the user to input using a valid email format. A valid email format should be allowed	The user must type an email which uses both an @ symbol and a dot after the @ symbol	PASS
8	2.6	First names and last names should only be accepted if they are under 50 characters each	A user should not be able to create a profile that has a firstname and/or a lastname	When a user has a firstname or lastname 50 or more, it fails the validation. For longer names ... is put in place if it is too long	PASS
9	2.7	Passwords made should be a minimum of 7 characters and a max of 99	A user should not be required to set a password that is between 7 and 99 characters	The user is required to set a password that is between 7 and 99 characters	PASS
10	2.8	Passwords should contain a letter and a number	Forces user to enter at least one letter and one number	The password form field is validated if it contains one letter and	PASS

				one number	
11	2.21	A user should be able to sign in with their profile using multiple devices	Should be able to successfully log in to their account simultaneously on multiple devices	Successfully signed in on multiple devices	PASS
12	2.22	Information of the profile should be synced to the cloud	Changes made are synced across devices	The data is synced successfully	PASS
13	2.23	Each profile should be able to access their favourites	Should be able to access favourites	Each profile has its own favourites which you can add and remove	PASS
14	6.1	The user should be able to open the application using an app icon with the correct logo	The app should have an icon associated with it	Successfully shows	PASS
15	6.2	By default, the user should be brought to the home page when the app is opened	On app opening, it brings the user to the home page	Opens app to home page if the user has completed setup	PASS

Testing Criteria (1.1)

All inputs should be checked for SQL injection, checking for key terms that use SQL

Justification

This ensures that the database cannot be edited by the user and means that the database is kept secure from malicious actions

Expected result

When the user inputs SQL, it should prevent the user from saving the data and force them to retype the form field

Actual result

My program currently does not allow the user to input ' or " in order to prevent the closing of the form field and therefore prevents the use of SQL injection. In the future, I would like to allow the user to user to input ' & " but for now this prevents the use of SQL injection. The result means that the typed key ' or " does not get registered by the app.

```
padding: EdgeInsets.only(top: 15.0),
        child: Icon(Icons.lock))), // Padding //
inputFormatters: [
    FilteringTextInputFormatter.allow(
        RegExp('[a-zA-Z0-9!@#%^&*(),.?:{}}|<>]')) //
],
```

← Register Profile

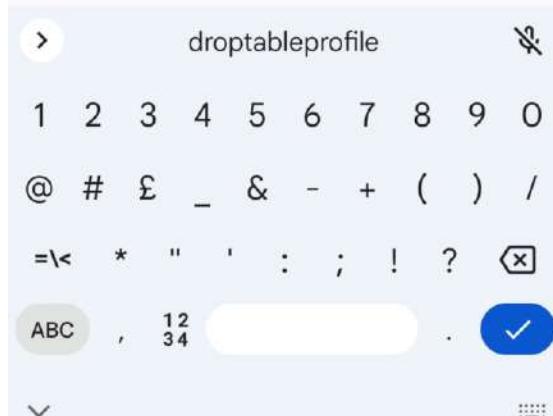
 First Name _____ Surname _____

 Email droptableprofile

 Password _____

 Confirm Password _____

Create Profile



Pass or Fail?

— PASS — (Revisit in future)

Testing Criteria (1.2)

Passwords used to make and log into a profile should be kept secure by using a password hash

Justification

This ensures that if there is a database leak, the original passwords cannot be decoded.

Expected result

The database should have hashed passwords stored not plaintext passwords

Actual result

The passwords are hashed but the hash uses a generic key and iv not a random one

<input type="checkbox"/>				98	Max	Gething	maxgething@cpur.net	\$2y\$10\$gMFQLhqGMdoqKbkS8VL3eirHw2vWAptdFAGbXAyZf...
<input type="checkbox"/>				99	Robbie	Sucho	robbiesucho@cpur.net	\$2y\$10\$5ewk8WeHGGWHEvMD.4l0aOj3NtI2UINPhP3wKqgDhZ...
<input type="checkbox"/>				100	Sam	Ripley	samripley@cpur.net	\$2y\$10\$zFaAxW5Co9V9.o9v/G0vLu1n6p0X9edB0oweQi.b6DG...
<input type="checkbox"/>				101	Stevie	Turner	stevieturner@cpur.net	\$2y\$10\$QGB2BPF0j.KSGQ7/M59TUOB6rCUUKC9jVtuJesAo60T...
<input type="checkbox"/>				102	john	tron	johntron@cpur.net	\$2y\$10\$7vKuimsw4qD0mZoAENsAQexJYB3bQTiyKGzDRH1f41n...

Pass or Fail?

— PASS — (Revisit in future)

Testing Criteria (1.3)

The database should be in third-normal form and normalised

Justification

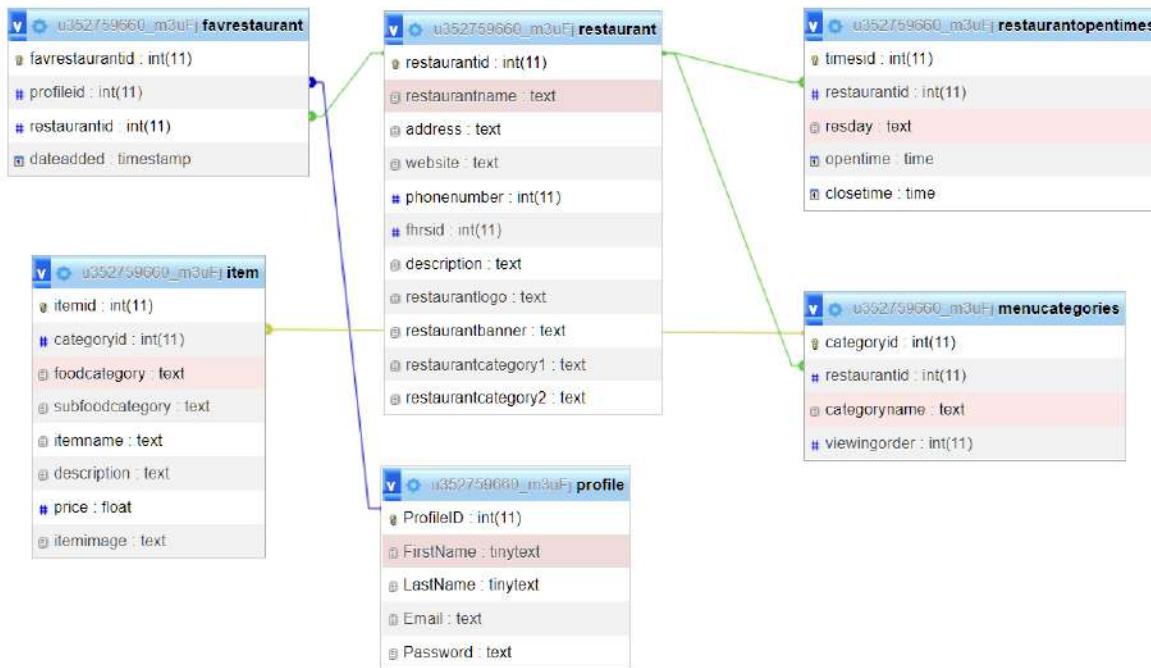
This makes the database not have repeated data so that query time is reduced

Expected result

The data is third-normal form

Actual result

The database uses third-normal, using foreign keys to create relationships between the tables



Pass or Fail?

— PASS — (Revisit in future)

Testing Criteria (2.1)

When there is no profiles on the app, the user should be presented with the login/profile creation page

Justification

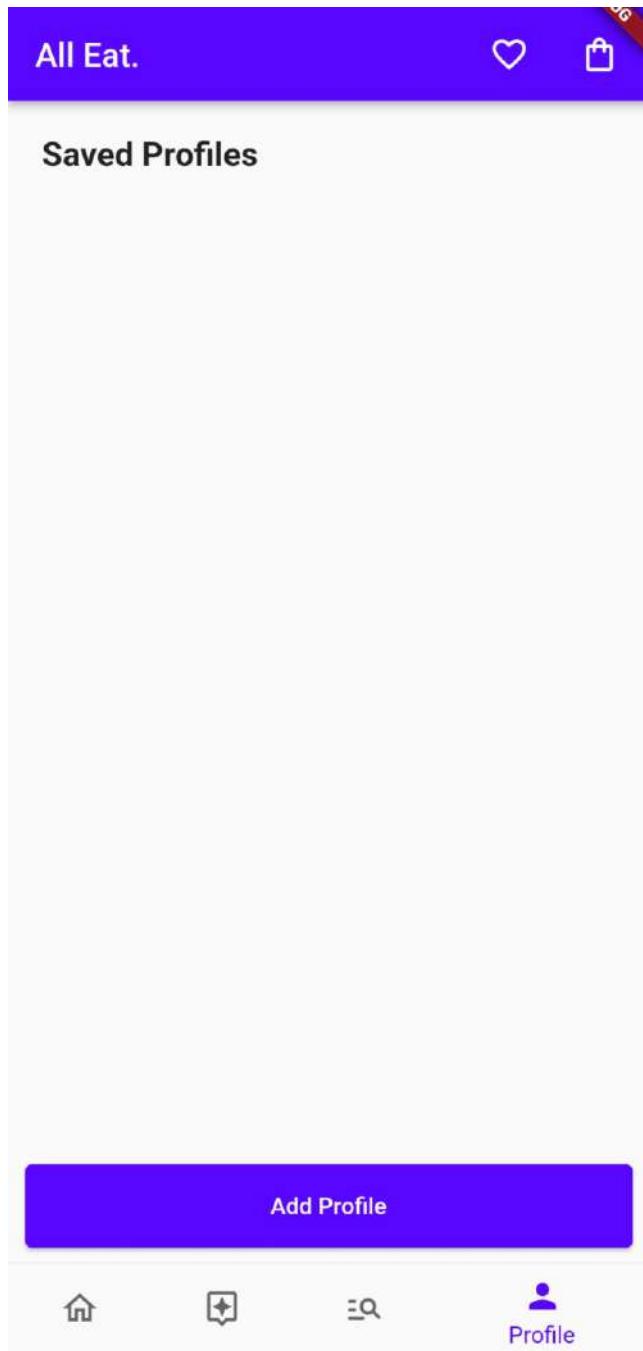
This means that the user cannot order food without having a profile associated with it

Expected result

Both on setup and when the user removes the profiles, the user should be brought to the setup page

Actual result

Upon removing all the profiles, the app does not automatically go to the setup page. You need to reopen the app for it to check if there is no profiles logged in



Pass or Fail?

— FAIL —

Fixing Issues

While developing the fix to the issue, I noticed that the app was using generic error snackbars. At the time of making them, I had no idea how to make them dynamic to the error but now I did so added code to display the error code so that the user can see the error instead of logging it in the console

```
    SnackBar(
      content:
        Text("Error ${res.statusCode}: Please try again later.")), // Snackbar
    );
msg = "Connection to database failed.";
```

As well, while fixing the removal system, I found that if a user deletes a profile that has been selected it will not automatically change the selected profile meaning that none is selected. I fixed this by setting the first profile to selected if after the deletion there is no profiles selected. A check is also done when the profile page is opened. If there is not exactly 1 profile selected, the first profile is selected.

Another fix I did was redo the entire wrapper.dart file since previously, it was building within the future which isn't good practice.

```
import 'package:alleat/screens/fresh_app/fresh_profile.dart';

import 'package:alleat/services/sqlite_service.dart';

import 'package:alleat/widgets/apperror.dart';

import 'package:alleat/widgets/genericload.dart';

import 'package:alleat/widgets/navigationbar.dart';

import 'package:flutter/material.dart';

class SetupWrapper extends StatefulWidget {

  const SetupWrapper({Key? key}) : super(key: key);
```

```
    @override

    State<SetupWrapper> createState() => _SetupWrapperState();

}

class _SetupWrapperState extends State<SetupWrapper> {

    //Default setup not complete if something wrong happens

    bool setup = false;

    Future<bool> isSetupComplete() async {

        List<Map> profileInfo = await SQLiteLocalDB

            .getFirstProfile(); //Call Database for the first entry

        if (profileInfo.isEmpty) {

            //If the first entry is empty

            //Then setup is not complete (pass to build)

            return false;

        } else {

            //If the first entry exists

            //Setup is complete (pass to build)

            return true;

        }

    }

}
```

```
    @override

    Widget build(BuildContext context) => FutureBuilder<bool> (

        future: isSetupComplete(),

        builder: (context, snapshot) {

            if (!snapshot.hasData) {

                return const GenericLoading();

            }

            if (snapshot.hasError) {

                return const AppError();

            } else {

                bool setup = snapshot.data ?? [] as bool;

                if (setup == false) {

                    return const FreshProfile();

                } else if (setup == true) {

                    return const Navigation();

                } else {

                    return const AppError();

                }

            }

        } ,

    ) ;

}
```

I finally did the fix that was intended, making it automatically go to the setup page on profile removal.

```
Future<void> deleteProfile(index) async {
    await SQLiteLocalDB.deleteProfile(index);
    List profileInfo = await SQLiteLocalDB.getDisplayProfiles();
    setState(() {
        profileInfo = profileInfo;
    });
    List<Map> profileInfoCheck = await SQLiteLocalDB
        .getFirstProfile(); //Call Database for the first entry
    if (profileInfoCheck.isEmpty) {
        //If the first entry is empty
        setup = false; //Then setup is not complete (pass to build)
        setState(() {
            Navigator.push(context,
                MaterialPageRoute(builder: (context) => const FreshProfile()));
        });
    } else {
        index = await SQLiteLocalDB.getFirstProfile();
        index = index[0]["id"];
        await SQLiteLocalDB.setSelected(index);
        List profileInfoSelect = await SQLiteLocalDB.getDisplayProfiles();
        setState(() {
            profileInfoSelect = profileInfoSelect;
        });
    }
}
```

Recode Result

When there is no profiles logged in, the app automatically goes to the setup page

Pass or Fail?

— PASS — (Revisit in future)

Testing Criteria (2.4)

If the user closes the app or goes back during profile creation, the app should not process the profile

Justification

If a user does not fully creates a profile, it could cause a database error or cause the app to crash if some of the fields are missing

Expected result

If the profile login/creation page is not complete, it should not continue

Actual result

The image displays two side-by-side screenshots of a mobile application interface. The left screenshot shows the 'Login to Profile' screen with fields for 'Email' and 'Password'. The right screenshot shows the 'Register Profile' screen with fields for 'First Name', 'Surname', 'Email', 'Password', and 'Confirm Password'. Both screens feature a blue header bar with back arrows and a blue button at the bottom.

Screen	Field	Type	Placeholder	Required
Login to Profile	Email	Text		Required
	Password	Text		Required
Register Profile	First Name	Text		Required
	Surname	Text		Required
	Email	Text		Required
	Password	Text		Required
	Confirm Password	Text		Required

The required form fields have to be completed before the app allows for the button performs any actions. As well, all the data is sent at once over the internet.

Pass or Fail?

— PASS — (Revisit in future)

Testing Criteria (2.5)

When creating a profile using an email, it should only allow emails that contain an @ symbol

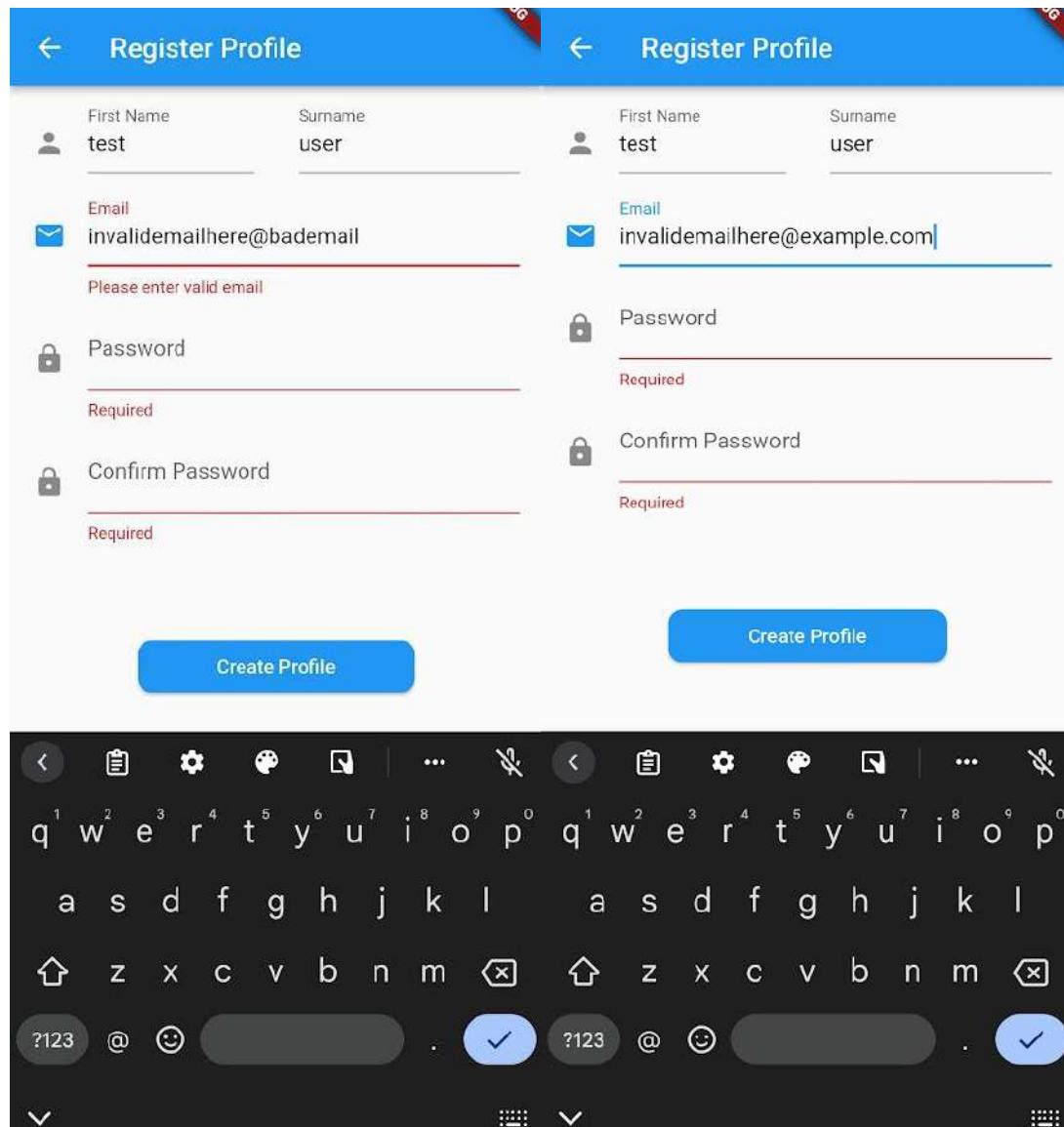
Justification

This ensures that for email verifications, it gets sent to a real email.

Expected result

Invalid emails should not be allowed and force the user to input using a valid email format. A valid email format should be allowed

Actual result



The user must type an email which uses both an @ symbol and a dot after the @ symbol

Pass or Fail?

— PASS —

Testing Criteria (2.6)

First names and last names should only be accepted if they are under 50 characters each

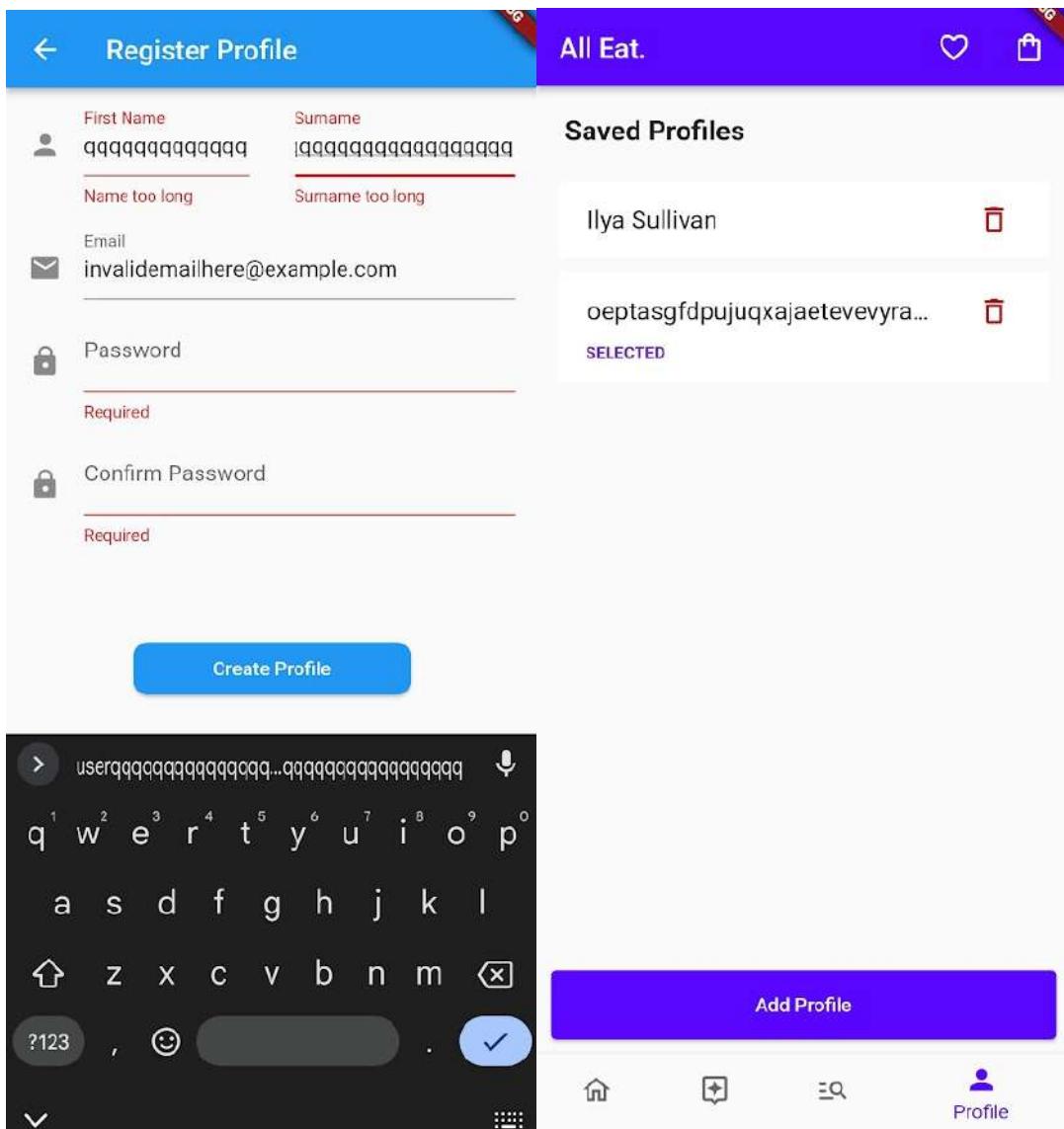
Justification

This ensures that there are no major issues displaying the name on the app. A name would therefore be a maximum of 101 characters.

Expected result

A user should not be able to create a profile that has a firstname and/or a lastname

Actual result



When a user has a firstname or lastname 50 or more, it fails the validation. For longer names ... is put in place if it is too long

Pass or Fail?

— PASS —

Testing Criteria (2.7)

Passwords made should be a minimum of 7 characters and a max of 99

Justification

This ensures that a password is encrypted well and makes it harder for people to guess

Expected result

A user should not be required to set a password that is between 7 and 99 characters

Actual result

The user is required to set a password that is between 7 and 99 characters

First Name: hello Surname: me

Email: hellome@example.com

Password:

Confirm Password:

Create Profile

First Name: hello Surname: me

Email: hellome@example.com

Password:

Confirm Password:

Required

Create Profile

Pass or Fail?

— PASS —

Testing Criteria (2.8)

Passwords should contain a letter and a number

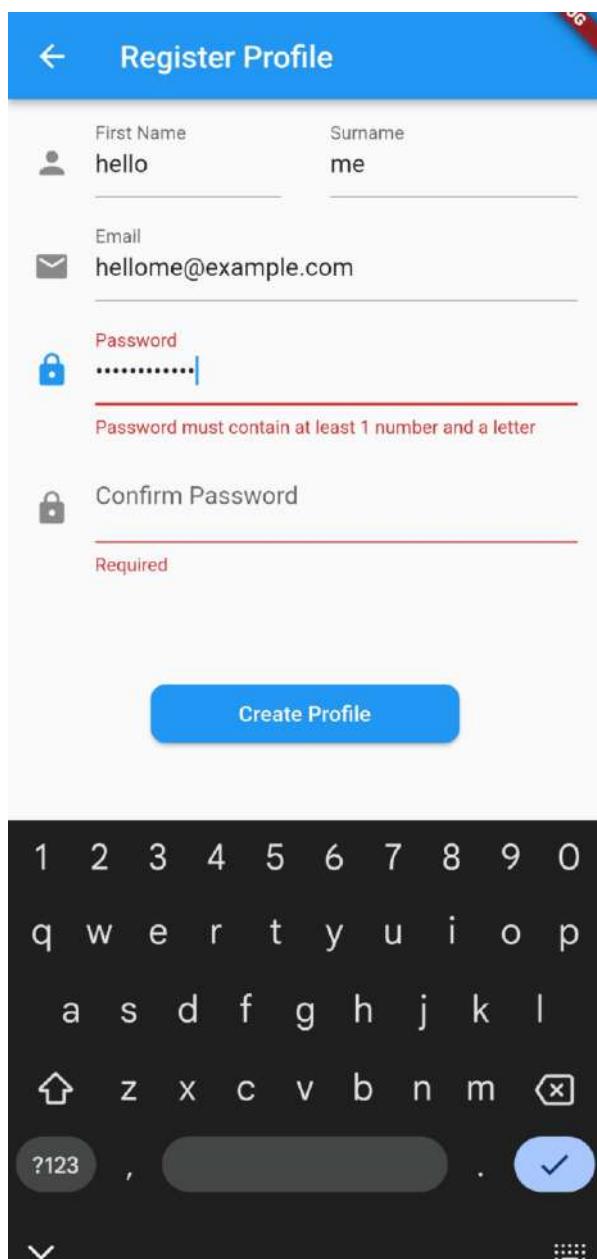
Justification

This ensures that the password is harder to guess and makes the encryption better

Expected result

Forces user to enter at least one lesser and one number

Actual result



The password form field is validated if it contains one letter and one number

```
    }
    if (!password.contains(RegExp(r'[0-9]'))) ||
       !password.contains(RegExp(r'[a-z]'))) {
      return "Password must contain at least 1 number and a letter";
    }
```

Pass or Fail?

— PASS —

Testing Criteria (2.21)

A user should be able to sign in with their profile using multiple devices

Justification

This allows for a user to be able to view restaurant items while they are not on their main device.

Expected result

Should be able to successfully log in to their account simultaneously on multiple devices

Actual result

Successfully signed in on multiple devices

Pass or Fail?

— PASS —

Testing Criteria (2.22)

Information of the profile should be synced to the cloud

Justification

This allows changes to be synced with the cloud for each devices and for them to update

Expected result

Changes made are synced across devices

Actual result

The data is synced successfully

← T →	▼	favrestaurantid	profileid	restaurantid	dateadded	
<input type="checkbox"/>	 Edit	 Copy	 Delete	65	102	1 2022-09-06 12:45:55

Pass or Fail?

— PASS —

Testing Criteria (2.23)

Each profile should be able to access their favourites

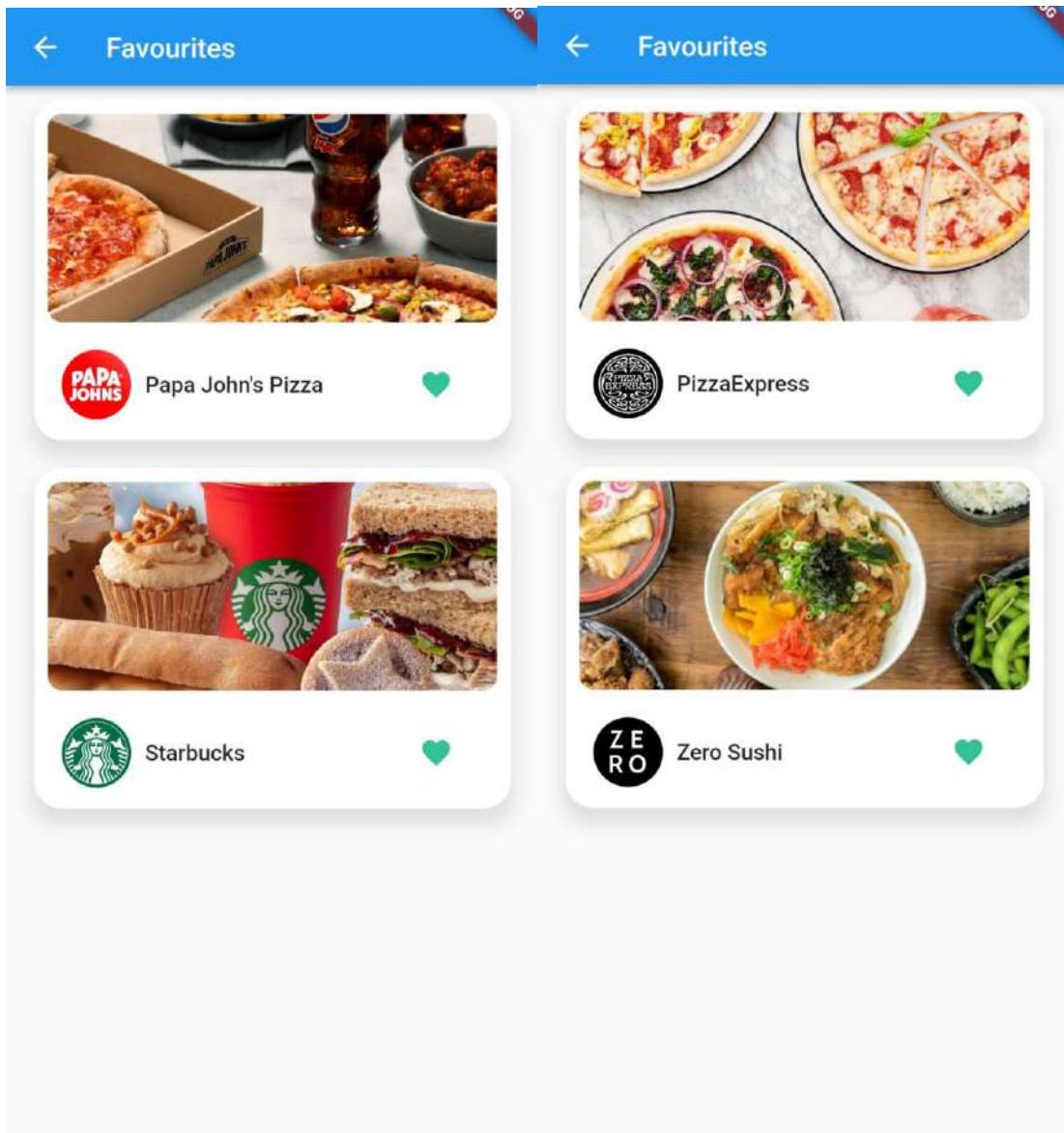
Justification

This means that because there are lots of restaurants in the world, users should be able to favourite their favourite restaurant.

Expected result

Should be able to access favourites

Actual result



Each profile has its own favourites which you can add and remove

Pass or Fail?

— PASS —

Testing Criteria (6.1)

The user should be able to open the application using an app icon with the correct logo

Justification

This means it is easily identifiable

Expected result

The app should have an icon associated with it

Actual result



Pass or Fail?

— PASS —

Testing Criteria (6.2)

By default, the user should be brought to the home page when the app is opened

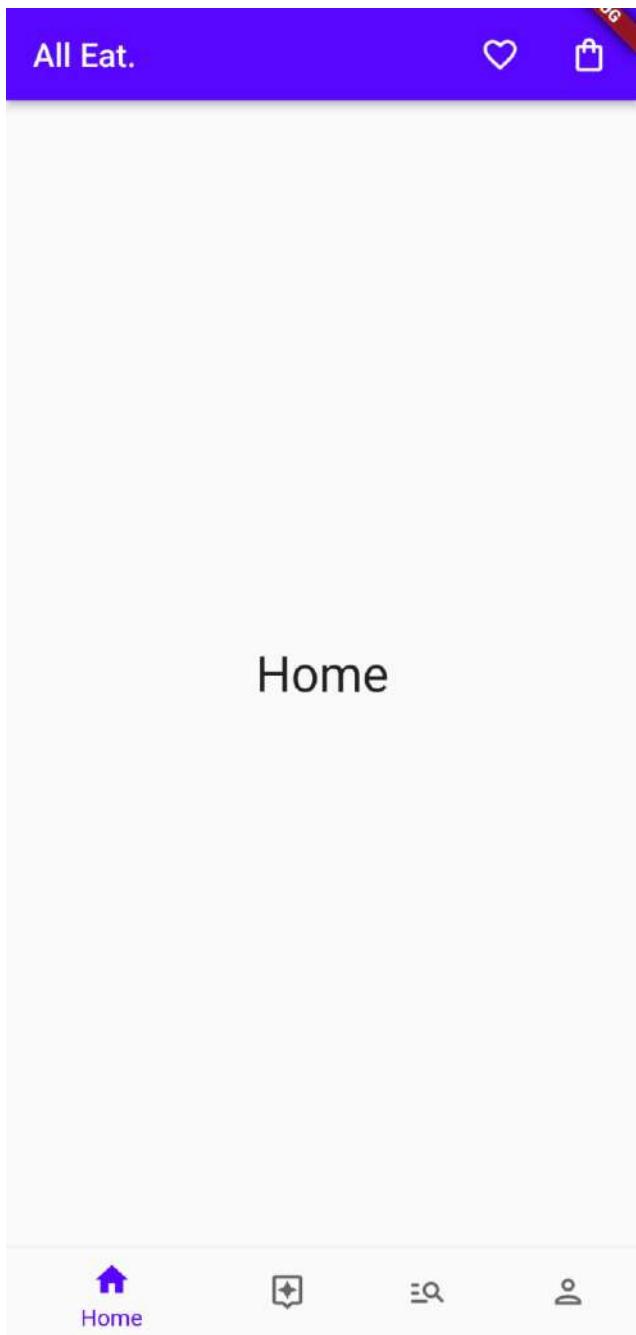
Justification

This means that the user is able to see the most useful information

Expected result

On app opening, it brings the user to the home page

Actual result

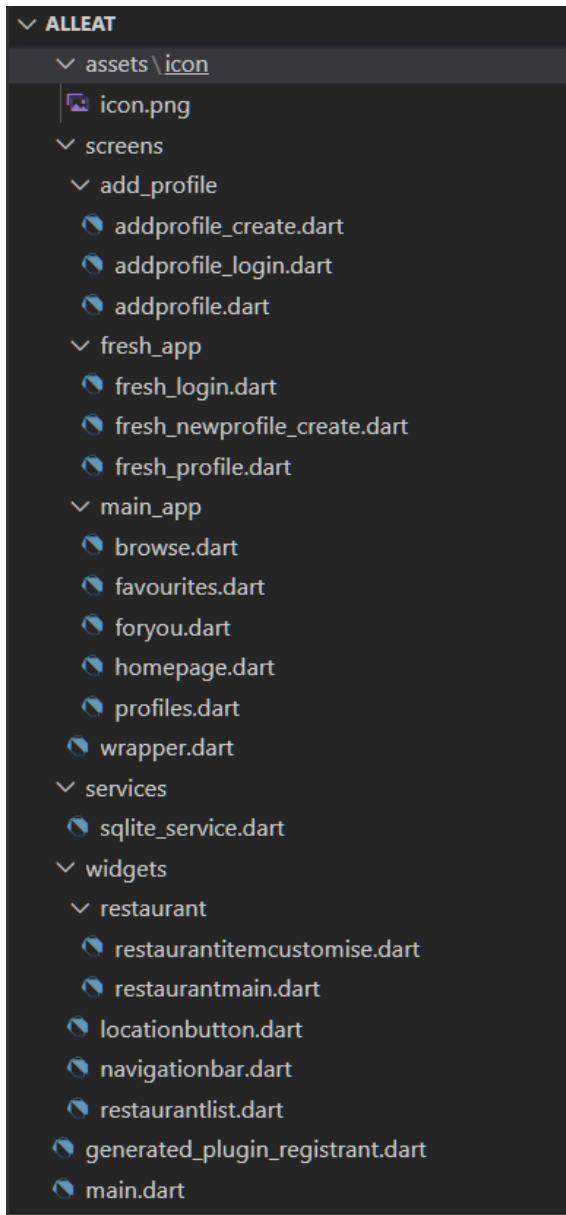


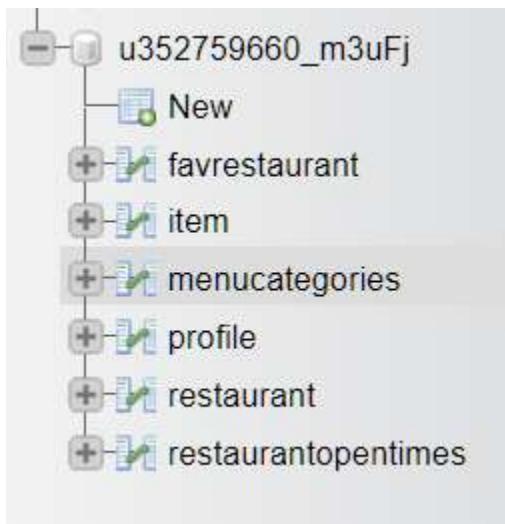
Pass or Fail?

— PASS —

Prototype 1A Final Code

File Structure





public_html > alleat > query				
Name ↑	Size	Last modified		
< > favouriterestaurant.php	2.38 KB	4 days ago	-rwxr-xr-x	
< > favouriterestaurantdata.php	2.49 KB	4 days ago	-rwxr-xr-x	
< > favouriterestaurantlist.php	1.81 KB	4 days ago	-rwxr-xr-x	
< > login.php	2.81 KB	4 days ago	-rwxr-xr-x	
< > register.php	2.42 KB	4 days ago	-rwxr-xr-x	
< > restaurantlist.php	1.35 KB	4 days ago	-rwxr-xr-x	
< > restaurantmenucategories.php	1.25 KB	3 days ago	-rwxr-xr-x	
< > restaurantmenuitems.php	1.36 KB	2 days ago	-rwxr-xr-x	

public_html > ... > assets > images > restaurant_banner				
Name ↑	Size	Last modified		
dojoasianfusionbanner.jpg	183.35 KB	4 days ago	-rwxr-xr-x	
papajohnsbanner.jpeg	238.94 KB	4 days ago	-rwxr-xr-x	
pizzaexpressbanner.jpg	272.39 KB	4 days ago	-rwxr-xr-x	
riversidefishandchipsbanner.jpeg	164.12 KB	4 days ago	-rwxr-xr-x	
starbucksbanner.jpg	248.01 KB	4 days ago	-rwxr-xr-x	
thefatbuddhabanner.jpg	309 KB	4 days ago	-rwxr-xr-x	
themeatingroombanner.jpg	302.55 KB	4 days ago	-rwxr-xr-x	
zerosushibanner.jpg	188.37 KB	4 days ago	-rwxr-xr-x	

Assets / images / restaurant_item

Name ↑	Size	Last modified	
 7free.webp	11.32 KB	2 days ago	-rwxr-xr-x
 allthemeats.webp	60.34 KB	2 days ago	-rwxr-xr-x
 american.webp	63.52 KB	2 days ago	-rwxr-xr-x
 americanhot.webp	33.65 KB	2 days ago	-rwxr-xr-x
 bbqchickenandbaconpapadias.webp	20.61 KB	2 days ago	-rwxr-xr-x
 bbqchickenclassic.jpg	57.9 KB	2 days ago	-rwxr-xr-x
 bbqdip.webp	8.66 KB	2 days ago	-rwxr-xr-x
 bbqmeatfeast.webp	34.64 KB	2 days ago	-rwxr-xr-x
 bolognese.webp	37.05 KB	2 days ago	-rwxr-xr-x
 calzonenduja.webp	53.82 KB	2 days ago	-rwxr-xr-x
 calzoneverdure.webp	48.51 KB	2 days ago	-rwxr-xr-x
 cheeseandtomato.jpg	50.46 KB	2 days ago	-rwxr-xr-x
 cheesegarlicsticks.webp	39.76 KB	2 days ago	-rwxr-xr-x
 chickenpoppers.webp	21.5 KB	2 days ago	-rwxr-xr-x
 chocolatefudaebrownie.webp	19.53 KB	2 days ago	-rwxr-xr-x

Assets / images / restaurant_logo

Name ↑	Size	Last modified	
 dojoasianfusionlogo.png	41.3 KB	4 days ago	-rwxr-xr-x
 papajohnslogo.png	166.55 KB	4 days ago	-rwxr-xr-x
 pizzaexpresslogo.png	376.2 KB	4 days ago	-rwxr-xr-x
 riversidefishandchipslogo.jpg	49.94 KB	4 days ago	-rwxr-xr-x
 starbuckslogo.png	552.05 KB	4 days ago	-rwxr-xr-x
 thefatbuddhalogo.png	38.42 KB	4 days ago	-rwxr-xr-x
 themeetingroomlogo.png	253.55 KB	4 days ago	-rwxr-xr-x
 zerosushilogo.png	22.01 KB	4 days ago	-rwxr-xr-x

Files - Application

main.dart

```
import 'package:alleat/screens/wrapper.dart';

import 'package:flutter/material.dart';

void main() { //Start App

  runApp(const MyApp());
}
```

```
}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return const MaterialApp( //Create main app

      title: 'AllEat.',

      home: SetupWrapper(), //Check if setup is complete for app
      (reference)

    );
  }
}
```

wrapper.dart

```
import 'package:alleat/screens/fresh_app/fresh_profile.dart';

import 'package:alleat/services/sqlite_service.dart';

import 'package:alleat/widgets/apperror.dart';

import 'package:alleat/widgets/genericload.dart';

import 'package:alleat/widgets/navigationbar.dart';

import 'package:flutter/material.dart';
```

```
class SetupWrapper extends StatefulWidget {

  const SetupWrapper({Key? key}) : super(key: key);

  @override

  State<SetupWrapper> createState() => _SetupWrapperState();
}

class _SetupWrapperState extends State<SetupWrapper> {

  //Default setup not complete if something wrong happens

  bool setup = false;

  Future<bool> isSetupComplete() async {

    List<Map> profileInfo = await SQLiteLocalDB

      .getFirstProfile(); //Call Database for the first entry

    if (profileInfo.isEmpty) {

      //If the first entry is empty

      //Then setup is not complete (pass to build)

      return false;

    } else {

      //If the first entry exists

      //Setup is complete (pass to build)

      return true;

    }
  }
}
```

```
}

@Override

Widget build(BuildContext context) => FutureBuilder<bool>(

    future: isSetupComplete(),

    builder: (context, snapshot) {

        if (!snapshot.hasData) { //While loading, go to loading page

            return const GenericLoading();

        }

        if (snapshot.hasError) { //If the app has an error, go to error
page

            return const AppError();

        } else {

            bool setup = snapshot.data ?? [] as bool; //Get data from
Future

            if (setup == false) { //If setup is not complete

                return const FreshProfile(); //Go to Setup page

            } else if (setup == true) { //If setup is complete

                return const Navigation(); //Go to main page

            } else { //If there is an error

                return const AppError(); // Go to error page

            }

        }

    }

),
```

```
) ;  
}
```

sqlite_service.dart

```
import 'package:flutter/foundation.dart';  
  
import 'package:flutter/rendering.dart';  
  
import 'package:sqflite/sqflite.dart' as sql;  
  
  
  
class SQLiteLocalDB {  
  
  //-----  
  
  // Default Database  
  
  //-----  
  
  
  
  static Future<void> createTableProfile(sql.Database database) async {  
  
    //Create localprofiles table (Used to store local logged in profiles)  
  
    await database.execute("""CREATE TABLE localprofiles(  
  
      id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
  
      firstname TEXT,  
  
      lastname TEXT,  
  
      email TEXT,  
  
      password TEXT,  
  
      selected TEXT,  
  
      createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
```

```

        )

        """);

    }

    static Future<void> createTablePreferences(sql.Database database) async {
        //Create systempreferences table (Used to store app preferences
        //including current location)

        await database.execute("""
CREATE TABLE systempreferences(
    sysid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    profileid INT,
    apptheme TEXT,
    savedaddressextra TEXT,
    savedaddressstreet TEXT,
    savedaddresscity TEXT,
    savedaddresspostcode TEXT,
    FOREIGN KEY(profileid) REFERENCES localprofiles(id)
)
        """);

    }

    static Future<sql.Database> db() async {
        //If tables don't exist, create tables

        return sql.openDatabase(

```



```
'SELECT savedaddressextra, savedaddressstreet, savedaddresscity,
savedaddresspostcode FROM systempreferences WHERE profileid = $selectedID
LIMIT 1'); //Get the saved address from the database

}

static Future<void> setAddress(String extraAddress, String
streetAddress,
String cityAddress, String postcodeAddress) async {

//Save address to profile

if (extraAddress.isEmpty) {

//If a part of the address is null, replace with empty string

extraAddress = "";

}

if (streetAddress.isEmpty) {

streetAddress = "";

}

if (cityAddress.isEmpty) {

cityAddress = "";

}

if (postcodeAddress.isEmpty) {

postcodeAddress = "";

}

final db = await SQLiteLocalDB.db();

List selectedProfile = await db.rawQuery(
```

```

        'SELECT id FROM localprofiles WHERE selected = True LIMIT 1');
//Get current selected profile so that saved address associated with
current profile can be replaced with new one

String selectedID = selectedProfile[0]["id"].toString();

List<Map> profilesavedpreferences = await db.rawQuery(
    'SELECT sysid FROM systempreferences WHERE profileid = $selectedID
ORDER BY sysid ASC LIMIT 1');

if (profilesavedpreferences.isEmpty) {

    //If there is no preference record, create a new one

    await db.insert("systempreferences", {
        "profileid": selectedProfile[0]["id"],
        "savedaddressextra": extraAddress,
        "savedaddressstreet": streetAddress,
        "savedaddresscity": cityAddress,
        "savedaddresspostcode": postcodeAddress,
    });
}

} else {

    //If there is a preference record associated with the selected
    profile, replace saved profile

    await db.rawUpdate(
        'UPDATE systempreferences SET savedaddressextra =
"$extraAddress", savedaddressstreet = "$streetAddress", savedaddresscity =
"$cityAddress", savedaddresspostcode = "$postcodeAddress" WHERE profileid
= $selectedID');

}
}

```

```
static Future<void> deletePreference(id) async {

    //Delete preference record associated with specified id

    final db = await SQLiteLocalDB.db();

    try {

        //Try to delete record

        await db

            .delete("systempreferences", where: "profileid = ?", whereArgs:
[id]);

    } catch (err) {

        //If failed

        debugPrint(
            "Something went wrong when deleting an item: $err"); //print
error

    }

}

//-----
// Profile
//-----

// Create new profile

static Future<void> createProfile(
```

```

        String firstname, String lastname, String email, String password)
async {

    //Create profile using firstname, lastname, email and encrypted
password

    final db = await SQLiteLocalDB.db();

    db.insert("localprofiles", {
        //Insert data into localprofiles table
        "firstname": firstname,
        "lastname": lastname,
        "email": email,
        "password": password
    });
}

// Get profiles

static Future<List<Map<String, dynamic>>> getProfiles() async {

    final db = await SQLiteLocalDB.db();

    return db.query('localprofiles', orderBy: "id"); //get local profiles
}

// Get display profiles (Id, firstname, lastname, selected status)

static Future<List> getDisplayProfiles() async {

    List checkSelected = await SQLiteLocalDB.getProfileSelected();

    if (checkSelected.length != 1) {

```

```

    //Check to make sure there is only one profile with the selected tag
is true

    int index = (await SQLiteLocalDB.getFirstProfile())[0][
        "id"]; //Make reset the selected profile to be the first entry
that is currently logged in

    await SQLiteLocalDB.setSelected(index);

}

final db = await SQLiteLocalDB.db();

List<Map> result = await db.rawQuery(
    'SELECT id, firstname, lastname, selected FROM localprofiles ORDER
BY id ASC'); //Get all the profiles logged in, sending id, firstname,
lastname and selected status back

return result;

}

// Set profile to be selected (returns profile)

static Future<List<Map<String, dynamic>>> setProfileSelected(int id)
async {

    //Set selected profile to the one inputted

    final db = await SQLiteLocalDB.db();

    db.execute(
        'UPDATE localprofiles SET selected = "FALSE"'; //Set all profiles
selected status to false

    db.execute(
        'UPDATE localprofiles SET selected = "TRUE" WHERE id = "$id"'; //
set inputted profile id to have selected status to true

```

```
return db.rawQuery(  
  
    'SELECT * FROM localprofiles WHERE selected = "TRUE" LIMIT 1');  
//Get the profile info from the selected profile and send it back  
  
}  
  
  
// Get selected profile  
  
static Future<List> getProfileSelected() async {  
  
    final db = await SQLiteLocalDB.db();  
  
  
  
    List<Map> result = await db  
  
        .rawQuery('SELECT * FROM localprofiles WHERE selected = 1 LIMIT  
1');  
  
    return result; //Return the first profile that has the selected status  
to true (should only be 1)  
  
}  
  
  
  
// Get profile info from id  
  
static Future<List<Map<String, dynamic>>> getProfileFromID(int id) async  
{  
  
    final db = await SQLiteLocalDB.db();  
  
    return db.query('localprofiles',  
  
        where: "id = ?", whereArgs: [id], limit: 1);  
  
}
```

```
// Get profile info from email

static Future<List<Map<String, dynamic>>> getProfileFromEmail(
    String email) async {

    final db = await SQLiteLocalDB.db();

    return db
        .rawQuery('SELECT * FROM localprofiles WHERE email = "$email"
LIMIT 1');

}

// Get first profile in profiles table

static Future<List<Map<String, dynamic>>> getFirstProfile() async {

    final db = await SQLiteLocalDB.db();

    return db.rawQuery('SELECT * FROM localprofiles ORDER BY id ASC LIMIT
1');

}

// Change selected to specific id (does not return)

static Future<void> setSelected(int id) async {

    final db = await SQLiteLocalDB.db();

    db.rawUpdate(
        'UPDATE localprofiles SET selected = false'); //Set all profiles
selected status to false

    db.rawUpdate(

```

```
'UPDATE localprofiles SET selected = true WHERE id = "$id");  
//Set the inputted profile it to be true  
  
}  
  
  
// Update an profile by id  
  
static Future<void> updateProfile(int id, String firstname, String  
lastname,  
String email, String password) async {  
  
    //Update profile using firstname, lastname, email and encrypted  
password  
  
    final db = await SQLiteLocalDB.db();  
  
  
  
    final data = {  
  
        'firstname': firstname,  
  
        'lastname': lastname,  
  
        'email': email,  
  
        'password': password,  
  
    };  
  
  
  
    await db.update('localprofiles', data, where: "id = ?", whereArgs:  
[id]);  
  
}  
  
  
// Delete profile
```

```

static Future<void> deleteProfile(int id) async {

    //Delete profile associated with the id inputted

    final db = await SQLiteLocalDB.db();

    try {

        //Try to delete profile

        await db.delete("localprofiles", where: "id = ?", whereArgs: [id]);

    } catch (err) {

        // If it fails, prints the error to the console

        debugPrint("Something went wrong when deleting an item: $err");

    }

}

}

```

fresh_profile.dart

```

import 'package:alleat/screens/fresh_app/fresh_login.dart';

import 'package:alleat/screens/fresh_app/fresh_newprofile_create.dart';

import 'package:flutter/material.dart';

class FreshProfile extends StatefulWidget {

    const FreshProfile({Key? key}) : super(key: key);

    @override

    State<StatefulWidget> createState() {

```

```
        return _FreshProfile();  
    }  
}  
  
class _FreshProfile extends State<FreshProfile> {  
  
    @override  
  
    Widget build(BuildContext context) {  
  
        return Scaffold( //Create new screen  
  
            resizeToAvoidBottomInset: false, //Allow resize  
  
            appBar: AppBar(  
  
                title: const Text('Welcome to All Eat.'),  
  
                backgroundColor: Theme.of(context).primaryColor,  
            ),  
  
            body: Column(  
  
                mainAxisAlignment: MainAxisAlignment.start,  
  
                children: [  
  
                    Column( //Create 2 buttons which allows you to either create a  
new profile or login with an existing profile  
  
                    children: [  
  
                        ElevatedButton(  
  
                            onPressed: () => (Navigator.push(  
  
                                context,  
  
                                MaterialPageRoute(  

```

```

        builder: (context) => const
FreshNewProfile()))),
            child: const Text("New Profile")),
ElevatedButton(
    onPressed: () => (Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => const FreshLogin()))),
    child: const Text("Login")),
    ],
)
],
),
);
}
}

```

fresh_newprofile_create.dart

```

import 'package:alleat/services/sqlite_service.dart';

import 'package:alleat/widgets/navbar.dart';

import 'package:flutter/services.dart';

import 'package:email_validator/email_validator.dart';

import 'package:http/http.dart' as http;

```

```
import 'package:flutter/material.dart';

import 'dart:convert' as convert;
import 'dart:async';

import 'package:encrypt/encrypt.dart' as encrypt;
import 'package:crypto/crypto.dart' as crypto;
```



```
class FreshNewProfile extends StatelessWidget {

  const FreshNewProfile({super.key});

  @override

  Widget build(BuildContext context) {
    const appTitle = 'Register Profile';

    return MaterialApp(
      //Create new screen
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(appTitle),
        ),
        body: const FreshRegisterProfilePage(),
      ),
    );
  }
}
```

```
        }

    }

// Create a Form widget.

class FreshRegisterProfilePage extends StatefulWidget {

    const FreshRegisterProfilePage({super.key});

    @override

    FreshRegisterProfilePageState createState() {
        return FreshRegisterProfilePageState();
    }
}

class FreshRegisterProfilePageState extends State<FreshRegisterProfilePage> {

    final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

    static final TextEditingController passwordController = TextEditingController();

    static TextEditingController firstname =
        TextEditingController(); //Create text controllers to allow for dynamic variable for form fields

    static TextEditingController lastname = TextEditingController();

    static TextEditingController email = TextEditingController();

    static TextEditingController password = TextEditingController();
}
```

```
dynamic plainText;

static dynamic encryptPassword;

late bool error, sending, success, serverOffline;

late String msg;

String phpurl =
    "https://alleat.cpur.net/query/register.php"; //Default values for
sending data for registering a user

@Override

void initState() {

    error = false;

    sending = false;

    success = false;

    msg = "";

    super.initState();
}

Future<bool> sendData() async {

    //Send Data

    //On submit, start

    plainText = password.text;

    String strkey =
```

```

'ZC8cegCGG45d1IjIACETrfypDXtkgJ1rA+4JABPncUE='; //Static key and
iv

String striv = 'yvhsTTh739b2yUW9NNWrcKmHTtLTZNbjbiV3F/cSRzM=';

var iv = cryptojs.sha256 //Grab the substring of the key and iv

    .convert(converty.utf8.encode(striv))

    .toString()

    .substring(0, 16);

var key = cryptojs.sha256

    .convert(converty.utf8.encode(strkey))

    .toString()

    .substring(0, 16);

encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);

encrypty.Key keyObj = encrypty.Key.fromUtf8(key);

final encrypter = encrypty.Encrypter(encrypty.AES(keyObj,
mode: encrypty.AESMode

    .cbc)); //Use the key and iv to encrypt the plaintext password

encryptPassword = encrypter.encrypt(plainText, iv: ivObj);

encryptPassword = encryptPassword

    .base64; //Set password to be in base64 instead of type encrypted
so that it can be sent

try {

  var res = await http.post(Uri.parse(phiurl), body: {

```

```
//Send data to register.php on server with firstname, lastname,
email and encrypted password

    "firstname": firstname.text,
    "lastname": lastname.text,
    "email": email.text,
    "password": encryptPassword,
  });
//Sending post request with data

if (res.statusCode == 200) {

  var data = converty.json.decode(res.body); //Decode to array

  if (data["error"]) {

    setState(() {
      //refresh the UI when error is received from server
      sending = false;
      error = true;
      msg = data["message"]; //error message from server
    });
    return success = false;
  } else {

    await SQLiteLocalDB.createProfile(
      firstname.text,
      lastname.text,
      email.text,
      data[
```

```
        "hash"]); //On success, create new profile entry on
localprofiles table within the app

List<Map> profileInfoTemp =
    await SQLiteLocalDB.getProfileFromEmail(email.text);
    await SQLiteLocalDB.setSelected(profileInfoTemp[0][
        "id"]); //Get the new registered user and set them to be the
selected profile using their stored id

firstname.text = ""; //Clear the form fields
lastname.text = "";
email.text = "";
passwordController.text = "";
password.text = "";
//clear form

setState(() {
    sending = false;
    success = true; //mark success and refresh UI with setState
    ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
            content: Text(
                'Profile Successfully Created.')), //Show success
    banner at bottom temporarily
```

```
) ;  
  
        Navigator.push(  
  
            context,  
  
            MaterialPageRoute(  
  
                builder: (context) =>  
  
                    const Navigation())); //Go to the homepage  
  
) ;  
  
  
    return success = true;  
  
}  
  
} else {  
  
    //If the app was able to send the data but there was an issue with  
the database  
  
  
    setState(() {  
  
        error = true;  
  
        ScaffoldMessenger.of(context).showSnackBar(  
  
            SnackBar(  
  
                content: Text(  
  
                    "Error ${res.statusCode}: Failed to connect to  
server.")), //Display the error with the generated error code from the  
server  
  
) ;  
}
```

```
        msg = "Connection to database failed.";

        sending = false;

        //mark error and refresh UI with setState

    });

    return success = false;
}

} catch (e) {

    setState(() {

        //If there was an error sending the data causing an app error,
        catch the error

        ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
                content: Text(
                    'FAILED: $e')), //Convert to error message and display
        to the user to prevent crashing

    );

    error = true;

    msg = "Connection to database failed.";

    sending = false;

    //mark error and refresh UI with setState

});

    return success = false;
}

}
```

```
override

Widget build(BuildContext context) {

return Form(
  //Create form
  key: _formKey,
  child: SingleChildScrollView(
    //Create scrollable page
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Row(children: [
          // ----- Firstname
          Expanded(
            child: ListTile(
              title: TextFormField(
                controller:
                  TextEditingController(),
                keyboardType: TextInputType.name,
                decoration: (const InputDecoration(
                  labelText: 'First Name',
                )),
              ),
            ),
          ),
        ],
      ),
    ),
  ),
);
```

```
        icon: Padding(  
  
            padding: EdgeInsets.only(top: 15.0),  
  
            child: Icon(Icons.person)),  
  
) ,  
  
inputFormatters: [  
  
    //Only allows the input of letters a-z and A-Z and .  
and -  
  
    FilteringTextInputFormatter.allow(RegExp('[a-zA-Z.-] '))  
  
,  
  
validator: (forename) {  
  
    //Required field with a minimum of 2 characters.  
Cannot be over 50 to prevent display issues  
  
    if (forename == null || forename.isEmpty) {  
  
        return "Required";  
  
    }  
  
    if (forename.length > 50) {  
  
        return "Name too long";  
  
    }  
  
    if (forename.length < 2) {  
  
        return "Name must be a minimum of 2 characters";  
  
    }  
  
    return null;  
  
) ,
```

```

) ) ,  

)  

// ----- Lastname  

Expanded(  

    child: ListTile(  

        title: TextFormField(  

            controller:  

                lastname, //Form data lastname collected and sent to  

database  

            keyboardType: TextInputType.name,  

            decoration: (const InputDecoration(  

                labelText: 'Surname',  

) ) ,  

            inputFormatters: [  

                //Only allows the input of letters a-z and A-Z and . and  

-  

                FilteringTextInputFormatter.allow(RegExp('^[a-zA-Z.-]'))  

] ,  

            validator: (surname) {  

                //Required field with a minimum of 2 characters. Cannot  

be over 50 to prevent display issues  

                if (surname == null || surname.isEmpty) {  


```

```
        return "Required";

    }

    if (surname.length > 50) {

        return "Surname too long";

    }

    return null;

} ,

) ) )

] ) ,



// ----- Email


ListTile(
    title: TextFormField(
        controller:
            email, //Form data lastname collected and sent to
database

        keyboardType: TextInputType.emailAddress,
        decoration: (const InputDecoration(
            labelText: 'Email',
            icon: Padding(
                padding: EdgeInsets.only(top: 15.0),
                child: Icon(Icons.email)))),

```



```
        }

        if (password.length > 99) {
            return "Password must be under 99 characters";
        }

        if (!password.contains(RegExp(r'[0-9]'))) ||
            !password.contains(RegExp(r'[a-z]')))) {
            return "Password must contain at least 1 number and a
letter";
        }

        return null;
    } ,
),
),
),

// ----- Confirm Password

ListTile(
    title: TextFormField(
        keyboardType: TextInputType.visiblePassword,
        autofillHints: const [AutofillHints.newPassword],
        decoration: (const InputDecoration(
            labelText: 'Confirm Password',
            icon: Padding(

```

```
padding: EdgeInsets.only(top: 15.0),  
  
        child: Icon(Icons.lock))),  
  
    inputFormatters: [  
  
        FilteringTextInputFormatter.allow(  
  
            RegExp('^[a-zA-Z0-9!@#%^&*(),.?:{ }|<>]'))  
  
    ],  
  
    obscureText: true, //Password not visible  
  
    controller: password,  
  
    validator: (confirmPassword) {  
  
        //Checked to make sure that the password is the same as  
        the other password. Has to follow password rules  
  
        if (confirmPassword == null || confirmPassword.isEmpty)  
  
        {  
  
            return "Required";  
  
        }  
  
        if (confirmPassword != passwordController.text) {  
  
            return "Passwords not the same";  
  
        }  
  
        return null;  
  
    },  
  
),  
  
) ,  
  
) ,  
  
const SizedBox(height: 50), //Gap
```

```
// ----- Submit button

Center(
    child: Container(
        height: 70,
        width: 200,
        padding: const EdgeInsets.symmetric(vertical: 16.0),
        child: ElevatedButton(
            //submit button
            style: ButtonStyle(
                shape:
MaterialStateProperty.all<RoundedRectangleBorder>(
                    RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10.0),
                    )),
            onPressed: () {
                if (_formKey.currentState!.validate()) {
                    //If fields have no errors and is validated
                    ScaffoldMessenger.of(context).showSnackBar(
                        const SnackBar(
                            content: Text(
                                'Creating profile...')), //Display creating
                    profile and start Future to send data to the server
                }
            },
        ),
    ),
)
```

```
) ;  
  
    setState( () {  
  
        //Change status to sending to verify  
  
        sending = true;  
  
    }) ;  
  
    sendData(); //run  
  
}  
  
} ,  
  
child: const Text('Create Profile') ,  
) ,  
) )  
  
] ,  
  
) ,  
  
) ;  
  
}  
}
```

fresh_login.dart

```
import 'package:alleat/widgets/navbar.dart';  
  
import 'package:flutter/material.dart';  
  
import 'package:email_validator/email_validator.dart';  
  
import 'package:alleat/services/sqlite_service.dart';
```

```
import 'package:flutter/services.dart';

import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

import 'dart:async';

import 'package:encrypt/encrypt.dart' as encrypt;

import 'package:crypto/crypto.dart' as crypto;
```



```
class FreshLogin extends StatelessWidget {

  const FreshLogin({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    const appTitle = 'Login to Profile';

    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(appTitle),
        ),
        body: const FreshLoginPage(),
      ),
    );
}
```

```
        }

    }

}

class FreshLoginPage extends StatefulWidget {

    const FreshLoginPage({Key? key}) : super(key: key);

    @override

    State<FreshLoginPage> createState() => _FreshLoginPageState();
}

class _FreshLoginPageState extends State<FreshLoginPage> {

    final _formKey = GlobalKey<FormState>();

    static TextEditingController email =
        TextEditingController(); //Create text controllers to allow for
    dynamic variable for form fields

    static TextEditingController password = TextEditingController();

    dynamic plainText;

    static dynamic encryptPassword;

    late bool error, sending, success, serverOffline;

    late String msg;

    late dynamic profileInfoImport;
```

```

String phpurl =
    "https://alleat.cpur.net/query/login.php"; //Default values for
sending data for logging in a user

@Override
void initState() {
    error = false;
    sending = false;
    success = false;
    msg = "";
    super.initState();
}

Future<bool> loginUser() async {
    //send Data
    //On submit, start
    plainText = password.text;
    String strkey =
        'ZC8cegCGG45d1IjIACTrfypDXtkgJ1rA+4JABPncUE='; //Static key and
    iv
    String striv = 'yvhsTTh739b2yUW9NNWrcKmHTtLTZNbjbiV3F/cSRzM=';
    var iv = cryptojs.sha256 //Grab the substring of the key and iv
        .convert(cryptojs.utf8.encode(striv))
        .toString()
        .substring(0, 16);
}

```

```

var key = crypto.sha256

    .convert(converty.utf8.encode(strkey))

    .toString()

    .substring(0, 16);

encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);

encrypty.Key keyObj = encrypty.Key.fromUtf8(key);

final encrypter =

    encrypty.Encrypter(encrypty.AES(keyObj, mode:
encrypty.AESMode.cbc));

encryptPassword = encrypter.encrypt(plainText,

    iv: ivObj); //Use the key and iv to encrypt the plaintext password

encryptPassword = encryptPassword

    .base64; //Set password to be in base64 instead of type encrypted
so that it can be sent

try {

    var res = await http.post(Uri.parse(phiurl), body: {

        //Send data to login.php on server with email and encrypted
password

        "email": email.text,

        "password": encryptPassword,

    }); //Sending post request with data

}

if (res.statusCode == 200) {

    var data = converty.json.decode(res.body); //Decode to array
}

```

```

    if (data["error"]) {
        setState(() {
            //refresh the UI when error is received from server
            sending = false;
            error = true;
            msg = data["message"]; //error message from server
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text("Error: ${data["message"]}")),
            );
        });
    }

    return success = false;
} else {
    if (data["exists"]) {
        profileInfoImport = data["profile"];
        email.text = "";
        password.text = "";
        //clear form
        String profileInfoImportPassword =
        profileInfoImport[3].toString();
    }
}

await SQLiteLocalDB.createProfile(
    profileInfoImport[0],

```

```

        profileInfoImport[1],
        profileInfoImport[2],
        profileInfoImportPassword);
    }

    List<Map> profileInfoTemp =
        await
    SQLiteLocalDB.getProfileFromEmail(profileInfoImport[2]);
    await SQLiteLocalDB.setSelected(profileInfoTemp[0]["id"]);

}

setState(() {
    sending = false;
    success = true; //mark success and refresh UI with setState
    ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Successfully logged in.')),
    );
    Navigator.push(context,
        MaterialPageRoute(builder: (context) => const
Navigation()));
});

}

return success = true;
} else {
    setState(() {
        sending = false;
        success = false;
    });
}

```

```
ScaffoldMessenger.of(context).showSnackBar(  
  
    const SnackBar(content: Text('Incorrect email or  
password')),  
  
);  
  
});  
  
return success = false;  
  
}  
  
}  
  
}  
  
} else {  
  
//there is error  
  
  
  
  
setState(() {  
  
error = true;  
  
ScaffoldMessenger.of(context).showSnackBar(  
  
SnackBar(  
  
content: Text(  
  
"Error ${res.statusCode}: Failed to connect to  
server.")),  
  
);  
  
msg = "Connection to database failed."  
  
sending = false;  
  
//mark error and refresh UI with setState  
  
});  
  
return success = false;
```

```
        }

    } catch (e) {
        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text('FAILED: $e')),
            );
            error = true;
            msg = "Connection to database failed.";
            sending = false;
            //mark error and refresh UI with setState
        });
        return success = false;
    }
}

@Override
Widget build(BuildContext context) {
    return Form(
        key: _formKey,
        child: SingleChildScrollView(
            child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [

```

```
ListTile(  
    title: TextFormField(  
        controller:  
            email, //Form data lastname collected and sent to  
database  
        keyboardType: TextInputType.emailAddress,  
        decoration: (const InputDecoration(  
            labelText: 'Email',  
            icon: Padding(  
                padding: EdgeInsets.only(top: 15.0),  
                child: Icon(Icons.email))),  
        inputFormatters: [  
            //Only allows the input of letters a-z and A-Z and @,.-  
            FilteringTextInputFormatter.allow(RegExp('[a-zA-Z0-9@,. -] '))  
        ],  
        validator: (email) {  
            //Required field and uses emailvalidator package to verify  
it is an email to simplify the code  
            if (email == null || email.isEmpty) {  
                return "Required";  
            }  
            if (EmailValidator.validate(email) == false) {  
                return "Please enter valid email";  
            }  
        },  
    ),  
);
```

```
        }

        return null;

    } ,

) ) ,



ListTile(


    title: TextFormField(


        keyboardType: TextInputType.visiblePassword,


        decoration: (const InputDecoration(


            labelText: 'Password',


            icon: Padding(


                padding: EdgeInsets.only(top: 15.0),


                child: Icon(Icons.lock))),


            inputFormatters: [


                //Password cannot use " or ' in order to prevent SQL
                injection


                FilteringTextInputFormatter.allow(


                    RegExp(' [a-zA-Z0-9!@#%^&*(),.?:{ }|<>] '))



            ] ,


            obscureText: true, //Password not visible


            controller:


                password, //Password copied and checked by confirm
                password


            validator: (password) {
```

```

        //Must be a minimum of 8 characters and contain a letter
and number to make sure there is variety and make it harder to guess. Must
be under 99 characters so that it reduces processing time on the system

    if (password == null ||
        password.isEmpty ||
        password.length < 8 ||
        password.length > 99 ||
        !password.contains(RegExp(r'[0-9]')))) ||
        !password.contains(RegExp(r'[a-z]')))) {
            return "Required";
        }
        return null;
    },
),
),
),
const SizedBox(height: 50), //Gap
Center(
    child: Container(
        height: 70,
        width: 200,
        padding: const EdgeInsets.symmetric(vertical: 16.0),
        child: ElevatedButton(
            //submit button
            style: ButtonStyle(

```

```

        shape:
MaterialStateProperty.all<RoundedRectangleBorder>(
    RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(10.0),
    ))),
onPressed: () {
    if (_formKey.currentState!.validate()) {
        loginUser();
    }
},
child: const Text('Login to Profile'),
),
),
],
),
),
);
}
}

```

addprofile.dart

```

import 'package:alleat/screens/add_profile/addprofile_create.dart';

import 'package:alleat/screens/add_profile/addprofile_login.dart';

```

```
import 'package:flutter/material.dart';

class AddProfile extends StatefulWidget {

  const AddProfile({Key? key}) : super(key: key);

  @override
  State<StatefulWidget> createState() {
    return _AddProfile();
  }
}

class _AddProfile extends State<AddProfile> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Add New Profile.',
      //Create new screen
      home: Scaffold(
        resizeToAvoidBottomInset: false, //Allow resize
        appBar: AppBar(
          title: const Text('Add New Profile'),
          leading: IconButton(
            icon: const Icon(Icons.arrow_back),

```

```
        onPressed: () => Navigator.of(context).pop(),
        backgroundColor: Theme.of(context).primaryColor,
    ),
    body: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
            Column(
                //Create 2 buttons which allows you to either create a new
                profile or login with an existing profile
                children: [
                    ElevatedButton(
                        onPressed: () => (Navigator.push(
                            context,
                            MaterialPageRoute(
                                builder: (context) => const
AddProfileCreate()))),
                        child: const Text("New Profile")),
                    ElevatedButton(
                        onPressed: () => (Navigator.push(
                            context,
                            MaterialPageRoute(
                                builder: (context) => const
AddProfileLogin()))),
                        child: const Text("Login")),
                ],
            ),
        ],
    ),
);
```

```
    ] ,  
    )  
    ] ,  
    ),  
)) ;  
}  
}
```

addprofile_newprofile_create.dart

```
import 'package:alleat/services/sqlite_service.dart';  
  
import 'package:alleat/widgets/navbar.dart';  
  
import 'package:flutter/services.dart';  
  
import 'package:email_validator/email_validator.dart';  
  
import 'package:http/http.dart' as http;  
  
import 'package:flutter/material.dart';  
  
import 'dart:convert' as convert;  
  
import 'dart:async';  
  
import 'package:encrypt/encrypt.dart' as encrypt;  
  
import 'package:crypto/crypto.dart' as crypto;
```



```
class AddProfileCreate extends StatelessWidget {  
  
  const AddProfileCreate({super.key});  
  
  final TextEditingController _nameController = TextEditingController();  
  final TextEditingController _emailController = TextEditingController();  
  final TextEditingController _passwordController = TextEditingController();  
  final TextEditingController _confirmPasswordController = TextEditingController();  
  
  final FocusNode _nameFocusNode = FocusNode();  
  final FocusNode _emailFocusNode = FocusNode();  
  final FocusNode _passwordFocusNode = FocusNode();  
  final FocusNode _confirmPasswordFocusNode = FocusNode();  
  
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();  
  
  void _submitForm() {  
    if (_formKey.currentState!.validate()) {  
      _formKey.currentState!.save();  
      _createProfile();  
    }  
  }  
  
  Future _createProfile() async {  
    final String encryptedPassword = encrypt.PBKDF2.withargon2().  
      compute(_passwordController.text);  
    final String encryptedConfirmPassword = encrypt.PBKDF2.withargon2().  
      compute(_confirmPasswordController.text);  
  
    final Map profileData = {  
      'name': _nameController.text,  
      'email': _emailController.text,  
      'password': encryptedPassword,  
      'confirm_password': encryptedConfirmPassword,  
    };  
  
    final response = await http.post(  
      Uri.parse('https://api.alleat.com/profiles'),  
      headers: {'Content-Type': 'application/json'},  
      body: json.encode(profileData),  
    );  
  
    if (response.statusCode == 201) {  
      Navigator.pop(context);  
    } else {  
      print(response.body);  
    }  
  }  
}
```

```
@override

Widget build(BuildContext context) {
  const appTitle = 'Register Profile';

  return Scaffold(
    appBar: AppBar(
      title: const Text(appTitle),
      leading: IconButton(
        icon: const Icon(Icons.arrow_back),
        onPressed: () => Navigator.of(context).pop(),
      ),
      body: const AddProfileRegisterPage(),
    );
}

}

// Create a Form widget.

class AddProfileRegisterPage extends StatefulWidget {
  const AddProfileRegisterPage({super.key});

  @override
  AddProfileRegisterPageState createState() {
    return AddProfileRegisterPageState();
  }
}
```

```
        }

    }

}

class AddProfileRegisterPageState extends State<AddProfileRegisterPage> {

    final _formKey = GlobalKey<FormState>();

    static final passwordController = TextEditingController();

    static TextEditingController firstname =
        TextEditingController(); //Create text controllers to allow for
    dynamic variable for form fields

    static TextEditingController lastname = TextEditingController();

    static TextEditingController email = TextEditingController();

    static TextEditingController password = TextEditingController();

    dynamic plainText;

    static dynamic encryptPassword;

    late bool error, sending, success, serverOffline;

    late String msg;

    String phpurl =
        "https://alleat.cpur.net/query/register.php"; //Default values for
    sending data for registering a user

    @override

    void initState() {
```

```

        error = false;
        sending = false;
        success = false;
        msg = "";
        super.initState();
    }

Future<bool> sendData() async {
    //On submit, start
    plainText = password.text;
    String strkey =
        'ZC8cegCGG45d1IjIACEtrfypDXtkgJ1rA+4JABPncUE='; //Static key and
    iv
    String striv = 'yvhsTTh739b2yUW9NNWrcKmHTtLTZNbJbiV3F/cSRzM=';
    var iv = crypto.sha256 //Grab the substring of the key and iv
        .convert(convertUtf8.encode(striv))
        .toString()
        .substring(0, 16);
    var key = crypto.sha256
        .convert(convertUtf8.encode(strkey))
        .toString()
        .substring(0, 16);

    encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);
}

```

```
encyrpyt.Key keyObj = encrpyt.Key.fromUtf8(key);

final encrypter = encrpyt.Encrypter(encrpyt.AES(keyObj,
    mode: encrpyt.AESMode
        .cbc)); //Use the key and iv to encrypt the plaintext password

encryptPassword = encrypter.encrypt(plainText, iv: ivObj);

encryptPassword = encryptPassword
    .base64; //Set password to be in base64 instead of type encrypted
so that it can be sent

try {
    var res = await http.post(Uri.parse(phurl), body: {
        //Send data to register.php on server with firstname, lastname,
        email and encrypted password
        "firstname": firstname.text,
        "lastname": lastname.text,
        "email": email.text,
        "password": encryptPassword,
    });
    //Sending post request with data
    if (res.statusCode == 200) {
        var data = convert.json.decode(res.body); //Decode to array
        if (data["error"]) {
            setState(() {
                //refresh the UI when error is received from server
                sending = false;
            });
        }
    }
}
```

```
        error = true;

        msg = data["message"]; //error message from server

    } );

    return success = false;
} else {

    await SQLiteLocalDB.createProfile(
        firstname.text, lastname.text, email.text, data["hash"]);

    List<Map> profileInfoTemp =
        await SQLiteLocalDB.getProfileFromEmail(email.text);

    await SQLiteLocalDB.setSelected(profileInfoTemp[0]["id"]);

    firstname.text = ""; //Clear the form fields

    lastname.text = "";

    email.text = "";

    passwordController.text = "";

    password.text = "";

    //clear form

}

setState(() {
    sending = false;

    success = true; //mark success and refresh UI with setState

    ScaffoldMessenger.of(context).showSnackBar(

```

```

        const SnackBar(content: Text('Profile Successfully
Created.')),

    ) ;




        Navigator.push(context,
            MaterialPageRoute(builder: (context) => const
Navigation())));
    } ) ;




return success = true;

}

} else {

//there is error


setState( () {
    error = true;
    ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
            content: Text('Failed to connect to server. Please try
again')),

    );
    msg = "Connection to database failed.";
    sending = false;
    //mark error and refresh UI with setState
}

```

```
    });

    return success = false;
}

} catch (e) {
    setState(() {
        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
                content: Text('Failed to connect to server. Please try again')),
        );
        error = true;
        msg = "Connection to database failed.";
        sending = false;
        //mark error and refresh UI with setState
    });
    return success = false;
}
}

@Override
Widget build(BuildContext context) {
    return Form(
        //Create form
```

```
key: _formKey,  
  
child: SingleChildScrollView(  
  
    //Create scrollable page  
  
    child: Column(  
  
        crossAxisAlignment: CrossAxisAlignment.start,  
  
        children: [  
  
            Row(children: [  
  
                Expanded(  
  
                    child: ListTile(  
  
                        title: TextFormField(  
  
                            controller:  
  
                                firstname, //Form data firstname collected and sent  
to database  
  
                            keyboardType: TextInputType.name,  
  
                            decoration: (const InputDecoration(  
  
                                labelText: 'First Name',  
  
                                icon: Padding(  
  
                                    padding: EdgeInsets.only(top: 15.0),  
  
                                    child: Icon(Icons.person)),  
  
                            )),  
  
                            inputFormatters: [  
  
                                //Only allows the input of letters a-z and A-Z and .  
and -
```

```
FilteringTextInputFormatter.allow(RegExp('[a-zA-Z.-] '))  
],  
  
validator: (forename) {  
  
    //Required field with a minimum of 2 characters.  
    Cannot be over 50 to prevent display issues  
  
    if (forename == null || forename.isEmpty) {  
  
        return "Required";  
  
    }  
  
    if (forename.length > 50) {  
  
        return "Name too long";  
  
    }  
  
    if (forename.length < 2) {  
  
        return "Name must be a minimum of 2 characters";  
  
    }  
  
    return null;  
},  
),  
) ,  
Expanded(  
child: ListTile(  
title: TextFormField(  
controller:
```

```
        lastname, //Form data lastname collected and sent to
database

        keyboardType: TextInputType.name,
decoration: (const InputDecoration(
labelText: 'Surname',
)),
inputFormatters: [
//Only allows the input of letters a-z and A-Z and . and
-
-
FilteringTextInputFormatter.allow(RegExp('[a-zA-Z.-]'))
],
validator: (surname) {
//Required field with a minimum of 2 characters. Cannot
be over 50 to prevent display issues
if (surname == null || surname.isEmpty) {
return "Required";
}
if (surname.length > 50) {
return "Surname too long";
}
return null;
},
),
),
]
),
```

```
ListTile(  
    title: TextFormField(  
        controller:  
            email, //Form data lastname collected and sent to  
database  
        keyboardType: TextInputType.emailAddress,  
        decoration: (const InputDecoration(  
            labelText: 'Email',  
            icon: Padding(  
                padding: EdgeInsets.only(top: 15.0),  
                child: Icon(Icons.email))),  
        inputFormatters: [  
            //Only allows the input of letters a-z and A-Z and @,.-  
            FilteringTextInputFormatter.allow(RegExp('[a-zA-Z0-9@,. -] '))  
        ],  
        validator: (email) {  
            //Required field and uses emailvalidator package to verify  
it is an email to simplify the code  
            if (email == null || email.isEmpty) {  
                return "Required";  
            }  
            if (EmailValidator.validate(email) == false) {  
                return "Please enter valid email";  
            }  
        },  
    ),  
);
```

```

        }

        return null;

    } ,
) ) ,  

ListTile(
    title: TextFormField(
        keyboardType: TextInputType.visiblePassword,
        autofillHints: const [AutofillHints.newPassword],
        decoration: (const InputDecoration(
            labelText: 'Password',
            icon: Padding(
                padding: EdgeInsets.only(top: 15.0),
                child: Icon(Icons.lock))),
        inputFormatters: [
            //Password cannnot use " or ' in order to prevent SQL
            injection
            FilteringTextInputFormatter.allow(
                RegExp('[a-zA-Z0-9!@#%^&*(),.?:{ }|<>]'))
        ],
        obscureText: true, //Password not visible
        controller:
            passwordController, //Password copied and checked by
        confirm password
        validator: (password) {

```

```

        //Must be a minimum of 8 characters and contain a letter
and number to make sure there is variety and make it harder to guess. Must
be under 99 characters so that it reduces processing time on the system

    if (password == null || password.isEmpty) {

        return "Required";

    }

    if (password.length < 7) {

        return "Password under 7 characters";

    }

    if (password.length > 99) {

        return "Password must be under 99 characters";

    }

    if (!password.contains(RegExp(r'[0-9]')) ||

        !password.contains(RegExp(r'[a-z]')))) {

        return "Password must contain at least 1 number and a
letter";

    }

    return null;

} ,


) ,


) ,


ListTile(

```

title: TextFormField(

```

        keyboardType: TextInputType.visiblePassword,

```

```
    autofillHints: const [AutofillHints newPassword],  
  
    decoration: (const InputDecoration(  
  
        labelText: 'Confirm Password',  
  
        icon: Padding(  
  
            padding: EdgeInsets.only(top: 15.0),  
  
            child: Icon(Icons.lock))),  
  
    inputFormatters: [  
  
        FilteringTextInputFormatter.allow(  
  
            RegExp('[a-zA-Z0-9!@#%^&*(),.?:{}}|<>]'))  
    ],  
  
    obscureText: true, //Password not visible  
  
    controller: password,  
  
    validator: (confirmPassword) {  
  
        //Checked to make sure that the password is the same as  
        the other password. Has to follow password rules  
  
        if (confirmPassword == null || confirmPassword.isEmpty)  
        {  
  
            return "Required";  
  
        }  
  
        if (confirmPassword != passwordController.text) {  
  
            return "Passwords not the same";  
  
        }  
  
        return null;  
    },
```

```

) ,
) ,
const SizedBox(height: 50), //Gap
Center(
  child: Container(
    height: 70,
    width: 200,
    padding: const EdgeInsets.symmetric(vertical: 16.0),
    child: ElevatedButton(
      //submit button
      style: ButtonStyle(
        shape:
MaterialStateProperty.all<RoundedRectangleBorder>(
        RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10.0),
        )),
      onPressed: () {
        if (_formKey.currentState!.validate()) {
          //If fields have no errors
          ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Creating
profile...')),
        );
        setState(() {

```

```

        //Change status to sending to verify

        sending = true;

    } ) ;

    sendData() ; //run

}

} ,



child: const Text('Create Profile') ,

) ,



) )

] ,



) ,



) ,



) ;

}

}

```

addprofile_login.dart

```

import 'package:alleat/widgets/navbar.dart';

import 'package:flutter/material.dart';

import 'package:email_validator/email_validator.dart';

import 'package:alleat/services/sqlite_service.dart';

import 'package:flutter/services.dart';

```

```
import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

import 'dart:async';

import 'package:encrypt/encrypt.dart' as encrypt;

import 'package:crypto/crypto.dart' as crypto;
```



```
class AddProfileLogin extends StatelessWidget {

  const AddProfileLogin({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {
    const appTitle = 'Login to Profile';

    return Scaffold(
      appBar: AppBar(
        title: const Text(appTitle),
        leading: IconButton(
          icon: const Icon(Icons.arrow_back),
          onPressed: () => Navigator.of(context).pop(),
        ),
        body: const AddProfileLoginPage(),
      );
}
```

```
}

class AddProfileLoginPage extends StatefulWidget {

  const AddProfileLoginPage({Key? key}) : super(key: key);

  @override

  State<AddProfileLoginPage> createState() => _AddProfileLoginPageState();
}

class _AddProfileLoginPageState extends State<AddProfileLoginPage> {

  final _formKey = GlobalKey<FormState>();

  static TextEditingController email = TextEditingController(); //Create
text controllers to allow for dynamic variable for form fields

  static TextEditingController password = TextEditingController();

  dynamic plainText;

  static dynamic encryptPassword;

  late bool error, sending, success, serverOffline;

  late String msg;

  late dynamic profileInfoImport;

  String phpurl = "https://alleat.cpur.net/query/login.php"; //Default
values for sending data for logging in a user

  @override
```

```

void initState() {

    error = false;

    sending = false;

    success = false;

    msg = "";

    super.initState();

}

Future<bool> loginUser() async {

    //send Data

    //On submit, start

    plainText = password.text;

    String strkey = 'ZC8cegCGG45d1IjIACEtrfypDXtkgJ1rA+4JABPncUE=';
    //Static key and iv

    String striv = 'yvhsTTh739b2yUW9NNWrcKmHTtLTZNbjbiV3F/cSRzM=';

    var iv = crypto.sha256 //Grab the substring of the key and iv

        .convert(utf8.encode(striv))

        .toString()

        .substring(0, 16);

    var key = crypto.sha256

        .convert(utf8.encode(strkey))

        .toString()

        .substring(0, 16);
}

```

```
encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);

encrypty.Key keyObj = encrypty.Key.fromUtf8(key);

final encrypter =

    encrypty.Encrypter(encrypty.AES(keyObj, mode:
encrypty.AESMode.cbc));

    encryptPassword = encrypter.encrypt(plainText, iv: ivObj); //Use the
key and iv to encrypt the plaintext password

    encryptPassword = encryptPassword.base64; //Set password to be in
base64 instead of type encrypted so that it can be sent

try {

var res = await http.post(Uri.parse(phiurl), body: {

    //Send data to login.php on server with email and encrypted password

    "email": email.text,

    "password": encryptPassword,


}); //Sending post request with data

if (res.statusCode == 200) {

    var data = converty.json.decode(res.body); //Decode to array

    if (data["error"]) {

        setState(() {

            //refresh the UI when error is received from server

            sending = false;

            error = true;

            msg = data["message"]; //error message from server
        });
    }
}
}
```

```

    });

    return success = false;
} else {

    if (data["exists"]) {

        profileInfoImport = data["profile"];

        email.text = "";

        password.text = "";

        //clear form

        String profileInfoImportPassword =
profileInfoImport[3].toString();

        await SQLiteLocalDB.createProfile(
            profileInfoImport[0],
            profileInfoImport[1],
            profileInfoImport[2],
            profileInfoImportPassword);

        List<Map> profileInfoTemp =
            await
SQLiteLocalDB.getProfileFromEmail(profileInfoImport[2]);

        await SQLiteLocalDB.setSelected(profileInfoTemp[0]["id"]);

    }

    setState(() {
        sending = false;

        success = true; //mark success and refresh UI with setState
    });
}

```

```

        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Successfully logged in.')),
        );
    }

    Navigator.push(context,
        MaterialPageRoute(builder: (context) => const
Navigation()));
}

}

return success = true;
} else {
    setState(() {
        sending = false;
        success = false;
        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Incorrect email or
password'))),
        );
    });
}

return success = false;
}

}

} else {
//there is error
setState(() {

```

```

        error = true;

        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
                content: Text('Failed to connect to server. Please try
again')),

        );
    }

    msg = "Connection to database failed.";

    sending = false;

    //mark error and refresh UI with setState
} );

return success = false;
}

} catch (e) {
}

setState(() {
    ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
            content: Text('Failed to connect to server. Please try
again')),

    );
    error = true;
    msg = "Connection to database failed.";
    sending = false;

    //mark error and refresh UI with setState
}

```

```
    } );

    return success = false;
}

}

@Override
Widget build(BuildContext context) {
    return Form(
        key: _formKey,
        child: SingleChildScrollView(
            child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                    ListTile(
                        title: TextFormField(
                            controller:
                                TextEditingController(text: email), //Form data last name collected and sent to database
                            keyboardType: TextInputType.emailAddress,
                            decoration: (const InputDecoration(
                                labelText: 'Email',
                                icon: Padding(
                                    padding: EdgeInsets.only(top: 15.0),

```

```
        child: Icon(Icons.email))),

      inputFormatters: [
        //Only allows the input of letters a-z and A-Z and @,.-

        FilteringTextInputFormatter.allow(RegExp('^[a-zA-Z0-9@,.+-]*'))
      ],
      validator: (email) {
        //Required field and uses emailvalidator package to verify
        it is an email to simplify the code

        if (email == null || email.isEmpty) {
          return "Required";
        }

        if (EmailValidator.validate(email) == false) {
          return "Please enter valid email";
        }

        return null;
      },
    ),
  ),
  ListTile(
    title: TextFormField(
      keyboardType: TextInputType.visiblePassword,
      decoration: (const InputDecoration(
        labelText: 'Password',
        icon: Padding(

```

```
padding: EdgeInsets.only(top: 15.0),  
  
child: Icon(Icons.lock))),  
  
inputFormatters: [  
  
    //Password cannot use " or ' in order to prevent SQL  
injection  
  
    FilteringTextInputFormatter.allow(  
  
        RegExp('[a-zA-Z0-9!@#%^&*(),.?:{}}|<>]'))  
],  
  
obscureText: true, //Password not visible  
  
controller:  
  
password, //Password copied and checked by confirm  
password  
  
validator: (password) {  
  
    //Must be a minimum of 8 characters and contain a letter  
and number to make sure there is variety and make it harder to guess. Must  
be under 99 characters so that it reduces processing time on the system  
  
    if (password == null ||  
  
        password.isEmpty ||  
  
        password.length < 8 ||  
  
        password.length > 99 ||  
  
        !password.contains(RegExp(r'[0-9]'))) ||  
  
        !password.contains(RegExp(r'[a-z]')))) {  
  
        return "Required";  
    }  
}
```

```
        return null;

    } ,

) ,

) ,



const SizedBox(height: 50), //Gap

Center(

    child: Container(

height: 70,

width: 200,

padding: const EdgeInsets.symmetric(vertical: 16.0),

child: ElevatedButton(

//submit button

style: ButtonStyle(

    shape:

MaterialStateProperty.all<RoundedRectangleBorder>(

        RoundedRectangleBorder(
borderRadius: BorderRadius.circular(10.0),

) ) ,

onPressed: () {

if (_formKey.currentState!.validate()) {

loginUser();

}

} ,
```

```
        child: const Text('Login to Profile'),  
        ),  
      ),  
    ],  
  ),  
),  
) ;  
}  
}
```

homepage.dart

```
import 'package:flutter/material.dart';

class HomePage extends StatelessWidget {

  const HomePage({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return const Center(
      child: Text(
        'Home',
        style: TextStyle(fontSize: 30),
      ),
    );
  }
}
```

```
) ;  
}  
}
```

foryou.dart

```
import 'package:flutter/material.dart';  
  
class ForYouPage extends StatelessWidget {  
  
  const ForYouPage({Key? key}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) {  
    return const Center(  
      child: Text(  
        'For You',  
        style: TextStyle(fontSize: 30),  
      ),  
    );  
  }  
}
```

browse.dart

```
import 'package:alleat/widgets/locationbutton.dart';

import 'package:alleat/widgets/restaurantlist.dart';

import 'package:flutter/material.dart';

class BrowsePage extends StatefulWidget {

  const BrowsePage({Key? key}) : super(key: key);

  @override
  State<BrowsePage> createState() => _BrowsePageState();
}

class _BrowsePageState extends State<BrowsePage> {

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      //Enable scrollable screen
      child: Column(children: const [
        CurrentLocation(), //Show current location button widget
        RestaurantList(), //Show list of restaurants widget
      ]));
  }
}
```

profiles.dart

```
import 'package:alleat/screens/add_profile/addprofile.dart';

import 'package:alleat/screens/fresh_app/fresh_profile.dart';

import 'package:alleat/services/sqlite_service.dart';

import 'package:flutter/material.dart';

class ProfilePage extends StatefulWidget {

  const ProfilePage({Key? key}) : super(key: key);

  @override

  State<ProfilePage> createState() => _ProfilePageState();
}

class _ProfilePageState extends State<ProfilePage> {

  late dynamic profileInfo;

  bool setup = false;

  Future<List> getProfileInfo() async {
    List profileInfo = await SQLiteLocalDB

      .getDisplayProfiles(); //Get id, firstname, lastname and selected
    status to display

    return profileInfo;
}
```

```

        }

Future<void> selectProfile(index) async {

    await SQLiteLocalDB.setSelected(index); //Get selected profile

    List profileInfo = await SQLiteLocalDB.getDisplayProfiles();

    setState(() {
        profileInfo = profileInfo;
    });
}

Future<void> deleteProfile(index) async {

    await SQLiteLocalDB.deleteProfile(index);

    List profileInfo = await SQLiteLocalDB

        .getDisplayProfiles(); //Get profile info under id selected

    setState(() {
        profileInfo = profileInfo;
    });
}

List<Map> profileInfoCheck = await SQLiteLocalDB

    .getFirstProfile(); //Call Database for the first entry

if (profileInfoCheck.isEmpty) {

    //If the first entry is empty

    setup = false; //Then setup is not complete (pass to build)

    setState(() {

```

```

        Navigator.push(context,
            MaterialPageRoute(builder: (context) => const
FreshProfile())));
    } );
} else {
    index = await SQLiteLocalDB.getFirstProfile(); //Get first profile
    index = index[0]["id"];
    await SQLiteLocalDB.setSelected(
        index); //Set selected to the first profile
    List profileInfoSelect = await SQLiteLocalDB
        .getDisplayProfiles(); //Return list of profiles back
    setState(() {
        profileInfoSelect = profileInfoSelect;
    });
}
}

@Override
Widget build(BuildContext context) {
    getProfileInfo();
    return Scaffold(
        //Create subpage
        body: SizedBox(

```

```
        child: Column(children: <Widget>[

    Container(
        alignment: Alignment.topLeft,
        padding: const EdgeInsets.all(20),
        child: const Text(
            //Saved Profiles Heading
            "Saved Profiles",
            style: TextStyle(fontSize: 20, fontWeight: FontWeight.w700),
        ),
    ),
    Expanded(
        child: FutureBuilder<List>(
            //Future builder remakes the list of profiles when it is updated
            future: getProfileInfo(), //On Future getProfileInfo data update rebuild
            builder: (context, snapshot) {
                List profiles = snapshot.data ?? [];
                //Get data from Future
                return ListView.builder(
                    // Creates a new instance of contianer for each item
                    itemCount: profiles.length,
                    itemBuilder: (context, index) {
                        Map<String, dynamic> profile =
                            profiles[index];
                        //Map profile to profiles index
                    }
                );
            }
        )
    )
]);
```

```
var profileSelected = profile["selected"].toString();

return Center(
    child: LayoutBuilder(builder: (context, constraints) {
try {
    if (profileSelected == "0") {

        //If profile is not selected then

        return Container(
            //Create container with ListTile so that it can
be pressed

            margin: const EdgeInsets.only(
                top: 5, bottom: 5, left: 15, right: 15),
            color: const Color(0xffffffff),
            padding: const EdgeInsets.only(left: 5),
            child: ListTile(
                onTap: () {
                    selectProfile(profiles[index]["id"]);
                },
                title: Text(
                    profile["firstname"] +
                    " " +
                    profile["lastname"],
                    overflow: TextOverflow.ellipsis,
                    style: const TextStyle(

```

```

        color: Colors.black,
        fontWeight: FontWeight.w400,
        fontSize: 18,
    ) ),
trailing: IconButton(
    //Delete button to remove from database
    icon: const Icon(
        Icons.delete_outline,
        color: Color(0xffAF0E17),
    ),
    onPressed: () {
        deleteProfile(profiles[index] [
            "id"]); //Run Future to delete
        profile from local database (signed out)
    },
)) );
} else if (profileSelected == "1") {
    //If profile is selected then
    return Container(
        //Create container with ListTile but without tap
        function since it doesnt do anything
        margin: const EdgeInsets.only(
            top: 5, bottom: 5, left: 15, right: 15),
        color: const Color(0xffffffff),

```

```
padding: const EdgeInsets.only(left: 5),  
  
width: 500,  
  
height: 80,  
  
child: Column(  
  
    //Column includes the normal name and delete  
button but also the SELECTED text to show which one is selected  
  
    children: [  
  
        Expanded(  
  
            child: SizedBox(  
  
                child: ListTile(  
  
                    title: Text(  
  
                        profile["firstname"] +  
  
                        " " +  
  
                        profile["lastname"],  
  
                    overflow:  
TextOverflow.ellipsis,  
  
                    style: const TextStyle(  
  
                        color: Colors.black,  
  
                        fontWeight: FontWeight.w400,  
  
                        fontSize: 18,  
  
                    ),  
  
                    trailing: IconButton(  
  
                        icon: const Icon(  
  
                            Icons.delete_outline,
```

```
        color: Color(0xffAF0E17),  
        ),  
        onPressed: () {  
            deleteProfile(  
                profiles[index]["id"]);  
        },  
    )),  
,  
Expanded(  
    child: Container(  
        alignment: Alignment.bottomLeft,  
        padding: const EdgeInsets.all(5),  
        child: const Padding(  
            padding:  
                EdgeInsets.only(bottom: 10,  
left: 10),  
            child: Text(  
                "SELECTED",  
                style: TextStyle(  
                    color: Colors.deepPurple,  
                    fontWeight: FontWeight.w800,  
                    fontSize: 12,  
                ),  
            ),  
        ),  
    ),  
);
```

```

) ) ,  

)  

)  

],  

)) ;  

} else {  

    return Container(  

        //If none show no profiles (fallback. Should go  

        straight to setup page normally)  

        margin: const EdgeInsets.only(  

            top: 5, bottom: 5, left: 15, right: 15),  

            color: const Color(0xffffffff),  

            child: const ListTile(title: Text("No  

Profiles")));  

}  

} catch (e) {  

    //If fails to get profile info, show failed to grab  

profile  

    return Container(  

        margin: const EdgeInsets.only(  

            top: 5, bottom: 5, left: 15, right: 15),  

            color: const Color(0xffffffff),  

            child: const ListTile(  

                title: Text("Failed to grab profile.")));  

}

```

```
        }

    } ) ) ;

}

} ,

) ) ,



Container(



//At the bottom of the page, include a button that allows the user
to add a new profile


padding: const EdgeInsets.all(10),


child: ElevatedButton(


style: ElevatedButton.styleFrom(


minimumSize: const Size.fromHeight(50),


backgroundColor: const Color(0xff5806FF)),


onPressed: () {




Navigator.push(context,


MaterialPageRoute(builder: (context) => const


AddProfile()));



},


child: const Text("Add Profile"),


),


)

] ) ) );


}

}
```

favourites.dart

```
import 'package:alleat/services/sqlite_service.dart';

import 'package:alleat/widgets/navbar.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

class FavouritesPage extends StatefulWidget {

  const FavouritesPage({Key? key}) : super(key: key);

  @override
  State<FavouritesPage> createState() => _FavouritesPageState();
}

class _FavouritesPageState extends State<FavouritesPage> {

  late bool error, sending, success, serverOffline;
  late String msg;

  @override
  void initState() {
    //Default values
    error = false;
```

```

    sending = false;

    success = false;

    msg = "";

    super.initState();

}

Future<List> getFavouriteRestaurants() async {

    String phpurl =

        "https://alleat.cpur.net/query/favouriterestaurantdata.php"; //Get
list of favourites associated with profile email

    var profile = await SQLiteLocalDB.getProfileSelected();

    String email = profile[0]["email"].toString();

    try {

        var res = await http.post(Uri.parse(phpurl), body: {

            "profileemail": email.toString(),
        });

        if (res.statusCode == 200) {

            var data = convert.json.decode(res.body); //Decode to array

            if (data["error"]) {

                //If there is an error, return blank restaurants

                List error = [
                    {"error": "true", "restaurants": "[]"}
                ];
            }
        }
    }
}

```

```
        return error;

    } else {

        //If there is not an error, send the data in a list

        List listdata = [data];

        return listdata;

    }

} else {

    //If there is an error, return blank restaurants

    List error = [

        {"error": "true", "restaurants": "[]"}

    ];

    return error;

}

} catch (e) {

    //If there is an error, return blank restaurants

    List<Map<String, String>> error = [

        {"error": "true", "restaurants": "[]"}

    ];

    return error;

}

}

Future<String> favouriteRestaurant(restaurantID, action) async {
```

```

String phpurl =
    "https://alleat.cpur.net/query/favouriterestaurant.php";
//Favourite/unfavourite a restaurant using email as an identifier

var profile = await SQLiteLocalDB.getProfileSelected();

String email = profile[0]["email"].toString();

try {

    var res = await http.post(Uri.parse(phpurl), body: {

        "action": action.toString(), // Favourite or unfavourite

        "profileemail": email

            .toString(), // Profile's email for identification of who's
profile to modify

        "restaurantid": restaurantID,

    });

    if (res.statusCode == 200) {

        //On successfull send

        var data = convert.json.decode(res.body); //Decode to array

        if (data["error"]) {

            //If error querying

            return "failed";

        } else {

            //If success, request the favourites list

            getFavourites();

            return "success";
    }
}

```

```

        }

    } else {

        return "failed";
    }

} catch (e) {

    return "failed";
}

}

Future<List> getFavourites() async {

    String phpurl =
"https://alleat.cpur.net/query/favouriterestaurantlist.php";

    var profile = await SQLiteLocalDB.getProfileSelected();

    String email = profile[0]["email"].toString();

    try {

        var res = await http.post(Uri.parse(phpurl), body: {

            //Get fav restaurants list associated with profile's email

            "profileemail": email.toString(),
        });
    }

    if (res.statusCode == 200) {

        //On successfull send

        var data = convert.json.decode(res.body); //Decode to array

        if (data["error"]) {
    
```

```
//If error querying

List<Map<String, String>> error = [
    {"error": "true", "favouriterestaurants": "[]"} //Return blank
list

];

return error;

} else {

List listdata = [data];

return listdata;

}

} else {

List<Map<String, String>> error = [
    {"error": "true", "favouriterestaurants": "[]"}
];

return error;

}

} catch (e) {

List<Map<String, String>> error = [
    {"error": "true", "favouriterestaurants": "[]"}
];

return error;

}

}
```

```
override

Widget build(BuildContext context) {

return Scaffold(
    //Create favourites screen

    appBar: AppBar(
        title: const Text('Favourites'),
        leading: IconButton(
            icon: const Icon(Icons.arrow_back), //Back button
            onPressed: () {
                Navigator.of(context).pop();
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => const Navigation())));
            }
        ),
        backgroundColor: Theme.of(context).primaryColor,
    ),
    body: FutureBuilder<List>(
        //Listen for changes to future (favourite restaurants list )

        future: getFavouriteRestaurants(),
        builder: ((context, snapshot) {
            List restaurantsdata =

```

```
snapshot.data ?? [] ; //Save list to restaurant data

if (!snapshot.hasData) {

    //If nothing is received then showing loading bar

    return const LinearProgressIndicator(

        color: Color(0xff4100C4), backgroundColor:

Color(0xffEBE0FF)) ;

} else {

    // If there is data

    try {

        //Try to show list

        List restaurants = restaurantsdata[0]["restaurants"] ;

        return FutureBuilder<List> (

            //Get status of which restaurants is favourited

            future: getFavourites(),

            builder: ((context, snapshot) {

                if (!snapshot.hasData) {

                    //If there is no data, show loading bar

                    return const LinearProgressIndicator(

                        color: Color(0xff4100C4),

                        backgroundColor: Color(0xffEBE0FF)) ;

                } else {

                    List restaurantFavourites = snapshot.data ??

[] ; //Save favourite restaurant list to

restaurantFavourites
```

```
        return ListView.builder(  
  
            //For each restaurant in list of favourites  
            create a container  
  
            physics: const AlwaysScrollableScrollPhysics(),  
  
            scrollDirection: Axis.vertical,  
  
            shrinkWrap: true,  
  
            itemCount:  
restaurantsdata[0]["restaurants"].length,  
  
            itemBuilder: (context, index) {  
  
                return Center(child:  
  
                    LayoutBuilder(builder: (context,  
constraints) {  
  
                        if (restaurants.isEmpty) {  
  
                            //If there is no restaurants in the list  
then show container with text 'No restaurants'  
  
                            return Padding(  
  
                                padding: const EdgeInsets.only(  
  
                                    left: 20,  
  
                                    right: 20,  
  
                                    top: 10,  
  
                                    bottom: 10),  
  
                                child: Container(  
  
                                    decoration: BoxDecoration(  

```

```
borderRadius:  
  
    const BorderRadius.all(  
  
        Radius.circular(20)),  
  
    color: const  
Color(0xffffffff),  
  
    boxShadow: [  
  
        BoxShadow(  
  
            color: Colors.black  
  
.withOpacity(0.1),  
  
            spreadRadius: 2,  
  
            blurRadius: 10,  
  
            offset: const Offset(0,  
  
10), // changes  
position of shadow  
  
) ,  
  
]),  
  
child: Column(children: const [  
  
    Padding(  
  
        padding: EdgeInsets.all(30),  
  
        child: SizedBox(  
  
            width: double.infinity,  
  
            child: Center(  
  
                child: Text(  
)
```

```
        "No restaurants
found" ) ) ,
    )
] ) );
} else {
    //If there is a list of restaurants in
favourites
    return Padding(
        padding: const EdgeInsets.only(
            left: 20,
            right: 20,
            top: 10,
            bottom: 10),
        child: Container(
            decoration: BoxDecoration(
                borderRadius: const
BorderRadius.all(
                    Radius.circular(20)),
                color: const Color(0xffffffff),
                boxShadow: [
                    BoxShadow(
                        color:
Colors.black.withOpacity(0.1),

```

```
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0,  
            10), // changes position  
        of shadow  
  
    ),  
  
] ),  
  
child: Column(children: [  
  
    Padding(  
  
        //  
  
        padding: const  
        EdgeInsets.all(10),  
  
        child: Container(  
  
            //Container with the  
            restaurant image contained in it  
  
            width:  
            MediaQuery.of(context)  
  
                .size  
  
                .width,  
  
            height: 150,  
  
            decoration: BoxDecoration(  
  
                borderRadius:  
  
                const  
                BorderRadius.all(  
                   
```

```
Radius.circular(10)),  
  
        image: DecorationImage(  
  
            fit: BoxFit.fitWidth,  
  
            image: NetworkImage(  
  
restaurants[index][3]  
  
.toString()),  
  
),  
  
),  
  
),  
  
Padding(  
  
padding: const  
EdgeInsets.only(  
  
left: 20,  
  
right: 30,  
  
bottom: 15,  
  
top: 10),  
  
child: Row(  
  
mainAxisAlignment:  
  
MainAxisAlignment  
  
.spaceBetween,  
  
children: [  
  
]
```

```
//Main area of restaurant
container, with logo, name and favourite button icon

Row(
  children: [
    ClipOval(
      //Circular
      restaurant logo
      child:
      Image.network(
        restaurants[index]
        [2]

      .toString(),
      height: 50,
      width: 50)),
      const SizedBox(
        //Empty space
        width: 10,
      ),
      Text(
        //Restaurant Name
        restaurants[index][1],
        style: const
        TextStyle(
```

```
        fontSize: 16,  
        fontWeight:  
  
FontWeight.w500),  
    ),  
],  
,  
IconButton(  
//Favourite icon  
 onPressed: () {  
if  
(restaurantFavourites[  
0]  
  
["restaurantids"]  
  
.contains(restaurants[  
  
index][0]  
  
.toString())) {  
//If the  
restaurant id is in the list of favourite restaurant list  
  
favouriteRestaurant(  
//Send  
unfavourite action to future
```

```
restaurants[index]

[0]

.toString(),

"unfavourite");

setState(() {

getFavourites();

}) ;

} else {

//If the restuarnt

is not on the list of favourite restaurants list

favouriteRestaurant(


//Send the

favourite action to future

restaurants[index]

[0]

.toString(),

"favourite");

setState(() {

getFavourites();

}) ;
```

```
        }

    } , icon:
(restaurantFavourites[

    0] [

"restaurantids"]

.contains(restaurants[

index] [0]

.toString()))

? const Icon(

// ? =
favoured

Icons.favorite,

color: Color(

0xff33C496)

: const Icon(

// : = not
favoured

Icons

.favorite_border_outlined,
```

```
        color: Color(0xff000000))),

    ],
)

]) ,
) ) ;

}

}) ) ;

} ) ;

}

}) ,


) ;

} catch (e) {

//If there is an error, show box container that allows the user to try requesting the data again

return Padding(
padding: const EdgeInsets.only(
left: 20, right: 20, top: 10, bottom: 10),
child: Container(
decoration: BoxDecoration(
borderRadius:
const BorderRadius.all(Radius.circular(20)),

```

```
        color: const Color(0xffffffff),  
  
        boxShadow: [  
  
            BoxShadow(  
  
                color: Colors.black.withOpacity(0.1),  
  
                spreadRadius: 2,  
  
                blurRadius: 10,  
  
                offset: const Offset(  
  
                    0, 10), // changes position of shadow  
  
            ),  
  
        ]),  
  
        child: Column(children: [  
  
            Padding(  
  
                padding: const EdgeInsets.all(30),  
  
                child: SizedBox(  
  
                    width: double.infinity,  
  
                    child: Center(  
  
                        child: Column(children: [  
  
                            const Text("Error 400: Connection  
failed"),  
  
                            ElevatedButton(  
  
                                onPressed: () {  
  
                                    setState(() {  
  
                                        getFavouriteRestaurants();  
                                });  
                            },  
                        ]),  
                    ),  
                ),  
            ),  
        ]),  
    ),  
);
```

apperror.dart

```
import 'package:flutter/material.dart';

class AppError extends StatefulWidget {

  const AppError({super.key});

  @override

  State<AppError> createState() => _AppErrorHandler();
}

class _AppErrorHandler extends State<AppError> {
  late final TextEditingController _controller = TextEditingController();

  void _onSubmit() {
    final String value = _controller.text;
    if (value.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Please enter a value')),
      );
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Value entered: $value')),
      );
    }
  }

  void _onClear() {
    _controller.clear();
  }

  void _onFocusLost() {
    FocusScope.of(context).unfocus();
  }

  void _onFocusGained() {
    FocusScope.of(context).requestFocus(FocusNode());
  }

  void _onChanged(String value) {
    setState(() {
      _controller.text = value;
    });
  }
}
```

```
class _AppErrorHandler extends State<AppErrorHandler> {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(
      title: "Error Page", //Error page
      home: Scaffold(
        resizeToAvoidBottomInset: false,
        body: Center(
          child: Padding(
            padding: const EdgeInsets.all(50),
            child: Column(
              mainAxisAlignment: MainAxisAlignment
                  .center, //Align text to center of the screen
              crossAxisAlignment: CrossAxisAlignment.center,
              children: const [
                Text(
                  "All Eat.",
                  style: TextStyle(
                    fontSize: 21, fontWeight:
FontWeight.w600),
                ),
                Text(

```

```
//Display app failed

        "The app has failed to perform an action.
\nPlease reopen the app to try again.",

        textAlign: TextAlign.center,

    )

] )) ) );
}

}
```

genericload.dart

```
import 'package:flutter/material.dart';

class GenericLoading extends StatefulWidget {

    const GenericLoading({super.key});

    @override
    State<GenericLoading> createState() => _GenericLoadingState();
}

class _GenericLoadingState extends State<GenericLoading> {

    @override
    Widget build(BuildContext context) {

        return MaterialApp(
```

```
title: "Loading",

home: Scaffold(
    resizeToAvoidBottomInset: false,
    body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: const [
            CircularProgressIndicator()
        ])); //Show blank page with circular loading circle

}

}
```

locationbutton.dart

```
import 'package:alleat/services/sqlite_service.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:geocoding/geocoding.dart';

import 'package:geolocator/geolocator.dart' as geo;

class CurrentLocation extends StatefulWidget {

    const CurrentLocation({Key? key}) : super(key: key);

    @override
```

```
    @override

    State<CurrentLocation> createState() => _CurrentLocationState();

}

class _CurrentLocationState extends State<CurrentLocation> {

    Future<String> getSavedStreet() async {
        try {
            // Try to see if there is an address saved
            List address = await SQLiteLocalDB.getAddress(); // Get address

            setState(() {
                // Update the widget if there is a change
                address = address;
            });
        }

        return address[0]["savedaddressstreet"]; // Send back the street
        address
    } catch (e) {
        return "Set Location."; // Send back that no location is saved
    }
}

@Override

Widget build(BuildContext context) {
    return FutureBuilder<String>(

```

```
future: getSavedStreet(), //Get the saved street address

builder: (context, snapshot) {

    var location = snapshot.data ?? [];

    if (!snapshot.hasData) {

        //While getting the street address, add button with loading
bar

    return Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
            Container(
                padding: const EdgeInsets.only(top: 20, bottom: 10),
                child: Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: [
                        Container(
                            padding: const EdgeInsets.only(
                                top: 5, bottom: 7, left: 35, right: 20),
                            decoration: const BoxDecoration(
                                color: Color(0xffEBE0FF),
                                borderRadius:
```

```
BorderRadius.all(Radius.circular(30))),  
        child: Row(  
            children: [  
                const Padding(  
                    padding: EdgeInsets.only(right: 12),  
                    child: SizedBox(  
                        width: 80,  
                        height: 3,  
                        child: LinearProgressIndicator(  
                            color: Color(0xff4100C4),  
                            backgroundColor:  
Color(0xffEBE0FF),  
                        ),  
                    ),  
                ),  
                IconButton(  
                    onPressed: () {  
                        Navigator.push(  
                            context,  
                            MaterialPageRoute(  
                                builder: (context) =>  
                                const  
FromSavedManualLocation())); //Go to the manual location page  
                    },  
                ),  
            ],  
        ),  
    ),  
);
```

```
        } ,  
  
        icon: const Icon(  
  
          Icons.location_on,  
  
          color: Color(0xff4100C4),  
  
        ) )  
  
      ] ,  
  
    ) )  
  
  ] ,  
  
) )  
  
] );  
  
} else {  
  
  // Once the street address is received, output the street  
address instead of the loading bar  
  
  return Column(  
  
    mainAxisAlignment: MainAxisAlignment.start,  
  
    children: [  
  
      Container(  
  
        padding: const EdgeInsets.only(top: 20, bottom: 10),  
  
        child: Row(  
  
          mainAxisAlignment: MainAxisAlignment.center,  
  
          crossAxisAlignment: CrossAxisAlignment.center,  
  
          children: [  
  
            InkWell(  
             
```

```
onTap: () {  
  Navigator.push(  
    context,  
    MaterialPageRoute(  
      builder: (context) =>  
      const  
      FromSavedManualLocation()));  
  
},  
  
child: Container(  
  padding: const EdgeInsets.only(  
    top: 5, bottom: 7, left: 35, right:  
    20),  
  
  decoration: const BoxDecoration(  
    color: Color(0xffEBE0FF),  
    borderRadius: BorderRadius.all(  
      Radius.circular(30))),  
  
  child: Row(  
    children: [  
      Text(  
        location.toString(),  
        style: const TextStyle(  
          fontSize: 14,  
          color: Color(0xff4100C4),  
          fontWeight: FontWeight.w600),  
    ],  
  ),  
),  
),
```

```
        ) ,  
  
        IconButton(  
  
            onPressed: () {  
  
                Navigator.push(  
  
                    context,  
  
                    MaterialPageRoute(  
  
                        builder: (context) =>  
  
                            const  
FromSavedManualLocation())); //Go to the manual location page  
  
            } ,  
  
            icon: const Icon(  
  
                Icons.location_on,  
  
                color: Color(0xff4100C4),  
  
            ) )  
  
        ] ,  
  
    ) ) )  
  
] ,  
  
) )  
  
] ) ;  
  
}  
  
} ) ;  
  
}  
  
}
```

```

class FromSavedManualLocation extends StatefulWidget {

  const FromSavedManualLocation({Key? key}) : super(key: key);

  @override

  State<FromSavedManualLocation> createState() =>

      _FromSavedManualLocationState();
}

class _FromSavedManualLocationState extends State<FromSavedManualLocation>
{

  final _formKey = GlobalKey<FormState>();

  final verifyPostcode = RegExp(
    r"^(Gg|Ii|Rr) ([Aa]{2})|(([A-Za-z][0-9]{1,2})|(([A-Za-z][A-Ha-hJ-Yj-y][0-9]{1,2})|(([A-Za-z][0-9][A-Za-z])|(([A-Za-z][A-Ha-hJ-Yj-y][0-9]?[A-Za-z])))[0-9][A-Za-z]{2})$"); //Postcode formatting

  Future<List> getSavedAddress() async {

    return await SQLiteLocalDB.getAddress(); // get the saved address
  }

  Future<void> saveFilledAddress(String extraAddress, String streetAddress,
    String cityAddress, String postcodeAddress) async {

```

```
try {

    SQLiteLocalDB.setAddress(extraAddress, streetAddress, cityAddress,
        postcodeAddress); // Try to save address

    ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Successfully saved address.')));

} catch (e) {

    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
        content: Text(
            'Failed to save address'))); // If failed to saved, output
failed to save

}

}

@Override

Widget build(BuildContext context) {

    return FutureBuilder<List>(
        future: getSavedAddress(), //Get the saved address to fill the
fields

        builder: (context, snapshot) {

            if (snapshot.connectionState != ConnectionState.done) {

                return const Center(
                    child:

                    CircularProgressIndicator()); //While loading, display
loading page

```

```

        }

        if (snapshot.hasError) {

            return const Center(
                child: Text("Failed")); //If there is a failure, output
Failed

        }

        List savedAddress = snapshot.data ?? []; //Get address in list
form

        if (savedAddress.isEmpty) {

            //If it is an empty list (nothing is saved for profile) output
empty fields in the right format

            savedAddress = [
                {
                    "savedaddressextra": "",
                    "savedaddressstreet": "",
                    "savedaddresscity": "",
                    "savedaddresspostcode": ""
                }
            ];
        }

        final TextEditingController extraAddress =
TextEditingController(
    text: savedAddress[0]

    ["savedaddressextra"]); //For each field, fill with
saved data

```

```
    final TextEditingController streetAddress =
TextEditingController(
    text: savedAddress[0]["savedaddressstreet"]);

    final TextEditingController cityAddress =
TextEditingController(text:
savedAddress[0]["savedaddresscity"]);

    final TextEditingController postcodeAddress =
TextEditingController(
    text: savedAddress[0]["savedaddresspostcode"]);

return Scaffold(
    resizeToAvoidBottomInset: false,
    appBar: AppBar(
        title: const Text(
            'Address Details.'), //Create page for manual fill
        leading: IconButton(
            icon: const Icon(Icons.arrow_back),
            onPressed: () =>
                Navigator.of(context).pop()), //Go back button
        backgroundColor: Theme.of(context).primaryColor,
    ),
    body: Form(
        key: _formKey,
        child: SingleChildScrollView(
            child: Column(
```

```
crossAxisAlignment: CrossAxisAlignment.start,  
  
        children: [  
  
            Align(  
  
                alignment: Alignment.center,  
  
                child: SizedBox(  
  
                    width: 170,  
  
                    height: 40,  
  
                    child: ElevatedButton(  
  
                        // Button for getting current location  
  
                        style: ElevatedButton.styleFrom(  
  
                            minimumSize: const  
Size.fromHeight(50),  
  
                            backgroundColor: const  
Color(0xff5806FF)),  
  
                        onPressed: () {  
  
                            Navigator.pop(context);  
  
                            Navigator.push(  
  
                                context,  
  
                                MaterialPageRoute(  
  
                                    builder: (context) =>  
  
                                    const  
FromCurrentManualLocation()));  
  
                    },  
  
                    child: const Text("Use current Location"),  
                ),  
            ),  
        ),  
    ),  
);
```

```
        ) ,  
  
        ) ) ,  
  
        ListTile(  
  
            title: TextFormField(  
  
                keyboardType: TextInputType.name,  
  
                controller: extraAddress,  
  
                inputFormatters: [  
  
FilteringTextInputFormatter.allow(RegExp(  
  
    "[a-zA-Z0-  
9|\\.|'|#|@|%|&|/| ]") // Must only contain a-z, A-Z, 0-9,  
. , ' , # , @ , % , & , / , (space)  
  
] ,  
  
decoration: (const InputDecoration(  
  
labelText:  
  
"Apt, suite, unit, building, floor  
etc."))),  
  
        ListTile(  
  
            title: TextFormField(  
  
                keyboardType: TextInputType.name,  
  
                controller: streetAddress,  
  
                validator: (streetAddress) {  
  
                    if (streetAddress == null ||  
  
streetAddress.isEmpty) {
```

```

        // Must be filled

        return "Required";

    }

    return null;

} ,

inputFormatters: [

    FilteringTextInputFormatter.allow(RegExp(
        "[a-zA-Z0-9|\\.|\\'|\\#|@|%|&|/|\\| ]") // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)
    ) ,
    decoration: (const InputDecoration(
        labelText: "Street Address")),

) ,

ListTile(
    title: TextFormField(
        keyboardType: TextInputType.name,
        validator: (cityAddress) {

            if (cityAddress == null ||
cityAddress.isEmpty) {

                //Must be filled

                return "Required";
            }

            return null;
        } ,

```

```
controller: cityAddress,  
  
decoration:  
  
    (const InputDecoration(labelText: "City")) ,  
  
    inputFormatters: [  
  
        FilteringTextInputFormatter.allow(RegExp(  
  
            "[a-zA-Z0-9|\\.|\\'|\\#\\||@\\||%\\||&\\||/|\\|\\|") // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)  
        ] ,  
    ) ,  
  
    ListTile(  
  
        title: TextFormField(  
  
            controller: postcodeAddress,  
  
            keyboardType: TextInputType.name,  
  
            onChanged: (value) {  
  
                postcodeAddress.value = TextEditingValue(  
  
                    //Force capital letters  
                    text: value.toUpperCase() ,  
  
                    selection: postcodeAddress.selection);  
            } ,  
  
            inputFormatters: [  
  
                FilteringTextInputFormatter.allow(RegExp(  
  
                    '[a-zA-Z0-9]')) // Must only contain a-z, A-Z and 0-9  
            ] ,
```

```
decoration:

    (const InputDecoration(labelText:
"Postcode")),

    validator: (postcodeAddress) {

        if (postcodeAddress == null ||

            postcodeAddress.isEmpty ||

            postcodeAddress.length != 6) {

            // Must be filled and be exactly 6

characters

        return "Enter valid postcode";

    }

    return null;

},

),

Container(
padding: const EdgeInsets.all(10),

child: ElevatedButton(

style: ElevatedButton.styleFrom(

minimumSize: const Size.fromHeight(50),

backgroundColor: const Color(0xff5806FF)),

onPressed: () {

if (_formKey.currentState!.validate()) {

//If passed validation continue

saveFilledAddress(
```

```

        extraAddress.text,
        streetAddress.text,
        cityAddress.text,
        postcodeAddress.text);
    Navigator.pop(context); // Go back to main
page

    }

    },
    child: const Text("Save Address"),
),
)

]
))));

} );
}

}

class FromCurrentManualLocation extends StatefulWidget {
const FromCurrentManualLocation({Key? key}) : super(key: key);

}

@Override
State<FromCurrentManualLocation> createState() =>
_FromCurrentManualLocationState();
}

}

```

```
class _FromCurrentManualLocationState extends
State<FromCurrentManualLocation> {

    final _formKey = GlobalKey<FormState>();

    final verifyPostcode = RegExp(
        r"^(Gg|Ii|Rr) [Aa]{2}) | (([A-Za-z][0-9]{1,2}) | ([A-Za-z][A-Ha-
hJ-Yj-y][0-9]{1,2}) | ([AZa-z][0-9][A-Za-z]) | ([A-Za-z][A-Ha-hJ-Yj-y][0-
9]?[A-Za-z])) [0-9] [A-Za-z]{2})\$"); // Postcode format

    Future<void> saveFilledAddress(String extraAddress, String
streetAddress,
        String cityAddress, String postcodeAddress) async {
        try {
            SQLiteLocalDB.setAddress(extraAddress, streetAddress, cityAddress,
                postcodeAddress); // Try to save
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Successfully saved address.')));
        } catch (e) {
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Failed to save address')));
        }
    }

    Future<List> determinePosition() async {

```

```
bool serviceEnabled;

geo.LocationPermission permission;

// Test if location services are enabled.

serviceEnabled = await geo.Geolocator.isLocationServiceEnabled();

if (!serviceEnabled) {

    return Future.error('Location services are disabled.');
}

// Check if the location is denied

permission = await geo.Geolocator.checkPermission();

if (permission == geo.LocationPermission.denied) {

    permission = await geo.Geolocator.requestPermission();

    if (permission == geo.LocationPermission.denied) {

        return Future.error('Location permissions are denied');

    }

}

if (permission == geo.LocationPermission.deniedForever) {

    // Permissions are denied forever, handle appropriately.

    return Future.error('Location permissions are permanently denied.');

}
```

```

// Access the current possition of the device

geo.Position locationDevice = await
geo.Geolocator.getCurrentPosition();

List locationDeviceLatLong = [
    locationDevice.latitude,
    locationDevice.longitude
];

List<Placemark> placemarks = await placemarkFromCoordinates(
    locationDeviceLatLong[0],
    locationDeviceLatLong[1]); //Get the lat and long

Placemark place = placemarks[0];

List address = [
    //Convert to address
    {
        "savedaddressextra": "",
        "savedaddressstreet": place.street,
        "savedaddresscity": place.subAdministrativeArea,
        "savedaddresspostcode": place.postalCode?.replaceAll(' ', ''),
        '' //Remove the space between the each 3 characters to fit
with the constraint
    }
];

return address;

```



```
builder: (context, snapshot) {

    if (snapshot.connectionState != ConnectionState.done) {

        //While waiting to get location, show loading circle

        return const Center(child: CircularProgressIndicator());

    }

    if (snapshot.hasError) {

        //On future failure, show failed

        return const Center(child: Text("Failed"));

    }

    List savedAddress =

        snapshot.data ?? [] ; //Get the location from future

    if (savedAddress.isEmpty) {

        //If it is empty, replace with blanks

        savedAddress = [

            {

                "savedaddressextra": "",

                "savedaddressstreet": "",

                "savedaddresscity": "",

                "savedaddresspostcode": ""

            }

        ] ;

    }

}
```

```

    final TextEditingController extraAddress =
TextEditingController(
    text: savedAddress[0][
        "savedaddressextra"]); //For each field, fill with the
current address

    final TextEditingController streetAddress =
TextEditingController(
    text: savedAddress[0]["savedaddressstreet"]);

    final TextEditingController cityAddress =
TextEditingController(text:
savedAddress[0]["savedaddresscity"]);

    final TextEditingController postcodeAddress =
TextEditingController(
    text: savedAddress[0]["savedaddresspostcode"]);

return Scaffold(
    resizeToAvoidBottomInset: false,
    appBar: AppBar(
        title: const Text('Address Details.'),
        leading: IconButton(
            icon: const Icon(Icons.arrow_back),
            onPressed: () =>
                Navigator.of(context).pop()), //Go back button
        backgroundColor: Theme.of(context).primaryColor,
    ),
    body: Form(

```

```
key: _formKey,  
  
child: SingleChildScrollView(  
  
    child: Column(  
  
        mainAxisAlignment: MainAxisAlignment.start,  
  
        children: [  
  
            Align(  
  
                alignment: Alignment.center,  
  
                child: SizedBox(  
  
                    width: 170,  
  
                    height: 40,  
  
                    child: ElevatedButton(  
  
                        style: ElevatedButton.styleFrom(  
  
                            minimumSize: const  
Size.fromHeight(50),  
  
                            backgroundColor: const  
Color(0xff5806FF)),  
  
                        onPressed: () {  
  
                            //Recenter button to get current  
location again  
  
                            Navigator.pop(  
  
                                context); //Reopen current location  
class  
  
                            Navigator.push(  
  
                                context,
```

```

        MaterialPageRoute(
            builder: (context) =>
            const
FromCurrentManualLocation())));
}

),
child: const Text("Recenter"),
),
),
ListTile(
title: TextFormField(
keyboardType: TextInputType.name,
controller: extraAddress,
inputFormatters: [
FilteringTextInputFormatter.allow(RegExp(
"^[a-zA-Z0-9|\\.|'|\\#|@|%|&|/| ]") // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)
],
decoration: (const InputDecoration(
labelText:
"Apt, suite, unit, building, floor
etc.")),
ListTile(
title: TextFormField(

```

```
        keyboardType: TextInputType.name,  
  
        controller: streetAddress,  
  
        validator: (streetAddress) {  
  
            if (streetAddress == null ||  
  
                streetAddress.isEmpty) {  
  
                return "Required";  
  
            }  
  
            return null;  
  
        },  
  
        inputFormatters: [  
  
            FilteringTextInputFormatter.allow(RegExp(  
  
                "[a-zA-Z0-9|\\.|\\'|\\#\\||@\\||%\\||&\\||/|\\\\  
                ]")) // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)  
  
        ],  
  
        decoration: (const InputDecoration(  
  
            labelText: "Street Address")),  
  
        ),  
  
        ListTile(  
  
            title: TextFormField(  
  
                keyboardType: TextInputType.name,  
  
                validator: (cityAddress) {  
  
                    if (cityAddress == null ||  
cityAddress.isEmpty) {  
  
                        return "Required";  
  
                    }  
  
                    return null;  
  
                },  
  
                initialValue:  
            ),  
  
            subtitle: Text("City Address"),  
  
            trailing: IconButton(  
                icon: Icon(Icons.location_on),  
  
                onPressed: () {  
                    Navigator.push(context,  
                        MaterialPageRoute(builder: (context) =>  
                            CityAddressScreen());  
                },  
  
            ),  
  
        ),  
  
    ),  
  
    
```

```
        }

        return null;
    } ,
    controller: cityAddress,
    decoration:
        (const InputDecoration(labelText: "City")) ,
    inputFormatters: [
        FilteringTextInputFormatter.allow(RegExp(
            "[a-zA-Z0-9|\\.|\\'|\\#|@|%|&|/|\\| ]") // Must only contain a-z, A-Z, 0-9, ., ', #, @, %, &, /, (space)
    ] ,
) ),
ListTile(
    title: TextFormField(
        controller: postcodeAddress,
        keyboardType: TextInputType.name,
        onChanged: (value) {
            postcodeAddress.value = TextEditingValue(
                text: value.toUpperCase(),
                selection: postcodeAddress.selection);
        },
        inputFormatters: [
            FilteringTextInputFormatter.allow(RegExp(

```

```
'[a-zA-Z0-9]')) // Must only contain a-z,  
A-Z, 0-9  
],  
decoration:  
(const InputDecoration(labelText:  
"Postcode")) ,  
validator: (postcodeAddress) {  
if (postcodeAddress == null ||  
postcodeAddress.isEmpty ||  
postcodeAddress.length != 6) {  
return "Enter valid postcode";  
}  
return null;  
},  
) ,  
Container(  
padding: const EdgeInsets.all(10) ,  
child: ElevatedButton(  
style: ElevatedButton.styleFrom(  
minimumSize: const Size.fromHeight(50) ,  
backgroundColor: const Color(0xff5806FF)) ,  
onPressed: () {  
if (_formKey.currentState!.validate()) {
```

```
// If all validation correct, send to be
saved and close the page

    saveFilledAddress (

        extraAddress.text,
        streetAddress.text,
        cityAddress.text,
        postcodeAddress.text);

    Navigator.pop(context);

} else {

ScaffoldMessenger.of(context).showSnackBar (
    const SnackBar(
        content: Text('Invalid
Address')));
}

},
child: const Text("Save Address"),
),
)
]
))));
```

```
} ;
```

```
}
```

```
}
```

navigationbar.dart

```
import 'package:alleat/screens/main_app/browse.dart';

import 'package:alleat/screens/main_app/favourites.dart';

import 'package:alleat/screens/main_app/foryou.dart';

import 'package:alleat/screens/main_app/homepage.dart';

import 'package:alleat/screens/main_app/profiles.dart';

import 'package:flutter/material.dart';

class Navigation extends StatefulWidget {

  const Navigation({Key? key}) : super(key: key);

  @override

  State<Navigation> createState() => _NavigationState();
}

class _NavigationState extends State<Navigation> {

  int _selectedIndex = 0; //Start at Home page

  final List _screens = [
    {"screen": const HomePage()},
    {"screen": const ForYouPage()},
    {"screen": const BrowsePage()},
  ];
}
```

```
{"screen": const ProfilePage()}

] ;

void _onItemTapped(int index) {
    //When user click button on bottom navigation bar, change to that
    index
    setState(() {
        _selectedIndex = index;
    });
}

@Override
Widget build(BuildContext context) {
    return WillPopScope(
        onWillPop: () async => false,
        child: MaterialApp(
            title: "All Eat.",
            theme: ThemeData(
                scaffoldBackgroundColor: const Color(0xffFAFAFA),
                primaryColor: const Color(0xff5806FF)),
            home: Scaffold(
                appBar: AppBar(
                    title: const Text("All Eat.")),
            )));
}
```

```
backgroundColor: const Color(0xff5806FF),  
  
actions: [  
  
    //Top right action buttons on app bar  
  
    IconButton(  
  
        //Favourite page button  
  
        icon: const Icon(Icons.favorite_outline),  
  
        onPressed: () {  
  
            Navigator.of(context).pop();  
  
            Navigator.push(  
  
                context,  
  
                MaterialPageRoute(  
  
                    builder: (context) => const  
FavouritesPage()));  
  
        } ,  
  
    //Top navigation bar icons  
  
    const Padding(  
  
        //Cart page button  
  
        padding: EdgeInsets.symmetric(horizontal: 18),  
  
        child: Icon(Icons.shopping_bag_outlined)),  
  
    ] ,  
  
) ,  
  
body: _screens[_selectedIndex] ["screen"] ,  
  
bottomNavigationBar: BottomNavigationBar(  

```

```
// If button is selected, show purple. If button is not
selected show greyed out text and icon

    selectedItemColor: const Color(0xff5806FF),
    unselectedItemColor: const Color(0xff666666),
    items: <BottomNavigationBarItem>[
        //Bottom navigation bar items
        BottomNavigationBarItem(
            icon: Container(
                padding: const EdgeInsets.only(
                    bottom:
                        3), //Each have padding to separate from
            the text
            child: const Icon(
                Icons.home_outlined)), //Unselected icon is
            outlined
            label: 'Home',
            activeIcon: Container(
                padding: const EdgeInsets.only(bottom: 3),
                child:
                    const Icon(Icons.home)), // Selected icon is
            filled
        ),
        BottomNavigationBarItem(
            //Bottom navigation bar options
```

```
icon: Container(
    padding: const EdgeInsets.only(bottom: 3),
    child: const Icon(Icons.assistant_outlined)),
label: 'For You',
activeIcon: Container(
    padding: const EdgeInsets.only(bottom: 3),
    child: const Icon(Icons.assistant)),
),

BottomNavigationBarItem(
    icon: Container(
        padding: const EdgeInsets.only(bottom: 3),
        child: const Icon(Icons.manage_search_rounded)),
    label: 'Browse',
    activeIcon: Container(
        padding: const EdgeInsets.only(bottom: 3),
        child: const Icon(Icons.manage_search)),
),
BottomNavigationBarItem(
    icon: Container(
        padding: const EdgeInsets.only(bottom: 3),
        child: const Icon(Icons.person_outlined)),
    label: 'Profile',
    activeIcon: Container(
```

```

        padding: const EdgeInsets.only(bottom: 3),
        child: const Icon(Icons.person)),
    ) ,
],
onTap: _onItemTapped, //On tap, change selected option
currentIndex:
        _selectedIndex, //Make current index the one last
selected (defaults to home)
),
) );
}
}

```

restaurantlist.dart

```

import 'package:alleat/services/sqlite_service.dart';

import 'package:alleat/widgets/restaurant/restaurantmain.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

class RestaurantList extends StatefulWidget {

```

```

@Override

State<RestaurantList> createState() => _RestaurantListState();

}

class _RestaurantListState extends State<RestaurantList> {

late bool error, sending, success, serverOffline;

late String msg;

@Override

void initState() {

//Default values

error = false;

sending = false;

success = false;

msg = "";

super.initState();

}

Future<List> getRestaurants() async {

String phpurl =

"https://alleat.cpur.net/query/restaurantlist.php"; //Get

restaurant list

try {

```

```

var res = await http.post(Uri.parse(phiurl), body: {

    "sort": "id", //Send sort type (Currently permanently by id)

}) ;

if (res.statusCode == 200) {

    //If sends successfully

    var data = convert.json.decode(res.body); //Decode to array

    if (data["error"]) {

        //If fails to perform query

        List error = [

        {

            "error": "true",

            "restaurants": "[]"

        } //Send blank list of restaurants

    ];

    return error;

} else {

    List listdata = [

        data

   ]; //If success, send the list of restaurants back

    return listdata;

}

} else {

    List error = [

```

```

        {"error": "true", "restaurants": "[]"}  

    ];  

    return error;  

}  

} catch (e) {  

List<Map<String, String>> error = [  

{"error": "true", "restaurants": "[]"}  

];  

return error;  

}  

}  

Future<String> favouriteRestaurant(restaurantID, action) async {  

String phpurl =  

"https://alleat.cpur.net/query/favouriterestaurant.php";  

//Favourite & unfavourite restaurant for sepcific profile using their  

email  

var profile = await SQLiteLocalDB.getProfileSelected();  

String email = profile[0]["email"].toString();  

try {  

var res = await http.post(Uri.parse(phpurl), body: {  

"action": action

```

```
        .toString(), //Action determines if it will favourite or
unfavourite depending on input

        "profileemail": email.toString(),

        "restaurantid": restaurantID,
    }) ;

    if (res.statusCode == 200) {

        //On successfull send

        var data = converty.json.decode(res.body); //Decode to array

        if (data["error"]) {

            return "failed";

        } else {

            getFavourites(); //get favourite restaurant list

            return "success";

        }

    } else {

        return "failed";

    }

} catch (e) {

    return "failed";

}

}

Future<List> getFavourites() async {
```

```

String phpurl =
    "https://alleat.cpur.net/query/favouriterestaurantlist.php"; //Get
favourite restaurant ids using profile email

var profile = await SQLiteLocalDB.getProfileSelected();

String email = profile[0]["email"].toString();

try {

    var res = await http.post(Uri.parse(phpurl), body: {
        "profileemail": email.toString(),
    });

    if (res.statusCode == 200) {

        //If successfull send data

        var data = convert.json.decode(res.body); //Decode to array

        if (data["error"]) {

            //If error querying data

            List error = [
                {
                    "error": "true",
                    "favouriterestaurants": "[]"
                } //Send empty favourite restaurant ids list
            ];
        }
        return error;
    } else {

        //If successful query
    }
}

```

```
        List listdata = [data]; //Send list of restaurant ids back

        return listdata;

    }

} else {

    List error = [

        {"error": "true", "favouriterestaurants": "[]"}

   ];

    return error;
}

} catch (e) {

    List<Map<String, String>> error = [

        {"error": "true", "favouriterestaurants": "[]"}

   ];

    return error;
}

}

@Override

Widget build(BuildContext context) {

    return FutureBuilder<List>(

        //Get list of restaurants data from future (rebuild on change of
        data)

        future: getRestaurants(),

```



```

    //If data received

    List restaurantFavourites = snapshot.data ?? [];

    return ListView.builder(
        //Show number of containers depending on number of
        restaurants

        physics:

            const NeverScrollableScrollPhysics(), //Dont
            allow scrolling (Done by main page)

            scrollDirection: Axis.vertical,
            shrinkWrap: true,
            itemCount: restaurantsdata[0]["restaurants"]
                .length, //For each restaurant

            itemBuilder: (context, index) {

                return Center(child:
                    LayoutBuilder(builder: (context, constraints)
{

                if (restaurants.isEmpty) {

                    //If there are no restaurants show container
                    saying no restaurants

                    return Padding(
                        padding: const EdgeInsets.only(
                            left: 20, right: 20, top: 10, bottom:
                            10),
                        child: Container(

```

```
decoration: BoxDecoration(  
    borderRadius: const  
BorderRadius.all(  
    Radius.circular(20)),  
    color: const Color(0xffffffff),  
    boxShadow: [  
        BoxShadow(  
            color:  
  
Colors.black.withOpacity(0.1),  
            spreadRadius: 2,  
            blurRadius: 10,  
            offset: const Offset(0,  
                10), // changes position  
of shadow  
        ),  
    ],  
    child: Column(children: const [  
        Padding(  
            padding: EdgeInsets.all(30),  
            child: SizedBox(  
                width: double.infinity,  
                child: Center(  
                    child: Text(  
                        text: 'Hello World!',  
                        style: TextStyle(fontSize: 24),  
                    ),  
                ),  
            ),  
    ],  
),  
);
```

```

        "No restaurants
found"))), //Display no restaurants text

    )

])));

} else {

    // If there is restaurant

    return InkWell(
        //Create clickable container

        onTap: () {

            List restaurantinfodata =
restaurantsdata[0]["restaurants"][index];

            Navigator.push(
                context,
                MaterialPageRoute(
                    builder:
                        (context) => //On tap, send
restaurant data associated with container to main page and go to that new
screen

                    Restaurant MainPage(
                        resid:
restaurantinfodata[0],


                    resname:

```

```
restaurantinfodata[1],  
  
                                reslogo:  
  
restaurantinfodata[2],  
  
                                resbanner:  
  
restaurantinfodata[3],  
  
)) );  
  
},  
  
child: Padding(  
  
padding: const EdgeInsets.only(  
  
left: 20,  
  
right: 20,  
  
top: 10,  
  
bottom: 10),  
  
child: Container(  
  
decoration: BoxDecoration(  
  
borderRadius: const  
BorderRadius.all(  
  
Radius.circular(20)),  
  
color: const Color(0xffffffff),  
  
boxShadow: [  
  
BoxShadow(  
)
```

```
        color:

Colors.black.withOpacity(0.1),

        spreadRadius: 2,

        blurRadius: 10,

        offset: const Offset(0,
            10), // changes position
        of shadow

    ) ,

] ) ,

child: Column(children: [
    //Main restaurant data here

Padding(
    //Restaurant Banner

padding: const
EdgeInsets.all(10),

    child: Container(
        //Container filled with
        restaurant banner

        width:
        MediaQuery.of(context)

        .size
        .width,
        height: 150,
        decoration: BoxDecoration(
```

```
borderRadius:  
    const  
BorderRadius.all(  
  
Radius.circular(10)),  
  
    image: DecorationImage(  
  
        fit: BoxFit.fitWidth,  
  
        image: NetworkImage(  
  
restaurants[index][3]  
  
.toString()),  
  
) ,  
  
) ,  
  
) ,  
  
Padding(  
  
//Restaurant logo, text and  
favourites button below restaurant banner  
  
padding: const  
EdgeInsets.only(  
  
    left: 20,  
  
    right: 30,  
  
    bottom: 15,  
  
    top: 10),  
  
child: Row(  
  
mainAxisAlignment:
```

```
        MainAxisAlignment  
        .spaceBetween,  
        children: [  
          Row(  
            children: [  
              ClipOval(  
                //Restaurant logo  
                child:  
                Image.network(  
                  restaurants[index]  
                  [2]  
  
.toString(),  
                height: 50,  
                width: 50)),  
              const SizedBox(  
                width: 10,  
              ),  
              Text(  
                //Restaurant Text  
                restaurants[index][1],  
                style: const  
                TextStyle(  
              ),  
            ),  
          ),  
        ],  
      ),  
    ),  
  ),  
),  
);
```

```
        fontSize: 16,  
        fontWeight:  
  
FontWeight.w500),  
    ),  
],  
,  
),  
 IconButton(  
 //Favourites button  
 onPressed: () {  
 if  
(restaurantFavourites[  
 0]  
  
["restaurantids"]  
  
.contains(restaurants[  
  
index] [0]  
  
.toString())) {  
  
favouriteRestaurant(  
  
restaurants[index]  
[0]
```

```
.toString() ,  
  
"unfavourite");  
  
        setState(() {  
            getFavourites();  
        } );  
    } else {  
  
        favouriteRestaurant(  
  
restaurants[index]  
            [0]  
  
.toString() ,  
            "favourite");  
        setState(() {  
            getFavourites();  
        } );  
    }  
},  
icon:  
(restaurantFavourites[  
            0] [  
  
"restaurantids"]
```

```
.contains(restaurants[  
  
index] [0]  
  
.toString()))  
?  
const Icon(  
  
Icons  
  
.favorite,  
// ? = favoured  
  
color: Color(  
  
0xff33C496))  
  
:  
const Icon(  
  
// : =  
unfavoured  
  
Icons  
  
.favorite_border_outlined,  
  
color: Color(  
  
0xff000000))),  
  
],  
)  
)  
]  
,  
))  
);
```

```
        }

    } ) ;

} ) ;

}

} ) ,


) ;

} catch (e) {

return Padding(


padding: const EdgeInsets.only(


left: 20, right: 20, top: 10, bottom: 10),


child: Container(


decoration: BoxDecoration(


borderRadius:


const BorderRadius.all(Radius.circular(20)),


color: const Color(0xffffffff),


boxShadow: [


BoxShadow(


color: Colors.black.withOpacity(0.1),


spreadRadius: 2,


blurRadius: 10,


offset: const Offset(


0, 10), // changes position of shadow


),



```

```

] ) ,  

child: Column(children: [  

Padding(  

padding: const EdgeInsets.all(30) ,  

child: SizedBox(  

width: double.infinity,  

child: Center(  

child: Column(children: [  

const Text("Error 400: Connection failed") ,  

ElevatedButton(  

onPressed: () {  

setState(() {  

getRestaurants();  

} ) ;  

} ,  

child: const Text("Retry"))  

])) ,  

)
]
)) );
}
}
}
);

```

}

restaurantitemcustomise.dart

```
import 'package:flutter/material.dart';

class RestaurantItemCustomisePage extends StatefulWidget {

    final String itemid;

    final String foodcategory;

    final String subfoodcategory;

    final String itemname;

    final String description;

    final String price;

    final String itemimage;

    const RestaurantItemCustomisePage(
        {Key?
            key, //Get the items from the restaurant main page when an item
is clicked

            required this.itemid,
            required this.foodcategory,
            required this.subfoodcategory,
            required this.itemname,
            required this.description,
```



```
        icon: const Icon(Icons.arrow_back),  
        onPressed: () => Navigator.of(context).pop(),  
        backgroundColor: Theme.of(context).primaryColor,  
    ),  
  
    body: SingleChildScrollView(  
        child: SizedBox(  
            child: Column(children: [  
                Padding(  
                    //Restaurant Banner  
                    padding: const EdgeInsets.only(bottom: 10),  
                    child: Container(  
                        width: MediaQuery.of(context).size.width,  
                        height: 200,  
                        decoration: BoxDecoration(  
                            borderRadius: const BorderRadius.only(  
                                bottomLeft: Radius.circular(10),  
                                bottomRight: Radius.circular(10)),  
                            image: DecorationImage(  
                                fit: BoxFit.fitWidth,  
                                image: NetworkImage(widget.itemimage),  
                            ),  
                        ),  
                    ),  
                ),  
            ),  
        ),  
    ),
```

```
) ,  
  
    Align (  
  
        alignment: Alignment.topLeft,  
  
        child: Column (  
  
            crossAxisAlignment: CrossAxisAlignment.start,  
  
            children: [  
  
                Padding (  
  
                    padding: const EdgeInsets.only(  
  
                        left: 20, right: 20, top: 20, bottom:  
5) ,  
  
                    child: Row (  
  
                        mainAxisAlignment:  
  
                        MainAxisAlignment.spaceBetween,  
  
                        children: [  
  
                            Flexible (  
  
                                flex: 8, // 8/10 assigned to item  
name  
  
                                child: Text (  
  
                                    //Restaurant Text  
  
                                    widget.itemname,  
  
                                    textAlign: TextAlign.left,  
  
                                    style: const TextStyle (  
  
                                        fontSize: 21,  

```

```
        fontWeight:  
FontWeight.w600)) ,  
  
        Flexible(  
  
            //Restaurant price  
  
            flex: 2, // 2/10 assigned to price  
  
            child: Text("£${widget.price}" ,  
  
                textAlign: TextAlign.left,  
  
                style: const TextStyle(  
  
                    fontSize: 21,  
  
                    fontWeight:  
FontWeight.w600,  
  
                    color:  
Color(0xff5806FF) ) ) ,  
  
        ] ,  
  
    ) ,  
  
    Padding(  
  
        //Categories  
  
        padding:  
        const EdgeInsets.only(left: 20, right:  
20) ,  
  
        child: Row(children: [  
  
            Text(  
  
                widget  
  
.foodcategory, //Must have food  
category
```

```
        style: const TextStyle(
```

```
            fontSize: 14,
```

```
            fontWeight: FontWeight.w400,
```

```
            color: Color(0xff5806FF))),
```

```
        LayoutBuilder(
```

```
            builder: (context, constraints) {
```

```
                if (widget.subfoodcategory != "") {
```

```
                    //If there is a sub food category,
```

```
show it
```

```
                    return Text(
```

```
                        " · ${widget.subfoodcategory}",
```

```
                        style: const TextStyle(
```

```
                            fontSize: 14,
```

```
                            fontWeight: FontWeight.w400,
```

```
                            color: Color(0xff5806FF)));
```

```
                } else {
```

```
                    //If there isnt a sub-food category,
```

```
dont show it
```

```
                    return const Text("");
```

```
                }
```

```
            } )
```

```
        ])),
```

```
        Padding(
```

```

    //Restaurant logo, text and favourites
button below restaurant banner

padding: const EdgeInsets.only(
    left: 20, right: 30, bottom: 15, top:
20) ,
    child: Text(
        //Restaurant Text
        widget.description,
        textAlign: TextAlign.left,
        style: const TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.w400)))
    ])) ,
    ]),
),
),
));
}

}

```

restaurantmain.dart

```

import 'package:alleat/services/sqlite_service.dart';

import 'package:alleat/widgets/restaurant/restaurantitemcustomise.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

```

```
import 'dart:convert' as convert;

class RestaurantMainPage extends StatefulWidget {

    final String resid;
    final String resname;
    final String reslogo;
    final String resbanner;

    const RestaurantMainPage(
        //Get restaurant info from the restuarant list widget (passed from
        the container)
        {Key? key,
        required this.resid,
        required this.resname,
        required this.reslogo,
        required this.resbanner})
        : super(key: key);

    @override
    State<RestaurantMainPage> createState() => _RestaurantMainPageState();
}

class _RestaurantMainPageState extends State<RestaurantMainPage> {
```

```
Future<List> getMenuCategories() async {

    String phpurl =
        "https://alleat.cpur.net/query/restaurantmenucategories.php";
    //Get the list of menu categories associated with the restaurant id

    try {

        var res = await http.post(Uri.parse(phpurl), body: {
            "restaurantid": widget.resid,
        });

        if (res.statusCode == 200) {

            //If successfully sent

            var data = convert.json.decode(res.body); //Decode to array

            if (data["error"]) {

                //If failed to query

                List error = [
                    {
                        "error": "true",
                        "restaurantcategories": "[]"
                    } //Return no menu categories
                ];
            }

            return error;
        } else {

            List listdata = [data];

            return listdata;
        }
    }
}
```

```

        }

    } else {

        List error = [
            {"error": "true", "restaurantcategories": "[]"}
        ];

        return error;
    }

} catch (e) {

    List<Map<String, String>> error = [
        {"error": "true", "restaurantcategories": "[]"}
    ];

    return error;
}

}

Future<String> favouriteRestaurant(restaurantID, action) async {

    String phpurl =
        "https://alleat.cpur.net/query/favouriterestaurant.php"; //Action
    of favourite/unfavourite restaurant by id (Unused in this version)

    var profile = await SQLiteLocalDB.getProfileSelected();

    String email = profile[0]["email"].toString();

    try {

```

```

var res = await http.post(Uri.parse(phiurl), body: {

    "action": action.toString(),

    "profileemail": email.toString(),

    "restaurantid": restaurantID,

}) ;

if (res.statusCode == 200) {

    var data = convert.json.decode(res.body); //Decode to array

    if (data["error"]) {

        return "failed";

    } else {

        getFavourites();

        return "success";

    }

} else {

    return "failed";

}

} catch (e) {

    return "failed";

}

}

Future<List> getFavourites() async {

String phiurl =

```

```

"https://alleat.cpur.net/query/favouriterestaurantlist.php"; //Get
favourite restaurant list associated with email of profile (unused in this
version)

var profile = await SQLiteLocalDB.getProfileSelected();

String email = profile[0]["email"].toString();

try {

var res = await http.post(Uri.parse(phppurl), body: {

"profileemail": email.toString(),

}) ;

if (res.statusCode == 200) {

var data = converty.json.decode(res.body); //Decode to array

if (data["error"]) {

List error = [

{"error": "true", "favouriterestaurants": "[]"

];

return error;

} else {

List listdata = [data];

return listdata;

}

} else {

List error = [

{"error": "true", "favouriterestaurants": "[]"

];

```

```

        return error;
    }

} catch (e) {
    List<Map<String, String>> error = [
        {"error": "true", "favouriterestaurants": "[]"}
    ];
    return error;
}

}

Future<List> getMenuItems(categoryid) async {

    String phpurl =
        "https://alleat.cpur.net/query/restaurantmenuitems.php"; //Get
    list of menu items, with the inforamtion associated with it. Grabs using
    the category id it falls under

    try {
        var res = await http.post(Uri.parse(phpurl), body: {
            "categoryid": categoryid,
        });
        if (res.statusCode == 200) {
            //If fails to send data
            var data = convert.json.decode(res.body); //Decode to array
            if (data["error"]) {
                //If fails the query

```

```

List error = [
    {"error": "true", "menuitems": "[]"} //Send nothing
];

return error;

} else {

    List listdata = [
        data
    ]; //Send back the list of items associated with the category

    return listdata;
}

} else {

    List error = [
        {"error": "true", "menuitems": "[]"}
    ];

    return error;
}

} catch (e) {
    List<Map<String, String>> error = [
        {"error": "true", "menuitems": "[]"}
    ];

    return error;
}
}

```

```
override

Widget build(BuildContext context) {

  return MaterialApp(
    title: widget.resname, //Set appbar title to restaurant name
    home: Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        title: Text(widget.resname),
        leading: IconButton(
          //Back button
          icon: const Icon(Icons.arrow_back),
          onPressed: () => Navigator.of(context).pop(),
        ),
        backgroundColor: Theme.of(context).primaryColor,
      ),
      body: SingleChildScrollView(
        //Make page scrollable
        child: Column(children: [
          Padding(
            //Restaurant banner
            padding: const EdgeInsets.only(bottom: 10),
            child: Container(
              width: MediaQuery.of(context).size.width,
```



```

        height: 50, width: 50)),

    const SizedBox(
        width: 10,
    ) ,
    Text(
        //Restaurant Name
        widget.resname,
        style: const TextStyle(
            fontSize: 16, fontWeight:
FontWeight.w500),
    ) ,
    ] ,
) ,
] ,
) ) ,
FutureBuilder<List>(
    //Checks for updates in the restaurant menu category
    future: getMenuCategories(),
    builder: ((context, snapshot) {
        if (!snapshot.hasData) {
            //If no data has been received, show loading bar
            return const LinearProgressIndicator(
                color: Color(0xff4100C4),

```

```

        backgroundColor: Color(0xffEBE0FF));

    }

    if (snapshot.hasError) {

        //If there is an error, grabbing data, show error

        return const Text("An Error occured. Please try
again");

    } else {

        //If the data has been received

        List restaurantcategories = snapshot.data ??

        [] ; //Categories data associated with
restaurantcategories

        return ListView.builder(
            //For each category

            physics:

                const NeverScrollableScrollPhysics(),
//Disable scrolling. Scroll with whole page

                scrollDirection: Axis.vertical,
                shrinkWrap: true,
                itemCount: restaurantcategories[0]

                ["restaurantcategories"]

                .length, //For each restaurant

                itemBuilder: (context, index) {

                    return LayoutBuilder(
                        builder: (context, constraints) {

```

```
if (restaurantcategories[0]

    ["restaurantcategories"]

    .isEmpty) {

    //If there is no categories, display menu
    unavailable

    return const Text("Menu unavailable");

} else {

    //If there are categories
    return Column(children: [ //Add text of
category name

    Padding(
        padding: const EdgeInsets.only(
            left: 30,
            top: 30,
            right: 10,
            bottom: 10),
        child: Text(
            restaurantcategories[0]

["restaurantcategories"][index][0],
            style: const TextStyle(
                fontSize: 21,
                fontWeight: FontWeight.w500),
        )));
}
```

```

        FutureBuilder<List>(<!--For each
category, use a future to get the list of items

        future: getMenuItems(
            restaurantcategories[0]
                ["restaurantcategories"]
                [index][1]),

        builder: (context, snapshot) {
            if (!snapshot.hasData) {<!--While
no data received, show loading bar

            return const
LinearProgressIndicator(
                color: Color(0xff4100C4),
                backgroundColor:
                    Color(0xffEBE0FF));
}

            if (snapshot.hasError) {<!--If
there is an error, show failed to get items

            return const Text(
                "Failed to get items");

        } else {
            //List of items for category
            List restaurantitems =
                snapshot.data ??
[]
;<!--Get data from
Future</pre>

```

```
        return ListView.builder(  
  
            physics:  
  
                const  
NeverScrollableScrollPhysics(),  
  
            scrollDirection:  
Axis.vertical,  
  
            shrinkWrap: true,  
  
            itemCount: (restaurantitems[  
  
                0] //For each  
item, create a container  
  
                ["menuitems"]  
  
.length), //For each  
restaurant  
  
            itemBuilder: (context,  
index) {  
  
                return  
LayoutBuilder(builder:  
  
(context, constraints)  
{  
  
                if (restaurantitems[0]  
  
                    ["menuitems"]  
  
.isEmpty) {  
  
//If there is no items  
for a category  
  
                return const Text(  
  
"No items found");  
}
```

```
        } else {

            // If there is items
            to display

            return InkWell(
                //Clickable items

                onTap: () {

                    Navigator.push(
                        context,
                        MaterialPageRoute(
                            builder:
                                (context) => RestaurantItemCustomisePage(
                                    itemid: restaurantitems[0]["menuitems"][index]

                                    [0],

                                    foodcategory: restaurantitems[0]["menuitems"]

                                    [index]

                                    [1],

                                    subfoodcategory: restaurantitems[0]

                                    ["menuitems"] [index]

                                    [2],

```

```
itemname: restaurantitems[0]["menuitems"][index]

[3],

description: restaurantitems[0]

["menuitems"][index]

[4],

price: restaurantitems[0]

["menuitems"][index]

[5],

itemimage:

restaurantitems[0]["menuitems"][index]

[6])));

},

child: Container(

//Create

clickable container

width: double

.infinity,
```

```
margin:  
        const  
EdgeInsets  
  
.only(  
        left:  
10,  
        right:  
10,  
        top:  
5,  
        bottom: 5),  
        color: const  
Color(  
0xffffffff),  
        child: Row(  
  
mainAxisAlignment:  
        MainAxisAlignment  
  
.spaceBetween,  
        //Create a  
row containing item image, name and price  
        children: [  
        Flexible(  
)
```

```
flex:  
4, // 4/13 of container for item image  
  
child:  
Padding(  
  
//Item Image  
  
padding: const EdgeInsets.all(10),  
  
child: Container(  
  
width:  
70,  
  
height:  
70,  
  
decoration:  
  
BoxDecoration(  
  
borderRadius:  
  
const BorderRadius.all(Radius.circular(10)),  
  
image:
```

```
DecorationImage(
```

```
    fit: BoxFit.cover,
```

```
    image: NetworkImage(restaurantitems[0]["menuitems"][index][6].toString()),
```

```
) ,
```

```
) ,
```

```
) ) ,
```

```
        Flexible(
```

```
            flex:
```

```
6, // 6/13 of container for item name
```

```
            child:
```

```
Align(
```

```
        alignment: Alignment.center,
```

```
        child: Padding(
```

```
            padding: const EdgeInsets.only(top: 10, bottom: 10, left: 5),
```

```
            child: Column(children: [
```

```
                //Item Name
```

```
                Text(
```

```
restaurantitems[0]["menuitems"][index][3].toString(),

textAlign: TextAlign.center,

style: const TextStyle(fontWeight: FontWeight.w600, fontSize: 16),

),

const SizedBox(height: 10),

Text(overflow: TextOverflow.ellipsis,
restaurantitems[0]["menuitems"][index][4].toString(), textAlign:
TextAlign.center, style: const TextStyle(fontWeight: FontWeight.w400,
fontSize: 12))

]]))),

Flexible(
flex: 3,
child:
Align(
alignment: Alignment.centerRight,
child: Padding(
padding: const EdgeInsets.only(left: 10, right: 20),
child: Text(

```

```
"£${restaurantItems[0]["menuitems"][index][5]}",
textAlign: TextAlign.end,
) ) )
] ,
) ) ;
}

} ) ;
} ) ;
}

} )
] ) ;
}

} ) ;
} ) ;
} ) ;
}

} )
} ) )
] ) ) ) ;
}

}
```

Files - Server

favouriterestaurant.php

```
<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDflr"; //database username
$dbpassword = "FO7&Ruvr;OG"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";
$return["success"] = false;

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["action"]) && ($_POST["profileemail"]) && isset($_POST["restaurantid"]);

if($val){
    $email = $_POST["profileemail"]; //grabing the data from headers
    $restaurantid = $_POST["restaurantid"];
    $action = $_POST["action"];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row['ProfileID'];
            }
            // Free result set
            mysqli_free_result($result);
            if ($action == "unfavourite"){
                $sqlunfavourite = "DELETE FROM favrestaurant WHERE profileid = $ProfileID AND restaurantid = $restaurantid";
                if ($link->query($sqlunfavourite) === TRUE) {
                    $return["success"] = true;
                } else{
                    $return["error"] = true;
                }
            }
            else if ($action == "favourite"){
                $sqlfavourite = "INSERT INTO favrestaurant (profileid, restaurantid) VALUES ($ProfileID, $restaurantid)";
                if ($link->query($sqlfavourite) === TRUE) {
                    $return["success"] = true;
                }
            }
        }
    }
}
```

```

} else{
    $return["error"] = true;
}

}

else{
    $return["error"] = true;
    $return["message"] = "action unspecified";
}

}

else{
    $return["error"] = true;
    $return["message"] = "duplicate found";
}
}

else{
    $return["error"] = true;
    $return["message"] = "failed";
}
}

else{$return["error"] = true;
    $return["message"] = "data not specified";}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

favouriterestaurantdata.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDFIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

```

```

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurants" ] = array();
$return[ "temp" ] = array();
$restaurantids = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileemail"]);

if($val){
    $email = $_POST["profileemail"];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row[ 'ProfileID' ];
            }
            // Free result set
            mysqli_free_result($result);
        }
    } else{
        $return[ "error" ] = true;
        $return[ "message" ] = "duplicate found";
    }
    $sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
profileid = $ProfileID";
    if($result2 = mysqli_query($link, $sqlfavrestaurant)){
        while($row2 = mysqli_fetch_assoc($result2)) {
            array_push($restaurantids, $row2[ "restaurantid" ]);
        }
        mysqli_free_result($result2);
    }

    foreach ($restaurantids as &$value) {
        $sqlrestaurantinfo = "SELECT res.restaurantid,
res.restaurantname, res.restaurantlogo, res.restaurantbanner FROM
restaurant res WHERE res.restaurantid = $value";

```

```

$return[ "temp" ] = $sqlrestaurantinfo;
if($result3 = mysqli_query($link, $sqlrestaurantinfo)){
    while($row3 = mysqli_fetch_assoc($result3)) {
        array_push($return["restaurants"],
[$row3["restaurantid"], $row3["restaurantname"], $row3["restaurantlogo"],
$row3["restaurantbanner"]]);
    }
    mysqli_free_result($result3);
}
}

else{
    $return[ "error" ] = true;
    $return[ "message" ] = "failed";
}

}

else{
    $return[ "error" ] = true;
    $return[ "message" ] = "data not specified 2";
}

mysqli_close($link); //close mysqli

header( 'Content-Type: application/json' );
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

favouriterestaurantlist.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDFIr"; //database username

```

```

$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurantids" ] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileemail"]);

if($val){
    $email = $_POST["profileemail"];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row[ 'ProfileID' ];
            }
            // Free result set
            mysqli_free_result($result);
        }
    } else{
        $return[ "error" ] = true;
        $return[ "message" ] = "duplicate found";
    }
}

else{
    $return[ "error" ] = true;
    $return[ "message" ] = "failed";
}

$sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
profileid = $ProfileID";
if($result2 = mysqli_query($link, $sqlfavrestaurant)){
    while($row2 = mysqli_fetch_assoc($result2)) {
        array_push($return[ "restaurantids" ],
$row2[ "restaurantid" ]);
    }
}

```

```

        }
        mysqli_free_result($result2);

    }

}

else{
    $return[ "error" ] = true;
    $return[ "message" ] = "data not specified 2";
}

mysqli_close($link); //close mysqli

header( 'Content-Type: application/json' );
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

login.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "exists" ] = false;
$return[ "profile" ] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);
//connecting to database server

$val = isset($_POST["email"]) && isset($_POST["password"]);

```

```

if($val){
    //checking if there is POST data
    $email = $_POST["email"];
    $password = $_POST["password"];

    $secret_iv = '5fd0b31f582beb1f';
    $secret_key = 'e16a3f356b2366c1';
    $cipher = "AES-128-CBC";
    $decryptedString = openssl_decrypt ($password, $cipher,
$secret_key, 0, $secret_iv);

    $sql = "SELECT FirstName, LastName, Email, Password FROM
profile WHERE Email = '$email'";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $FirstNameR = $row['FirstName'];
                $LastNameR = $row['LastName'];
                $EmailR = $row['Email'];
                $PasswordR = $row['Password'];
            }
            // Free result set
            mysqli_free_result($result);
            $ispasswordsame = password_verify($decryptedString,
$PasswordR);
            if ($ispasswordsame == true){
                $PasswordREncrypted = openssl_encrypt($PasswordR,
$cipher, $secret_key, 0, $secret_iv);
                $return["exists"] = true;
                $return["profile"] = array($FirstNameR, $LastNameR,
$EmailR, $PasswordREncrypted);
            }
            else{
                $return["error"] = false;
                $return["message"] = "Email or password incorrect";
                $return["exists"] = false;
                $return["profile"] = null;
            }
        } else{
            $return["error"] = false;
        }
    }
}

```

```

        $return[ "message" ] = "No matching profiles";
        $return[ "exists" ] = false;
        $return[ "profile" ] = null;
    }
} else{
    $return[ "error" ] = true;
    $return[ "message" ] = "Failed profile search";
}

}else{
    $return[ "error" ] = true;
    $return[ "message" ] = 'Send all parameters.';
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

register.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDFIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "hash" ] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);
//connecting to database server

```

```

$val = isset($_POST["firstname"]) && isset($_POST["lastname"])
&& isset($_POST["email"]) && isset($_POST["password"]);

if($val){
    //checking if there is POST data
    $firstname = $_POST["firstname"]; //grabing the data from headers
    $lastname = $_POST["lastname"];
    $email = $_POST["email"];
    $password = $_POST["password"];

    $secret_iv = '5fd0b31f582beb1f';
    $secret_key = 'e16a3f356b2366c1';
    $cipher = "AES-128-CBC";
    $decryptedString = openssl_decrypt ($password, $cipher,
$secret_key, 0, $secret_iv);
    $hashpassword = password_hash($decryptedString, PASSWORD_DEFAULT);
    $base64hash = base64_encode($hashpassword);
    $encryptedhashedpassword = openssl_encrypt ($base64hash, $cipher,
$secret_key, 0, $secret_iv);
    $return[ "hash" ] = $encryptedhashedpassword;

    if($return[ "error" ] == false){
        $firstname = mysqli_real_escape_string($link, $firstname);
        $lastname = mysqli_real_escape_string($link, $lastname);
        $email = mysqli_real_escape_string($link, $email);
        $hashpassword = mysqli_real_escape_string($link,
$hashpassword);
        //escape inverted comma query conflict from string

        $sql = "INSERT INTO profile SET
                FirstName = '$firstname',
                LastName = '$lastname',
                Email = '$email',
                Password = '$hashpassword'";

        $res = mysqli_query($link, $sql);
        if($res){
            //write success
        }else{
            $return[ "error" ] = true;
            $return[ "message" ] = "Database error";
        }
    }
}

```

```

        }
    }else{
        $return[ "error" ] = true;
        $return[ "message" ] = 'Send all parameters.';
    }

    mysqli_close($link); //close mysqli

    header('Content-Type: application/json');
    // tell browser that its a json data
    echo json_encode($return);
    //converting array to JSON string
?>

```

restaurantlist.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurants" ] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["sort"]);

if($val){
    $sort = $_POST["sort"];

    if ($sort == "id"){

        $sql = "SELECT res.restaurantid, res.restaurantname,
res.restaurantlogo, res.restaurantbanner FROM restaurant res LIMIT 20";

```

```

        if($result = mysqli_query($link, $sql)){
            while($row = mysqli_fetch_assoc($result)) {
                array_push($return["restaurants"],
                [$row["restaurantid"], $row["restaurantname"], $row["restaurantlogo"],
                $row["restaurantbanner"]]);
            }
            mysqli_free_result($result);
        }
    }
    else{
        $return["error"] = true;
        $return["message"] = "sort method not specified";
    }
}
else{
    $return["error"] = true;
    $return["message"] = "data not specified";
}
mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

restaurantmenucategories.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";
$return["restaurantcategories"] = array();

```

```

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["restaurantid"]);

if($val){
    $resid = $_POST["restaurantid"];

    $sql = "SELECT categoryname, categoryid FROM menucategories WHERE
restaurantid = '$resid' ORDER BY viewingorder ASC";
    if($result = mysqli_query($link, $sql)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($return[ "restaurantcategories" ],
[$row["categoryname"], $row[ "categoryid" ]]);
        }
        mysqli_free_result($result);
    }
    else{
        $return[ "error" ] = true;
        $return[ "message" ] = "restaurant not sepcified";
    }
}
else{
    $return[ "error" ] = true;
    $return[ "message" ] = "data not specified";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

restaurantmenuitems.php

```

<?php

$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "menuitems" ] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["categoryid"]);

if($val){
    $rescategory = $_POST["categoryid"];

    $sql = "SELECT itemid, foodcategory, subfoodcategory, itemname,
description, price, itemimage FROM item WHERE categoryid = '$rescategory'
ORDER BY itemid ASC";
    if($result = mysqli_query($link, $sql)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($return[ "menuitems" ], [$row[ "itemid" ],
$row[ "foodcategory" ], $row[ "subfoodcategory" ], $row[ "itemname" ],
$row[ "description" ], $row[ "price" ], $row[ "itemimage" ]]);
        }
        mysqli_free_result($result);
    }
    else{
        $return[ "error" ] = true;
        $return[ "message" ] = "category not specified";
    }
}
else{
    $return[ "error" ] = true;
    $return[ "message" ] = "data not specified";
}

mysqli_close($link); //close mysqli

header( 'Content-Type: application/json' );

```

```
// tell browser that its a json data  
echo json_encode($return);  
//converting array to JSON string  
?>
```

Summary

During the creation of prototype 1A, I have successfully made an application that allows you to see a list of restaurants and view what items they serve, giving you the option to favourite the restaurants which you like. As well, the application allows you to create and login to profiles with the option to switch between the profiles which you have logged in to. I have also made a button which allows you to either select your location manually or automatically based upon your current location. This feature will be used in the future to sort the restaurant by distance as well as for ordering your food.

Many measures were also put in place to prevent app crashes, using error catches in order to show the user what has gone wrong so that they can get the app working again.

Evaluation

Before the making of the app, I had little experience with the language Dart but over the course of prototype 1A, I ended up developing my skills at building applications, learning how to create new screens, custom app bars, use AES encryption and sending data over the HTTP protocol.

prototype 1A was all about creating the visual experience for the user and getting the relationships between the server and the client app working. This was completed successfully, with the user being able to create their profile and it storing it successfully. The tables used 3rd normal form allowing for a relationship between the restaurant, its categories and their items. This not only saves time but also computing time since it the server only needs to process the same data once.

During the development of the application, many databases were made. In the future, I will only make a table once I need it. While developing the database in prototype 1A, I found that I kept making alterations to the app features and usability causing the database to have to be remade from scratch.

Since the app is expected to get a lot larger, comments have been added in order to better interpret the code and not need to read multiple files to understand where you are. For now, the parts worked on has been tested against the success criteria and after some iteration, has passed all the checks which means I will move onto the next section. In prototype 2 I aim to build on the current app by implementing item customisation, cart and checkout pages, the custom home page and filtering.

Prototype 1B

Design

After analysing the main application from prototype 1A, it has been decided that it will be redesigned in order to make it more user-friendly. As well, with the development from prototype 1A, I found that it had many issues. These include

- Duplicated encryption
- No theming (eg. dark mode and light mode)
- Inefficient algorithms
- Dead code

In order to fix these issues I designed how the app will look in final iterations. This aided in future development by allowing me to have a reference on how to build each part.

In prototype 1B I will use polymorphic techniques in order to remove the amount of duplicate code as well as redesign the app to have a more user-friendly design.

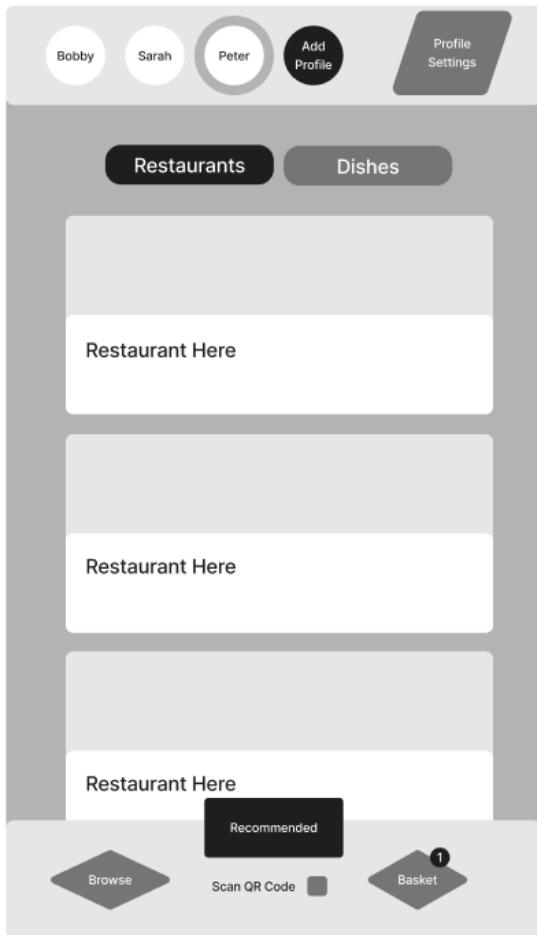
Concept Art

To design it, I used a prototyping software called Figma; this software allows you to use shapes and boards to recreate how the app would look.

Over the next month, I spent a month creating multiple concepts until I eventually made one that I liked and had all the potential to be an app.

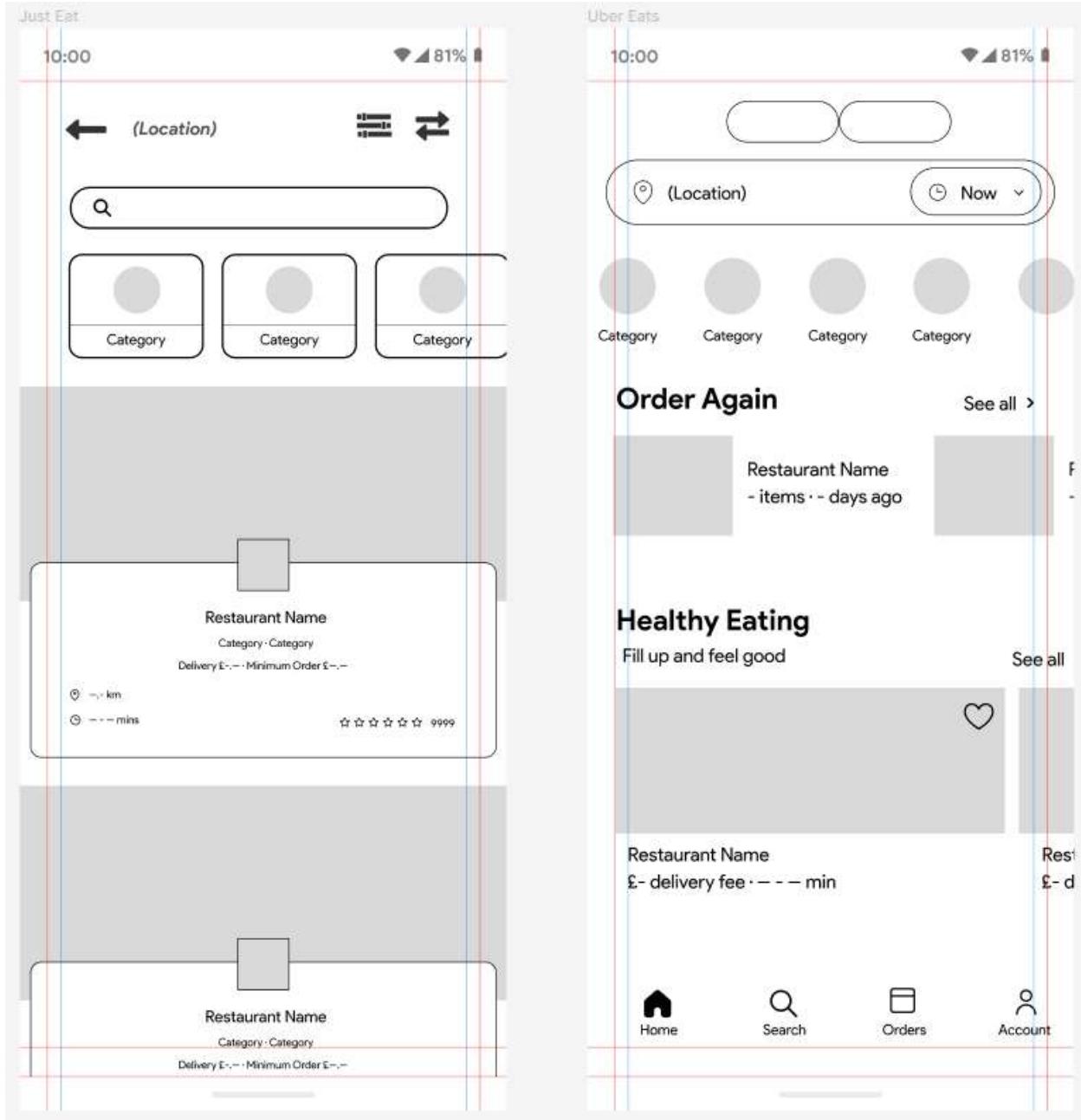
Concept 1

For this one, I was just trying to get a picture of how food delivery apps work, basing it off Uber Eats. This was never meant to be the final concept so I went back to work designing the next one



Concept 2

After concept 1, I decided to create a box layout of the leading food delivery apps and how they are both good and bad



What I found was that the restaurant banners usually have priority over the rest, being front and centre. Each restaurant also had the delivery price and most obviously, their name. What I found was that with Uber Eats, they never included the restaurant logo on their browse pages. Taking all this into account, I designed concept 2.

10:00

81%



11 Canal Reach, London
N1C 4BE

P



Try Searching "Burgers"



Pizza



Ramen



American



Papa Johns

Pizza • American



🕒 --- mins

📍 --- km

Delivery £---

Minimum Order £---

★★★★★ 9999

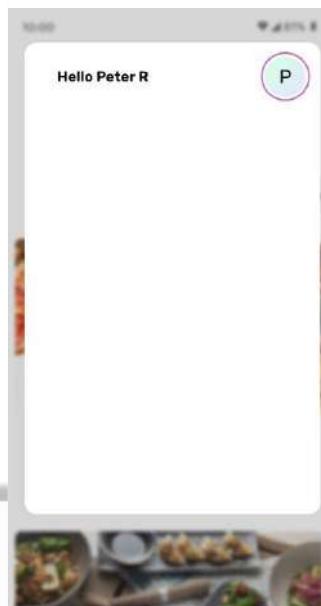


Home

For You

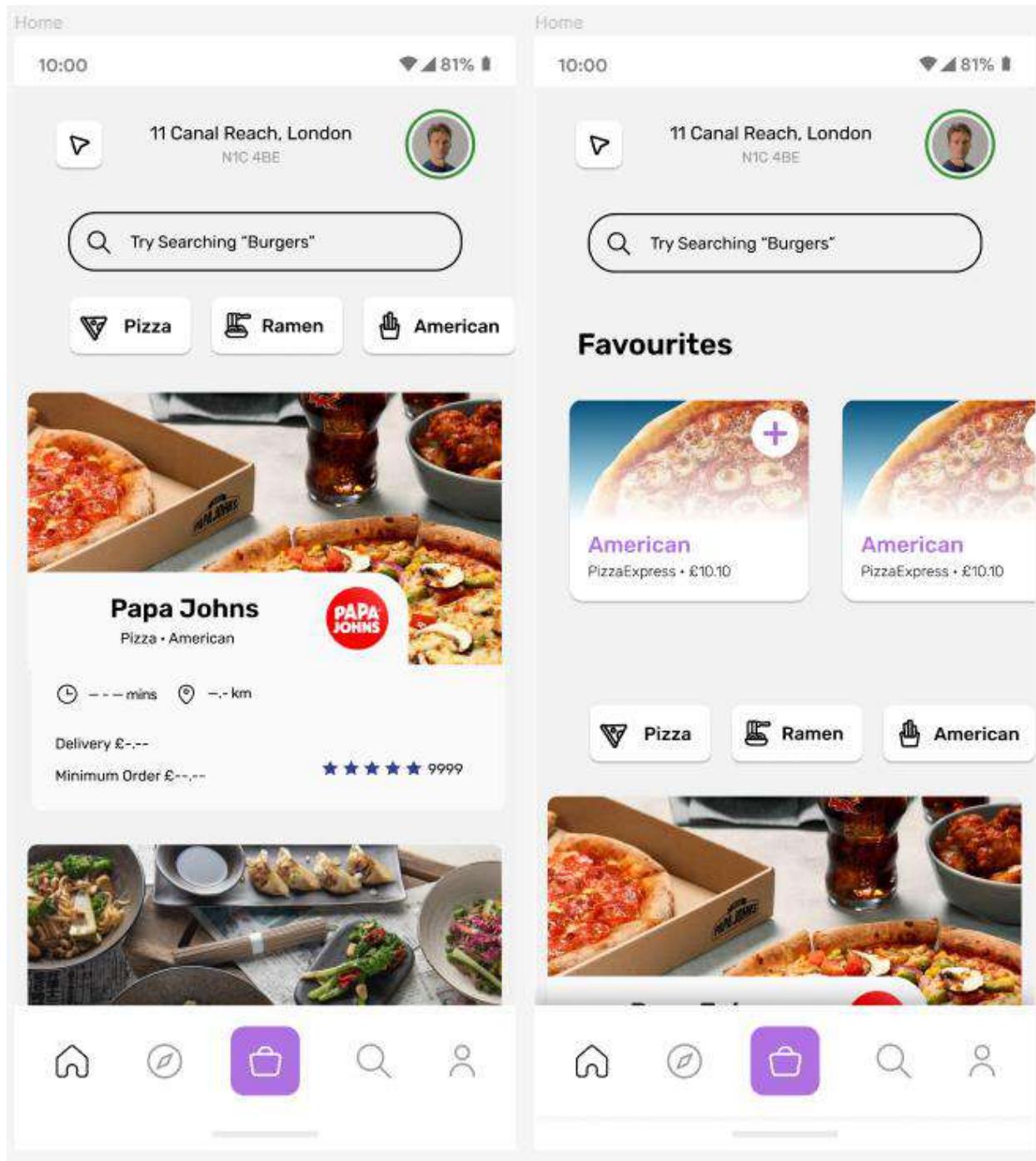


Cart



Concept 3

Concept 3 was done immediately after concept 2 and attempts to fix the menu bar and the colour scheme, there were two different designs for it but both were stopped quickly after making it due to the lack of ideas on how to continue.



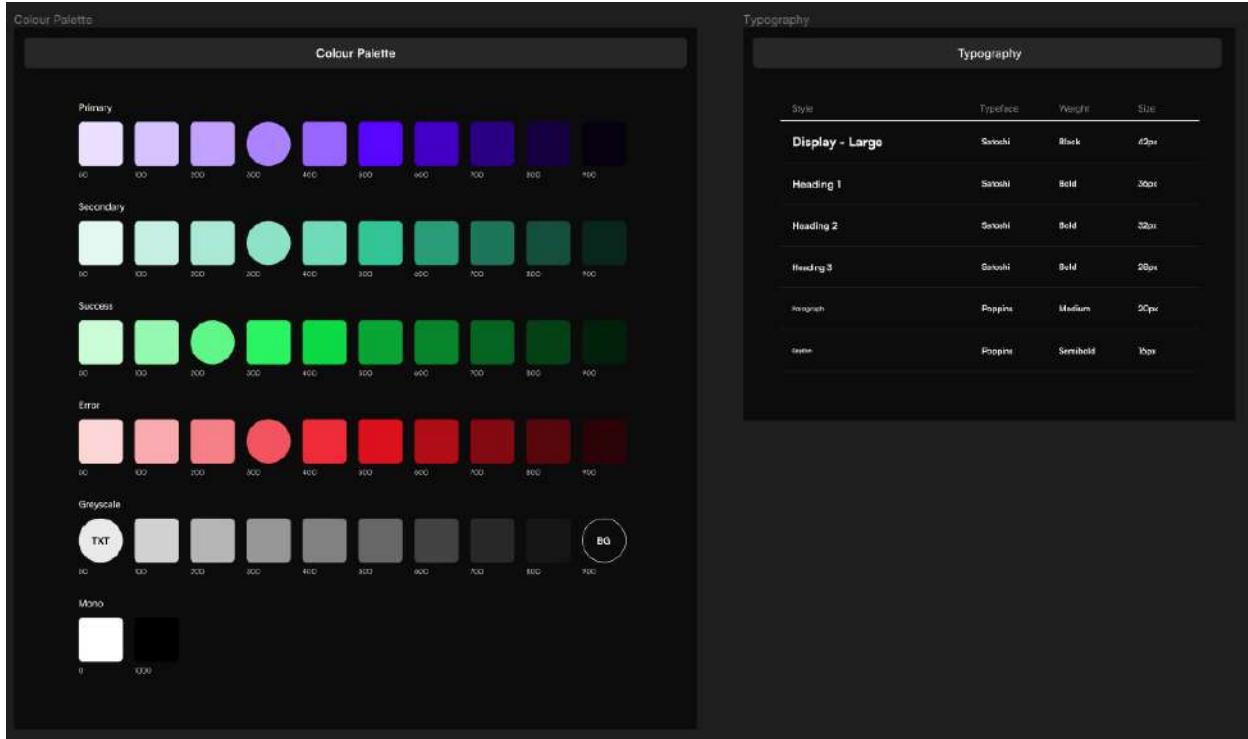
Concept 4 (Final)

When it came to concept 4, I merged all my previous concepts together, taking the best things from each one. From concept 1, I took the restaurant and food buttons which were used to easily change how you view the foods. From concept 2, I took the colour scheme. From concept 3 I took the favourites section and the simple clean menu icons. Putting all of these together, gave me concept 4, a polished and clean design, using a solid foundation.

Theme

Before creating the app boards, I built the theme, allowing me to keep the whole app unified. During this journey of making the theme, I learnt some colour science, learning that you should have values from 50 to 900. 500 would be the solid colour. Anything less than 500 would be a decrease in saturation while anything more than 500 would increase the darkness closer to 100% black and 1000. On top of this when you work with a dark theme instead of using shadows, you use saturation and different tones of grey to portray depth (which is the reason why there is no shadow theme for dark mode).

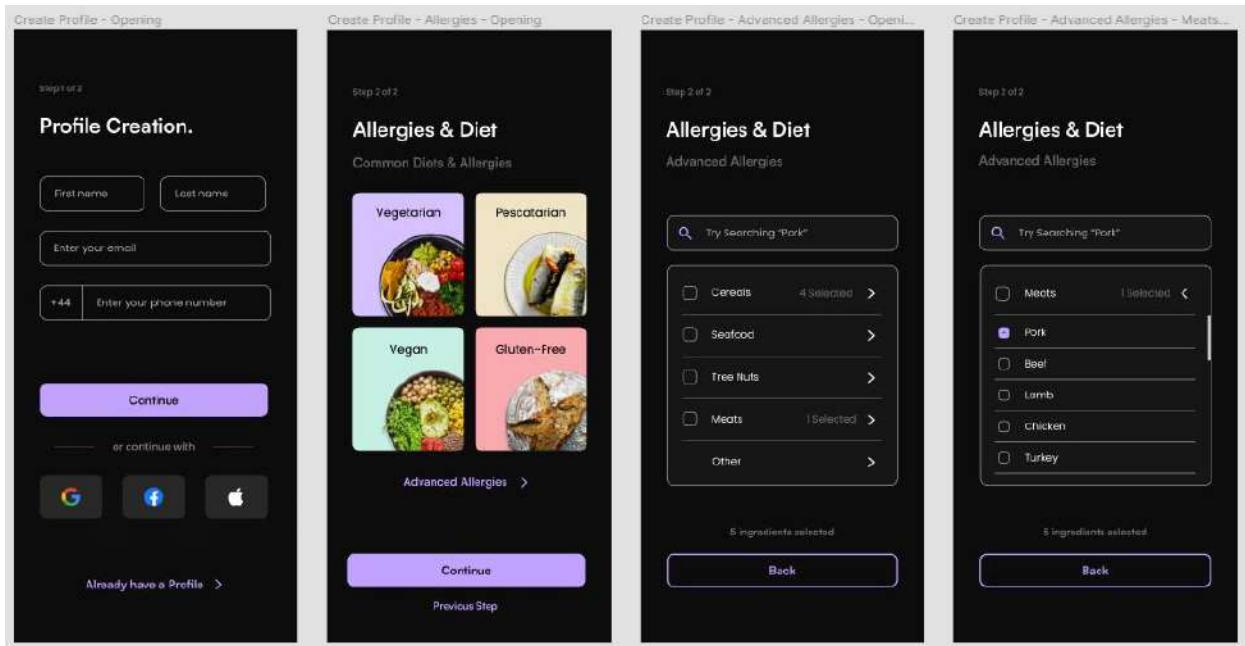
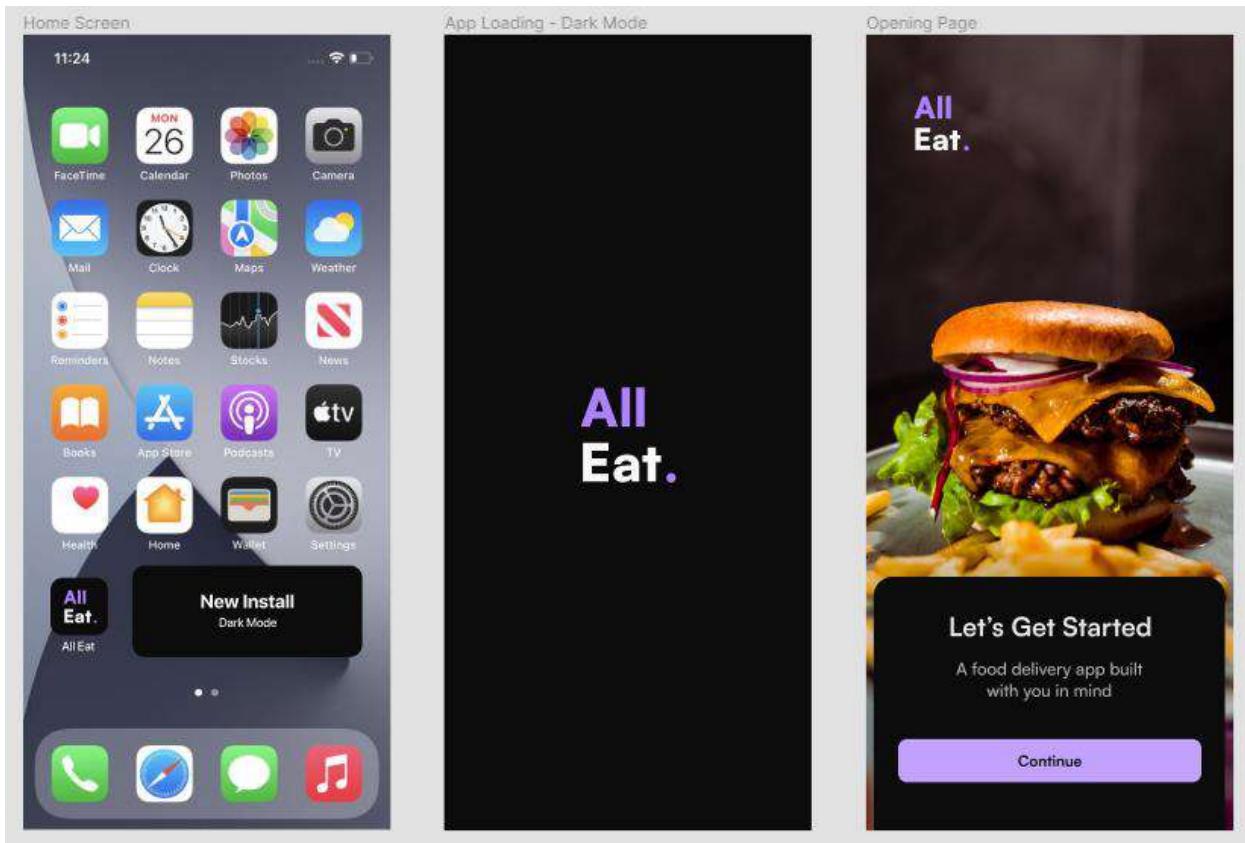
The image displays two design tools side-by-side. The left tool is a 'Colour Palette' editor with sections for Primary, Secondary, Success, Error, Greyscale, and Mono colors. Each section contains a 10-step slider from 50 to 900. The right tool is a 'Typography' editor showing styles like Display - Large, Heading 1, Heading 2, Heading 3, Paragraph, and Body, with details for font family (Sawasdee), weight (Black or Bold), and size (36px, 32px, 28px, 20px, 16px). Below these is a 'Shadows' editor showing four examples with X and Y offsets, blur radius, spread, and opacity values.



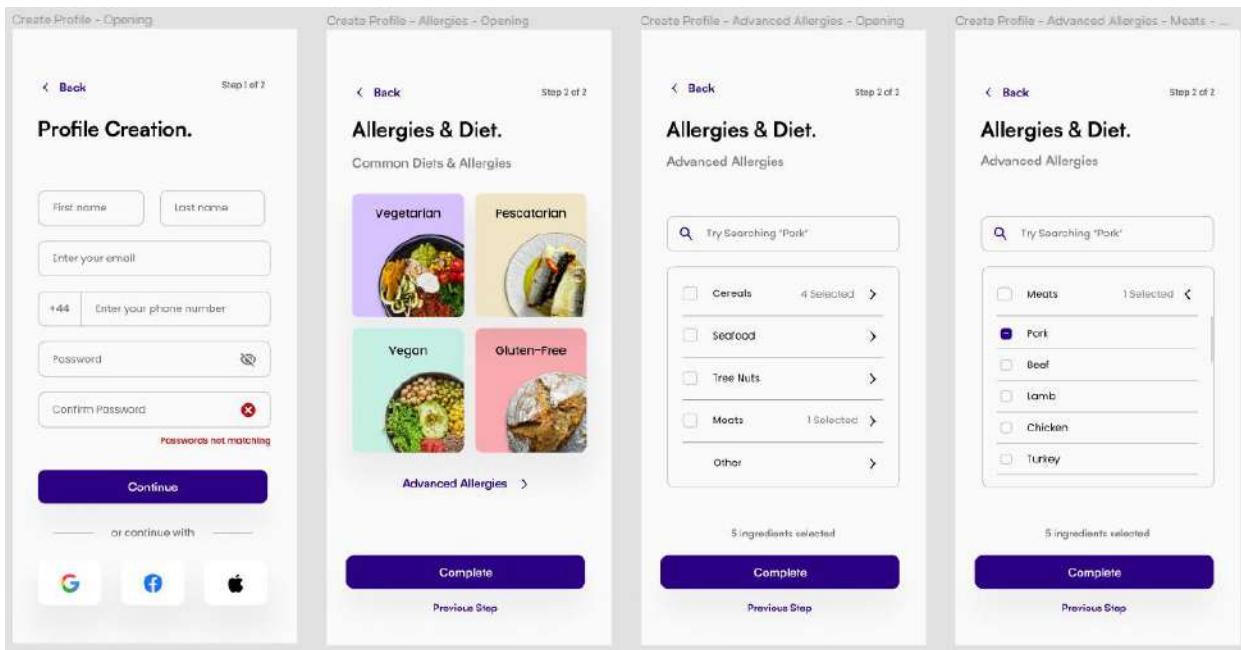
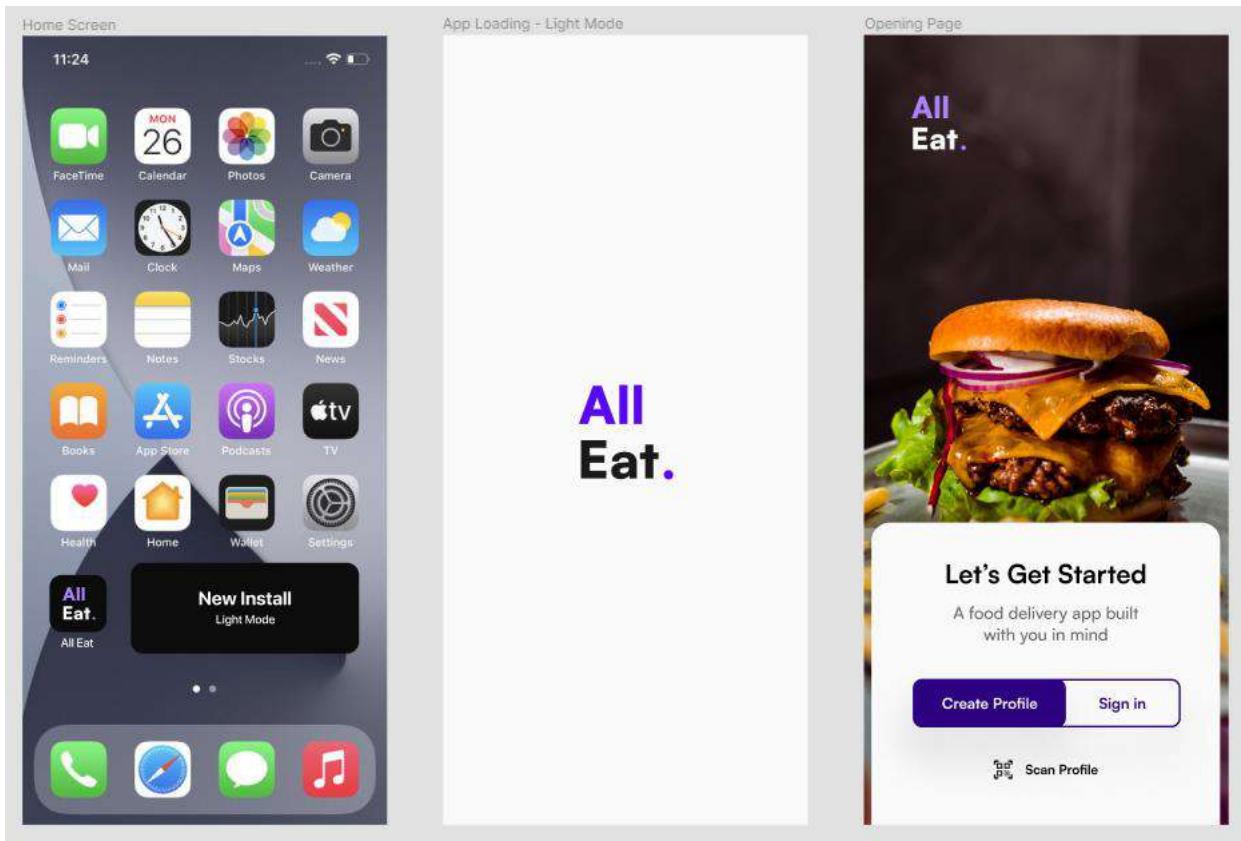
Clean Install

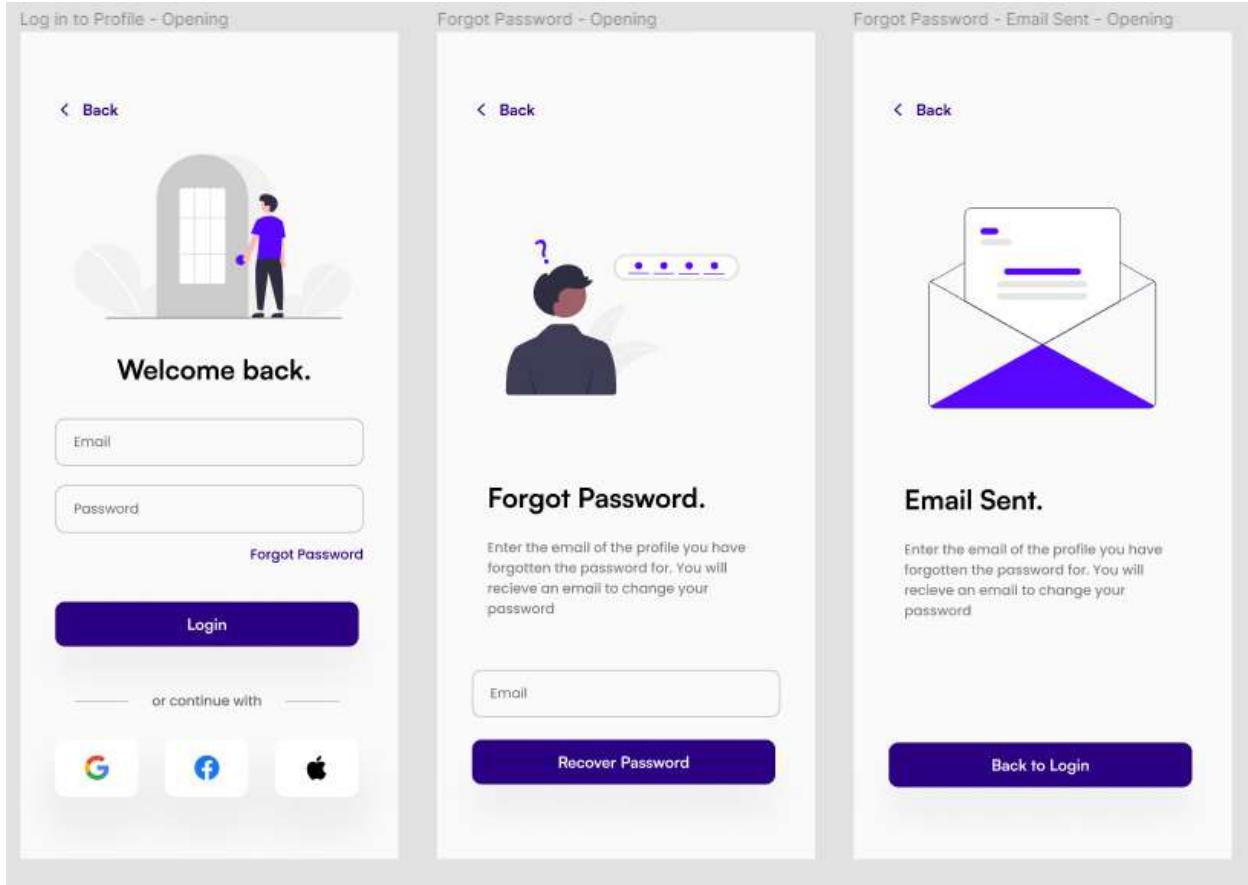
When you first open the app, it should be different from when you reopen it, so I designed boards for this in both light and dark mode. During this time, I also changed the name of the app as when I discussed with colleagues “Allivery” they were either not able to pronounce it correctly or found it off putting. The new name became “All Eat.”, a more simple but easier to understand name.

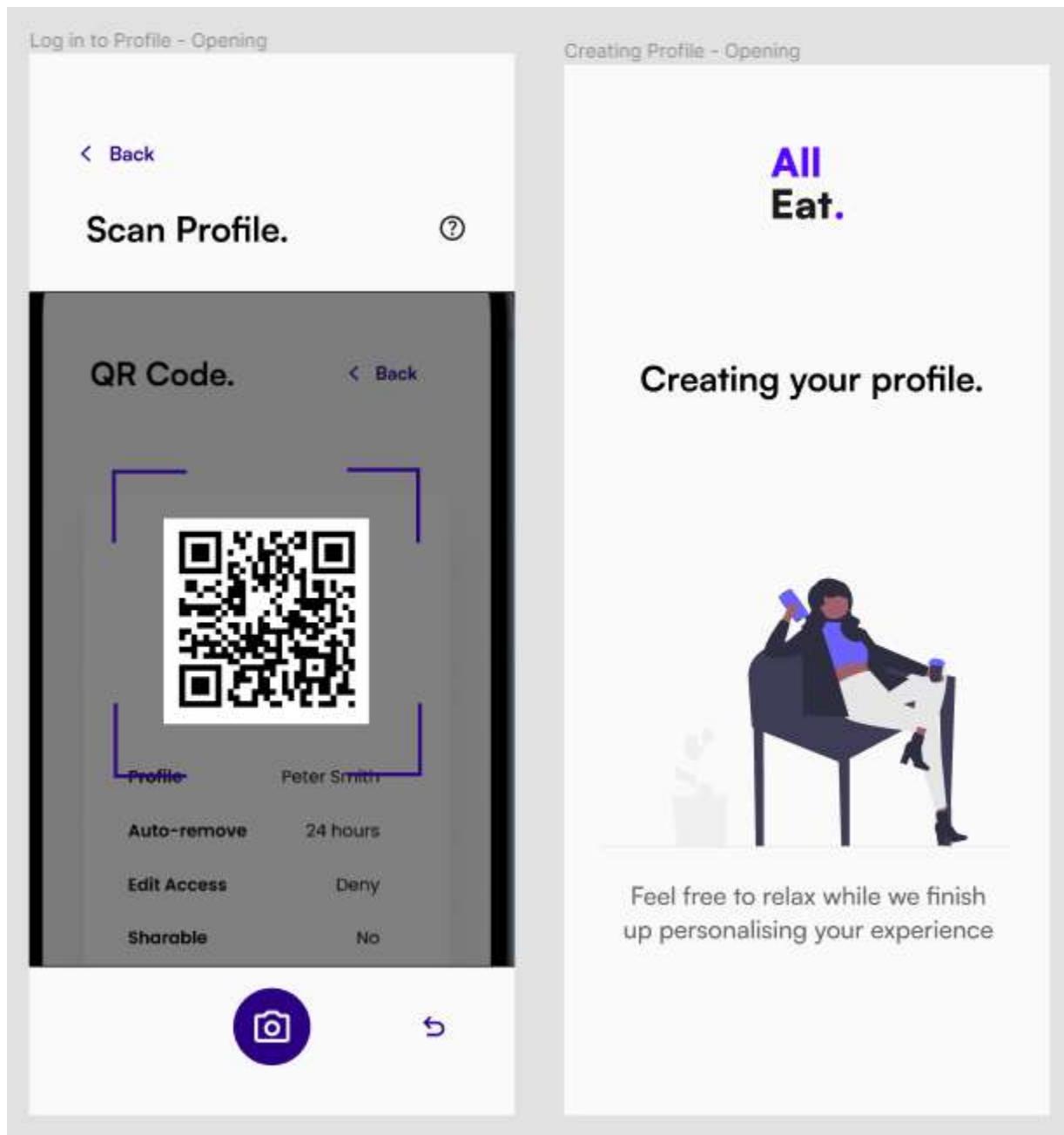
Dark Mode



Light Mode

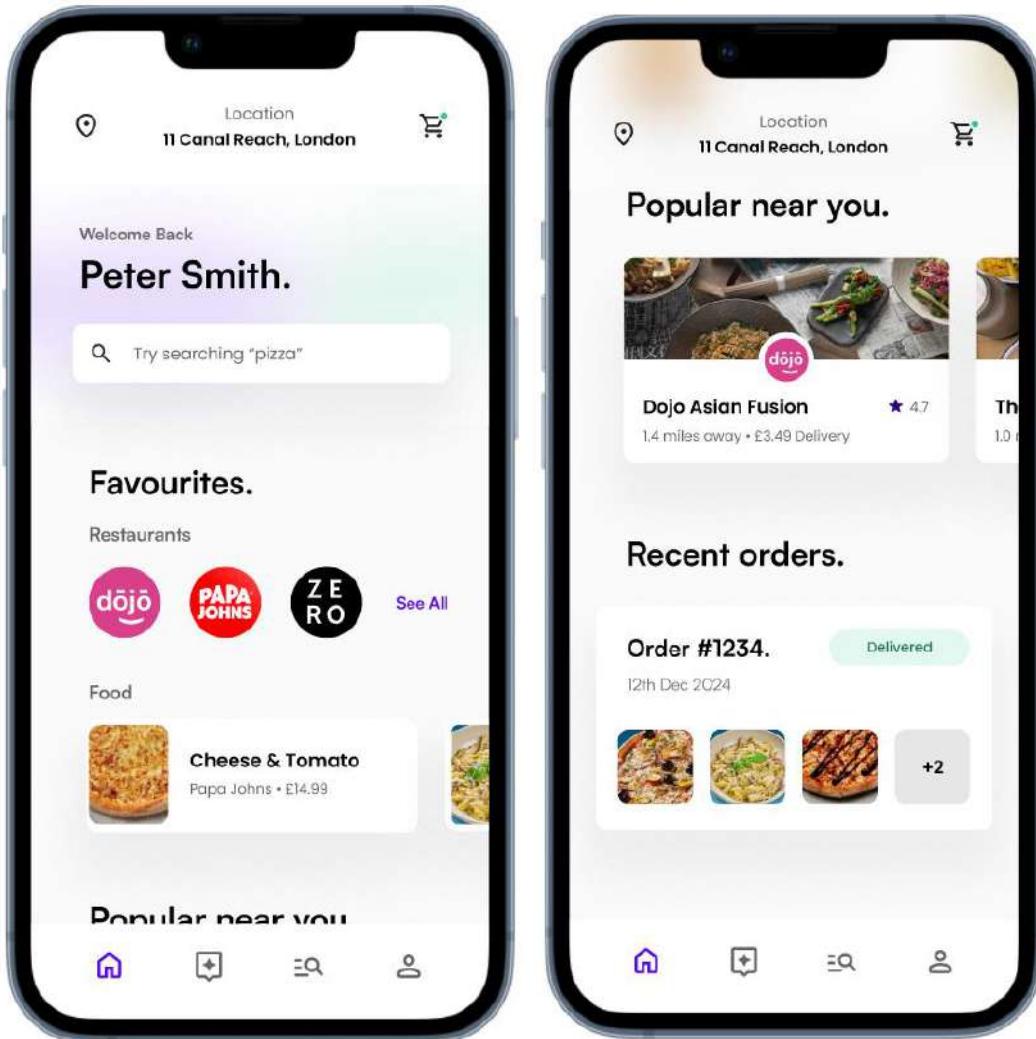






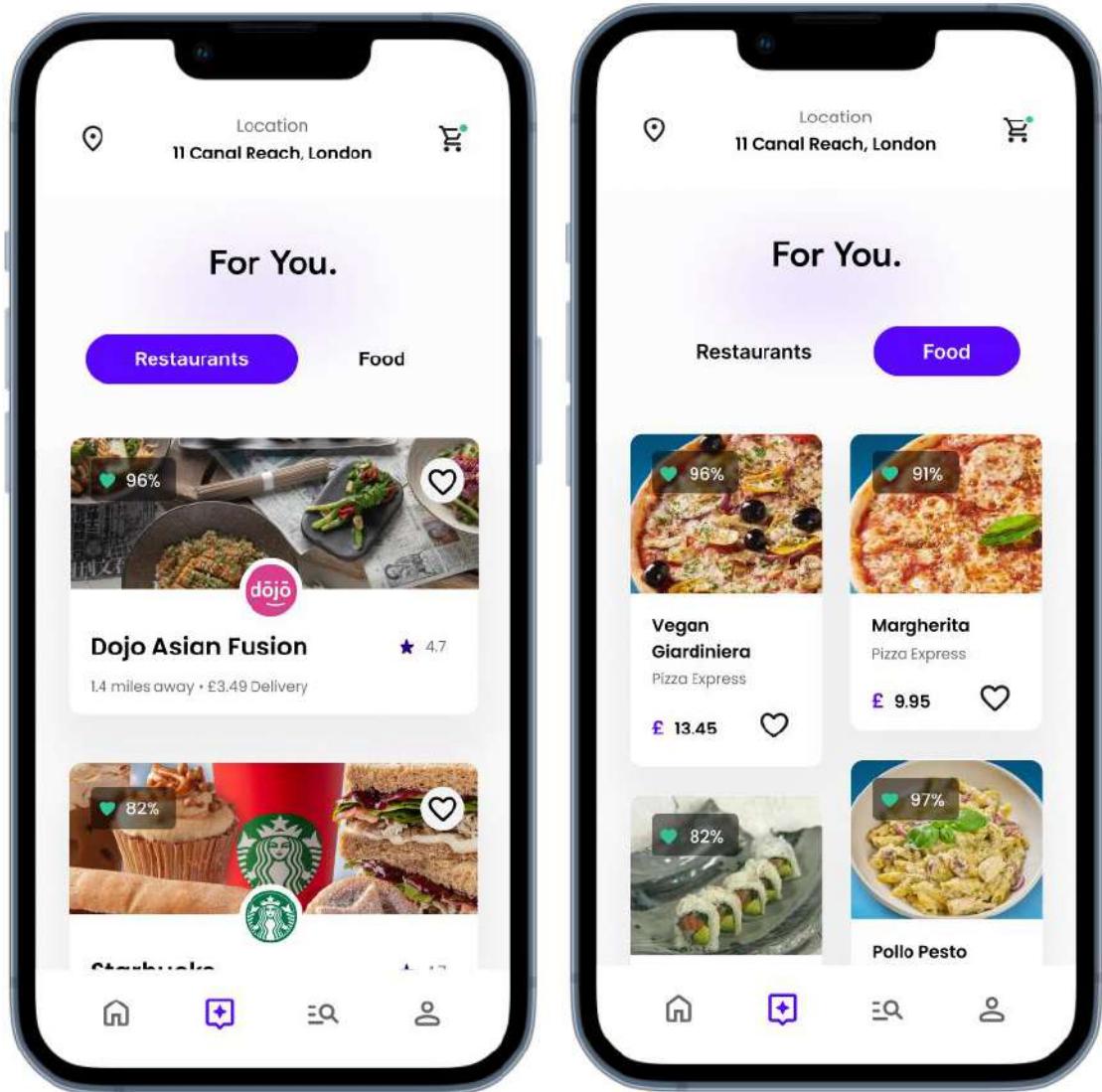
Home Page

The home page contains a quick way to get back to your favourite restaurants and dishes as well as many others. On this page, instead of having restaurants, it is more like a control centre where you can quickly access everything related to that profile.



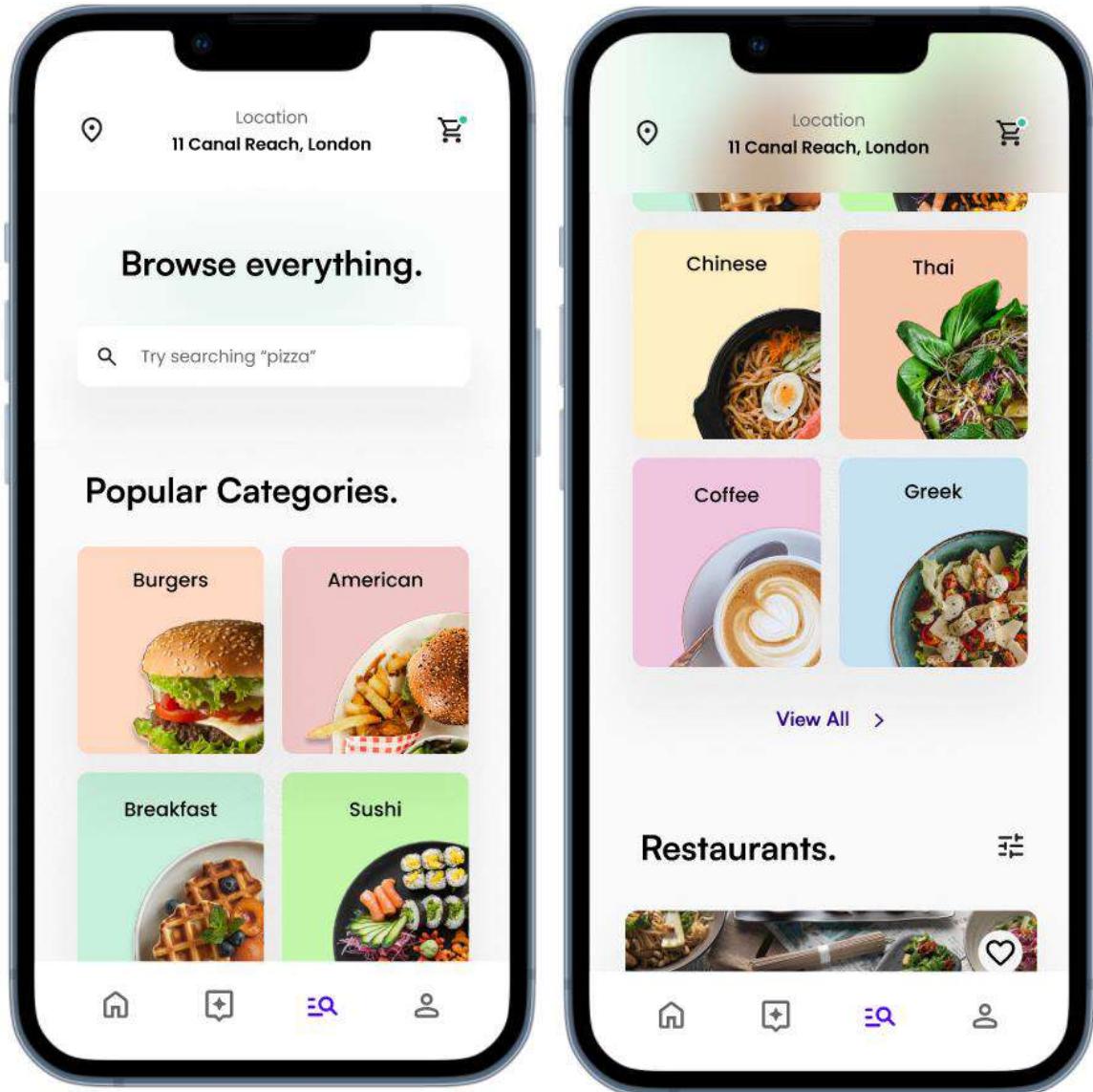
For You Page

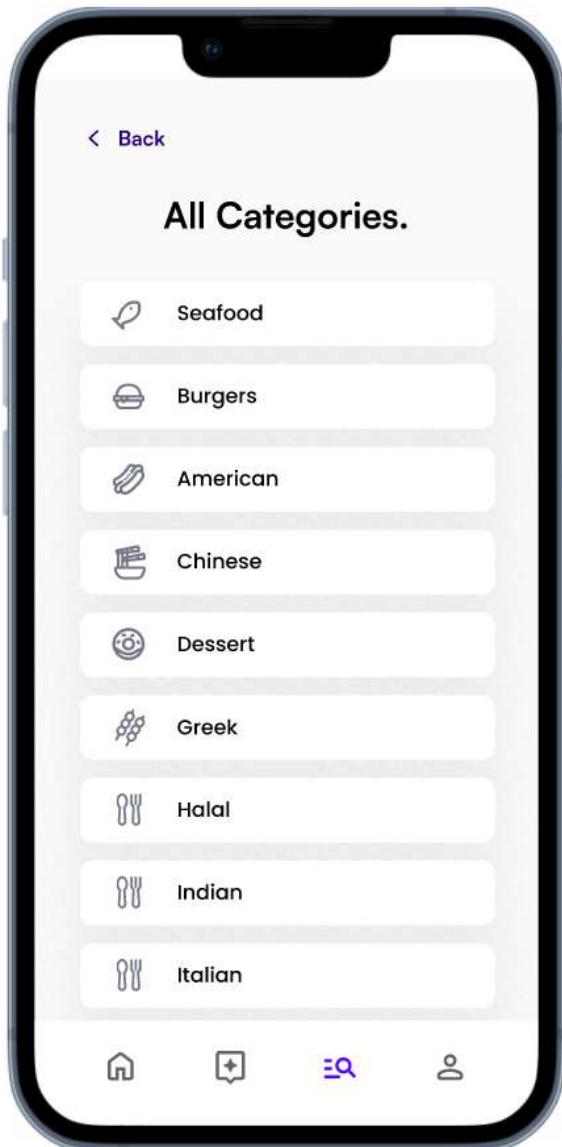
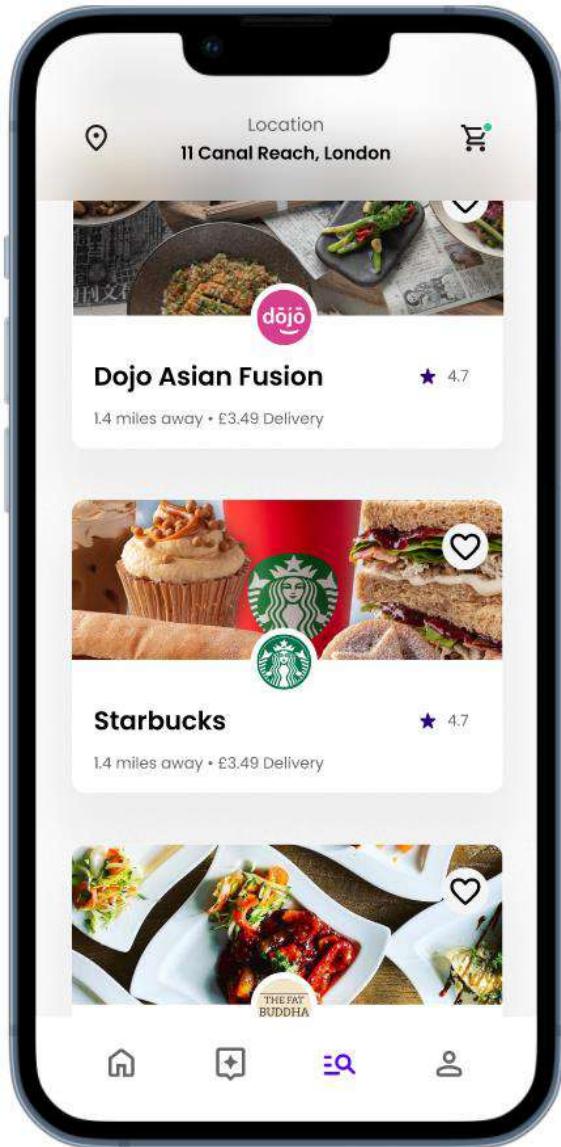
The For you page is a place to find food that the app thinks you will like, in relation to previous orders, allergies/diets and ratings.

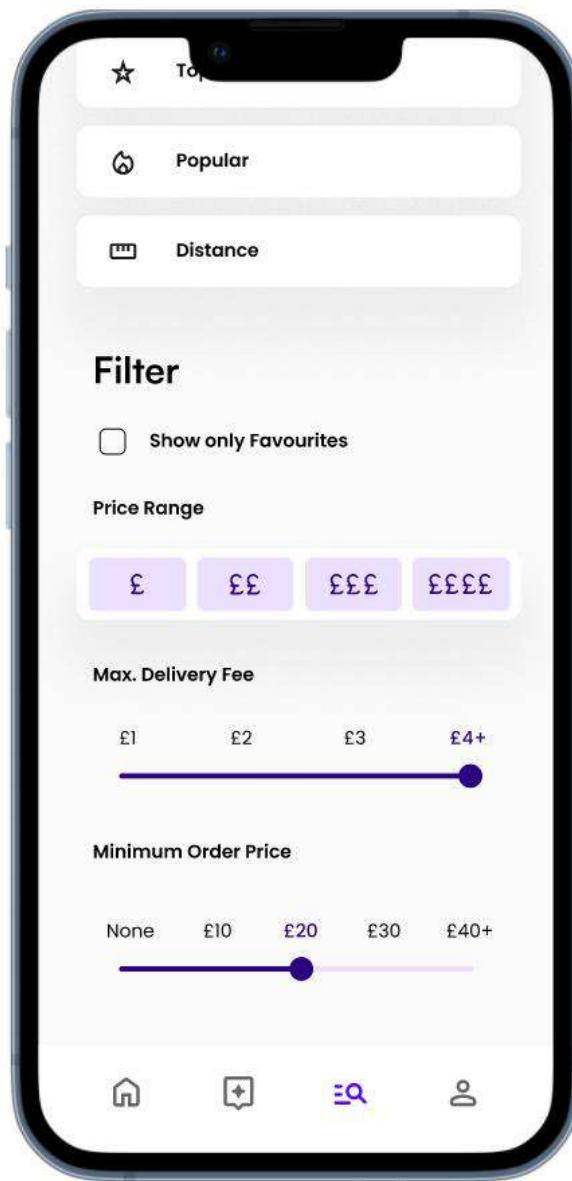
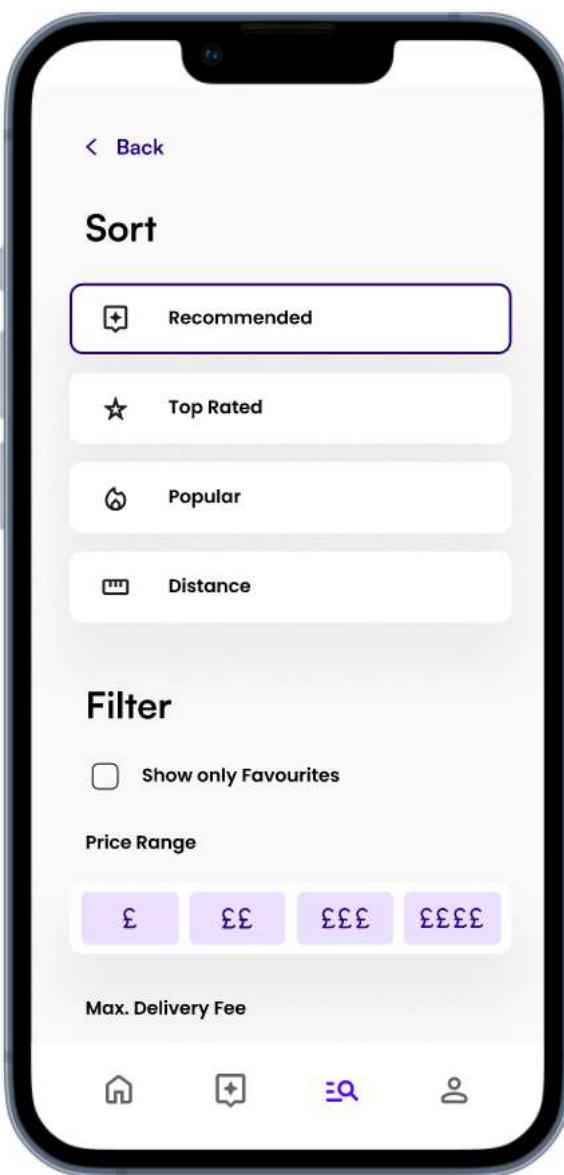


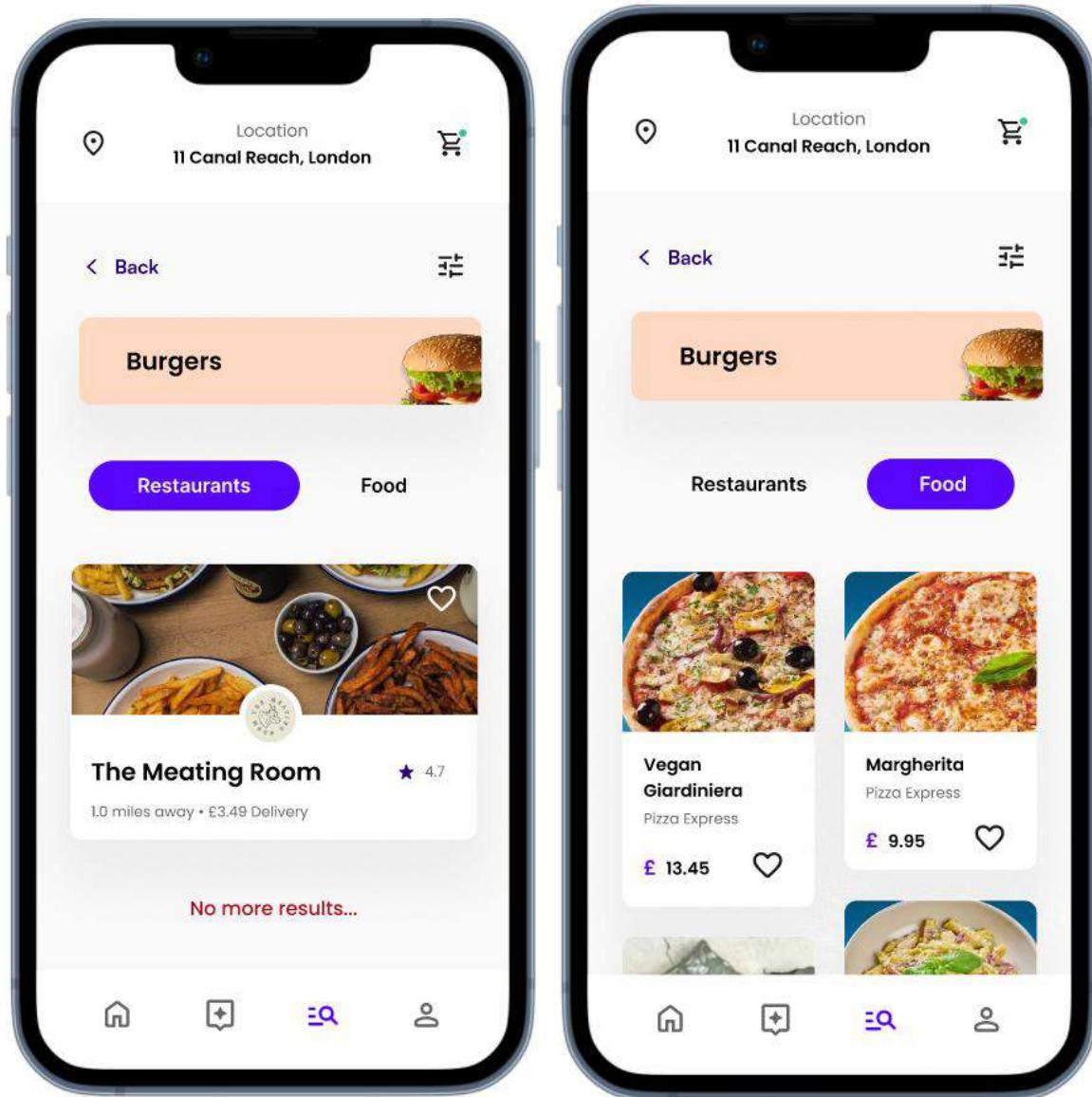
Browse Page

The Browse page is the place to find all the restaurants and to find what you are looking for, either by searching or clicking on a category



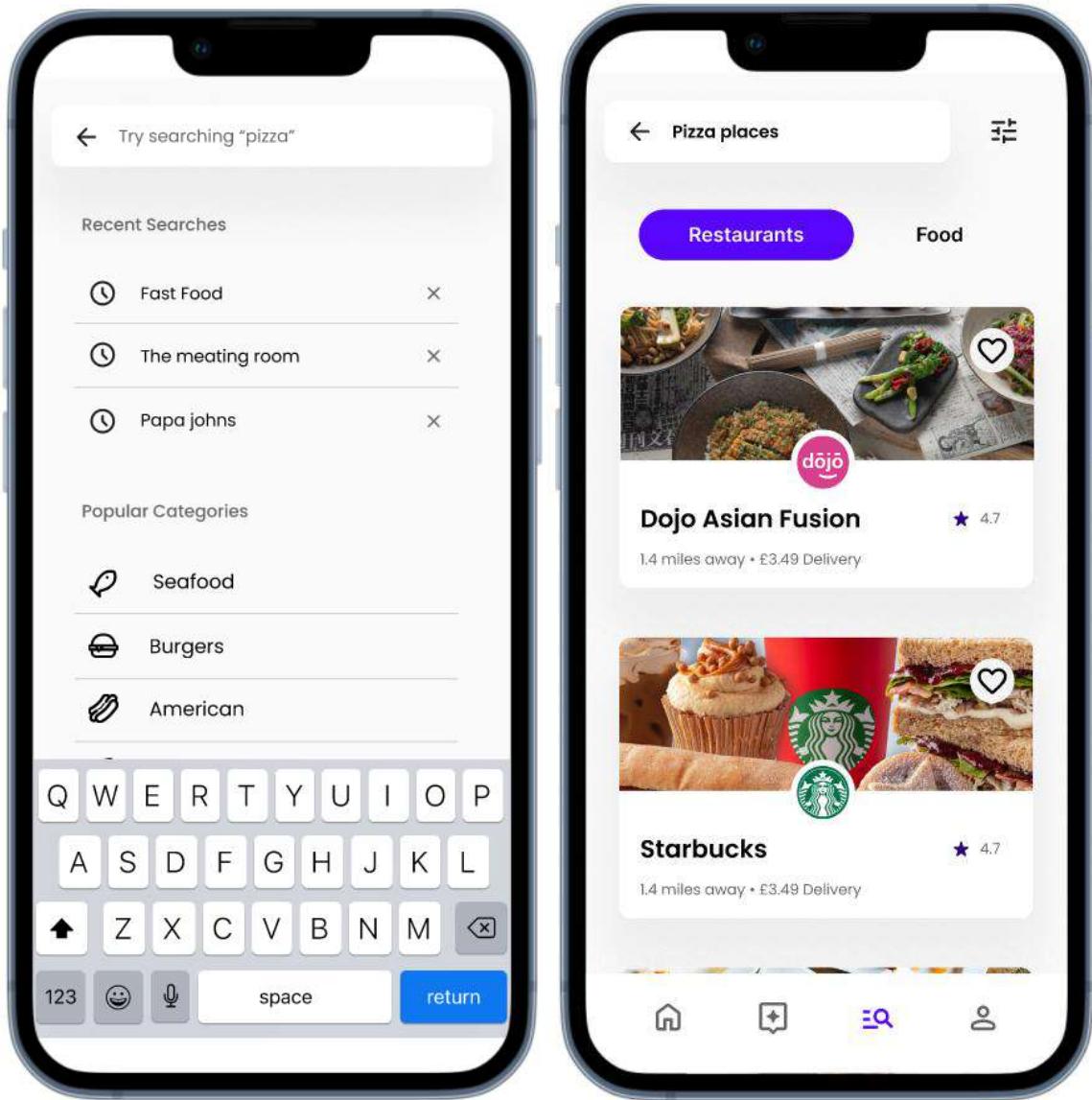






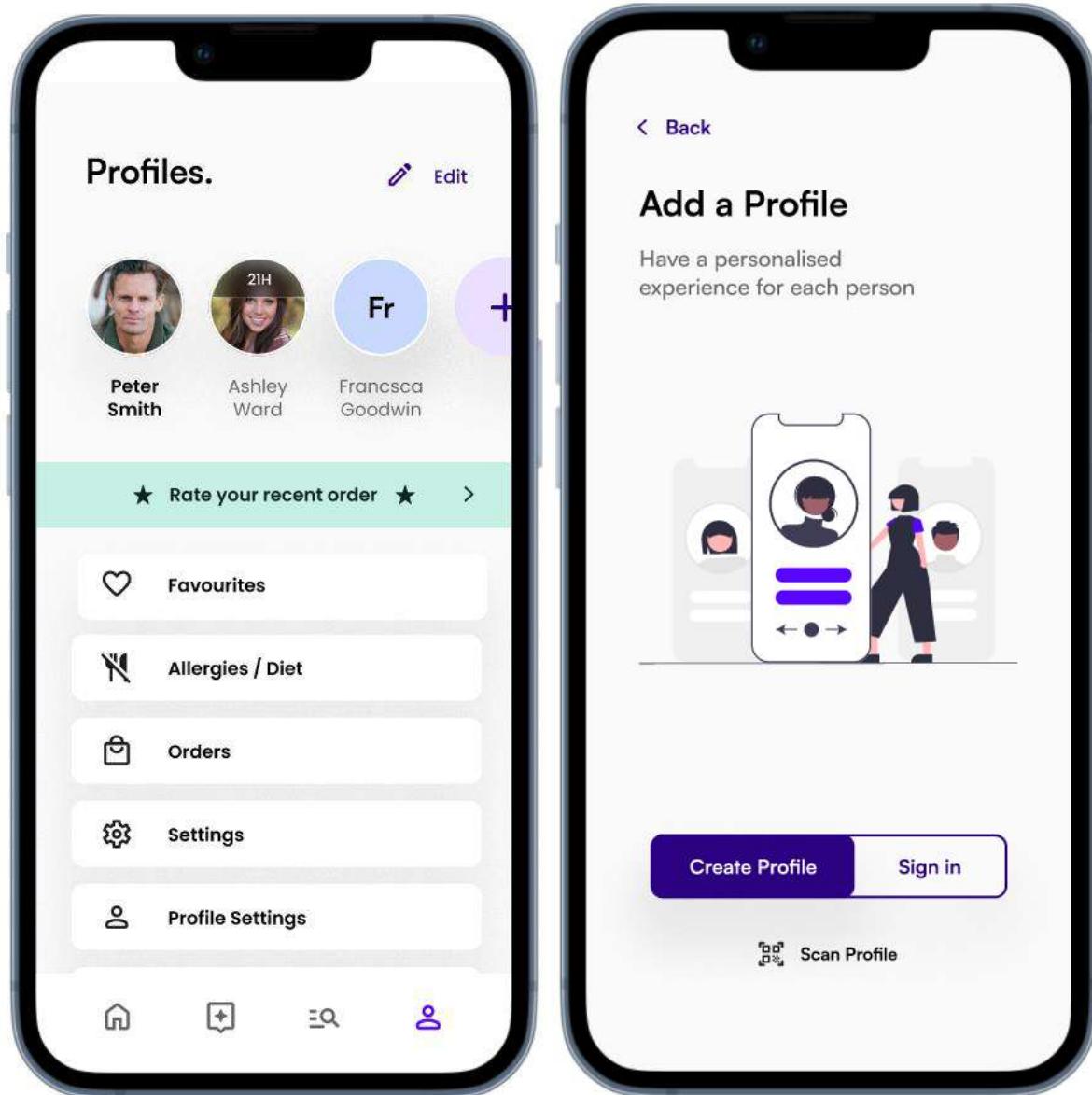
Search Page

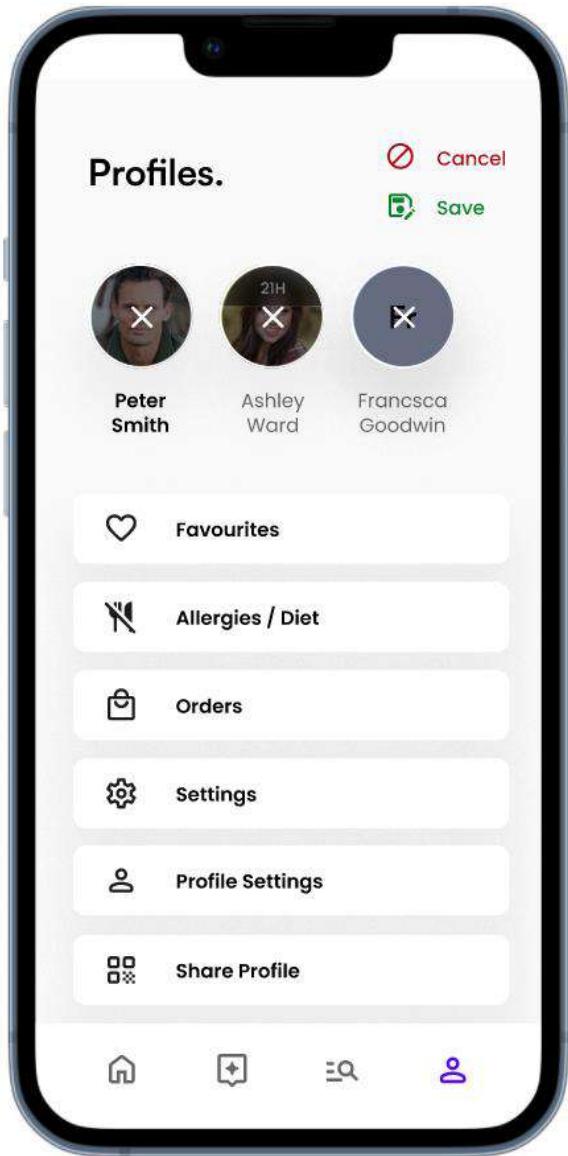
The search section shows previous searches as well as popular categories. Upon searching, it uses the keywords to find the most relevant answers and allows you to either look at restaurants related to the search or food related to it.



Profile Page

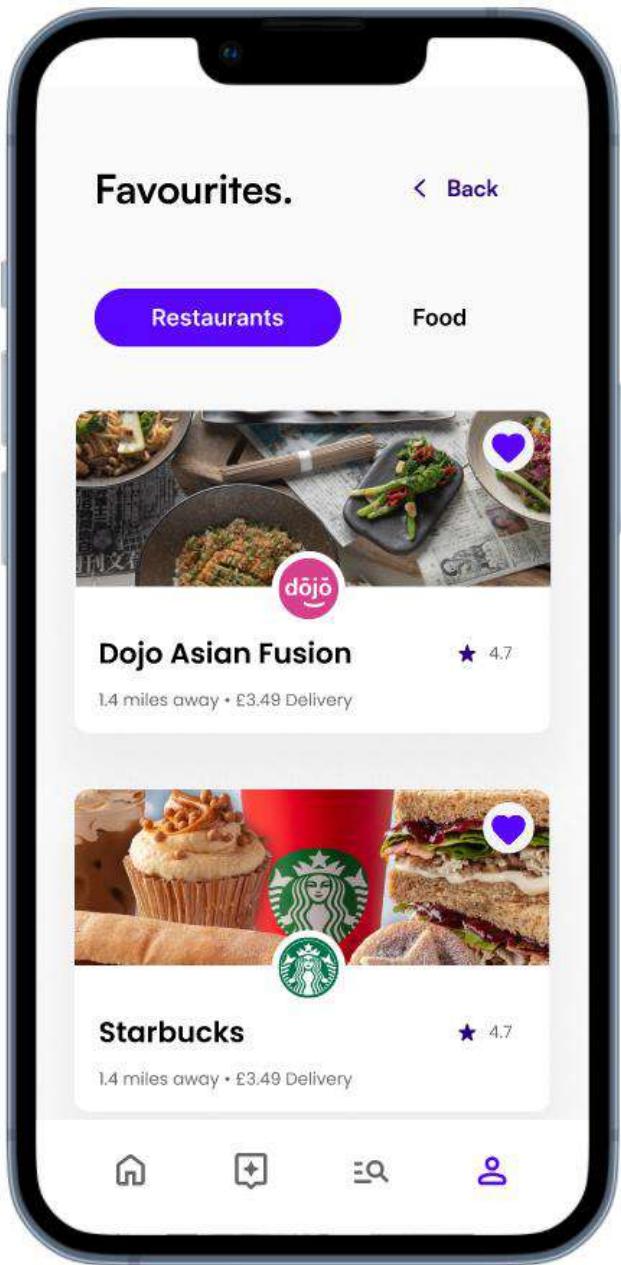
From the profile page, you can do a variety of things including switching profiles and editing your settings. Unlike the home page, this section has all the information as well as managing more than the current selected profile. If you would like to add a profile, just press the add button and it will bring you to a similar section to when you originally opened the app but allows you to go back to the profiles page. On this page, if you haven't reviewed the recent order, it will allow you to review it, a banner indicating that it hasn't been done yet. On top of this, it will only show the foods that were associated with that profile.





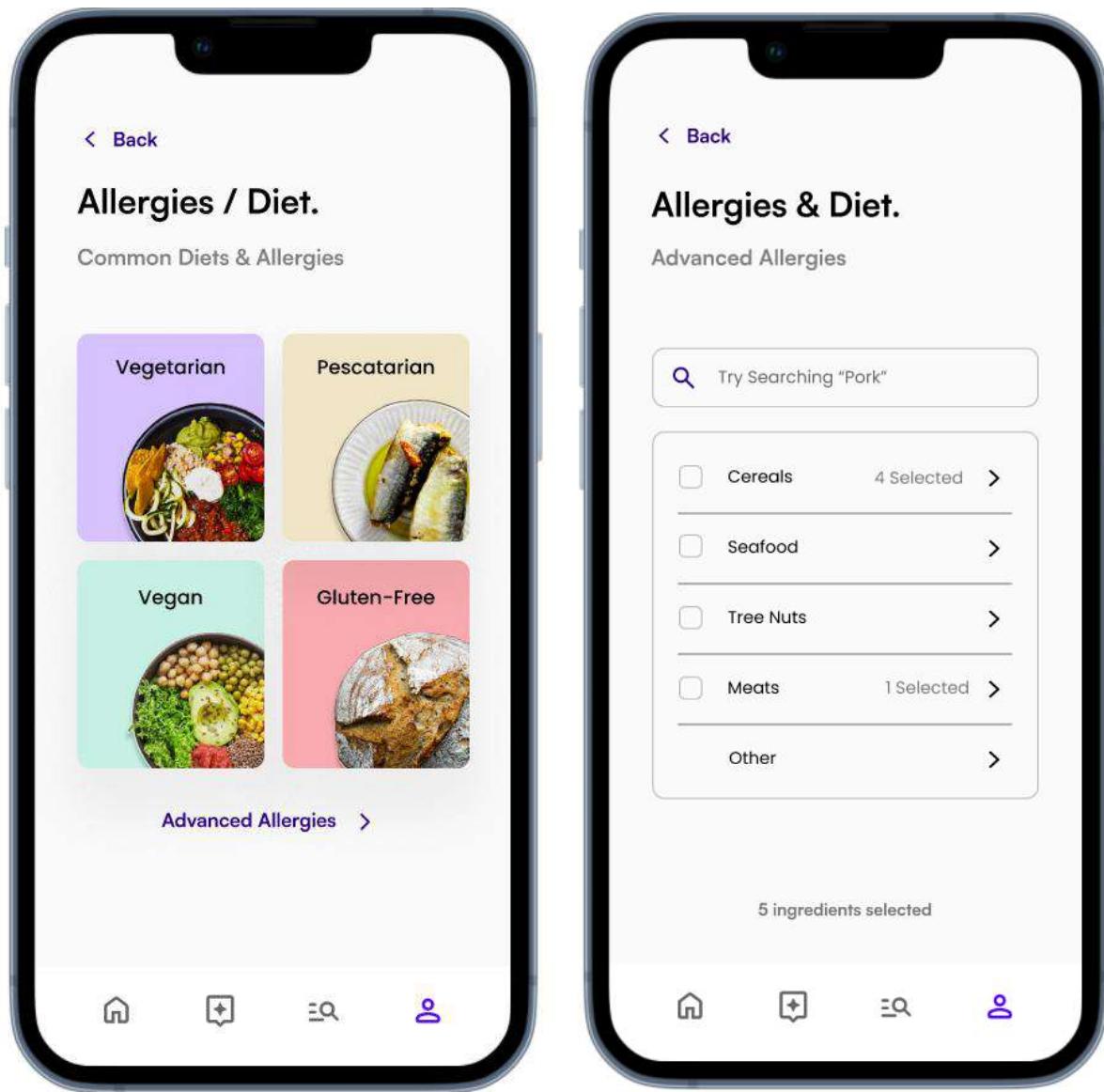
Favourites Page

The favourites page can be used throughout the page and is a quick place to find things you want again. The favourites can be found both here or from the home page where each person has their own favourites. A user can favourite both the food or the restaurant



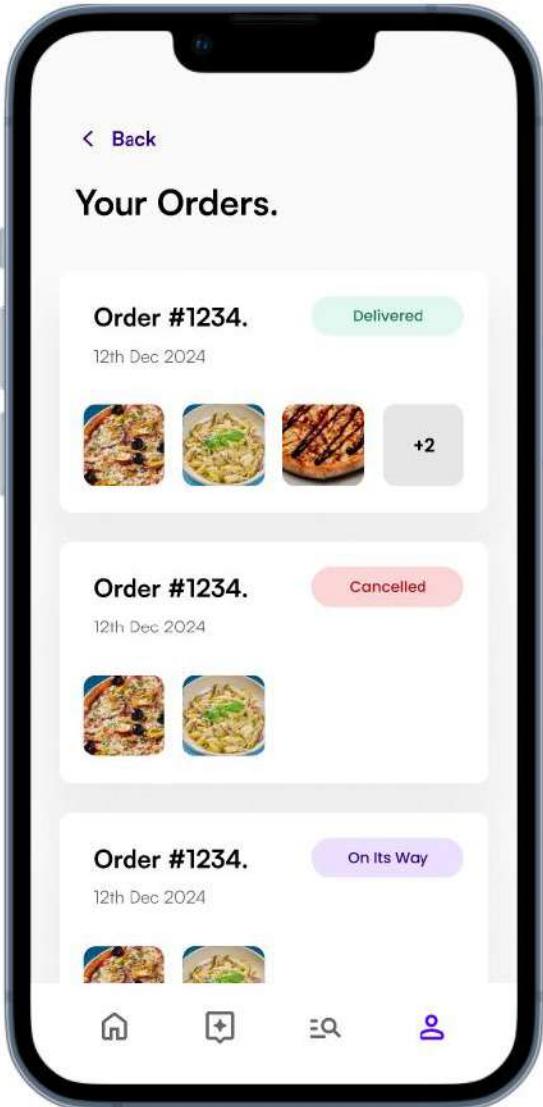
Allergies/Diet Page

The allergies and diet page is first shown when the user creates a profile but can be viewed and edited on this page.



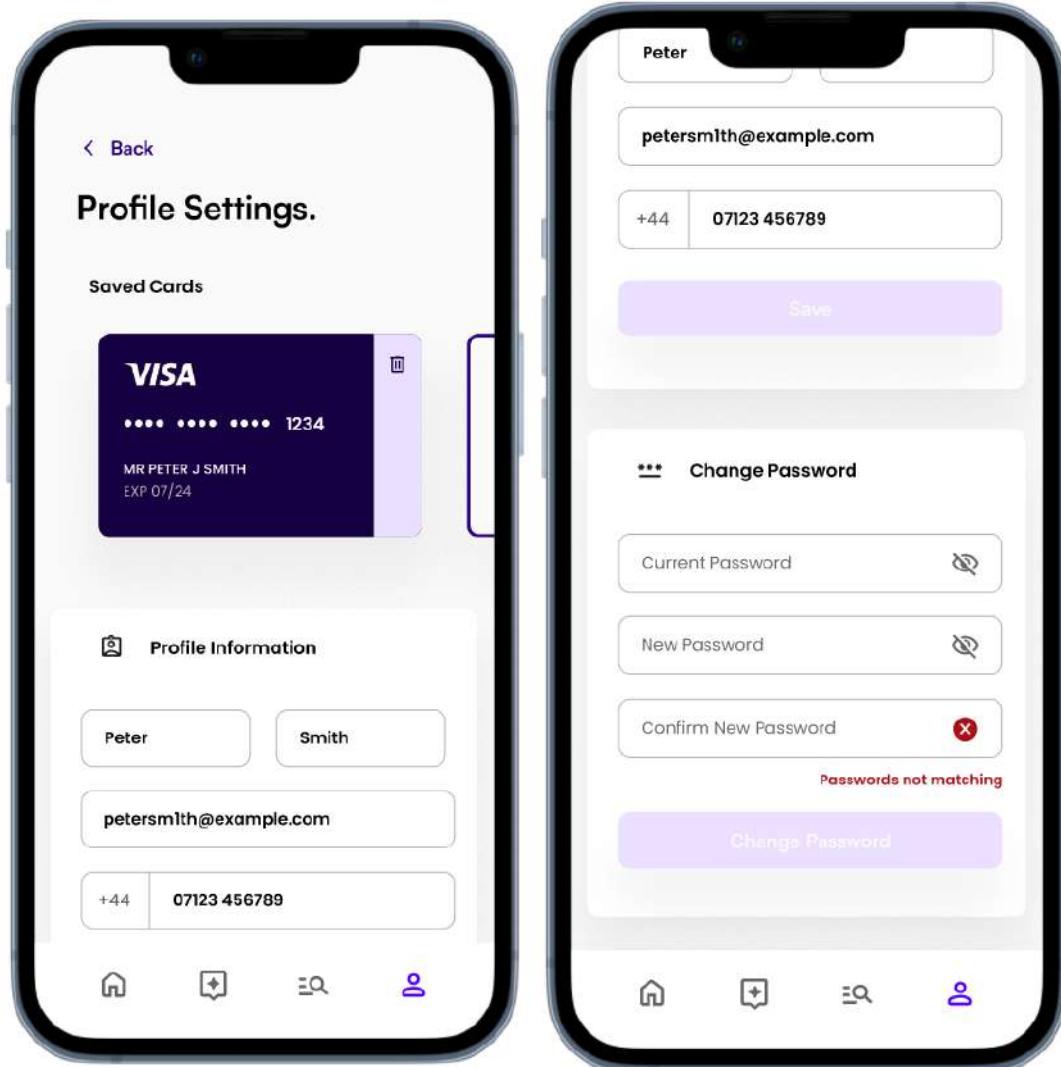
Your Orders Page

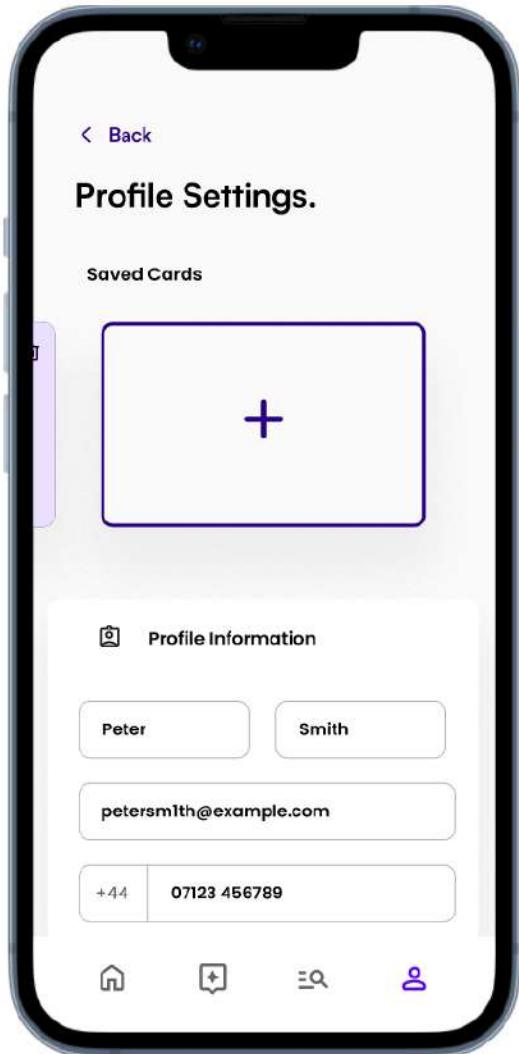
On this page you can see all your previous orders including those that are not complete yet. As well, there are images with a preview of what has been ordered. The most recent 3 also show up on the homepage.



Profile Settings Page

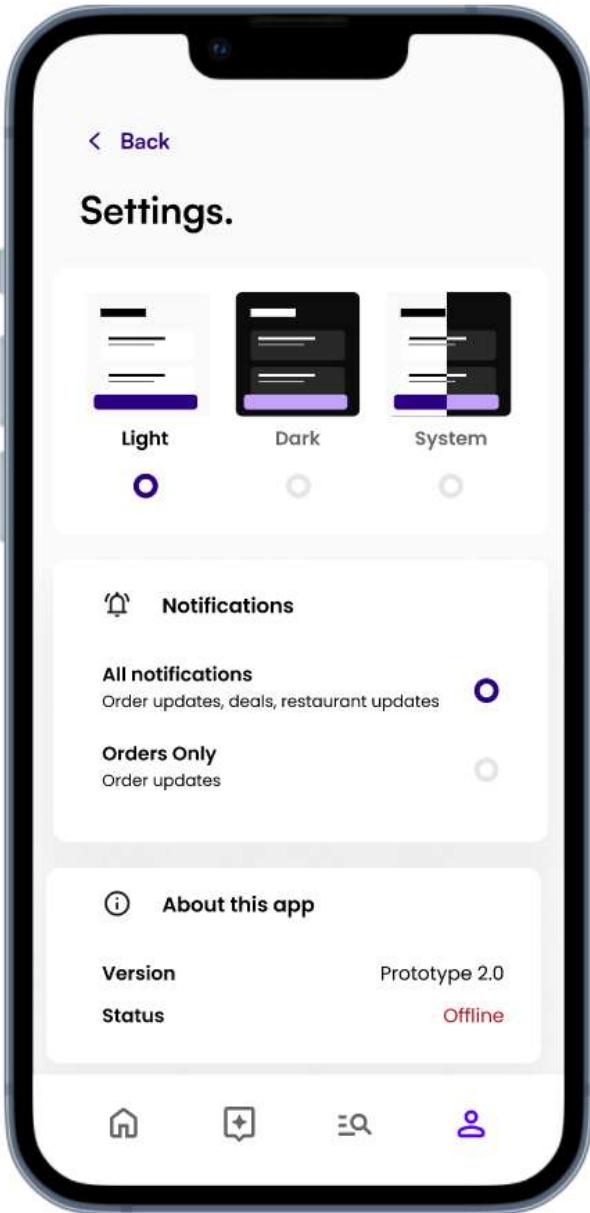
This page is used to view and edit the information for the profile selected. You are able to edit the card information, add a new card, Change personal details including password.





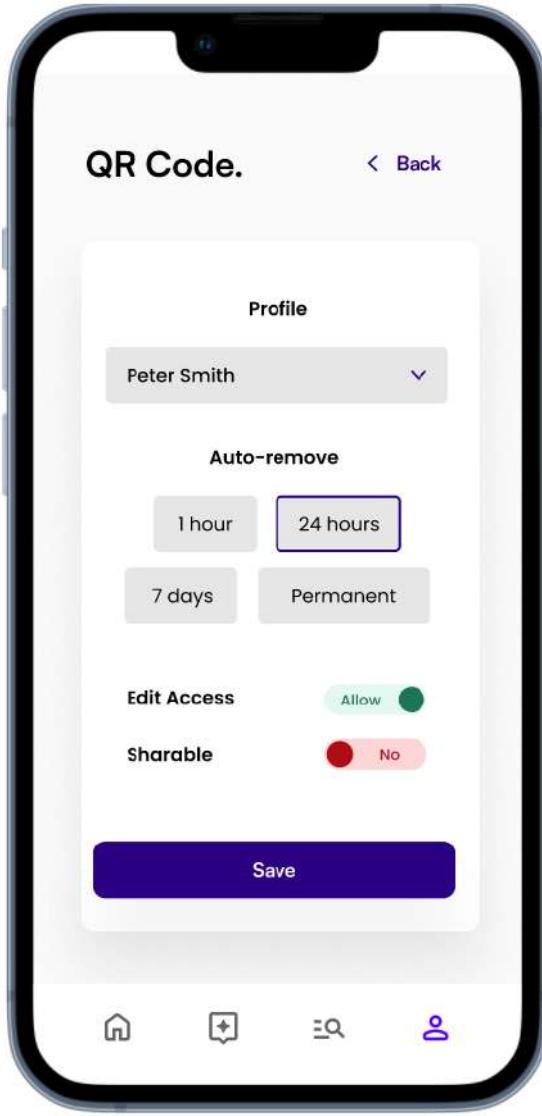
Settings Page

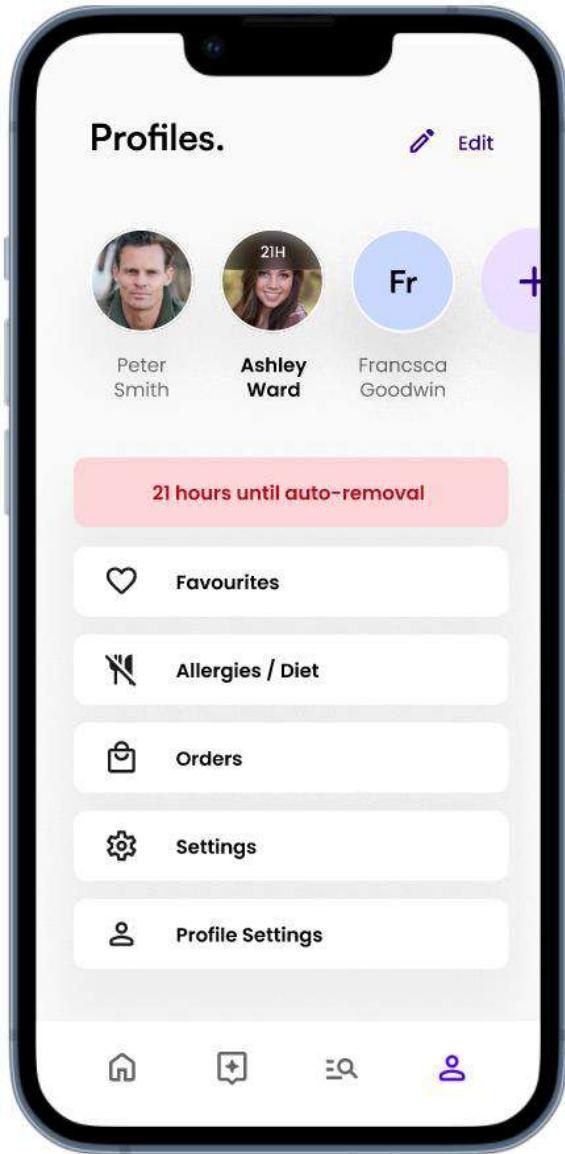
The settings page is accessible from all profiles. It is used to change the theme from the default system theme to either dark or light mode. As well, you can change the notification settings so you don't receive everything and can view the version number and see if the server is online.



Share Profile Page

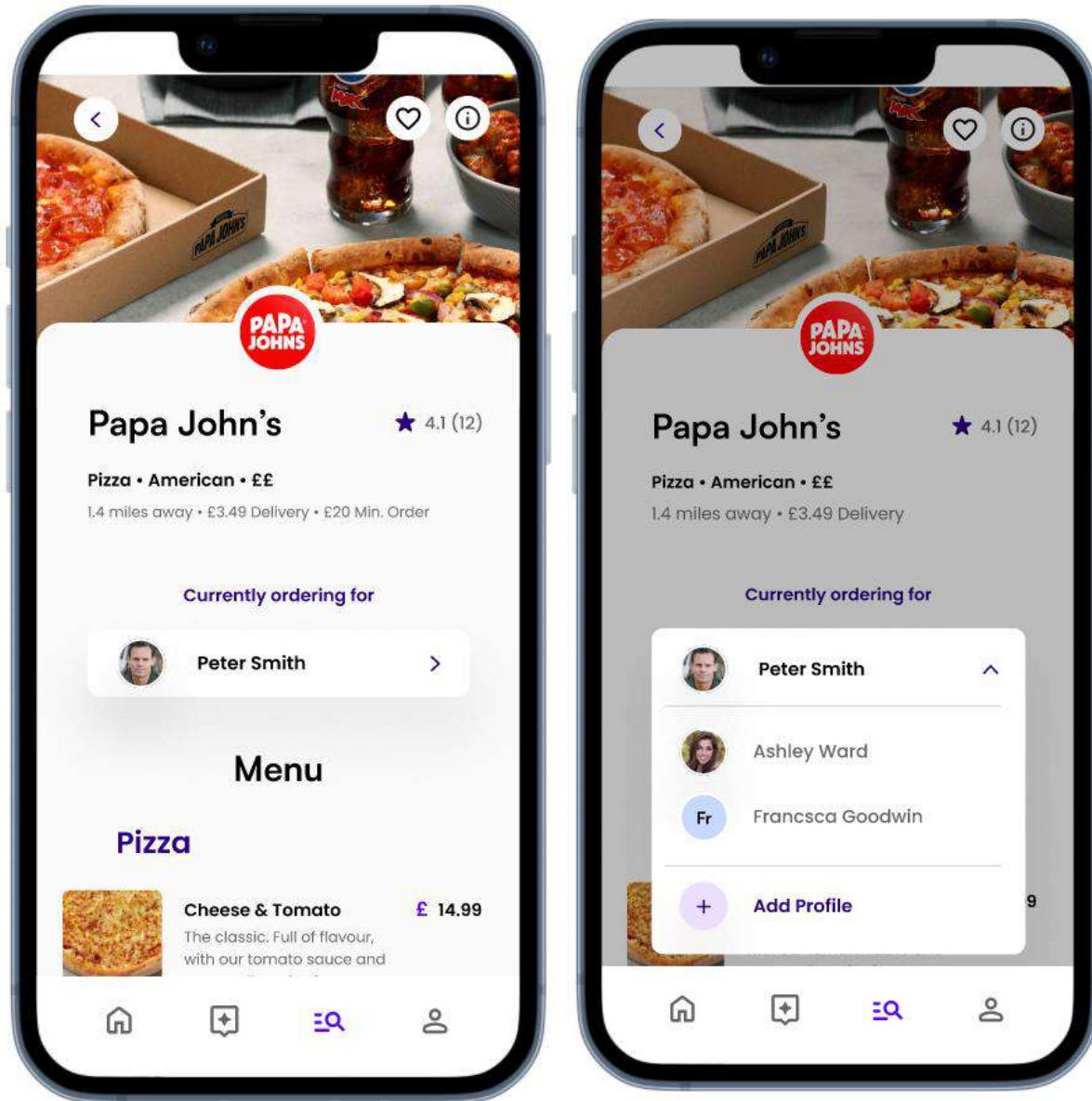
If you want to share your profile with another account, you can share it with a QR Code so you don't need to log in. With the share profile mode, it also allows you to change the settings of it so that they have less access to your account.

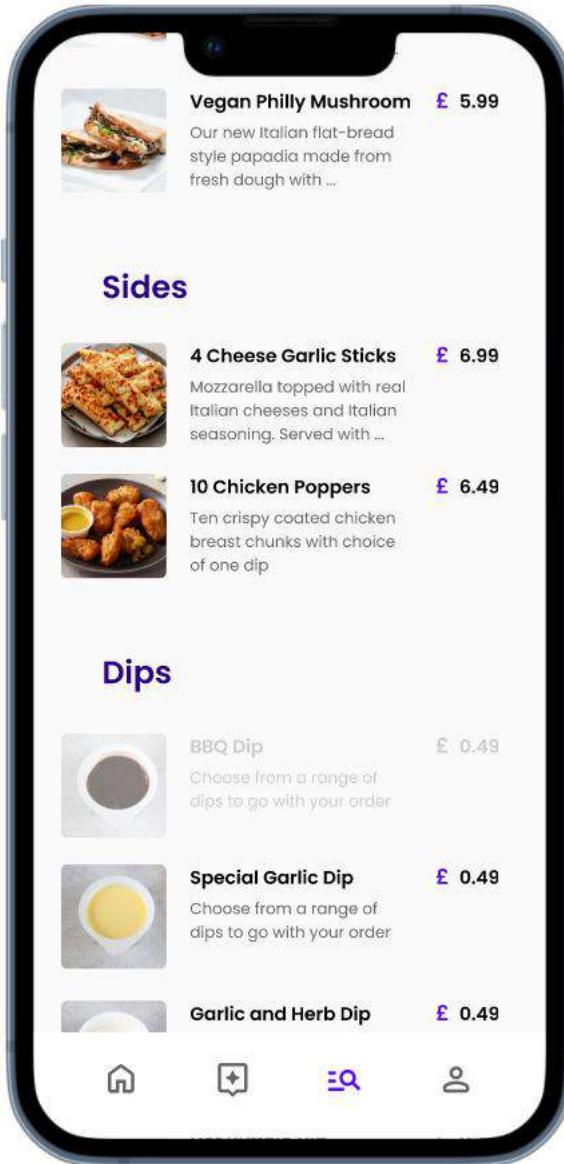
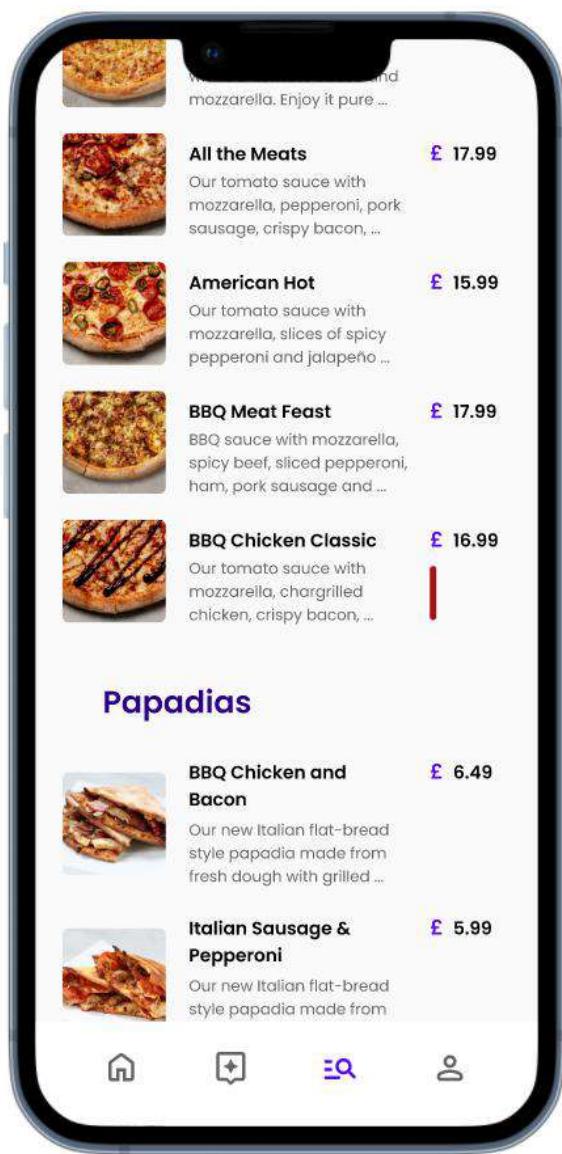


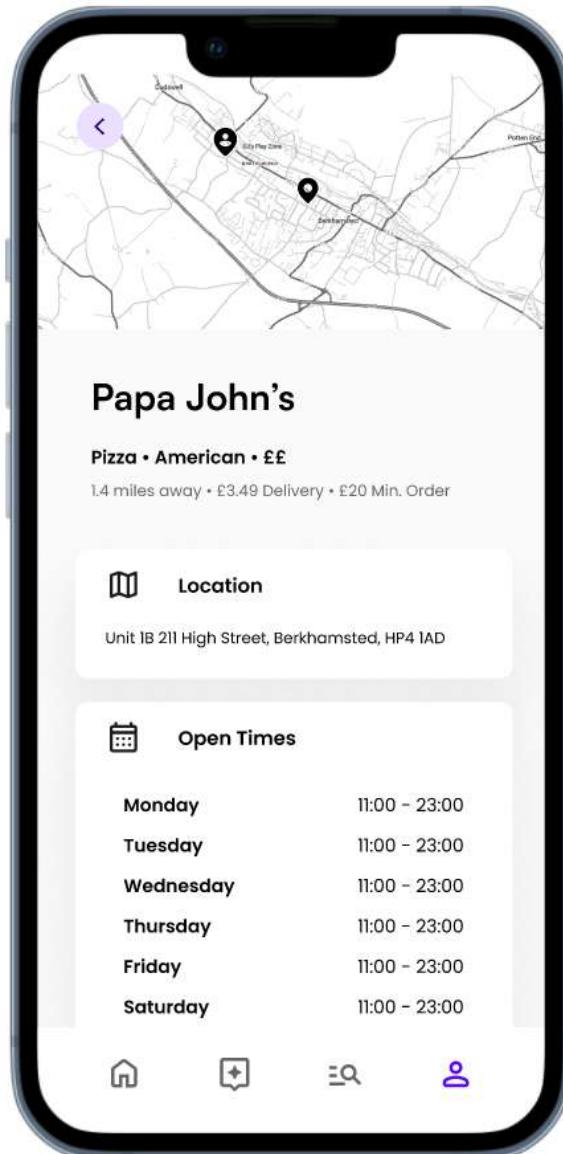
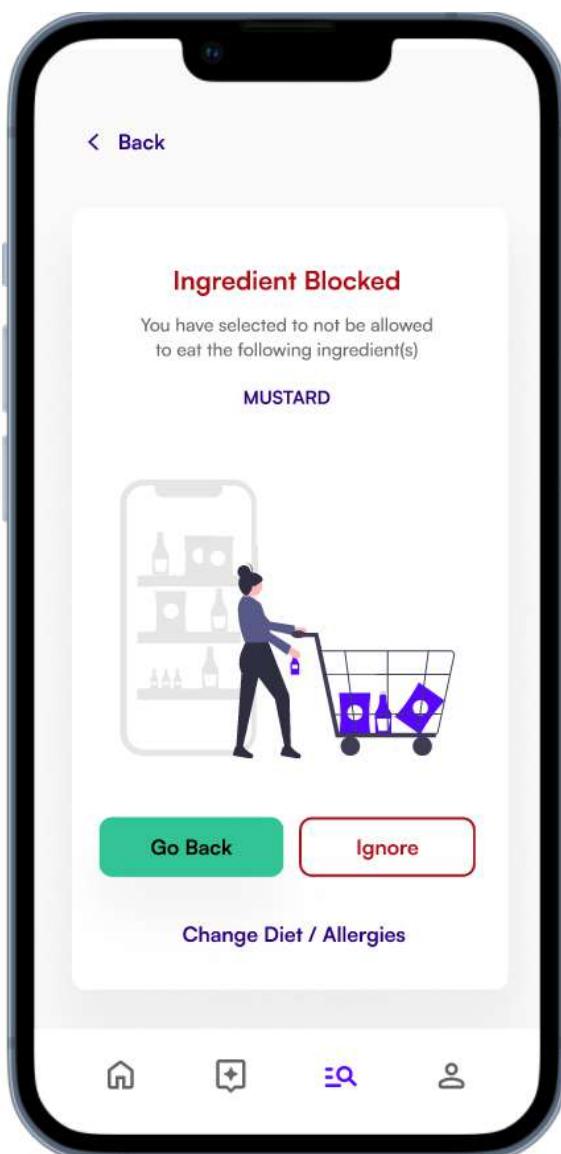


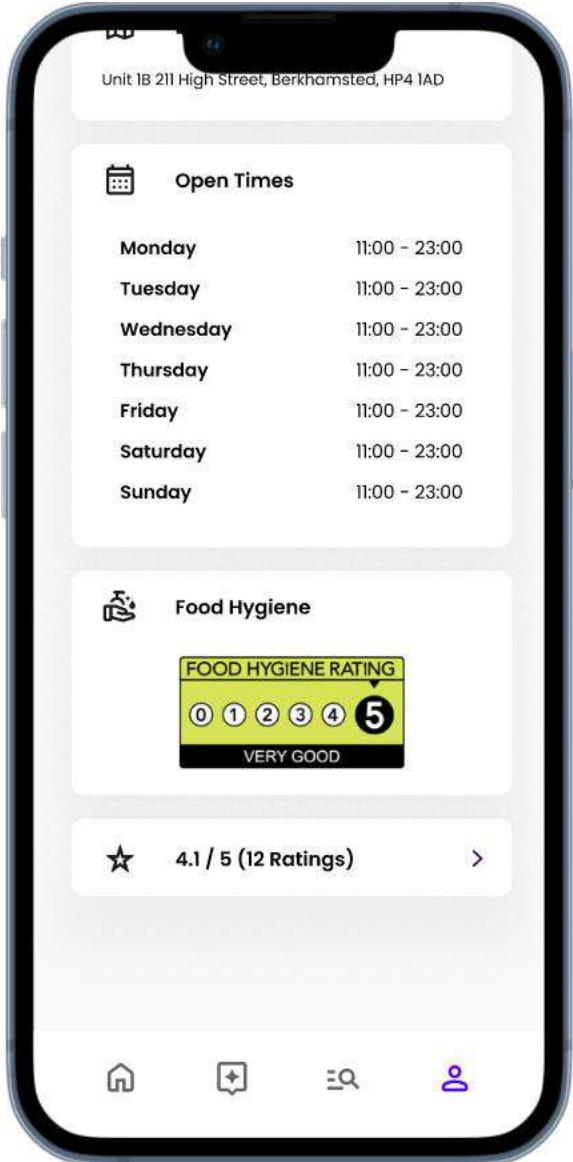
Restaurant Page

The restaurant page is a base template where it changes depending on the restaurant you are looking at. To ease using profiles, there is a place to switch profiles for when you want to order from the same restaurant. To make it easy to know what you can eat and what you can't, an item highlighted in red means that a required field has options removed because the profile cannot eat it. If an item has optionals which cannot be eaten, they will not have this highlight since it does not affect the user much and will be known when they are ordering. An item greyed out will mean that the item contains an ingredient they can't eat and it cannot be removed. Although something greyed out does mean that a user cannot order it, if they wish to, they can still select it, although it does show a warning and will show on the order that they bypassed the warning.



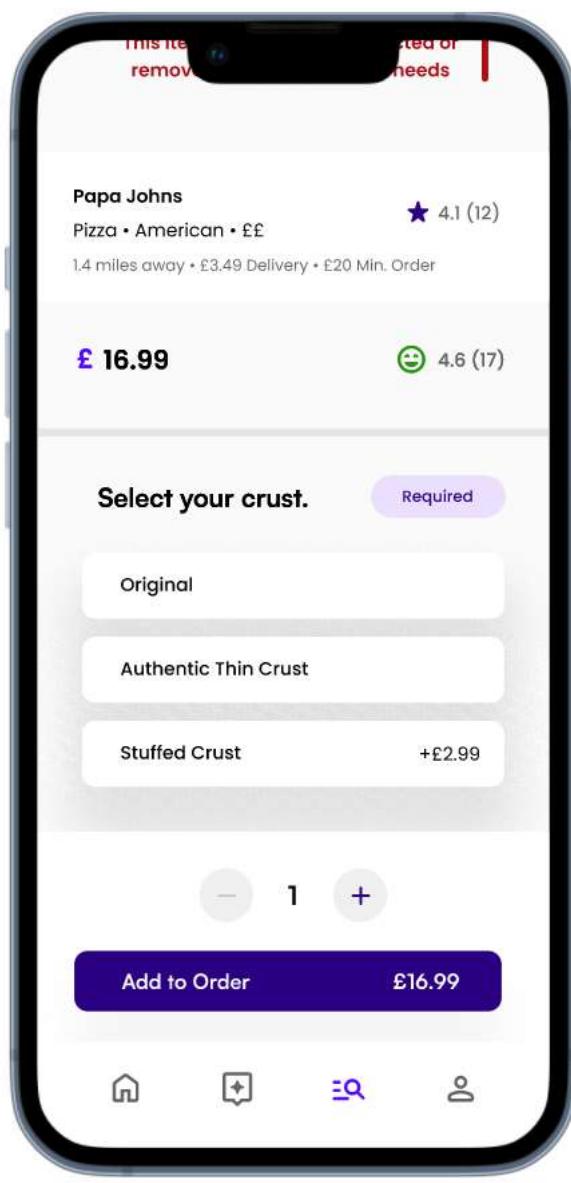


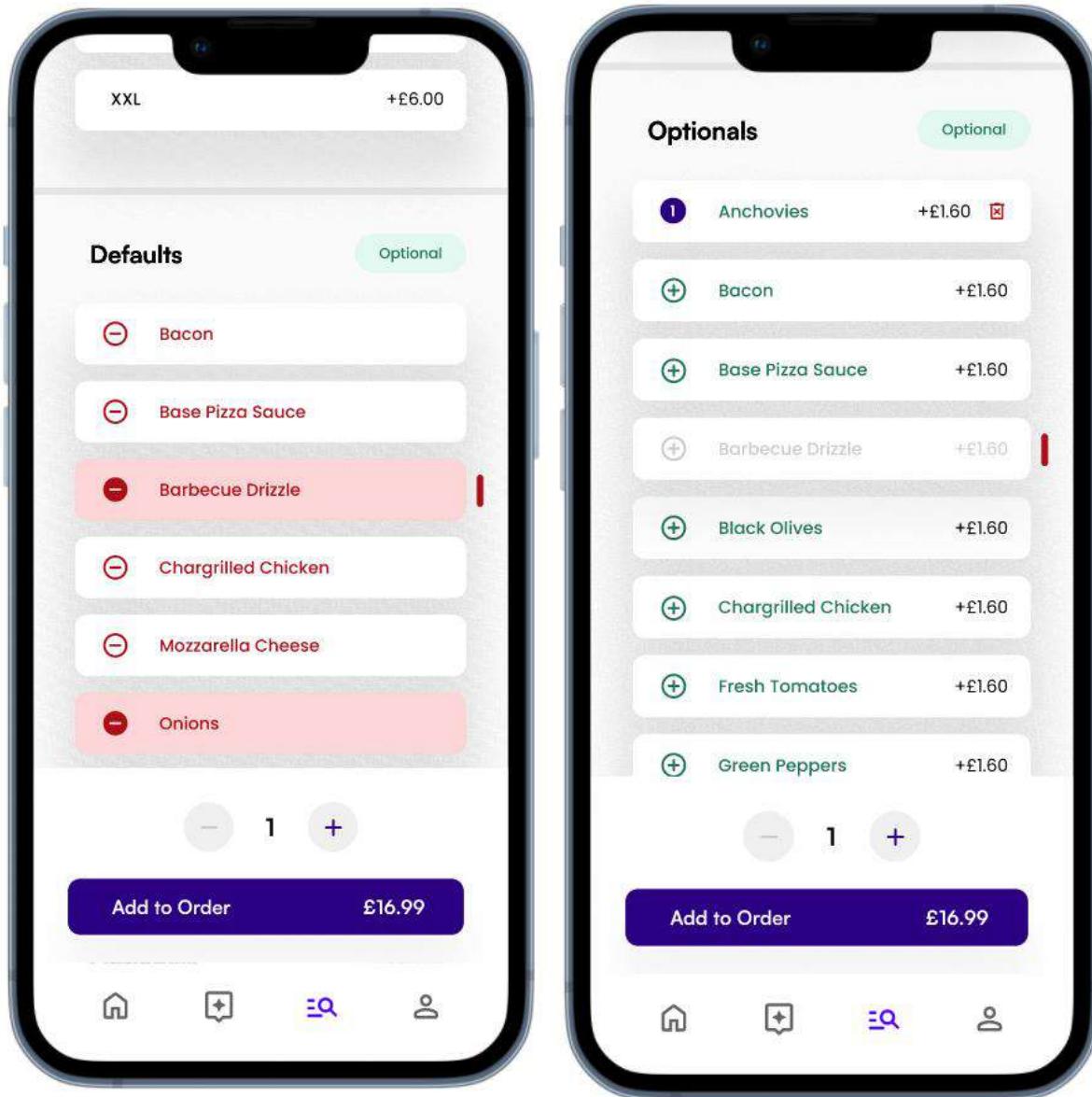




Item Page

The item page appears when a user either clicks on an item from browsing or from a restaurant.





Cart & Checkout

The cart gives a summary of the options in a similar style to the selection screen. During the checkout you can see approximately how long it will take and have the final opportunity to change the destination. At the payment section, you can choose between three options. Single will mean that one person pays with their card, split evenly means that the total is split evenly by the number of people give or take 1p. Individual means that each person pays for their food and the delivery is split.

Peter Smith's Cart:

- Vegan Giardiniera** (Pizza Express) - £13.45
Base: Romana
Toppings: Mushroom
- Pollo Pesto** (Pizza Express) - £13.75

Notes:

- This order contains ingredients which Peter is not allowed to consume.
- MILK

Ashley Ward's Checkout:

Papa Johns Order:

- Select your crust: Authentic Thin Crust
- Select the size: Medium
- Defaults: Barbecue Drizzle
- Optionals: Ham

Items:

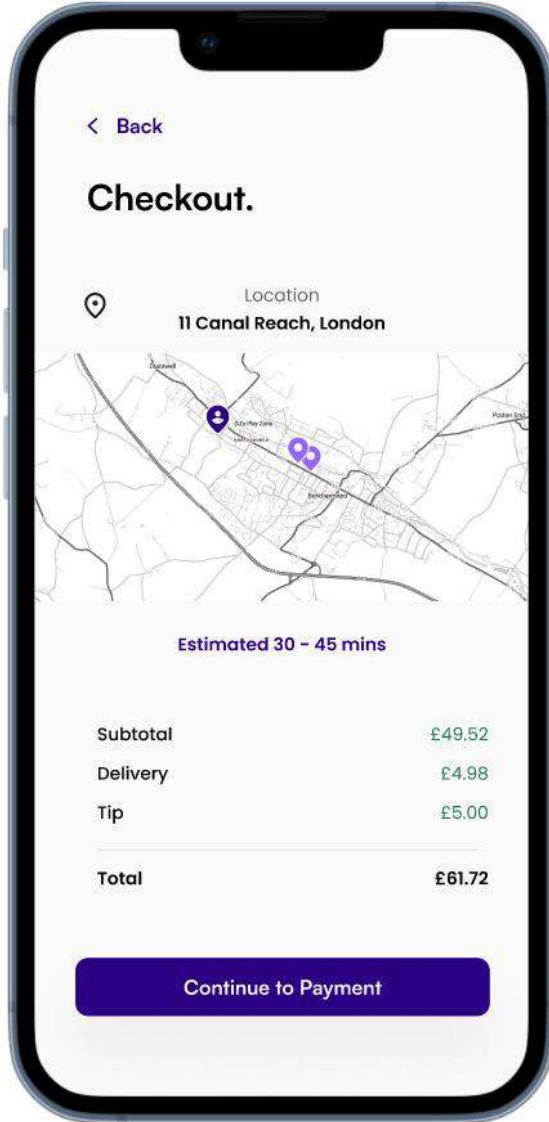
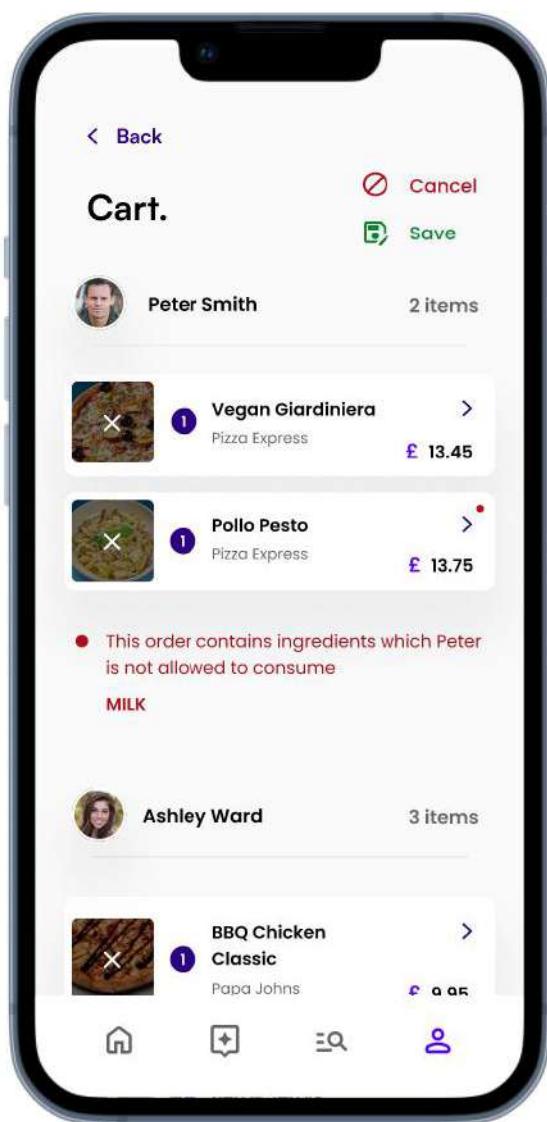
- Cookie Dough - Ben & Jerry's (Papa Johns) - £13.75
- Pepsi Max (Large) (Papa Johns) - £13.75

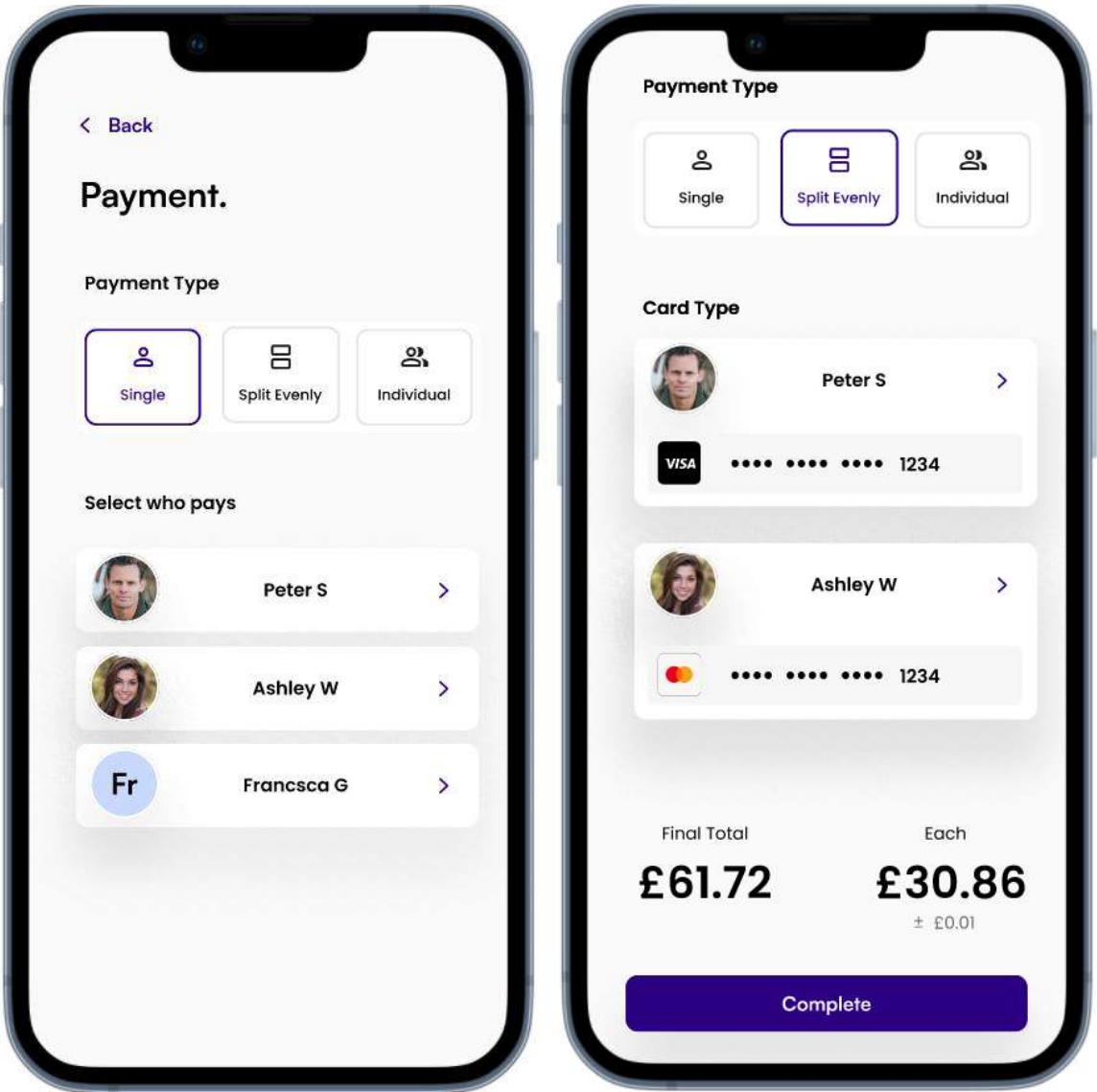
Delivery Summary:

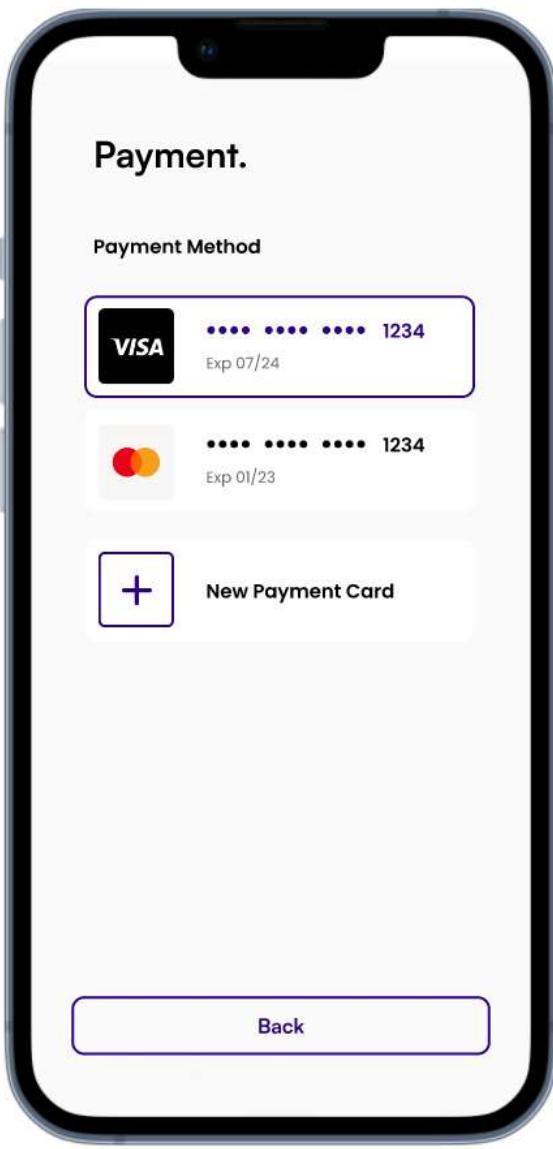
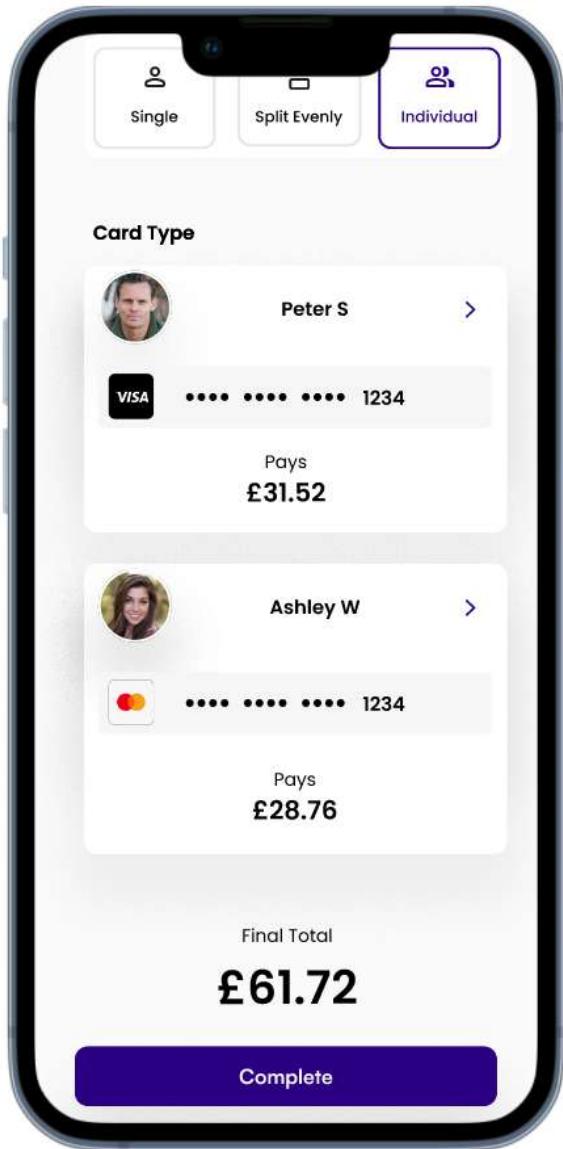
- Subtotal: £49.52
- Delivery: £4.98
- Current Total: £56.72

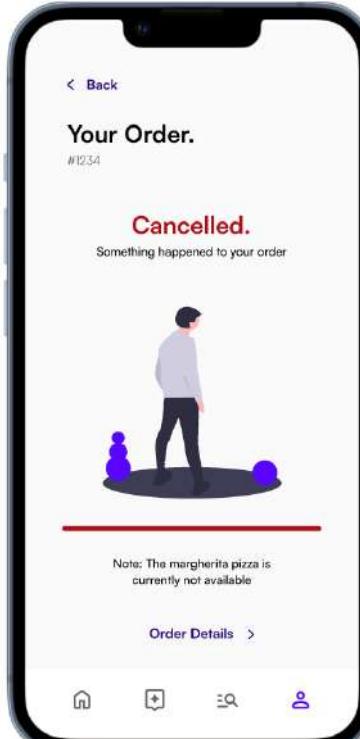
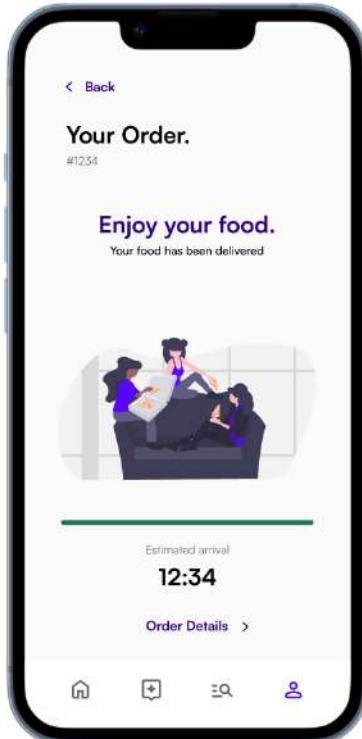
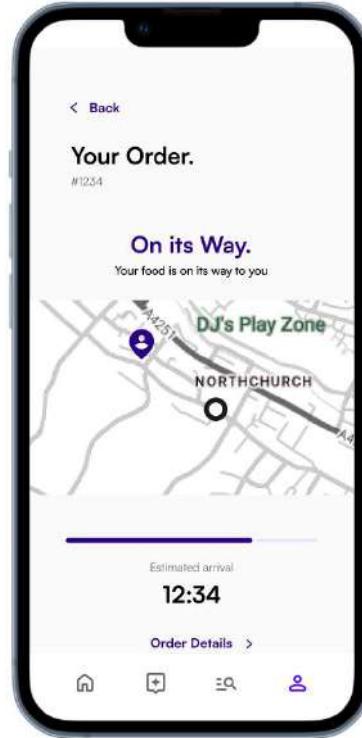
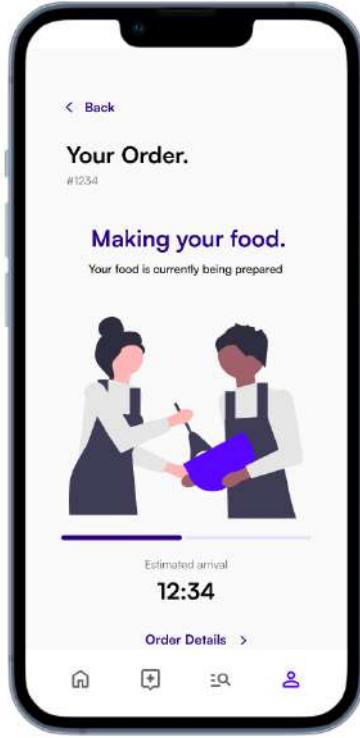
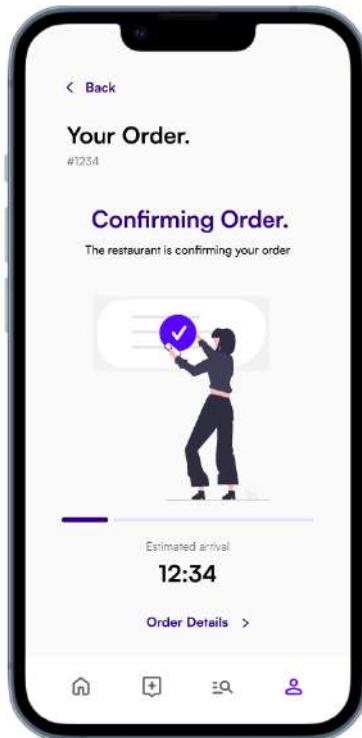
Buttons:

- Continue to Checkout









< Back

Order #1234.

12th Dec 2024

★ Rate this order ★ >

Peter Smith 2 items

1 Vegan Giardiniera Pizza Express £ 13.45
Which Base? Romana
Would you like to add any extra toppings? Mushroom

1 Pollo Pesto Pizza Express £ 13.75

● This order contains ingredients which Peter is not allowed to consume
MILK

Ashley Ward 3 items

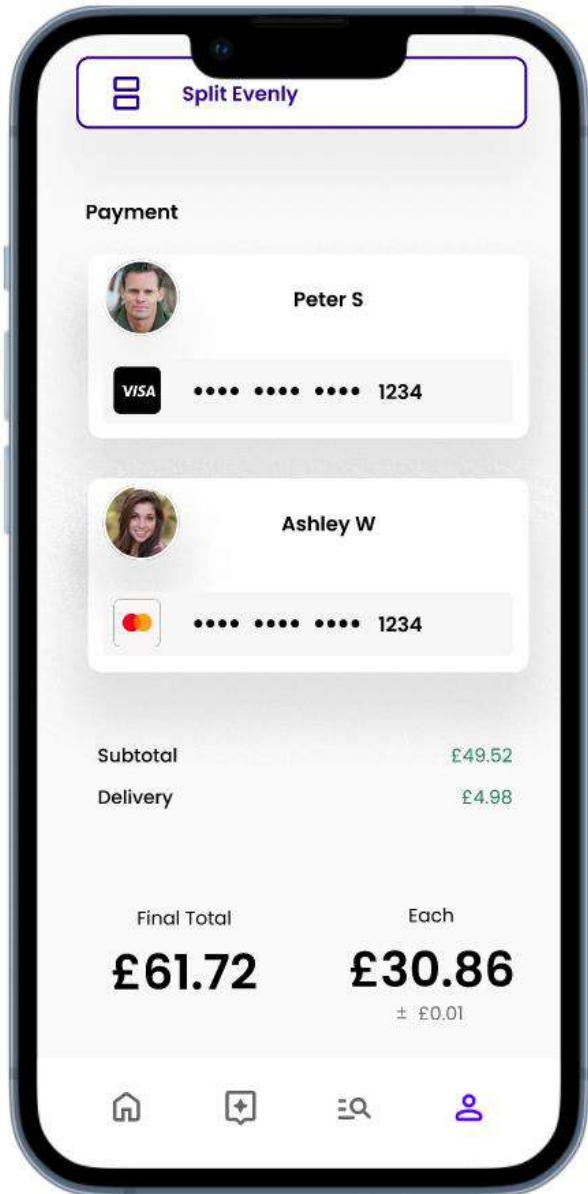
1 BBQ Chicken Classic Papa Johns £ 9.95
Select your crust. Authentic Thin Crust
Select the size. Medium
Defaults. Barbecue Drizzle
Optionals. Ham

1 Cookie Dough - Ben & Jerry's Papa Johns £ 13.75

1 Pepsi Max (Large) Papa Johns £ 13.75

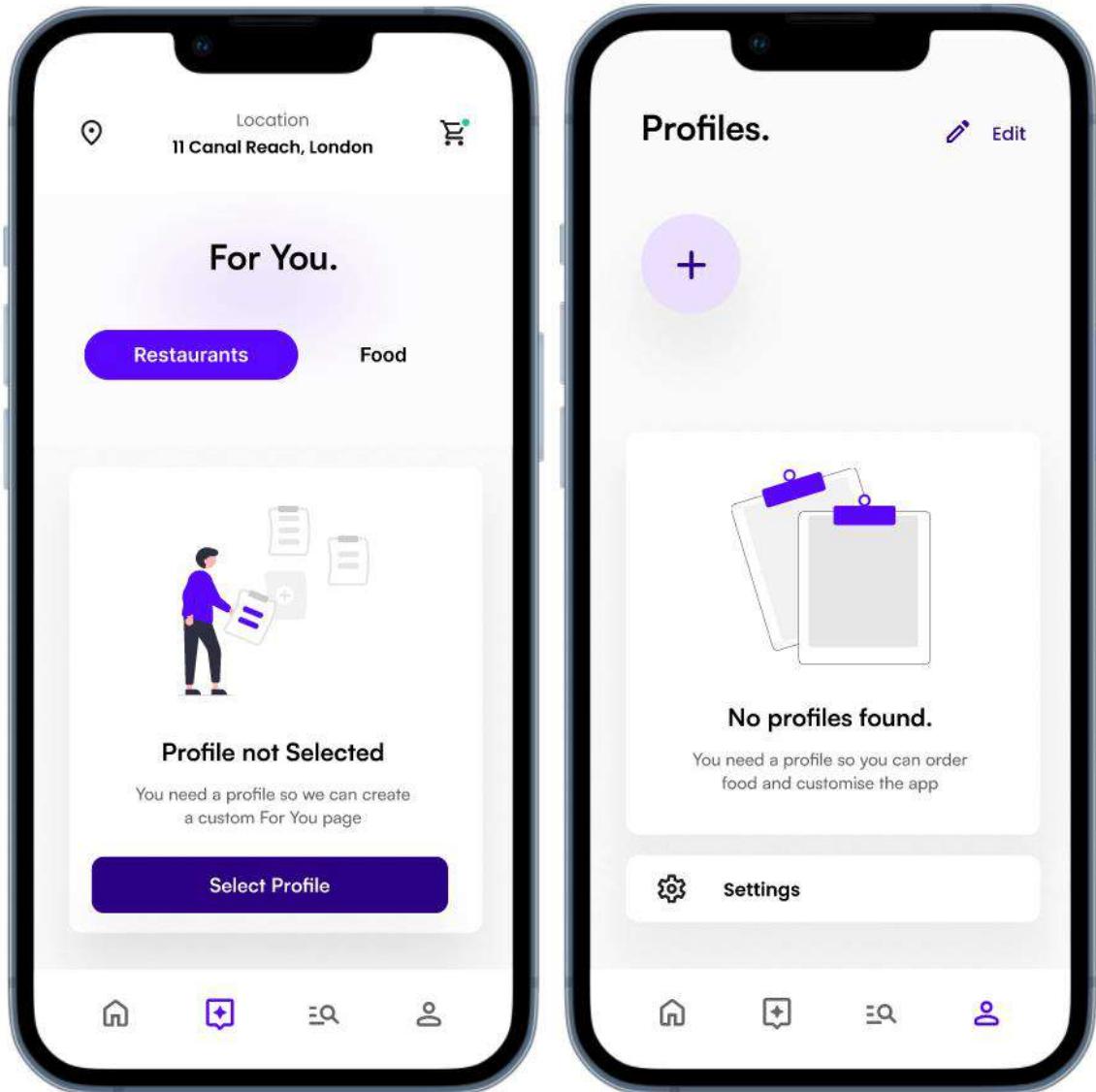
Payment Type

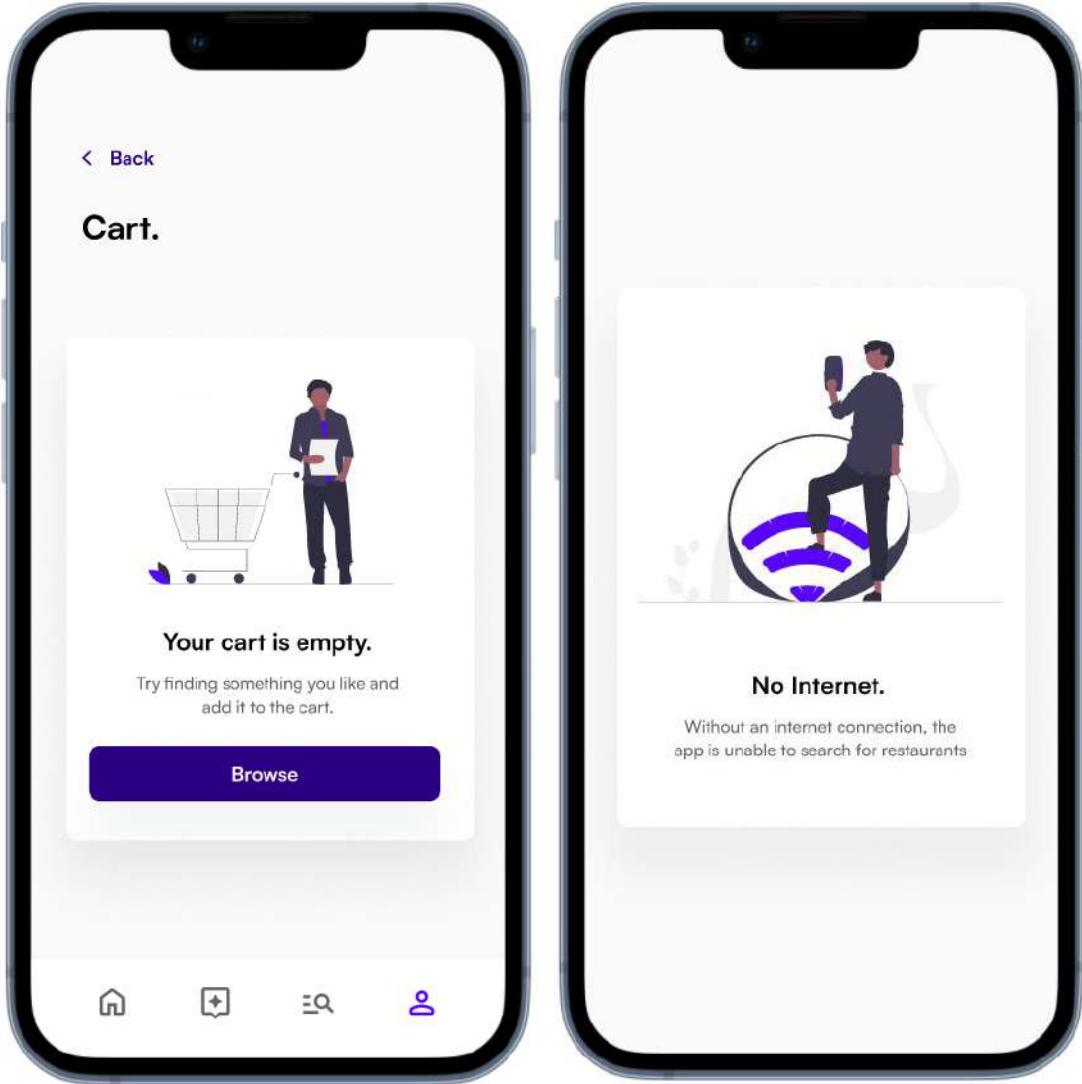
Home Search Profile



Blank Pages

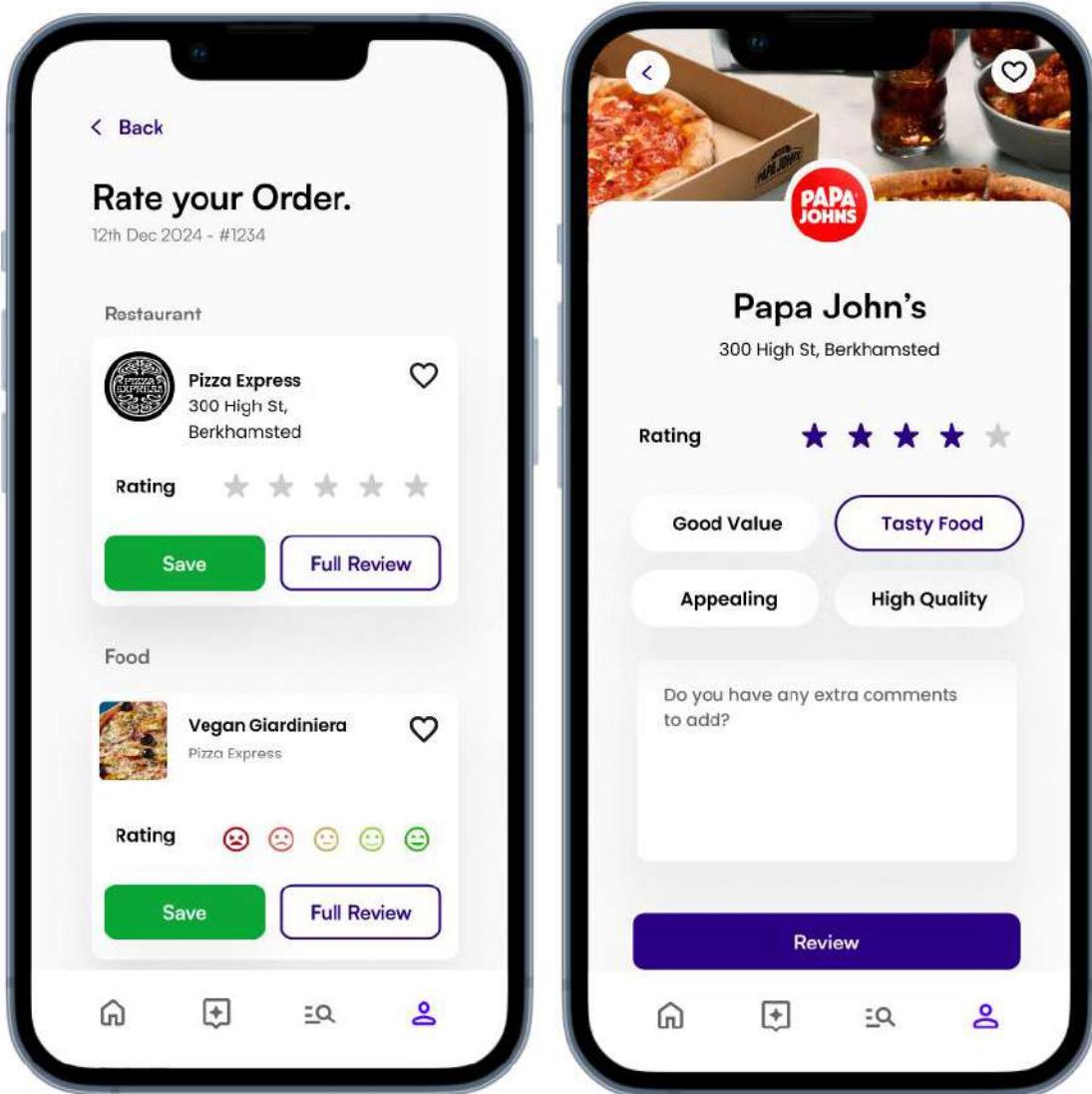
When nothing is selected or something happens, it is always important to have a page that indicates what has happened and how to fix it.



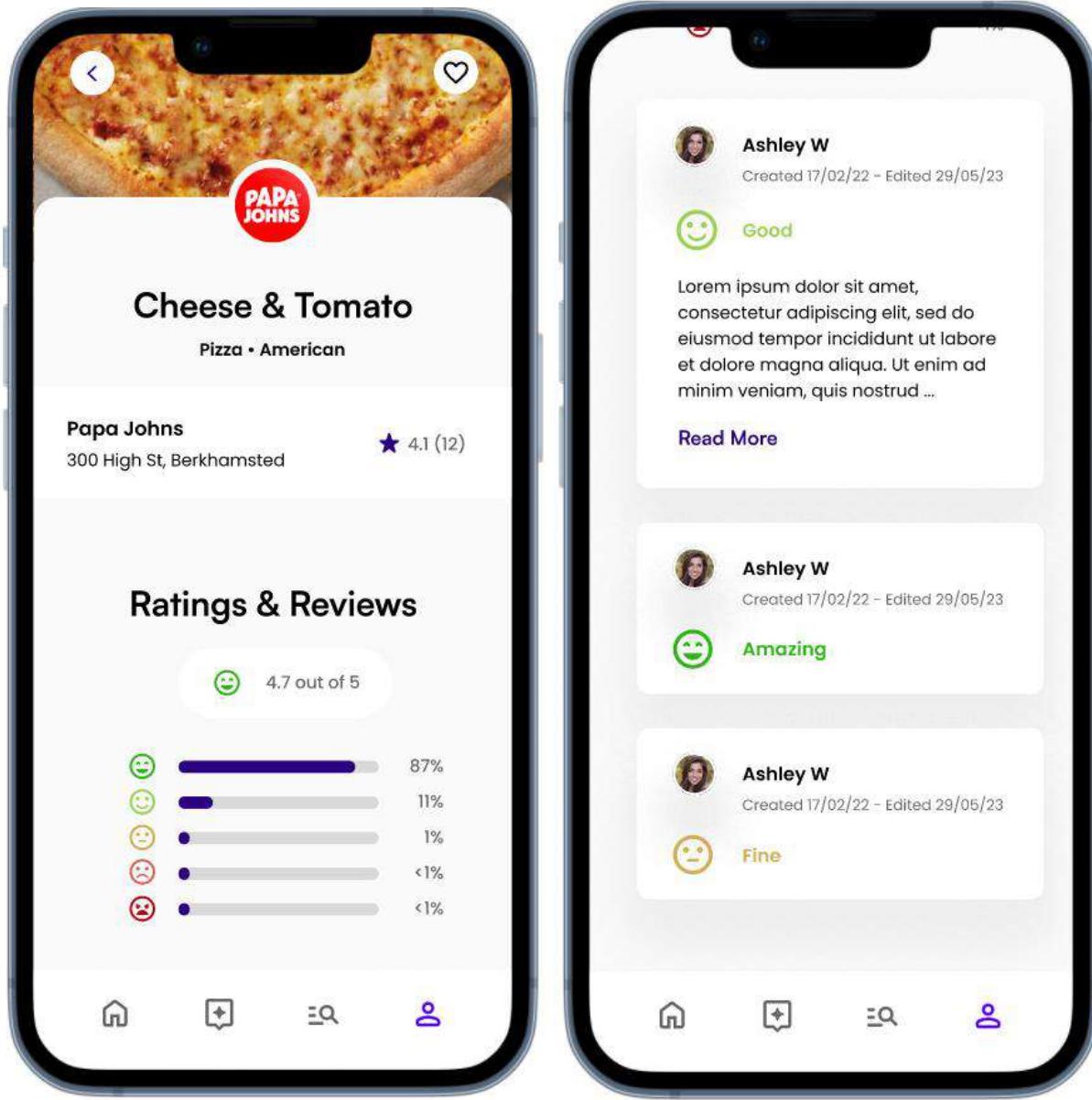


Rating Page

When it comes to ratings, it is important that it should be as easy as possible to rate food and restaurants so after each order, there is a banner which asks to rate the order. If the user wants to, they can rate quickly with a star system for restaurants and faces for food or do a full review. By having all the ratings in one area, it will allow a user to rate their previous order in a matter of seconds. To better differentiate food and restaurant ratings, food uses faces. On top of this, when a user rates food, it has a slight impact on the overall restaurant ratings. If a restaurant has some foods which consistently have a poor performance this will affect the overall star rating since it means that they do not care about the quality of food which they serve.







Development

I will be using shared preferences to store the current profile's basic information in locally stored variables which can be accessed by all files instead of calling the localprofiles table

https://pub.dev/packages/shared_preferences

To start with the remade application, I started by creating a fresh new project which only contained an Android capability. After doing some research, I found that the only way to make the app for iOS was by having a MacBook to develop it which I sadly did not have.

Adaptive App Icon

I then decided to give a fresh new look to the app by making it have an adaptive icon, giving the capability to have an icon pack put on it without it being stuck as a square icon. As I normally have an icon pack on my phone, it was very obvious that this needed to be addressed.



To fix this, I added a few lines to the pubspec.yaml file and added a foreground image which will show when the app icon has an icon pack applied

```
flutter_icons:
  # Use the following page to setup the app icon
  # https://pub.dev/packages/flutter\_launcher\_icons#mag-attributes
  # Use the following page to fix adaptive icons
  # https://dev.to/nombrekeff/how-to-make-your-flutter-app-feel-more-professional-with-adaptive-icons-31ji
  android: "launcher_icon"
  image_path: "lib/assets/launcher_icon/icon.png"
  adaptive_icon_foreground: "lib/assets/launcher_icon/foreground.png"
  adaptive_icon_background: "#27037d"
```

App Theme

During the redesign, I decided that it was a good idea to set the theme of the app. I used the following resource to not only design the light theme but also the dark theme so that it can be added in the future

<https://www.raywenderlich.com/16628777-theming-a-flutter-app-getting-started>

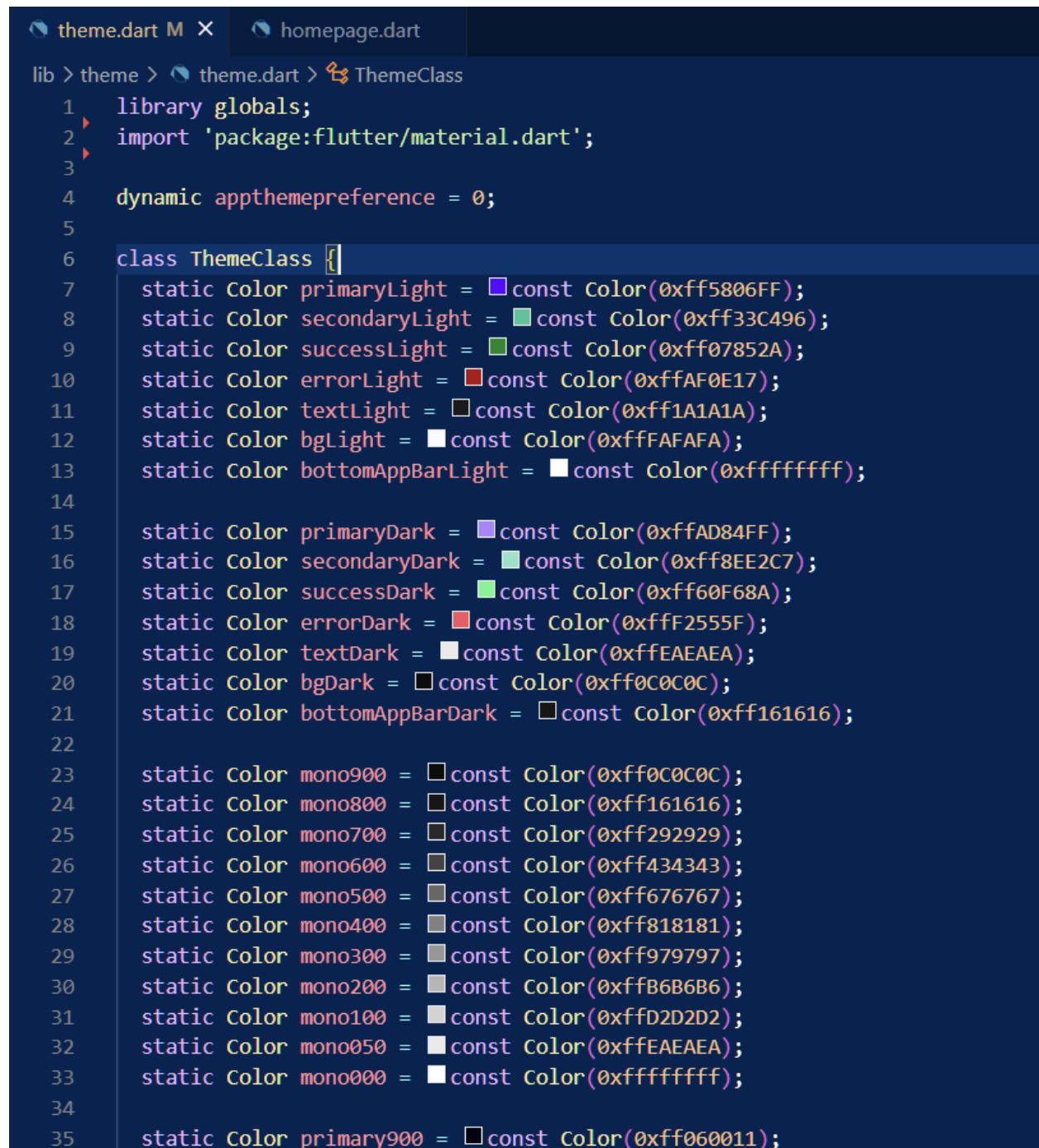
With the use of my original concept art of the final design, I was able to create a theme which closely met the same theme.

https://pub.dev/packages/flutter_native_splash

I also used the following article to aid with the making of universal headings and font types.

I used the variable `appthemepreference` as a global in order for it to be accessible by all files to find out if the user wants dark mode (2), light mode (1) or system style (null - default).

In order to also stick with the theme, I created static variables with all the different styles that were created in concept art 4. These can then be altered in the future if ever the theme changes which means that all the colours are centralised since they are referencing the theme.



```
theme.dart M X | homepage.dart

lib > theme > theme.dart > ThemeClass
1 library globals;
2 import 'package:flutter/material.dart';
3
4 dynamic appthemepreference = 0;
5
6 class ThemeClass {
7     static Color primaryLight = const Color(0xff5806FF);
8     static Color secondaryLight = const Color(0xff33C496);
9     static Color successLight = const Color(0xff07852A);
10    static Color errorLight = const Color(0xFFAF0E17);
11    static Color textLight = const Color(0xff1A1A1A);
12    static Color bgLight = const Color(0xffFAFAFA);
13    static Color bottomAppBarLight = const Color(0xffffffff);
14
15    static Color primaryDark = const Color(0xFFAD84FF);
16    static Color secondaryDark = const Color(0xff8EE2C7);
17    static Color successDark = const Color(0xff60F68A);
18    static Color errorDark = const Color(0xFFF2555F);
19    static Color textDark = const Color(0xffEAEAEA);
20    static Color bgDark = const Color(0xff0C0C0C);
21    static Color bottomAppBarDark = const Color(0xff161616);
22
23    static Color mono900 = const Color(0xff0C0C0C);
24    static Color mono800 = const Color(0xff161616);
25    static Color mono700 = const Color(0xff292929);
26    static Color mono600 = const Color(0xff434343);
27    static Color mono500 = const Color(0xff676767);
28    static Color mono400 = const Color(0xff818181);
29    static Color mono300 = const Color(0xff979797);
30    static Color mono200 = const Color(0xffB6B6B6);
31    static Color mono100 = const Color(0xffD2D2D2);
32    static Color mono050 = const Color(0xffEAEAEA);
33    static Color mono000 = const Color(0xffffffff);
34
35    static Color primary900 = const Color(0xff060011);
```

To determine the theme that will be used, when the app is being opened, it grabs the app preference from shared preferences and uses a case switch to direct which way the app will open, using either the light theme, dark theme or system theme.

```
class MyApp extends StatelessWidget {
    MyApp({super.key});
    Future getTheme() async {
        final prefs = await SharedPreferences.getInstance();
        final int? appTheme = prefs.getInt('appTheme');
        globals.appthemepreference = appTheme;
    }

    @override
    Widget build(BuildContext context) {
        WidgetsFlutterBinding.ensureInitialized();
        getTheme();
        switch (globals.appthemepreference) {
            case 1:
                return MaterialApp(
                    //Create main app
                    title: 'AllEat.',
                    themeMode: ThemeMode.light,
                    theme: ThemeClass.lightTheme,
                    home:
                        ||| const SetupWrapper(), //SetupWrapper(), //Check if
                ); // MaterialApp
            case 2:
                return MaterialApp(
                    //Create main app
                    title: 'AllEat.',
                    themeMode: ThemeMode.dark,
                    theme: ThemeClass.darkTheme,
                    home:
                        ||| const SetupWrapper(), //SetupWrapper(), //Check if
                ); // MaterialApp
        }
    }
}
```

To get the system theme, it uses a built-in function which does this automatically

```
    }
    return MaterialApp(
        //Create main app
        title: 'AllEat.',
        themeMode: ThemeMode.system,
        theme: ThemeClass.lightTheme,
        darkTheme: ThemeClass.darkTheme,
        home:
            ||| const SetupWrapper(), //Setup
    ); // MaterialApp
}
```

For each theme, there are set definitions which can be referenced to from a widget like a button or text.



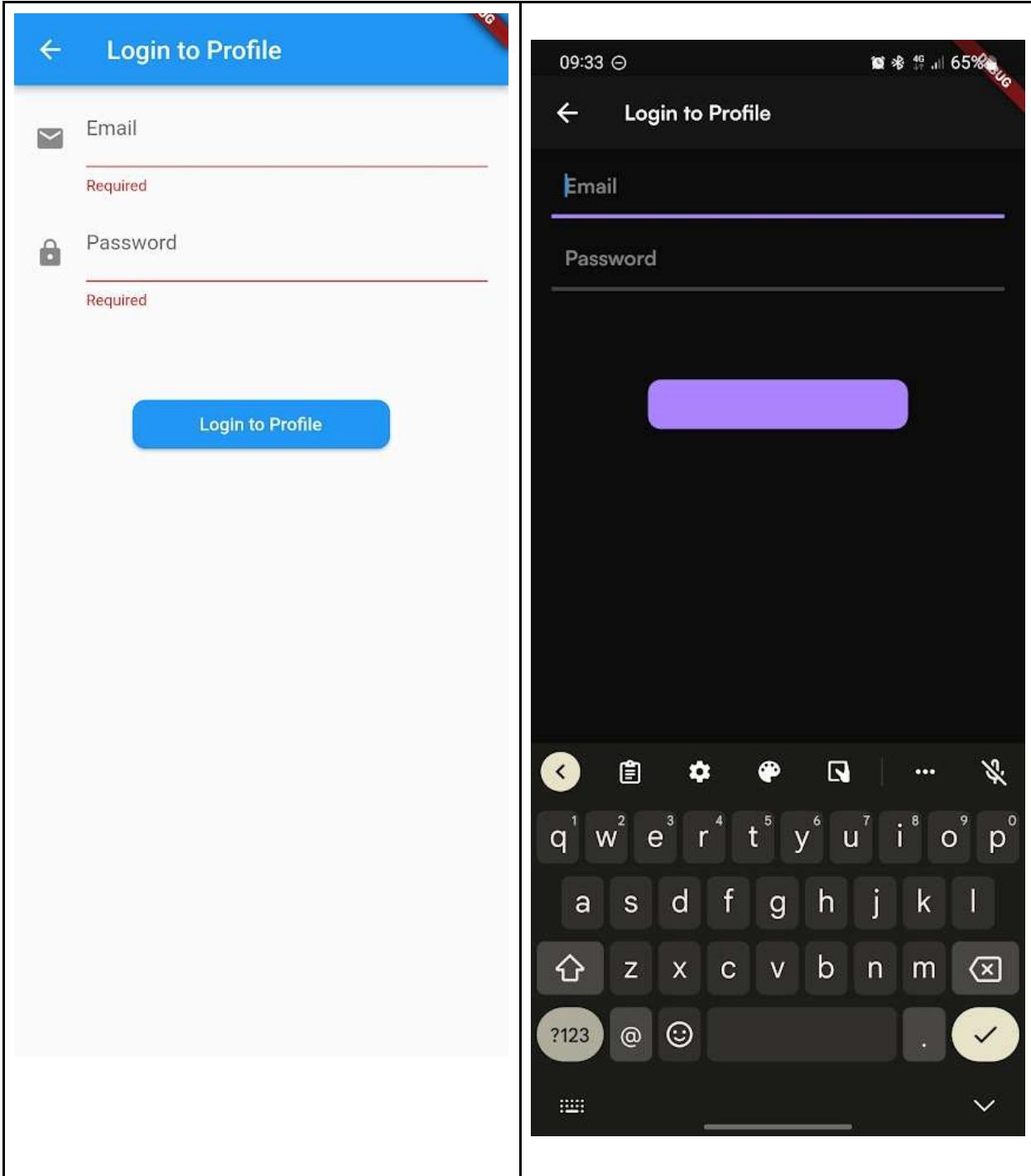
The screenshot shows a code editor with three tabs at the top: 'theme.dart M X', 'main.dart', and 'homepage.dart'. The 'theme.dart' tab is active. Below the tabs, the file path 'lib > theme > theme.dart' is shown, followed by the class name 'ThemeClass'. The code itself is a static ThemeData definition:

```
47 static ThemeData lightTheme = ThemeData(  
48   //COLOUR  
49   primaryColor: primaryLight,  
50   colorScheme: ColorScheme.fromSwatch().copyWith(  
51     secondary: secondaryLight,  
52     error: errorLight,  
53   ),  
54   // MAIN APP  
55   bottomNavigationBarTheme: BottomNavigationBarThemeData(  
56     backgroundColor: mono000,  
57     selectedItemColor: primaryLight,  
58     unselectedItemColor: mono400), // BottomNavigationBarThemeData  
59   appBarTheme: AppBarTheme(backgroundColor: primaryLight),  
60   backgroundColor: bgLight,  
61   scaffoldBackgroundColor: bgLight,  
62   // TEXT  
63   textTheme: TextTheme(  
64     headline1: TextStyle(  
65       color: textLight,  
66       fontFamily: 'Satoshi',  
67       fontWeight: FontWeight.w800,  
68       fontSize: 32), // TextStyle  
69     headline2: TextStyle(  
70       color: mono900,  
71       fontFamily: 'Satoshi',  
72       fontWeight: FontWeight.w700,  
73       fontSize: 28), // TextStyle  
74     headline3: TextStyle(  
75       color: mono800,  
76       fontFamily: 'Satoshi',
```

For example, the following uses a themed input box

```
children: [
  ListTile(
    title: TextFormField(
      controller:
        email, //Form data lastname collected and sent to database
      keyboardType: TextInputType.emailAddress,
      style: Theme.of(context).textTheme.bodyText2,
      decoration: (InputDecoration(
        hintText: "Email",
        contentPadding:
          Theme.of(context).inputDecorationTheme.contentPadding,
        border: Theme.of(context).inputDecorationTheme.border,
        focusedBorder:
          Theme.of(context).inputDecorationTheme.focusedBorder,
        enabledBorder:
          Theme.of(context).inputDecorationTheme.enabledBorder,
        floatingLabelBehavior: FloatingLabelBehavior.never)),
      inputFormatters: [
        //Only allows the input of letters a-z and A-Z and @,.-_
        FilteringTextInputFormatter.allow(RegExp('[a-zA-Z0-9@,. -]'))
      ],
      validator: (email) {
        //Required field and uses emailvalidator package to verify it is an email +
      }
    )
  )
]
```

Before	After
--------	-------



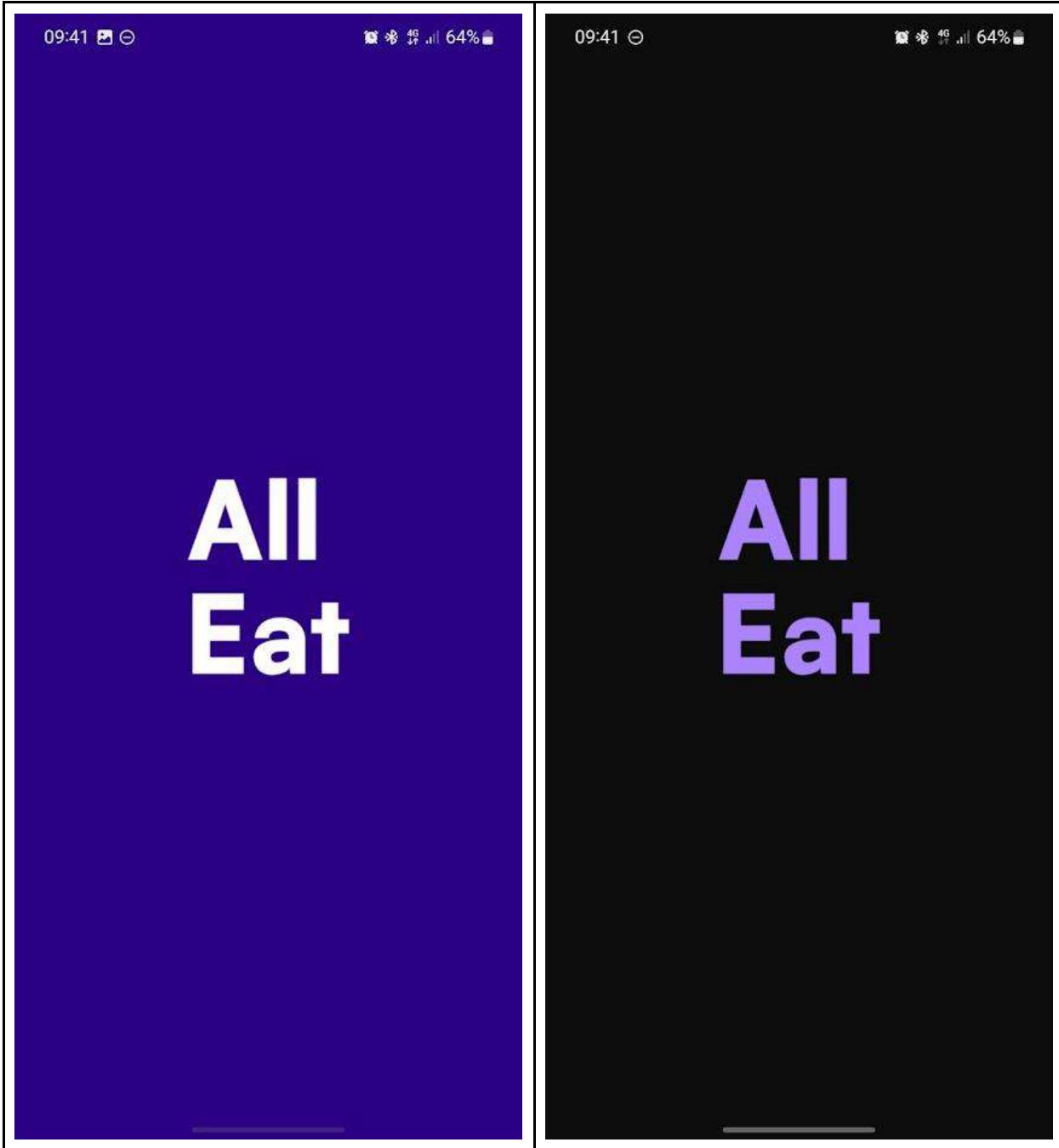
New Splash Screen

While getting the app theme and rendering the app, instead of having a large app icon with a white background for a splash screen, I made a custom splash screen. Depending on if the app

is in light mode or dark mode it will show a different splash screen in order to help reduce eye strain.

```
flutter_native_splash:  
  color: "#2B0082"  
  
  image: lib\assets\images\logo\splashscreen-android12.png  
  
  android_12:  
    image: lib\assets\images\logo\splashscreen-android12.png  
  
    # Splash screen background color.  
    color: "#2B0082"  
    # App icon background color.  
    icon_background_color: "#2B0082"  
  
    image_dark: lib\assets\images\logo\splashscreen-android12-dark.png  
    color_dark: "#0C0C0C"  
  
  ios: false  
  web: false
```

Light Mode	Dark Mode
-------------------	------------------



Force Portrait mode

While remaking the welcome screen, I found that if the screen rotates to landscape, the app would not have enough space to fit all elements. To fix this, I forced portrait mode by setting portrait mode when the app is opened.

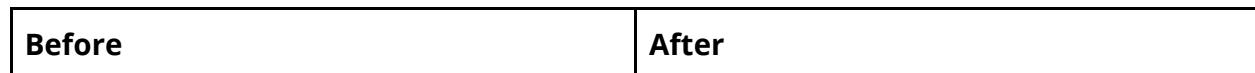
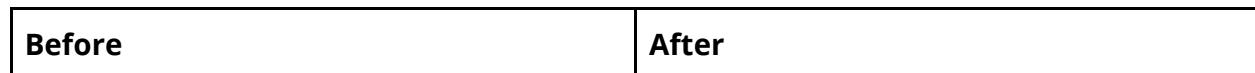
```
@override
Widget build(BuildContext context) {
    SystemChrome.setPreferredOrientations([ //Force portrait mode
        DeviceOrientation.portraitUp,
        DeviceOrientation.portraitDown,
    ]);
    WidgetsFlutterBinding.ensureInitialized();
}
```

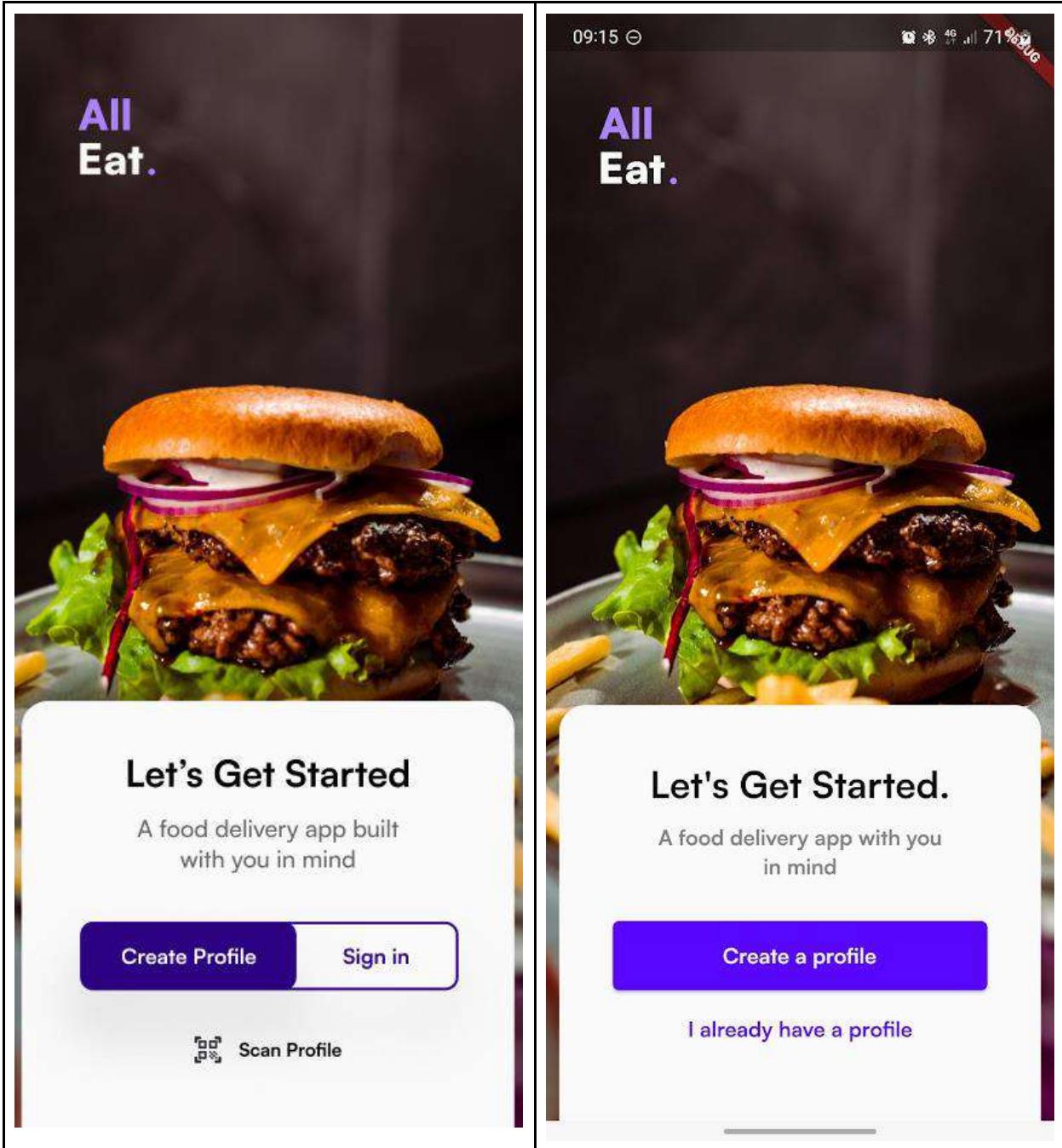
Redesigned Welcome Screens

When a user first opens the app and wants to get started, they should be greeted as if they do not have an account. Unlike existing users, these people do not know how to interact with the app, so it should be as user friendly as possible.

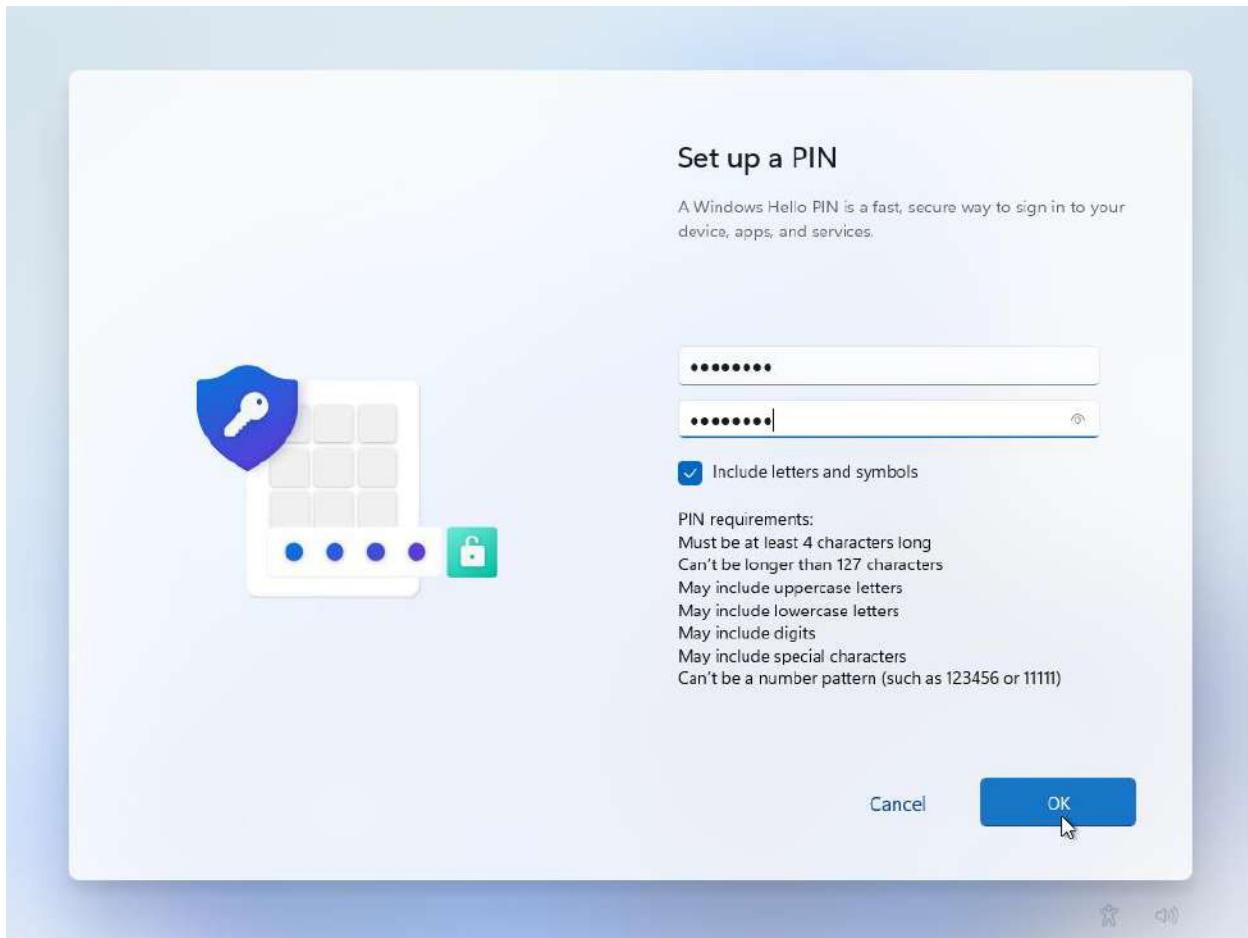
To do this, I used my concept art as a reference point for how it should look.

After trying to transfer the screens over to the app, what became apparent was that if there was a smaller screen size, it would not display properly. To fix this, I decided that each button would take its own line and the page would be adapted.

Before	After
	



As well as this issue, I decided, in order to not have lots of fields that the user needs to fill in to put their details in, I would use multiple pages with the name of the steps. I took this idea from the Windows setup page. When a user first sets up Windows, they are given multiple pages where each one has a single option to complete before continuing.

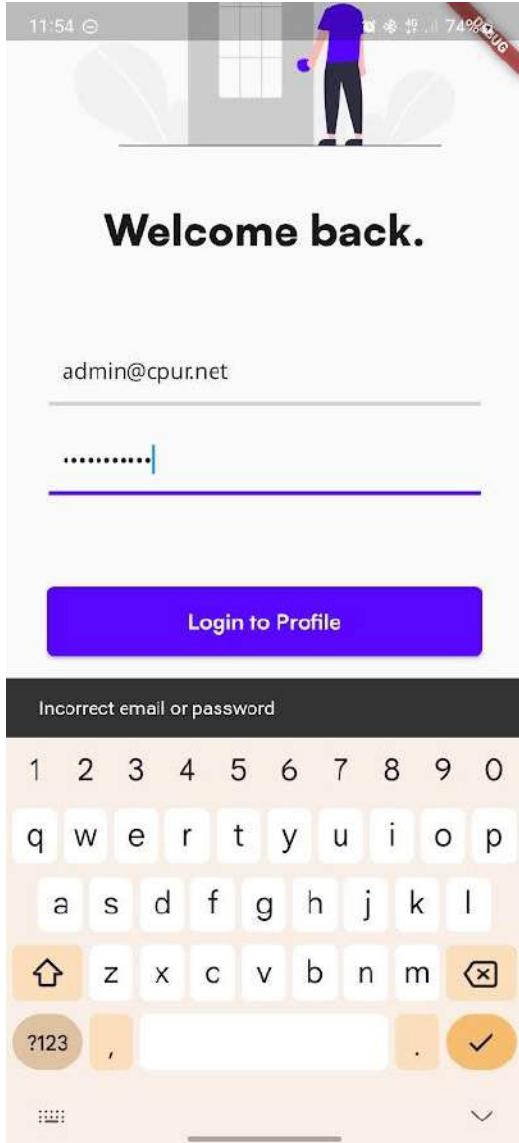


<https://uk.pcmag.com/migrated-3765-windows-10/135811/giving-windows-11-a-try-how-to-set-up-and-customize-your-installation>

While creating the multipage creation page, I had an issue where I forgot how to forward data from different classes to another. By using the following article, it assisted in moving data from one screen to another.

<https://stackoverflow.com/questions/50818770/passing-data-to-a-stateful-widget-in-flutter>

During testing the new more efficient login system, I kept getting errors with the server where it said that the "email or password was incorrect" even when it was.



To debug this issue, I used temporary variables to output the data passing through the server login file. Since the profile creation page was under development I decided to install prototype 1 on my phone and create a new profile via the app.

Once again, it still resulted in saying there was no user under that name.

```
#1178 [mainActivity] (28416): View of line pointer: 1  
: {error: false, message: Email or password incorrect, exists: false, profile: null}
```

To test if the body text was being forwarded, I outputted that but it was working correctly

```
04a178[MainActivity](28410): viewPostime pointer 1  
): {email: testuser100@cpur.net, password: CaGDH456REwCV/C/WXM5ww==}  
): {error: false, message: Email or password incorrect, exists: false, profile: null}
```

The server was also getting the data

```
: {error: false, message: Email or password incorrect, exists: false, profile: null, temp1: testuser100@cpur.net, temp2: CaGDH456REwCV/C/WXM5ww==}
```

The only solution that could be found was that the profile creation file on the server was inputting the incorrect data to the database.

As it turned out, the data was being checked incorrectly on the server

While testing the login page, when a password or email was not required, the text "Required" would show up but it would overflow the screen, causing an error. To fix this, I used a SingleChildScrollView in order to add extra scrolling to the screen.

Existing User Create Profile Page

Since a user can add multiple profiles, it is good for there to be a different page for adding profiles so it is distinguishable when it is set up and adding profiles. To do this, I made a copy of the welcome screen and changed the text and some of the widgets. I was considering forwarding text to change the widget but since the file size was small anyway and it needed a different widget to go back to the main area



Removal of Duplicate Users

In prototype 1A, I found if a user logged in twice it would glitch so in Prototype 1B I decided to fix this. Since the app uses emails as a key because they are unique, it is important that only one user holds this email.

For logins, to fix this issue, a simple check of each profile's email to see if it is already logged in can be implemented. Instead of going straight into the Future to verify a user, it can check the local database first.

```

Future<void> _checkDuplicate() async {
    List profileList = await SQLiteLocalProfiles.getProfiles();
    bool alreadyLogged = false;
    for (int i = 0; i < (profileList.length - 1); i++) {
        if (profileList[i]["email"].contains(email.text)) {
            alreadyLogged = true;
        }
    }
    if (alreadyLogged == false) {
        _loginUser();
    } else {
        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text("Profile is already logged in.")));
        });
    }
}

```

For registering a user, it is very different since it has to check the server to see if there is an email already as well as checking if the email already exists. Doing it this way means that checking the local database is unnecessary because if it doesn't exist there is no chance that it is stored locally.

To start with, I tried to use the following code to search the server for the email inputted but I got an error

```

$sqlverify = "SELECT Email FROM profile WHERE Email = '$email'";
if ($res = mysqli_query($link, $sql) > 0){
    $return["exist"] = true;
}
else{
    $sql = "INSERT INTO profile SET
        //try {
        List importedProfile = (recievedServerData["message"]["profile"]);

```

Exception has occurred. ×
TypeError (type 'Null' is not a subtype of type 'List<dynamic>')

As it turned out, I had just forgotten to change \$sql to \$sqlverify so it was running a check on nothing. After fixing this it worked and it was able to verify successfully.

```

} else {
  if ((recievedServerData["message"])["exist"] == true) {
    passwordText.text = "";
    confirmPasswordText.text = "";
    setState(() {
      Navigator.pop(context);
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Profile already exists')),
      );
    });
  } else {
    //try {
    List importedProfile = (recievedServerData["message"])["profile"];
    await solitelocalProfiles.createProfile(

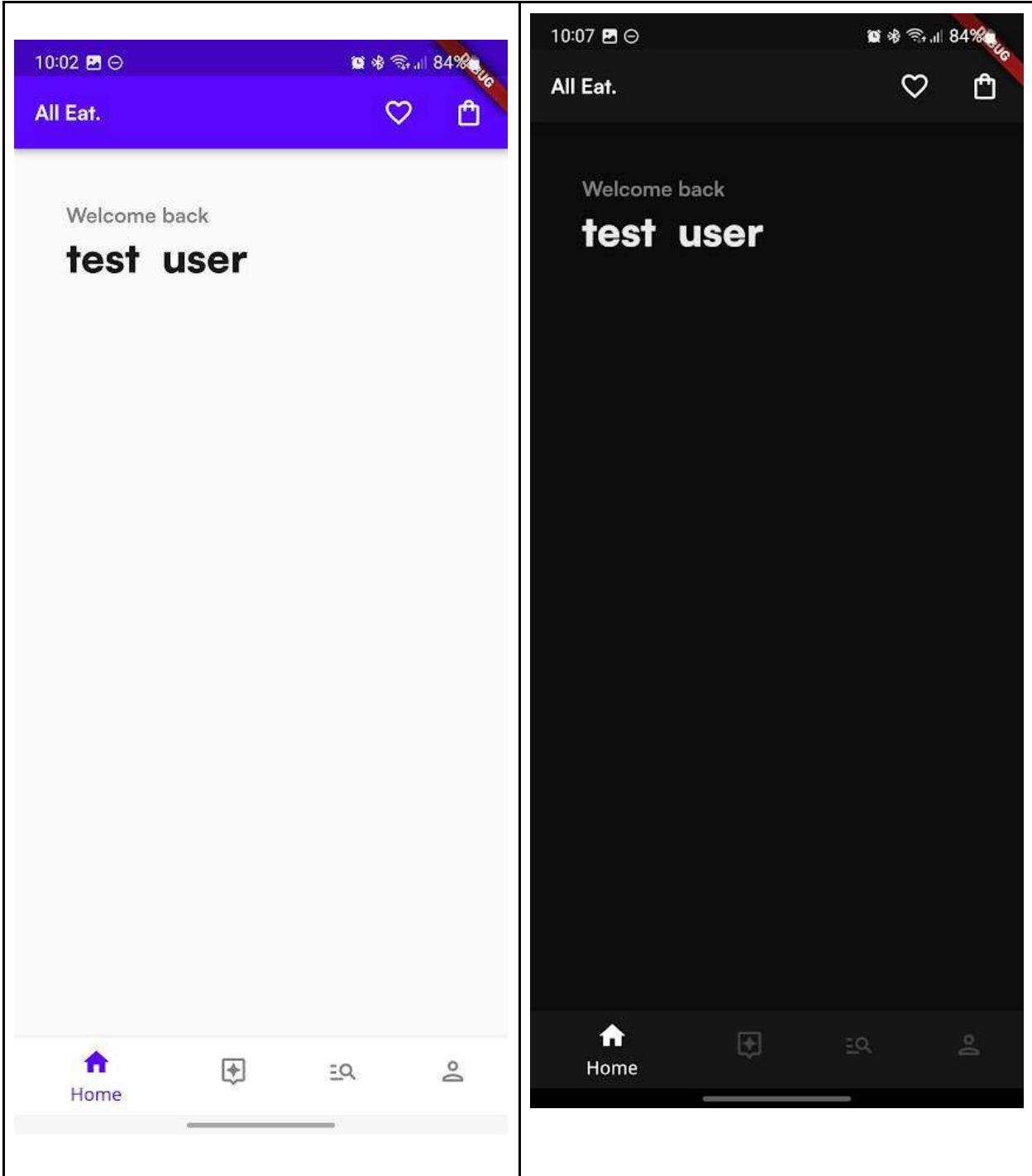
```

Redesigned Top Bar

To get started, I made some extra progress during the progression of Prototype 1B; while making the dark and light themes, I made a dark and light theme for the home page, using the homepage as a reference. I will be remaking the top bar though in order to make it look like the concept art.

Before - Light Mode

Before - Dark Mode



I will be aiming for the following look



Since I didn't know how to make a custom app bar, I used the following resource as a starting point to understand how to make app bar locked to the top of the page.

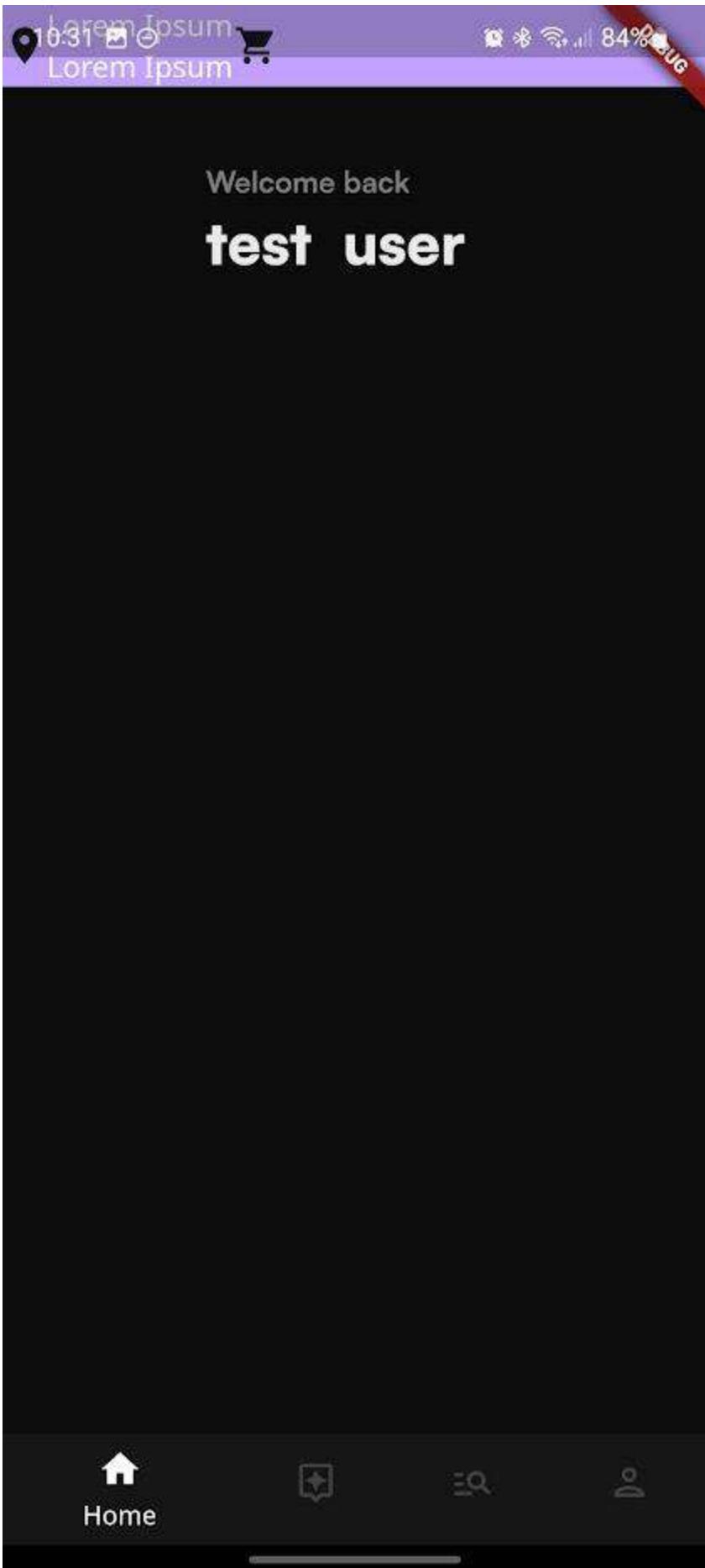
<https://medium.com/flutter-community/flutter-increase-the-power-of-your-appbar-sliverappbar-c4f67c4e076f>

```
import 'package:flutter/material.dart';

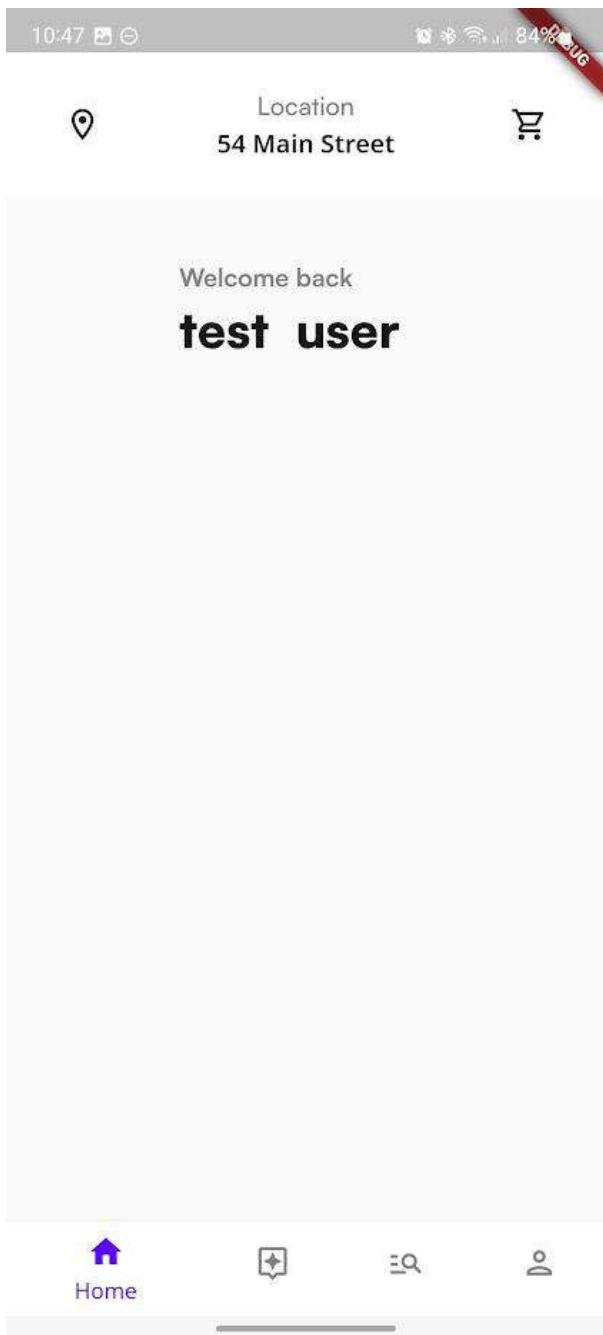
class MainAppBar extends StatelessWidget implements PreferredSizeWidget {
  final double height;
  const MainAppBar({Key? key, required this.height}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      color: Theme.of(context).primaryColor,
      child: Row(children: [
        const Icon(Icons.location_on),
        Column(children: const [Text("Lorem Ipsum"), Text("Lorem Ipsum")]),
        const Icon(Icons.shopping_cart),
      ]),
    );
  }

  @override
  Size get preferredSize => Size.fromHeight(height);
}
```



Since the app bar was overlapping with the notification bar, I added SafeArea to prevent overlap and moved the widgets to the correct area. Doing this, made the topbar almost identical



```
import 'package:flutter/material.dart';

class MainAppBar extends StatelessWidget implements PreferredSizeWidget {
```

```
final double height;

const MainAppBar({Key? key, required this.height}) : super(key: key);

@override

Widget build(BuildContext context) {

    return Container(
        height: 120,
        color: Theme.of(context).bottomNavigationBarTheme.backgroundColor,
        child: SafeArea(
            child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                    Icon(
                        Icons.location_on_outlined,
                        color: Theme.of(context).textTheme.headline1?.color,
                    ),
                    Column(mainAxisAlignment: MainAxisAlignment.center,
children: [
                    Text(
                        "Location",
                        style: Theme.of(context)

```

```
        .headline6  
        ?.copyWith(fontWeight: FontWeight.w500),  
    ),  
    const SizedBox(  
        height: 2,  
    ),  
    Text(  
        "54 Main Street",  
        style: Theme.of(context)  
            .textTheme  
            .bodyText2  
            ?.copyWith(fontWeight: FontWeight.w600),  
    )  
,  
    ]),  
    Icon(  
        Icons.shopping_cart_outlined,  
        color: Theme.of(context).textTheme.headline1?.color,  
    ),  
    ]),  
) );  
}  
  
@override
```

```
    Size get preferredSize => Size.fromHeight(height);  
}
```

After making this change, the profiles section broke. As I found out, it was because the profiles list was expanded and since the top had no defined start, it was an infinite height.

Profiles Page

In order to fix the previous issue, I would need to recode it to have a set size. Instead of doing that, I decided that it would be a good idea to recode it with the final look shown in the concept art.

Profiles.

 Edit



Peter
Smith



Ashley
Ward



Francsca
Goodwin



★ Rate your recent order ★ >



Favourites



Allergies / Diet



Orders



Settings



Profile Settings



Share Profile



Profile Selection

With profile selection, what the app needs to know is the colour of the profile icon and since it should be consistent for every load, it is added during profile creation. Since there are multiple places in which the profile is created, I decided to add the random colour generation to the SQLite Local profiles file when the profile info is being inputted into the database.

```
static Future<void> createProfile(dynamic profileid, String firstname,
|   String lastname, String email, String password) async {
//Create profile using firstname, lastname, email and encrypted password
final db = await SQLiteLocalProfiles.db();
db.insert("localprofiles", {
    //Insert data into localprofiles table
    "profileid": profileid,
    "firstname": firstname,
    "lastname": lastname,
    "email": email,
    "password": password,
    "profilecolor": Color((math.Random().nextDouble() * 0xFFFFFFFF).toInt())
        .withOpacity(1.0),
});
}
```

After doing a test, I realised that this could pick any colour, not just the pastel colours that were wanted. To fix this, a set list of colours was hand-picked and the result was picked.

While doing this, I tried to add it to the database and recall it within the profiles widget but it came out with an error that it needs to be an integer

```
Type: String
The argument type 'String' can't be assigned to the parameter type
'int'. dart(argument_type_not_assignable)
```

To fix this, instead of using a hex value, I used an RGB value and store the list

```

//Create profile using firstname, lastname, email and encrypted password
List colors = [
  [46, 41, 78],
  [239, 188, 213],
  [190, 151, 198],
  [134, 97, 193],
  [77, 104, 184],
  [112, 202, 209]
];
Random random = Random();
List randomProfileColor = colors[random.nextInt(colors.length)];
final db = await SQLiteLocalProfiles.db();
db.insert("localprofiles", {
  //Insert data into localprofiles table
  "profileid": profileid,
}

```

If a user selects a profile, the user id, first name, last name and email is grabbed and sent to a file where it will grab the favourite restaurants from the server and store them.

```

import 'package:alleat/services/queryserver.dart';
import 'package:shared_preferences/shared_preferences.dart';

class SetSelected {
  static Future<bool> selectProfile(id, firstname, lastname, email) async {
    dynamic favReataurantList = await QueryServer.query(
      'https://alleat.cpur.net/query/favouriterestaurantlist.php',
      {"profileemail": email});
    if (favReataurantList["Error"] == true) {
      return false;
    } else {
      final prefs = await SharedPreferences.getInstance();
      await prefs.setString('profileid', id);
      await prefs.setString('firstname', firstname);
      await prefs.setString('lastname', lastname);
      await prefs.setString('email', email);
      await prefs.setString('favrestaurants',
        ((favReataurantList["message"])["restaurantids"]).toString());
    }
    return true;
  }
}

```

To make the profile selection element, I created a file which has a scrollable section and uses a FutureBuilder to grab the profiles in the database and display them. The profile is then displayed.

```
import 'package:alleat/screens/profilesetup/profilesetup_existing.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:flutter/material.dart';

class ProfileList extends StatefulWidget {

  const ProfileList({super.key});

  @override

  State<ProfileList> createState() => _ProfileListState();
}

class _ProfileListState extends State<ProfileList> {

  bool edit = false;

  Future<List> getDisplayableProfiles() async {

    return await SQLiteLocalProfiles.getDisplayProfilesList();
  }

  @override

  Widget build(BuildContext context) {

    return Container(
```

```
//Create profile selection widget with height 150px

height: 150,

padding: const EdgeInsets.symmetric(horizontal: 10),

child: ListView(scrollDirection: Axis.horizontal, children:

<Widget>[

    FutureBuilder<List> (

        // Get selected profile

        future: getDisplayableProfiles(),

        builder: (context, snapshot) {

            if (!snapshot.hasData) {

                //While there is no information sent back

                return Column(

                    //Display empty circle

                    children: [

                        Container(

                            width: 100,

                            height: 100,

                            decoration: BoxDecoration(

                                shape: BoxShape.circle,

                                color: Theme.of(context)

                                    .navigationBarTheme

                                    .backgroundColor))

                ],

```

```
);

} else if (snapshot.hasData) {

    // Once there is information

    List? profileInfo = snapshot.data; // Store in
profileInfo

    print(profileInfo);

    if (profileInfo != null && profileInfo.isNotEmpty) {

        //If it contains a profile

        return SizedBox(
            //Set width of data to the number of profiles by
120px

            width: (profileInfo.length) * 100,

            child: ListView.builder(
                //For each profile

                itemCount: profileInfo.length,
                scrollDirection:
                    Axis.horizontal, //Make scrollable list

                itemBuilder: (context, index) {

                    Map<String, dynamic> profile = profileInfo[
index]; //Store current profile being
created in profile

                    if (index == 0) {

                        return Container(
                            padding: const EdgeInsets.symmetric(
```

```
        horizontal: 10),  
  
        child: Column(children: [  
  
          Container(  
  
            // Create a circle with a purple  
            outline and and the first letter of first and last name  
  
            width: 80,  
  
            height: 80,  
  
            decoration: BoxDecoration(  
  
              shape: BoxShape.circle,  
  
              border: Border.all(  
  
                color: Theme.of(context)  
  
                .primaryColor,  
  
                width: 3,  
  
                style:  
BorderStyle.solid),  
  
              color: Color.fromRGBO(  
  
profile['profilecolorred'],  
  
profile['profilecolorgreen'],  
  
profile['profilecolorblue'],  
  
          1)),  
  
          child: Align(  
  
            alignment: Alignment.center,
```

```
        child: Text(

'${profile['firstname'][0]}${profile['lastname'][0]}',

        style: Theme.of(context)

        .textTheme

        .headline3

        ?.copyWith(

            color: Theme.of(

                context)


        .backgroundColor))),

        const SizedBox(height: 20),

        SizedBox(

        //Display profile name

        width: 80,

        child: Text(

        "${profile['firstname']}",

        textAlign: TextAlign.center,

        style: Theme.of(context)

        .textTheme

        .headline5

        ?.copyWith(

            fontWeight:

        FontWeight.w600,

```

```
        color:  
Theme.of(context)  
  
            .textTheme  
            .headline1  
            ?.color),  
  
        overflow:  
TextOverflow.ellipsis,  
    ) )  
]);  
} else if (index > 0) {  
  
    return Container(  
  
        padding: const EdgeInsets.symmetric(  
            horizontal: 10),  
  
        child: Column(children: [  
  
            Container(  
  
                // Create a circle with a purple  
outline and and the first letter of first and last name  
  
                width: 80,  
  
                height: 80,  
  
                decoration: BoxDecoration(  
  
                    shape: BoxShape.circle,  
  
                    border: Border.all(  
  
                        color: Theme.of(context)  
                        .colorScheme
```

```
        .onSecondary,  
        width: 3,  
        style:  
BorderStyle.solid),  
        color: Color.fromRGBO(  
  
profile['profilecolorred'],  
  
profile['profilecolorgreen'],  
  
profile['profilecolorblue'],  
        1)),  
        child: Align(  
        alignment: Alignment.center,  
        child: Text(  
  
'${profile['firstname'][0]}${profile['lastname'][0]}',  
        style: Theme.of(context)  
        .textTheme  
        .headline3  
        ?.copyWith(  
        color: Theme.of(  
        context)  
  
.backgroundColor))));
```

```
        const SizedBox(height: 20),  
  
        SizedBox(  
  
            //Display profile name  
  
            width: 80,  
  
            child: Text(  
  
                "${profile['firstname']} ",  
  
                textAlign: TextAlign.center,  
  
                style: Theme.of(context)  
  
                    .textTheme  
  
                    .headline5  
  
                    ?.copyWith(  
  
                        fontWeight:  
FontWeight.w600,  
  
                        color:  
Theme.of(context)  
  
                    .textTheme  
  
                    .headline1  
  
                    ?.color),  
  
                    overflow:  
TextOverflow.ellipsis,  
  
            )  
  
        ]));  
  
    } else {  
  
        return Container(  
    }
```

```
padding: const EdgeInsets.symmetric(
    horizontal: 10),
child: Column(children: [
    Container(
        // Create a circle with a purple
outline and and the first letter of first and last name
        width: 80,
        height: 80,
        decoration: BoxDecoration(
            shape: BoxShape.circle,
            color: Theme.of(context)
                .colorScheme
                .error),
        child: Align(
            alignment: Alignment.center,
            child: Text(
                "Error",
                style: Theme.of(context)
                    .textTheme
                    .headline5
                    ?.copyWith(
                        color:
Color(0xffffffff)),
        ),
    ),
],
```

```
) )  
]  
}  
});  
}  
} else {  
    return Column(  
        children: [  
            Container(  
                width: 80,  
                height: 80,  
                decoration: BoxDecoration(  
                    shape: BoxShape.circle,  
                    color: Theme.of(context).errorColor)  
                ),  
            ),  
        ],  
    );  
}  
} else {  
    return Column(  
        children: [  
            Container(  
                width: 80,  
                height: 80,  
                decoration: BoxDecoration(  
                    shape: BoxShape.circle,  
                    color: Theme.of(context).primaryColorDark  
                ),  
            ),  
        ],  
    );  
}
```

```
        shape: BoxShape.circle,
        color: Theme.of(context).errorColor))

    ],
);

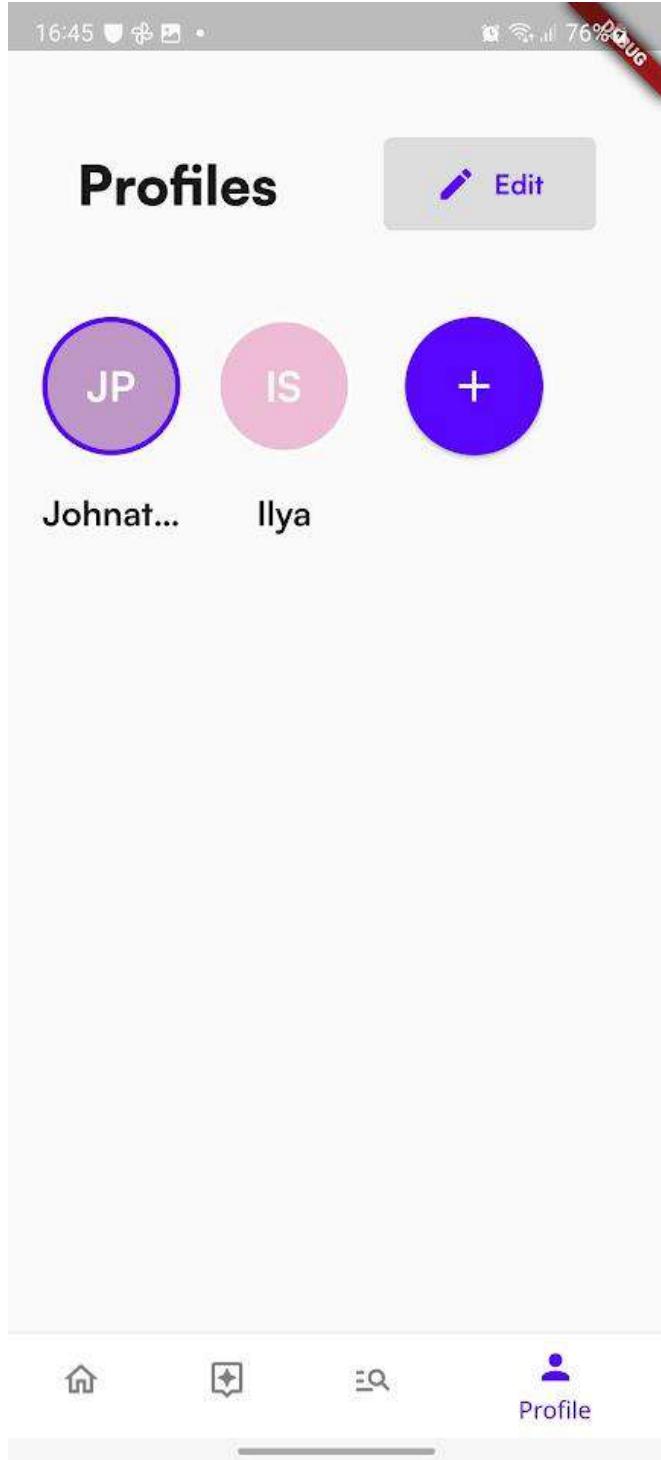
}

}),

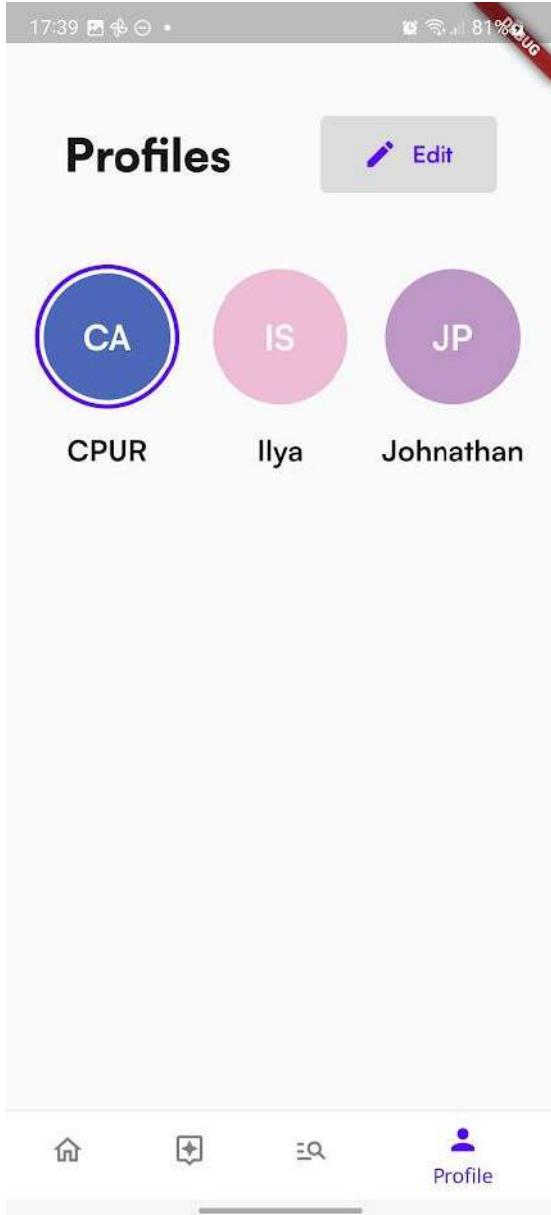
Column(
  children: [
    Padding(
      padding: EdgeInsets.symmetric(horizontal: 20),
      child: Column(children: [
        ElevatedButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>
                  const ProfileSetupExisting())));
          },
        ),
        style: ElevatedButton.styleFrom(
          shape: const CircleBorder(),
          padding: const EdgeInsets.all(25)),
        child: Icon(

```

```
Icons.add,  
color: Theme.of(context).backgroundColor,  
size: 30,  
) ,  
) ,  
const SizedBox(height: 20),  
]))  
] ,  
)  
]) );  
}  
}
```



After using a longer name, I found that this method of displaying profiles was not good so the dimensions were adjusted and the selected look was fixed in order to better distinguish the difference



I then made it so that clicking an unselected profile, would change the selected profile and refresh the profiles page.

In order to simplify development, the decision to have an edit button that would update the widgets was not put forward, and instead would be put within the profile settings section.

The next section that was completed was the buttons. These were easy to port over from the concept art since it had shadow information and padding. In order to reduce the amount of repeated code, a file was created for the information to be inputted into.

```
import 'package:flutter/material.dart';
```

```
class ProfileButton extends StatelessWidget {

    final dynamic icon;

    final dynamic name;

    final dynamic action;

    const ProfileButton({Key? key, this.icon, this.name, this.action})

        : super(key: key);

    @override

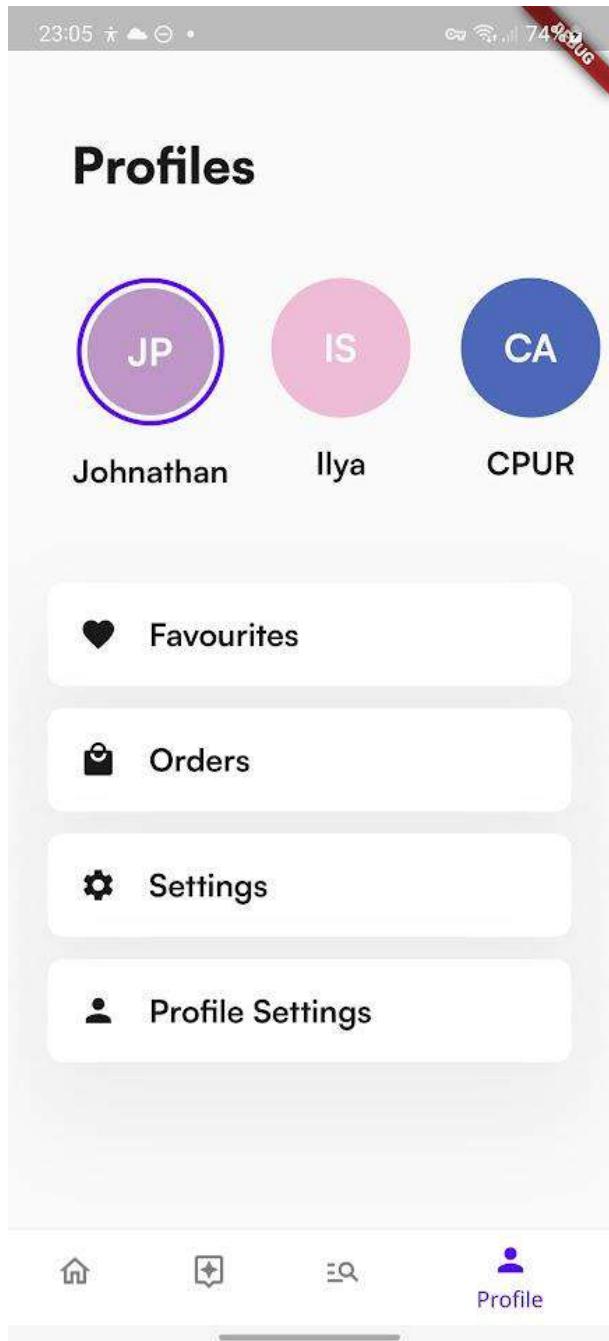
    Widget build(BuildContext context) {

        return Container(
            padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 15),
            width: double.infinity,
            height: 65,
            margin: const EdgeInsets.symmetric(vertical: 7, horizontal: 25),
            decoration: BoxDecoration(
                borderRadius: const BorderRadius.all(Radius.circular(10)),
                color: Theme.of(context).colorScheme.onSurface,
                boxShadow: [
                    BoxShadow(
                        color:
                        Theme.of(context).colorScheme.onBackground.withOpacity(0.05),
                        spreadRadius: 10,
```

```
        blurRadius: 45,  
  
        offset: const Offset(0, 30), // changes position of shadow  
    ),  
],  
,  
child: InkWell(  
  
    child: Row(  
  
        children: [  
  
            Icon(icon, color: Theme.of(context).colorScheme.onBackground),  
  
            const SizedBox(width: 20),  
  
            Text(  
  
                name,  
  
                style: Theme.of(context).textTheme.headline5?.copyWith(  
  
                    color: Theme.of(context).colorScheme.onBackground,  
  
                    fontWeight: FontWeight.w700),  
  
            )  
  
        ],  
    ),  
    onTap: () {  
  
        action;  
  
    },  
),  
);
```

```
    }  
}
```

```
| | ), // Row // Padding  
const SizedBox(  
| height: 50,  
) , // SizedBox  
const ProfileList(),  
const SizedBox(  
| height: 35,  
) , // SizedBox  
const ProfileButton(  
| icon: (Icons.favorite),  
name: "Favourites",  
action: null,  
) , // ProfileButton  
const ProfileButton(  
| icon: (Icons.local_mall),  
name: "Orders",  
action: null,  
) , // ProfileButton  
const ProfileButton(  
| icon: (Icons.settings),  
name: "Settings",  
action: null,  
) , // ProfileButton  
const ProfileButton(  
| icon: (Icons.person),  
name: "Profile Settings",  
action: null,  
) // ProfileButton
```



After testing where the internet is disconnected and the user tries to switch profiles and then enable the internet, an error occurs.

```
20 |     await prefs.setString('favrestaurants',
21 |       ((favReataurantList["message"]) D ["restaurantids"]).toString());
Exception has occurred. ×
_TypeError (type 'String' is not a subtype of type 'int' of 'index')
22 |   SQLiteLocalProfiles.selectProfile(email);
```

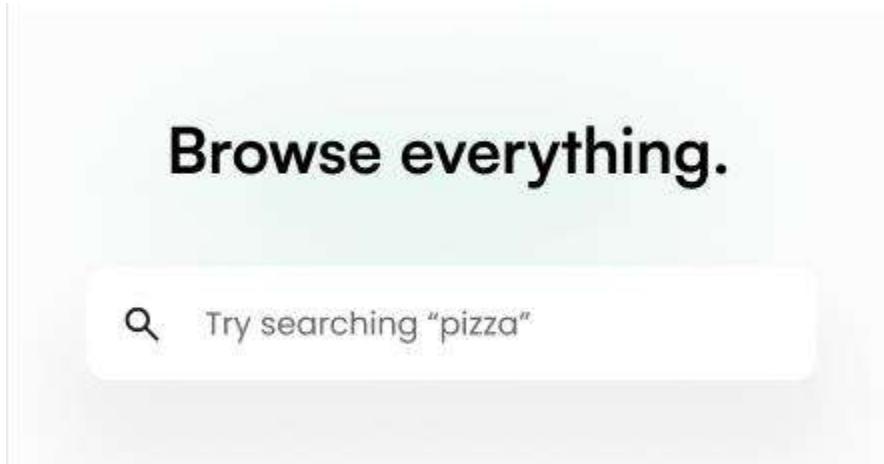
In order to the favourites section, similarly to prototype 1A, I would need to do the browse section.

Browse Page

Search Bar

To start off with, as the concept art shows, the browse page has a title and a search bar. This search bar will not be done in this prototype so a placeholder will be put in instead, where it will be an Inkwell so that it can be clicked in the future.

Concept Art



Since the search bar can be used on multiple pages, the search bar will be made as a widget.

After some consideration, the title of browse everything was unnecessary since a user would already know they were on the browse page by the bottom nav bar so the title was removed

New Concept Art



Location

11 Canal Reach, London



Try searching "pizza"

Popular Categories.

Burgers

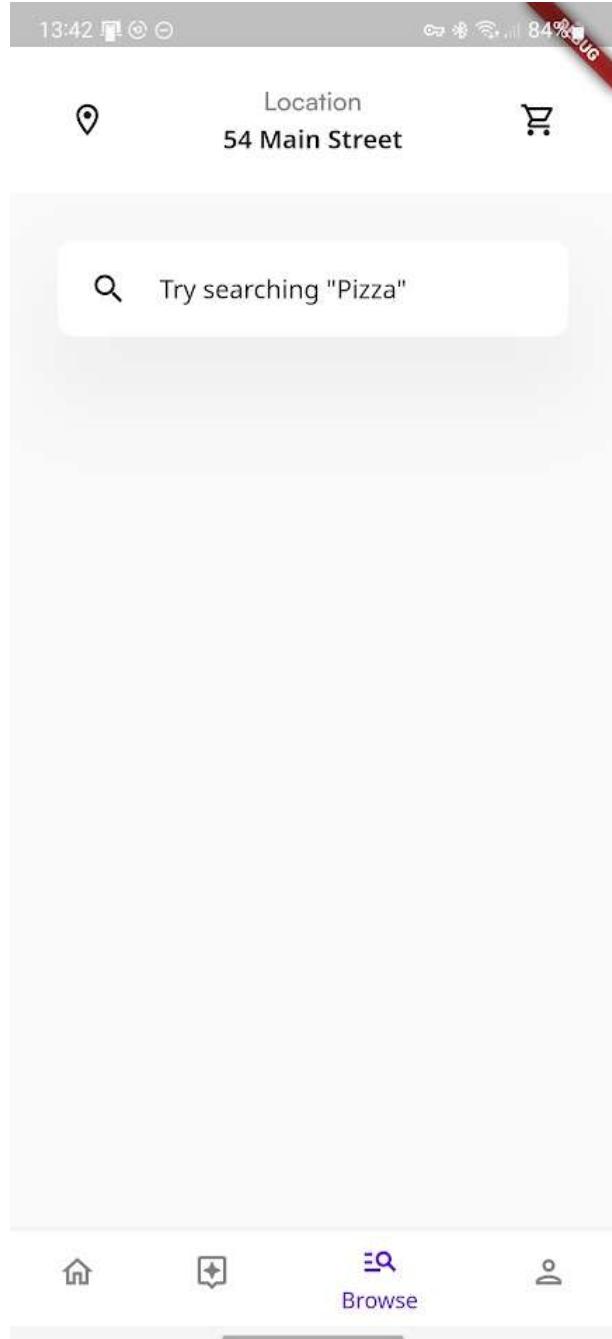


American



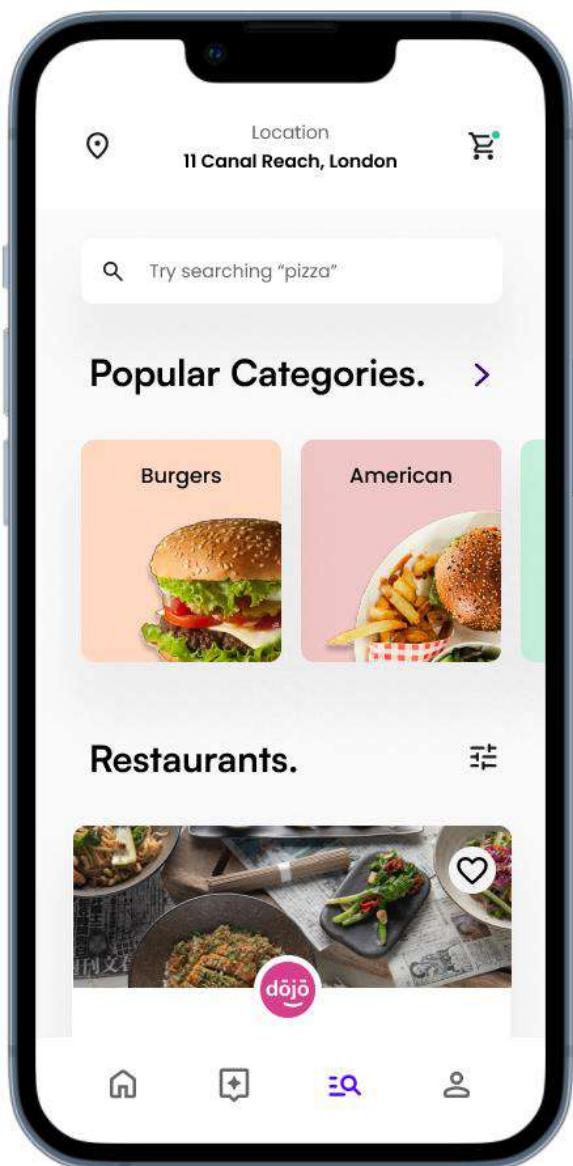
```
class SearchBar extends StatelessWidget {
  const SearchBar({super.key});

  @override
  Widget build(BuildContext context) {
    return InkWell(
      child: Container(
        width: double.infinity,
        height: 60,
        margin: const EdgeInsets.symmetric(vertical: 10, horizontal: 30),
        padding: const EdgeInsets.symmetric(horizontal: 20),
        decoration: BoxDecoration(
          borderRadius: const BorderRadius.all(Radius.circular(10)),
          color: Theme.of(context).colorScheme.onSurface,
          boxShadow: [
            BoxShadow(
              color: Theme.of(context).colorScheme.onBackground.withOpacity(0.05),
              spreadRadius: 10,
              blurRadius: 45,
              offset: const Offset(0, 30), // changes position of shadow
            ), // BoxShadow
          ],
        ), // BoxDecoration
        child: Row(children: [
          Icon(
            Icons.search,
            color: Theme.of(context).textTheme.headline1?.color,
          ), // Icon
          const SizedBox(
            width: 20,
          ), // SizedBox
          Text(
            'Try searching "Pizza"',
            style: Theme.of(context).textTheme.bodyText2,
          ) // Text
        ]), // Row
      )), // Container // InkWell
    }
}
```



Categories

One thing I wanted to include was categories and the ability to refresh the page with a swipe. The only way to do this was to make the categories be a horizontal scroll while the restaurants load vertically.



Using this format also allows the app to be denser and there to be more variety. The original format can instead now be used for the all categories page

[◀ Back](#)

All Categories.

Burgers



American



Breakfast



Sushi



Chinese



Thai



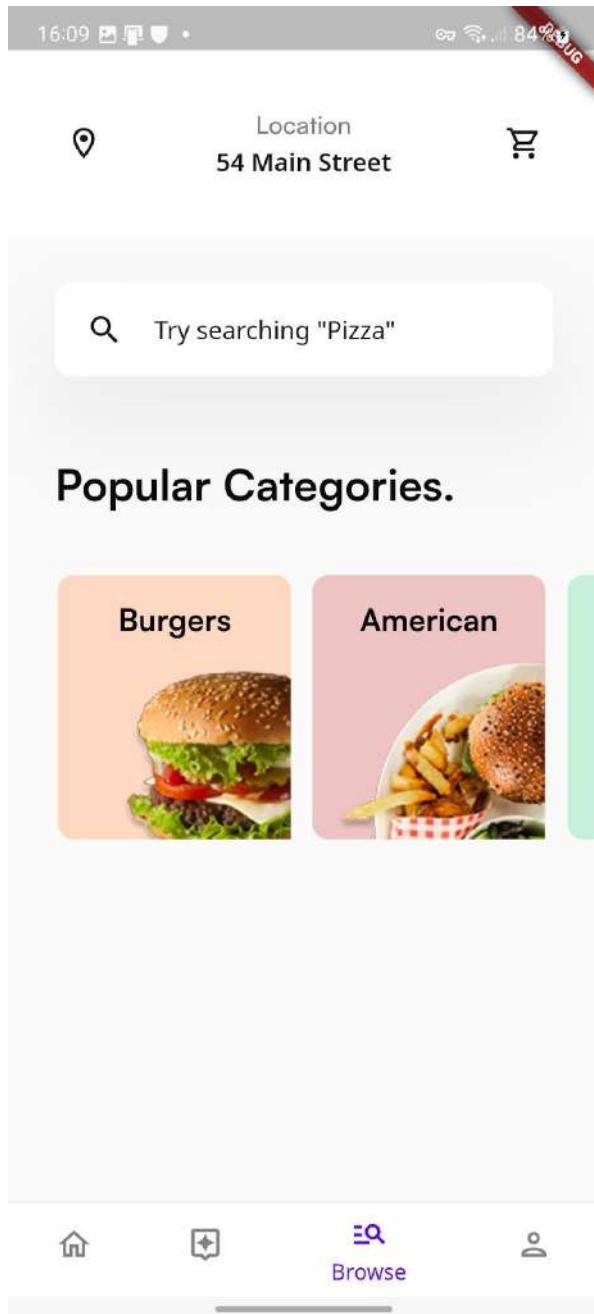
Coffee



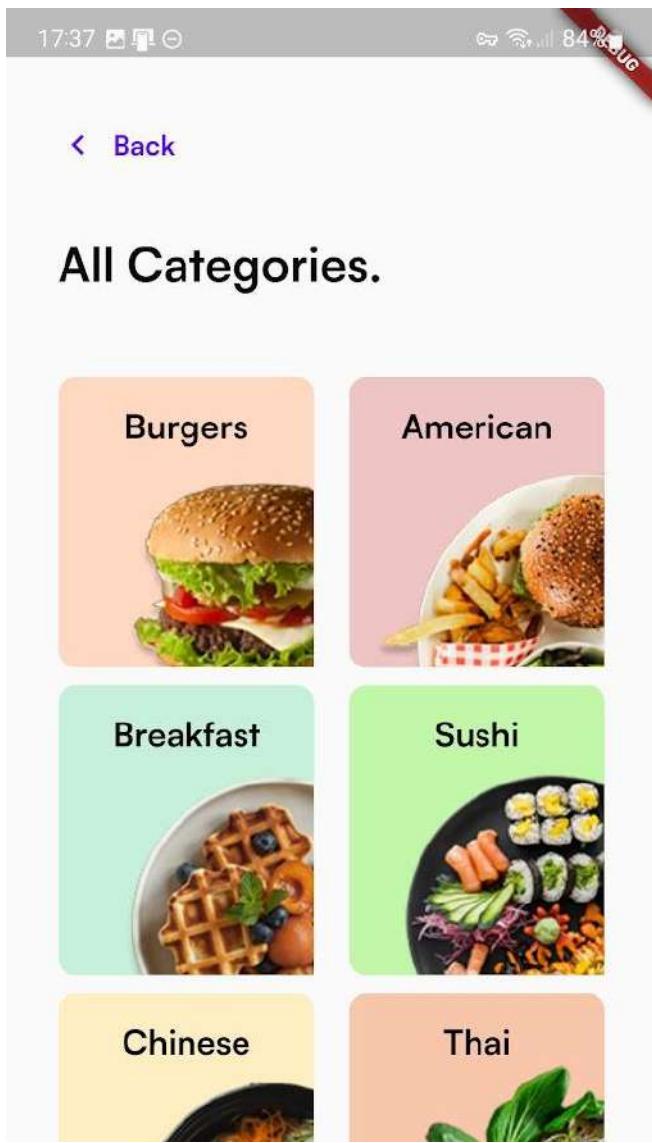
Greek



In order to make the scrollable categories, I created a 2D array containing the list of category names, colour and the image associated with it. I then made a ListView which picked the first 5 of the categories and displayed them in stacks



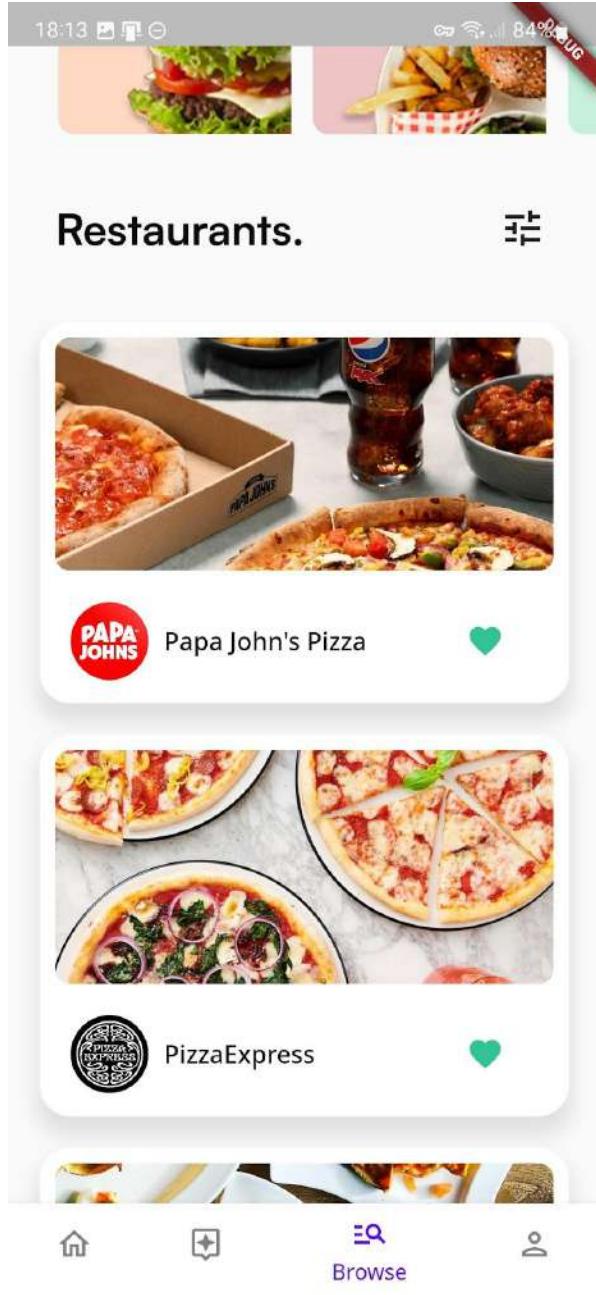
I then created a page which used the whole array containing the list of categories in a grid view.



Restaurant Summaries

To make the restaurant summaries, I used the code from prototype 1A as a building block. Copying the code.

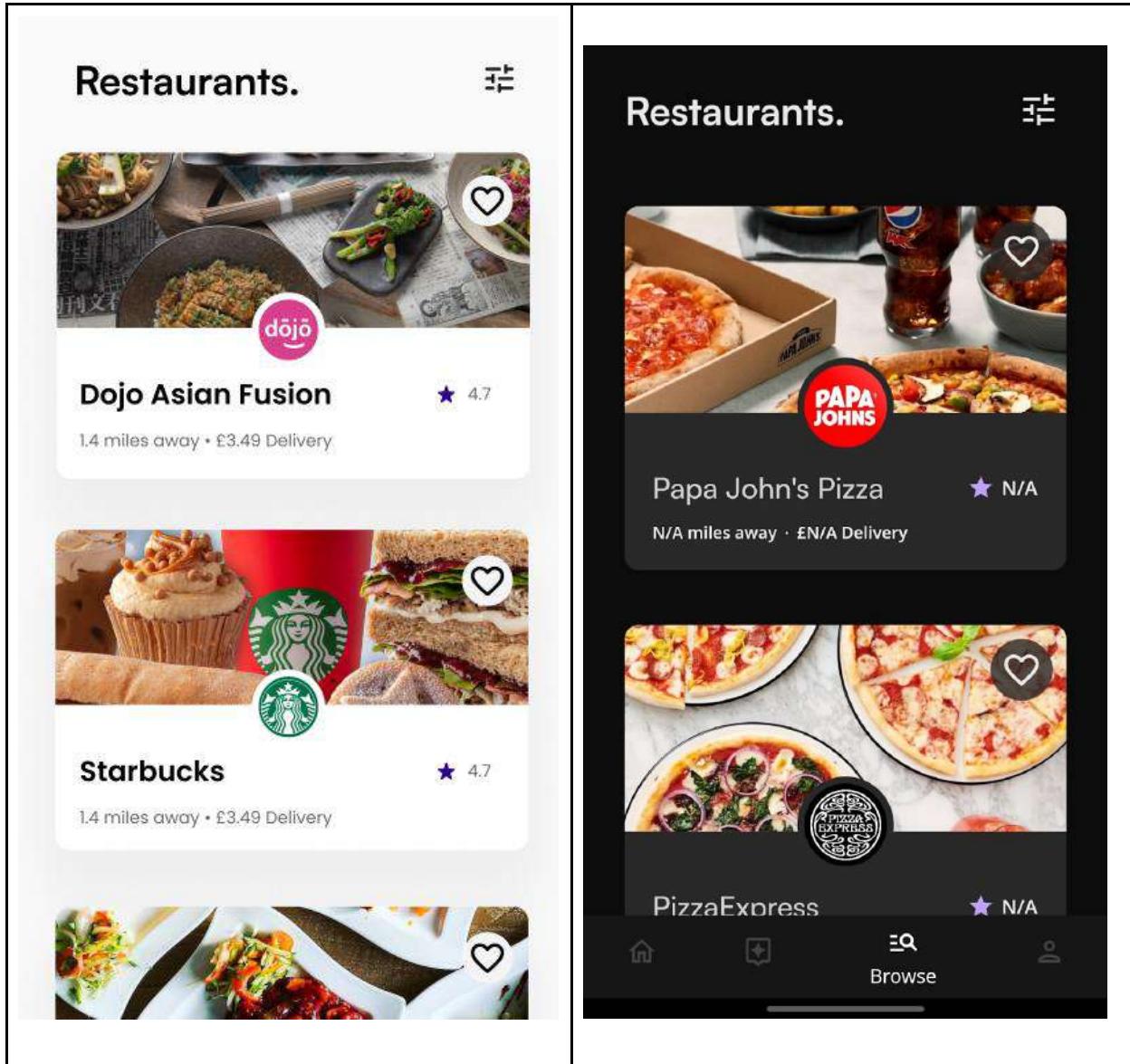
I first changed the selected profile code instead of checking the database, checking in shared preferences since it takes up less code.



I then used the dimensions of the concept art to remake each restaurant summary, using stacks.

Concept Art

In App



Restaurant Page

To make the restaurant page, the same strategy was used as the restaurant summary, altering the previous design to fit the concept art look. This was very successful, with few changes needed in order for it to work. While previously, I was using ratios in order to determine the sizes of each menu item, with the new one, I used a set size for the image and the price while the description was adaptive meaning on smaller devices everything could be seen without any overlap. The top of the page used a stack in order to fit everything on top of each other, with a container being used at the bottom of the image in order to have the image curve outwards.

Concept Art	In App
 <p>Papa John's ★ 4.1 (12)</p> <p>Pizza • American • ££</p> <p>1.4 miles away • £3.49 Delivery • £20 Min. Order</p> <p>Currently ordering for</p> <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">  Peter Smith > </div> <p>Menu</p> <p>Pizza</p> <ul style="list-style-type: none">  Cheese & Tomato £ 14.99 The classic, full of flavour, with our tomato sauce and mozzarella. Enjoy it pure ...  All the Meats £ 17.99 Our tomato sauce with mozzarella, pepperoni, pork sausage, crispy bacon, ... 	 <p>Papa John's Pizza ★ N/A</p> <p>N/A miles away • £N/A Delivery</p> <p>Pizza</p> <ul style="list-style-type: none">  All the Meats Our tomato sauce with mozzarella, pepperoni, pork sausage, crispy bacon, spicy ... £ 17.99  American Hot Our tomato sauce with mozzarella, slices of spicy pepperoni and jalapeño ... £ 15.99  BBQ Meat Feast BBQ sauce with mozzarella, spicy beef, sliced pepperoni, ham, pork sausage and ... £ 17.99  BBQ Chicken Classic Our tomato sauce with ... £ 14.99

An issue I faced was if the internet was disconnected after getting the restaurant summary, and the user would click on a restaurant, the app would crash because there would be no other info sent back from the server.

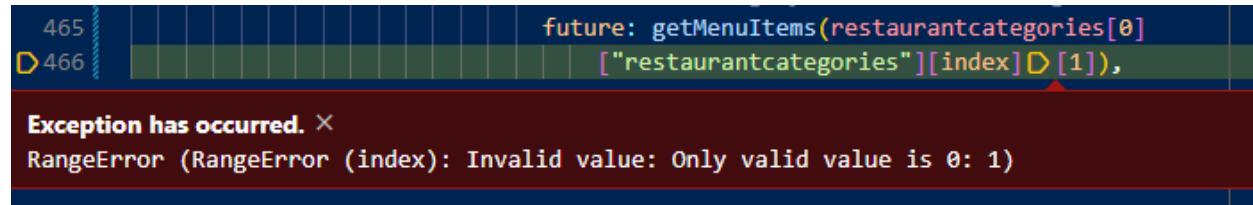


To fix this error, I needed to add the following code that would display an error icon if the server failed to answer the request.

```
List restaurantFavourites = snapshot.data ?? [];
if (restaurantFavourites[0]["error"] == true) {
    return SafeArea(
        child: Align(
            alignment: Alignment.topRight,
            child: Container(
                margin: const EdgeInsets.only(top: 20, right: 20),
                width: 45,
                height: 45,
                decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color: Theme.of(context)
                        .colorScheme
                        .onSurface
                        .withOpacity(0.8),
                ), // BoxDecoration
                child: Icon(
                    Icons.error,
                    color: Theme.of(context).colorScheme.error,
                )))); // Icon // Container // Align // SafeArea
} else {
    return SafeArea(

```

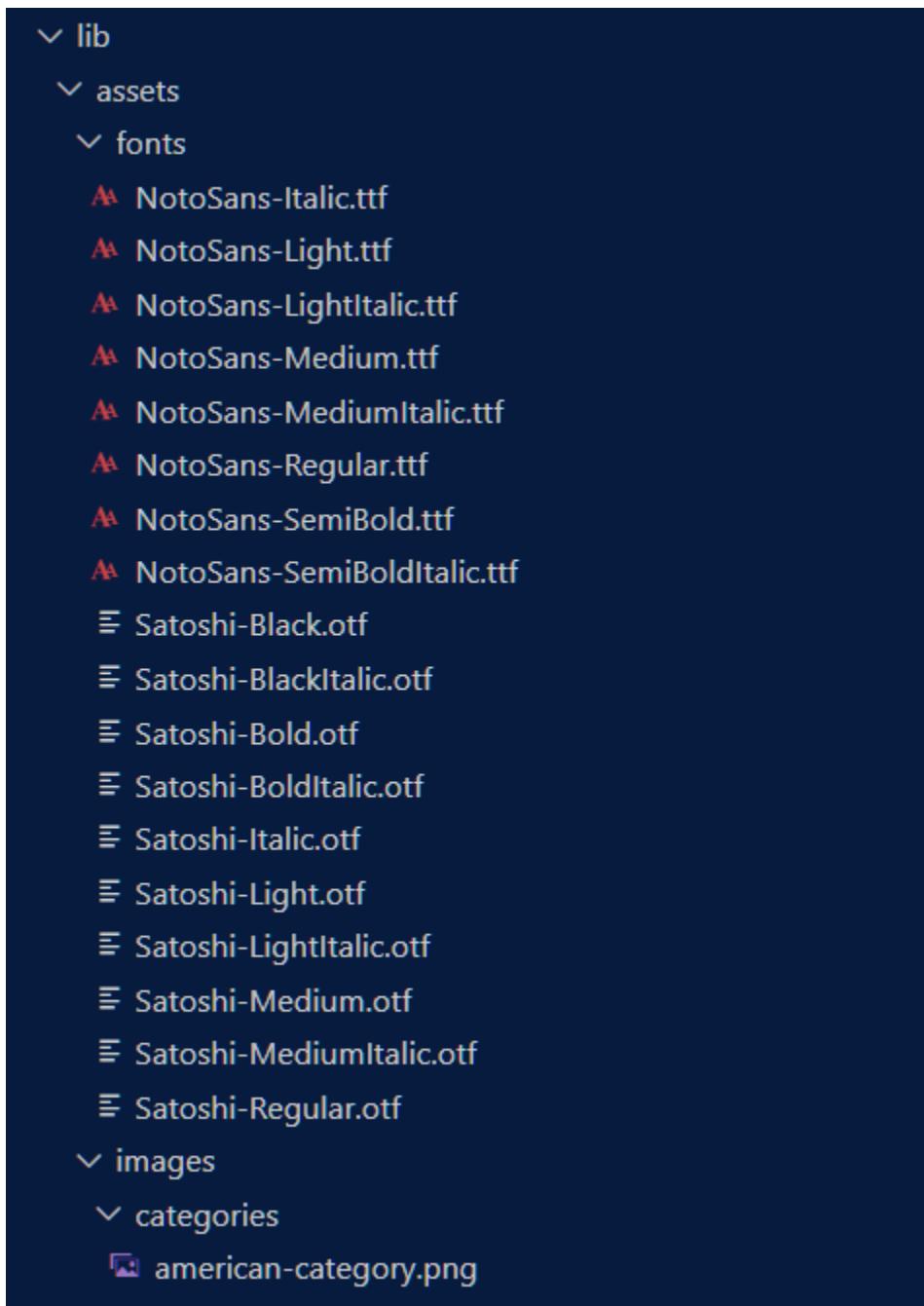
What would then occur is that it would infinitely load and once a connection was established, the app would crash and the following error would occur.



After some research, it seems that this would require me to create a session ID and store it until the user reconnects with the server and due to the size of the project I will not be doing it.

Prototype 1B Final Code

File Structure

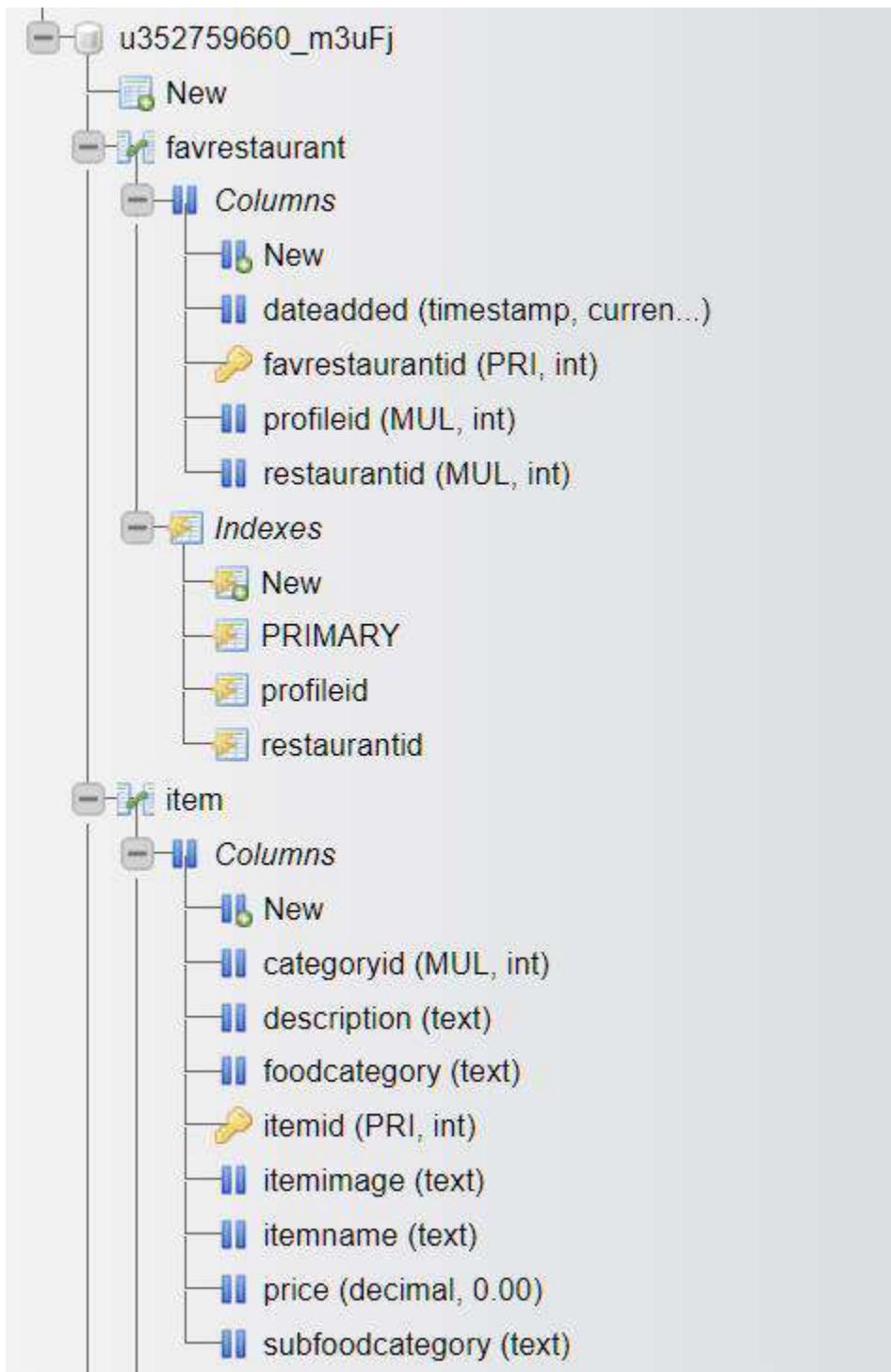


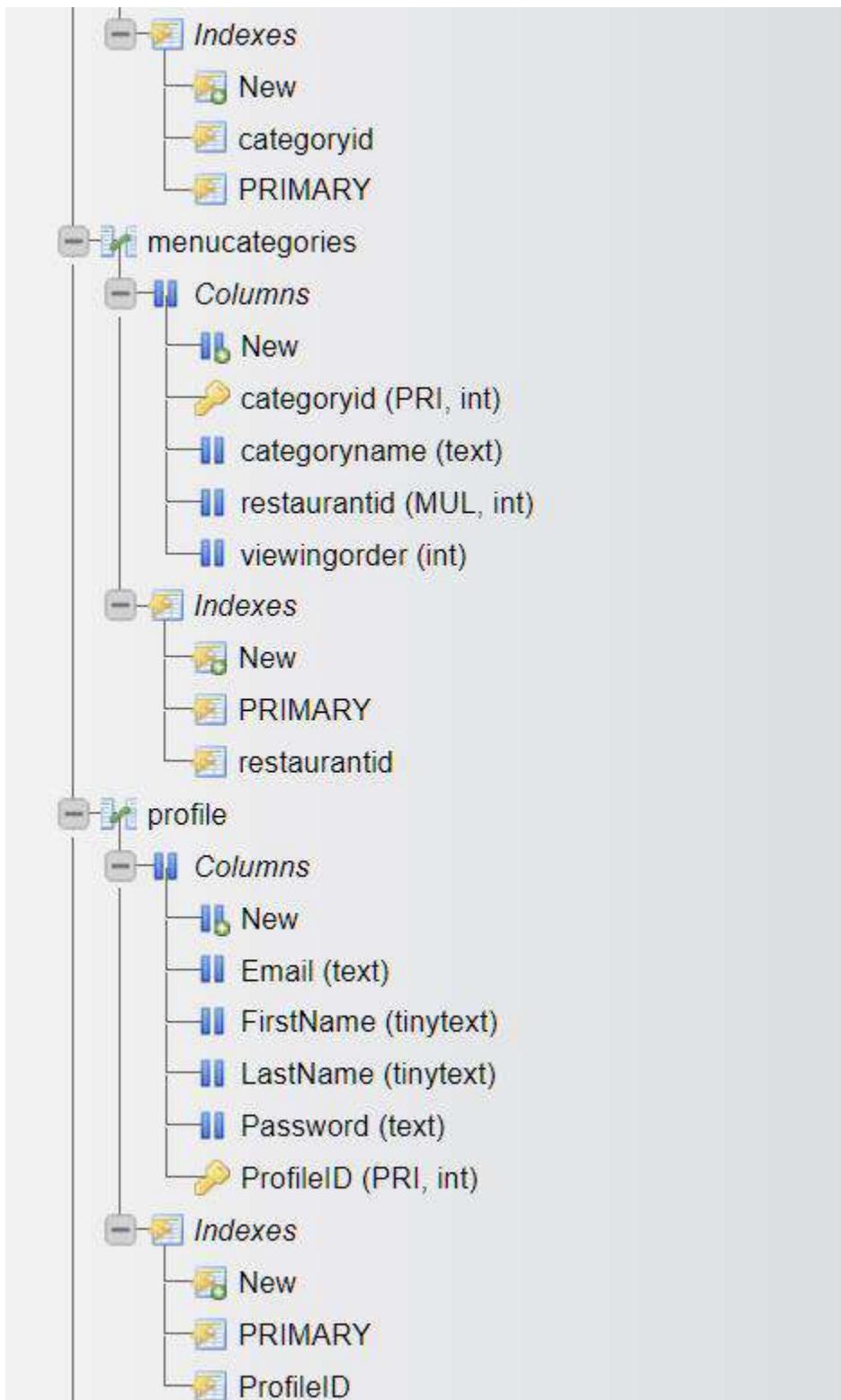
- ☒ breakfast-category.png
- ☒ burger-category.png
- ☒ chinese-category.png
- ☒ coffee-category.png
- ☒ greek-category.png
- ☒ sushi-category.png
- ☒ thai-category.png
- ✓ logo
 - ☒ foregroundicon.png
 - ☒ icon.png
 - ☒ logodark.png
 - ☒ splashscreen-android12-dark.png
 - ☒ splashscreen-android12.png
- ✓ screens
 - ✓ existingsetup
 - ☒ existingsetupscreenfood.jpg
 - ✓ profilesetup
 - ☒ login-illustration-dark.png
 - ☒ login-illustration-light.png
 - ✓ welcomescreen
 - ☒ welcomescreenfood.jpg

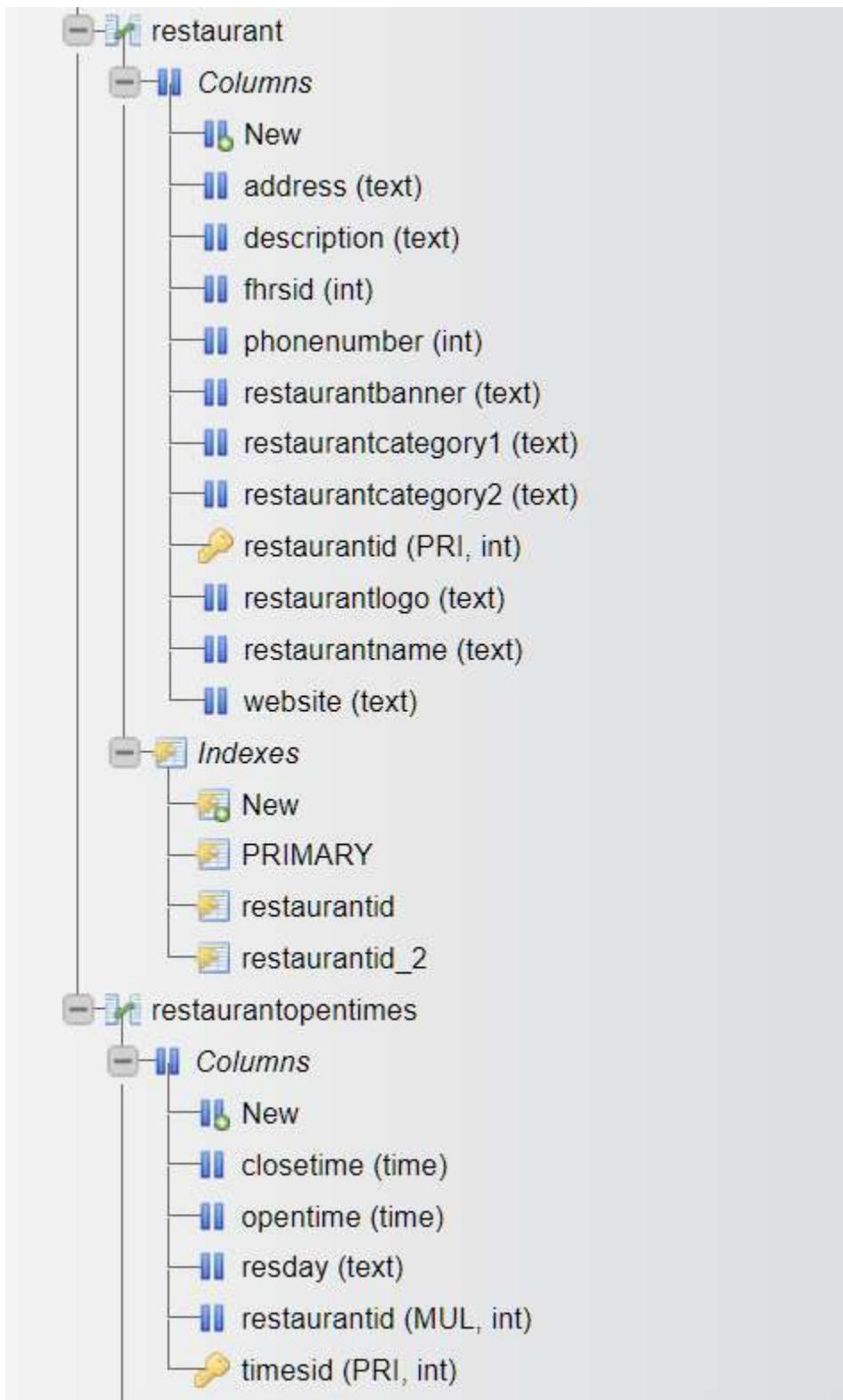
```
✓ screens
  ✓ navigationscreens
    ⚒ browse.dart
    ⚒ foryou.dart
    ⚒ homepage.dart
    ⚒ profiles.dart
  ✓ profilesetup
    ⚒ profilesetup_create.dart
    ⚒ profilesetup_existing.dart
    ⚒ profilesetup_login.dart
    ⚒ welcomescreen.dart
  ✓ restaurant
    ⚒ restaurant_customise.dart
    ⚒ restaurant_main.dart
    ⚒ setupverification.dart
  ✓ services
    ⚒ dataencryption.dart
    ⚒ localprofiles_service.dart
    ⚒ queryserver.dart
    ⚒ setselected.dart
  ✓ theme
    ⚒ theme.dart
  ✓ widgets
    ✓ elements
```

- browse_categories.dart
- elements.dart
- profiles_buttons.dart
- profiles_selection.dart
- search.dart
- genericlocading.dart
- navigationbar.dart
- restaurants_list.dart
- topbar.dart
- main.dart

- <> favouriterestaurant.php
- <> favouriterestaurantdata.php
- <> favouriterestaurantlist.php
- <> login.php
- <> register.php
- <> restaurantlist.php
- <> restaurantmenucategories.php
- <> restaurantmenuitems.php









Files - Application

Browse.dart

```
import 'package:alleat/widgets/elements/browse_categories.dart';

import 'package:alleat/widgets/elements/search.dart';

import 'package:alleat/widgets/restaurants_list.dart';

import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

class BrowsePage extends StatefulWidget {

  const BrowsePage({Key? key}) : super(key: key);

  @override

  State<BrowsePage> createState() => _BrowsePageState();
}

class _BrowsePageState extends State<BrowsePage> {

  @override

  Widget build(BuildContext context) {
```

```
return SafeArea(  
    //Keep within screen area  
  
    child: Scaffold(  
  
        body: SingleChildScrollView(  
  
            child: Column(children: [  
  
                const MainAppBar(  
  
                    height: 150,  
  
                ),  
  
                const SizedBox(height: 20),  
  
                const SearchBar(), //Search bar  
  
                const SizedBox(height: 40),  
  
                Padding(  
  
                    padding: const EdgeInsets.symmetric(horizontal: 30),  
  
                    child: Row(  
  
                        mainAxisAlignment: MainAxisAlignment.center,  
  
                        mainAxisSize: MainAxisSize.spaceBetween,  
  
                        children: [  
  
                            //Display row with the title popular categories and an  
                            //arrow pointing right to go to all the categories  
  
                            Text(  
  
                                "Popular Categories.",  
  
                                style: Theme.of(context).textTheme.headline2,  
  
                            ),  
                        ],  
                    ),  
                ),  
            ],  
        ),  
    ),  
);
```

```
InkWell(  
    child: Padding(  
        padding: const EdgeInsets.all(5),  
        child: Icon(  
            Icons chevron_right,  
            size: 30,  
            color: Theme.of(context).colorScheme.onBackground,  
        )),  
    onTap: () => (Navigator.push(  
        context,  
        MaterialPageRoute(  
            builder: (context) => const CategoriesPage(),  
        ))))  
,  
) ,  
  
const SizedBox(height: 40),  
const Categories(), // Display horizontally scrolling categories  
const SizedBox(height: 40),  
Padding(  
    padding: const EdgeInsets.symmetric(horizontal: 30),  
    child: Row(  
        mainAxisAlignment: MainAxisAlignment.spaceBetween,  
        mainAxisSize: MainAxisSize.max,
```

```
        children: [  
  
            // Display title restaurants with a filter icon to go to  
filtering options  
  
            Text(  
  
                "Restaurants.",  
  
                style: Theme.of(context).textTheme.headline2,  
            ),  
  
            InkWell(  
  
                child: Padding(  
  
                    padding: const EdgeInsets.all(5),  
  
                    child: Icon(  
  
                        Icons.tune,  
  
                        size: 30,  
  
                        color: Theme.of(context).colorScheme.onBackground,  
                    ),  
  
                    onTap: () => (null))  
  
            ],  
  
        )),  
  
        const SizedBox(height: 30),  
  
        const RestaurantList(),  
    ]))));  
}  
}
```

Foryou.dart

```
import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

class ForYouPage extends StatelessWidget {

  const ForYouPage({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Column(children: [
      const MainAppBar(
        height: 150,
      ),
      Center(
        child: Text(
          'For You',
          style: Theme.of(context).textTheme.headline1,
        ),
      )
    ]);
  }
}
```

Homepage.dart

```
import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

class HomePage extends StatelessWidget {

  const HomePage({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    Future<List> getName() async {

      // Get the name of the profile that is selected from shared
      preferences

      final prefs = await SharedPreferences.getInstance();

      final String? firstname = prefs.getString('firstname');

      final String? lastname = prefs.getString('lastname');

      return [firstname, lastname];
    }

    return Column(children: [
      const MainAppBar(
        //Display main app bar at the top
      )
    ]);
  }
}
```

```

        height: 150,
    ) ,
Padding(
    padding:
        const EdgeInsets.only(left: 40, top: 40, right: 60, bottom:
80),
child: Column(
    mainAxisAlignment: MainAxisAlignment.start,
children: [
Text(
    // Display welcome back
    'Welcome back',
    style: Theme.of(context).textTheme.headline6,
),
FutureBuilder<List>(
    // run future to get the name from shared preferences
future: getName(),
builder: (context, snapshot) {
if (snapshot.hasData) {
    //If the data is received
    List? name = snapshot.data; //Assign received data to
name
    if (name![0] != null) {
        // If the data is not null

```

```
        return Text(  
  
            // Display firstname and lastname  
  
            ('${name[0]} ${name[1]}'),  
  
            style: Theme.of(context).textTheme.headline1,  
        );  
  
    } else {  
  
        // If the data is null or anything else  
  
        return Text(  
  
            // Display Profile Unknown  
  
            'Profile Unknown.',  
  
            style: Theme.of(context).textTheme.headline1,  
        );  
  
    }  
  
} else {  
  
    // While waiting to get data or there is an error  
    while getting the data, display loading profile  
  
    return Text(  
  
        'Loading Profile...',  
  
        style: Theme.of(context).textTheme.headline1,  
    );  
  
}  
  
,  
)
```

))
]) ;
}
}

Profiles.dart

```
import 'package:alleat/widgets/elements/profiles_buttons.dart';

import 'package:alleat/widgets/elements/profiles_selection.dart';

import 'package:flutter/material.dart';

class ProfilePage extends StatefulWidget {

  const ProfilePage({super.key});

  @override

  State<ProfilePage> createState() => _ProfilePageState();
}

class _ProfilePageState extends State<ProfilePage> {

  @override

  Widget build(BuildContext context) {

    return SafeArea(
```

```
body: CustomScrollView(slivers: [  
    SliverFillRemaining(  
        hasScrollBody: false,  
        child: Padding(  
            padding: const EdgeInsets.only(top: 50, bottom: 50),  
            child: Column(children: [  
                Padding(  
                    padding: const EdgeInsets.only(left: 40, right: 40),  
                    child: Row(  
                        //Display text with title Profiles  
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,  
                        children: [  
                            Text(  
                                "Profiles",  
                                style: Theme.of(context).textTheme.headline1,  
                            ),  
                        ],  
                    )),  
                const SizedBox(  
                    height: 50,  
                ),  
                const ProfileList(), //Display slidable row of profiles  
                const SizedBox(  

```

```
    height: 35,  
),  
  
const ProfileButton(  
  
    // Display buttons for favourites, orders, settings and  
    profile settings with an icon using the widget ProfileButton  
  
    icon: (Icons.favorite),  
  
    name: "Favourites",  
  
    action: null,  
,  
  
const ProfileButton(  
  
    icon: (Icons.local_mall),  
  
    name: "Orders",  
  
    action: null,  
,  
  
const ProfileButton(  
  
    icon: (Icons.settings),  
  
    name: "Settings",  
  
    action: null,  
,  
  
const ProfileButton(  
  
    icon: (Icons.person),  
  
    name: "Profile Settings",  
  
    action: null,
```

```
    ) ,  
    ] ) ) )  
] ) ) ) ;  
}  
}
```

Profilesetup_create.dart

```
import 'package:alleat/services/dataencryption.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/queryserver.dart';

import 'package:alleat/services/setselected.dart';

import 'package:alleat/widgets/elements/elements.dart';

import 'package:alleat/widgets/navbar.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:email_validator/email_validator.dart';

class AddProfileCreationPageName extends StatefulWidget {

  const AddProfileCreationPageName({Key? key}) : super(key: key);

  @override

  State<AddProfileCreationPageName> createState() =>

    AddProfileCreationPageNameState();
}
```

```
}
```

```
class _AddProfileCreationPageNameState
```

```
    extends State<AddProfileCreationPageName> {
```

```
    final _formKey = GlobalKey<FormState>();
```

```
    static TextEditingController firstnameText = TextEditingController();
```

```
    static TextEditingController lastnameText = TextEditingController();
```

```
    final data = [
```

```
        firstnameText.text = "",
```

```
        lastnameText.text = ""
```

```
    ]; //Store the data to be reset if back button pressed
```

```
    @override
```

```
    Widget build(BuildContext context) {
```

```
        return Scaffold(
```

```
            body: CustomScrollView(slivers: [
```

```
                SliverFillRemaining(
```

```
                    hasScrollBody: false,
```

```
                    child: Column(
```

```
                        mainAxisAlignment: MainAxisAlignment.start,
```

```
                        crossAxisAlignment: CrossAxisAlignment.start,
```

```
                        children: <Widget>[
```

```
                            Flexible(
```

```
//Create flexible widgets to be able to resize with
keyboard

flex: 2,
fit: FlexFit.loose,
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
const ScreenBackButton(), //Back button to go to
starting screen

Padding(
padding: const EdgeInsets.only(
top: 20, left: 20, right: 20, bottom: 5),
child: Align(
alignment: Alignment.center,
child: Text(
"Step 1 of 3",
style:
Theme.of(context).textTheme.headline6,
textAlign: TextAlign.center,
))),
Padding(
padding: const EdgeInsets.only(
bottom: 20, left: 20, right: 20),
child: Align(
```



```
        title: TextFormField(
```

```
            controller:
```

```
                firstnameText, //Form data
```

```
lastname collected and sent to database
```

```
                keyboardType: TextInputType.name,
```

```
                style:
```

```
Theme.of(context).textTheme.bodyText2,
```

```
                decoration: (InputDecoration(
```

```
                    hintText: "Forename",
```

```
                    contentPadding: Theme.of(context)
```

```
                        .inputDecorationTheme
```

```
                        .contentPadding,
```

```
                    border: Theme.of(context)
```

```
                        .inputDecorationTheme
```

```
                        .border,
```

```
                    focusedBorder: Theme.of(context)
```

```
                        .inputDecorationTheme
```

```
                        .focusedBorder,
```

```
                    enabledBorder: Theme.of(context)
```

```
                        .inputDecorationTheme
```

```
                        .enabledBorder,
```

```
                    floatingLabelBehavior:
```

```
                        FloatingLabelBehavior.never)),
```

```
        inputFormatters: [  
            //Only allows the input of letters  
            a-z and A-Z and @, .-  
            FilteringTextInputFormatter.allow(  
                RegExp('^[a-zA-Z0-9@,.^-]$'))  
        ],  
  
        validator: (firstname) {  
  
            //Required field and uses  
            emailvalidator package to verify it is an email to simplify the code  
  
            if (firstname == null ||  
                firstname.isEmpty) {  
  
                return "Required";  
  
            }  
  
            return null;  
        },  
    ),  
    const SizedBox(  
        height: 10,  
    ),  
    ListTile(  
        title: TextFormField(  
            controller:
```

```
        lastnameText, //Form data lastname  
collected and sent to database  
  
        keyboardType: TextInputType.name,  
        style:  
  
Theme.of(context).textTheme.bodyText2,  
  
        decoration: (InputDecoration(  
  
            hintText: "Surname",  
  
            contentPadding: Theme.of(context)  
  
.inputDecorationTheme  
  
.contentPadding,  
  
border: Theme.of(context)  
  
.inputDecorationTheme  
  
.border,  
  
focusedBorder: Theme.of(context)  
  
.inputDecorationTheme  
  
.focusedBorder,  
  
enabledBorder: Theme.of(context)  
  
.inputDecorationTheme  
  
.enabledBorder,  
  
floatingLabelBehavior:  
  
FloatingLabelBehavior.never)),  
  
inputFormatters: [
```

```

        //Only allows the input of letters
a-z and A-Z and @,.-

        FilteringTextInputFormatter.allow(
            RegExp('^[a-zA-Z0-9@,.^-]$'))

        ] ,


        validator: (lastname) {

            //Required field and uses
emailvalidator package to verify it is an email to simplify the code

            if (lastname == null ||
                lastname.isEmpty) {

                return "Required";
            }

            return null;
        } ,
    ) ) ,
const SizedBox(
    height: 50,
) ,
] ,
) )
] )) ,
Flexible(
    flex: 1,
    fit: FlexFit.loose,

```



```
        firstname:  
  
firstnameText  
        .text,  
        lastname:  
  
lastnameText.text,  
        )));  
    } else {  
        null;  
    }  
},  
child: const Text("Continue")))),  
const SizedBox(  
    height: 50,  
)  
],  
))  
]))  
));  
}  
}  
  
class AddProfileCreationPageEmail extends StatefulWidget {
```

```

const AddProfileCreationPageEmail(
    {Key? key,
     this.firstname,
     this.lastname}) //Get firstname and lastname from previous screen
    : super(key: key);

final dynamic firstname;
final dynamic lastname;

@Override
State<AddProfileCreationPageEmail> createState() =>
    _AddProfileCreationPageEmailState();
}

class _AddProfileCreationPageEmailState
extends State<AddProfileCreationPageEmail> {
    final _formKey = GlobalKey<FormState>();
    static TextEditingController emailText = TextEditingController();
    static TextEditingController confirmemailText = TextEditingController();
    dynamic data = [
        emailText.text = "",
        confirmemailText.text = ""
    ]; // Store email and confirm email to be reset if back button pressed
    @override

```

```
Widget build(BuildContext context) {  
  
  return Scaffold(  
  
    body: CustomScrollView(slivers: [  
  
      SliverFillRemaining(  
  
        hasScrollBody: false,  
  
        child: Column(  
  
          mainAxisAlignment: MainAxisAlignment.spaceBetween,  
  
          children: <Widget>[  
  
            Flexible(  
  
              flex: 2,  
  
              fit: FlexFit.loose,  
  
              child: Column(  
  
                mainAxisAlignment: MainAxisAlignment.start,  
  
                children: [  
  
                  ScreenBackButton(  
  
                    data:  
  
                      data), //If back button pressed, reset  
inputted data  
  
                  Padding(  
  
                    padding: const EdgeInsets.only(  
  
                      top: 20, left: 20, right: 20, bottom: 5),  
  
                    child: Align(  
                
```

```
        alignment: Alignment.center,  
  
        child: Text(  
  
            "Step 2 of 3",  
  
            style:  
Theme.of(context).textTheme.headline6,  
  
            textAlign: TextAlign.center,  
  
        )),  
  
        Padding(  
  
            padding: const EdgeInsets.only(  
  
                bottom: 20, left: 20, right: 20),  
  
            child: Align(  
  
                alignment: Alignment.center,  
  
                child: Text(  
  
                    "Profile Setup.",  
  
                    style:  
Theme.of(context).textTheme.headline1,  
  
                    textAlign: TextAlign.center,  
  
                )),  
  
                const SizedBox(  
  
                    height: 20,  
  
                )  
  
            ],  
  
        )),  
  
        Flexible(  

```

```
        flex: 2,  
  
        fit: FlexFit.loose,  
  
        child: Column(  
  
            mainAxisAlignment: MainAxisAlignment.start,  
  
            children: [  
  
                Form(  
  
                    key: _formKey,  
  
                    child: Column(  
  
                        children: [  
  
                            ListTile(  
  
                                title: TextFormField(  
  
                                    controller:  
  
                                    emailText, //Form data lastname  
collected and sent to database  
  
                                    keyboardType:  
TextInputType.emailAddress,  
  
                                    style:  
  
Theme.of(context).textTheme.bodyText2,  
  
                                    decoration: (InputDecoration(  
  
                                        hintText: "Email",  
  
                                        contentPadding: Theme.of(context)  
                                            .inputDecorationTheme  
  
                                            .contentPadding,
```

```

border: Theme.of(context)

    .inputDecorationTheme

    .border,
focusedBorder: Theme.of(context)

    .inputDecorationTheme

    .focusedBorder,
enabledBorder: Theme.of(context)

    .inputDecorationTheme

    .enabledBorder,
floatingLabelBehavior:

    FloatingLabelBehavior.never)),

inputFormatters: [
    //Only allows the input of letters
    a-z and A-Z and @,.-

    FilteringTextInputFormatter.allow(
        RegExp('[a-zA-Z0-9@,. -]'))
],
validator: (email) {

    //Required field and uses
    emailvalidator package to verify it is an email to simplify the code

    if (email == null || email.isEmpty)
    {
        return "Required";
    }
}

```

```
        if (EmailValidator.validate(email)
==

            false) {

                return "Please enter valid email";

            }

            return null;

        } ,

    ) ,

const SizedBox(
    height: 10,
),
ListTile(
    title: TextFormField(
        controller:
        confirmemailText, //Form data
lastname collected and sent to database

        keyboardType:
TextInputType.emailAddress,

        style:

Theme.of(context).textTheme.bodyText2,
decoration: (InputDecoration(
        hintText: "Confirm email",
        contentPadding: Theme.of(context)
```

```

        .inputDecorationTheme

        .contentPadding,

        border: Theme.of(context)

        .inputDecorationTheme

        .border,
        focusedBorder: Theme.of(context)

        .inputDecorationTheme

        .focusedBorder,
        enabledBorder: Theme.of(context)

        .inputDecorationTheme

        .enabledBorder,
        floatingLabelBehavior:

        FloatingLabelBehavior.never)),

        inputFormatters: [
        //Only allows the input of letters
        a-z and A-Z and @,.-

        FilteringTextInputFormatter.allow(
        RegExp('^[a-zA-Z0-9@,.^-]$'))

        ],
        validator: (confirmemail) {
        //Required field and uses
        emailvalidator package to verify it is an email to simplify the code
        if (confirmemail == null ||
        confirmemail.isEmpty) {

```



```
        bottom: 60,  
    ),  
  
    child: Center(  
  
        child: SizedBox(  
  
            width: double.infinity,  
  
            child: ElevatedButton(  
  
                onPressed: () {  
  
                    if (_formKey.currentState!  
                        .validate()) {  
  
                        //If fields have no errors  
  
                        Navigator.pop(context);  
  
                        Navigator.push(  
  
                            // On continue, export  
                            inputted fields to the final screen  
  
                            context,  
  
                            MaterialPageRoute(  
  
                                builder: (context) =>  
  
AddProfileCreationPagePassword(  
  
    email:  
emailText.text,  
  
    confirmemail:  
  
confirmemailText
```

```
        .text,  
        firstname:  
  
widget.firstname,  
        lastname:  
  
widget.lastname,  
    )));  
  
} else {  
  
    null;  
  
}  
  
,  
  
child: const Text("Continue"))))  
  
],  
  
))  
  
]))  
));  
}  
}  
  
  
  
class AddProfileCreationPagePassword extends StatefulWidget {  
  
const AddProfileCreationPagePassword(  
  
{Key? key, this.email, this.confirmemail, this.firstname,  
this.lastname})
```

```
    : super(key: key);

    final dynamic email;

    final dynamic confirmemail;

    final dynamic firstname;

    final dynamic lastname;

    @override

    State<AddProfileCreationPagePassword> createState() =>

        _AddProfileCreationPagePasswordState();

}

class _AddProfileCreationPagePasswordState

    extends State<AddProfileCreationPagePassword> {

    final _formKey = GlobalKey<FormState>();

    static TextEditingController passwordText = TextEditingController();

    static TextEditingController confirmpasswordText =
    TextEditingController();

    static dynamic encryptPassword;

    dynamic data = [

        passwordText.text = "",

        confirmpasswordText.text = ""

    ]; //If back button pressed, reset data

    bool _passwordVisible = false;
```

```
@override

Widget build(BuildContext context) {

  return Scaffold(
    body: CustomScrollView(slivers: [
      SliverFillRemaining(
        hasScrollBody: false,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: <Widget>[
            Flexible(
              flex: 2,
              fit: FlexFit.loose,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                  ScreenBackButton(data: data),
                  Padding(
                    padding: const EdgeInsets.only(
                      top: 20, left: 20, right: 20, bottom: 5),
                    child: Align(
                      alignment: Alignment.center,
                      child: Text(
                        title,
                        style: TextStyle(
                          color: Colors.white,
                          fontSize: 18,
                          fontWeight: FontWeight.w500,
                        ),
                      ),
                    ),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    ],
  );
}
```

```
        "Step 3 of 3",
        style:
Theme.of(context).textTheme.headline6,
        textAlign: TextAlign.center,
    )),
Padding(
    padding: const EdgeInsets.only(
        bottom: 20, left: 20, right: 20),
child: Align(
    alignment: Alignment.center,
child: Text(
    "Secure Password.",
    style:
Theme.of(context).textTheme.headline1,
    textAlign: TextAlign.center,
)),
const SizedBox(
    height: 20,
)
],
),
Flexible(
    flex: 2,
    fit: FlexFit.loose,
```

```
        child: Column (

            mainAxisAlignment: MainAxisAlignment.start,

            children: [
                Form(
                    key: _formKey,
                    child: Column (
                        children: [
                            ListTile(
                                title: TextFormField(
                                    controller:
                                        passwordText, //Form data lastname
                                    keyboardType:
                                        TextInputType.visiblePassword,
                                    obscureText: !_passwordVisible,
                                    style:
                                        Theme.of(context).textTheme.bodyText2,
                                    decoration: (InputDecoration(
                                        hintText: "Password",
                                        contentPadding: Theme.of(context)
                                            .inputDecorationTheme
                                            .contentPadding,
                                        border: Theme.of(context)
                                            .borderRadius
                                            .copyWith(
                                                topRight: Radius.circular(0),
                                                bottomLeft: Radius.circular(0),
                                                bottomRight: Radius.circular(10),
                                                topLeft: Radius.circular(10),
                                            )
                                    ))
                            )
                        ],
                    ),
                )
            ],
        ),
    ),

```

```
.inputDecorationTheme  
  
.border,  
  
focusedBorder: Theme.of(context)  
  
.inputDecorationTheme  
  
.focusedBorder,  
  
enabledBorder: Theme.of(context)  
  
.inputDecorationTheme  
  
.enabledBorder,  
  
floatingLabelBehavior:  
  
FloatingLabelBehavior.never,  
  
suffixIcon: IconButton(  
  
//Button to toggle password  
visibility  
  
icon: Icon(  
  
_passwordVisible  
  
? Icons.visibility_off  
  
: Icons.visibility,  
  
color: Theme.of(context)  
  
.primaryColor  
  
.withOpacity(0.5),  
  
) ,  
  
onPressed: () {  
  
setState(() {
```

```

        _passwordVisible =
            !_passwordVisible;
    } );
},
))),

autofillHints: const [
    AutocompleteHints.newPassword
],
inputFormatters: [
    //Only allows the input of letters
    a-z and A-Z and @, .-
    FilteringTextInputFormatter.allow(
        RegExp(r'[a-zA-Z0-9@$.&#!#?]+'))
],
validator: (password) {
    //Required field and uses
    emailvalidator package to verify it is an email to simplify the code
    if (password == null ||
        password.isEmpty) {
        return "Required";
    } else if (!password
        .contains(RegExp(r'[0-9]')) ||
        ||

}

```

```
!password

    .contains(RegExp(r'[a-z]')))

{

    return "Password must contain at
least 1 number and 1 letter";

} else if (password.length < 8) {

    return "Password must be a minimum
of 8 characters";

} else if (password.length > 99) {

    return "Password must be a maximum
of 99 characters";

}

return null;

} ,

) ) ,

const SizedBox(
height: 10,
) ,
ListTile(
title: TextFormField(
controller:
confirmPasswordText, //Form data
lastname collected and sent to database

keyboardType:
TextInputType.visiblePassword,
```

```
        style:  
  
Theme.of(context).textTheme.bodyText2,  
  
        decoration: (InputDecoration(  
  
            hintText: "Confirm password",  
  
            contentPadding: Theme.of(context)  
  
                .inputDecorationTheme  
  
                .contentPadding,  
  
            border: Theme.of(context)  
  
                .inputDecorationTheme  
  
                .border,  
  
            focusedBorder: Theme.of(context)  
  
                .inputDecorationTheme  
  
                .focusedBorder,  
  
            enabledBorder: Theme.of(context)  
  
                .inputDecorationTheme  
  
                .enabledBorder,  
  
            floatingLabelBehavior:  
  
                FloatingLabelBehavior.never,  
  
        )),  
  
        obscureText: true,  
  
        autofillHints: const [
```

```
        AutofillHints.newPassword  
    ] ,  
    autovalidateMode:  
  
AutovalidateMode.onUserInteraction,  
  
    inputFormatters: [  
  
        //Only allows the input of letters  
        a-z and A-Z and @, .-  
  
        FilteringTextInputFormatter.allow(  
  
            RegExp(' [a-zA-Z0-9@,.^-] ') )  
    ] ,  
  
    validator: (confirmPassword) {  
  
        //Required field and uses  
        emailvalidator package to verify it is an email to simplify the code  
  
        if (confirmPassword == null ||  
  
            confirmPassword.isEmpty) {  
  
            return "Required";  
        } else if (confirmPassword !=  
  
            passwordText.text) {  
  
            return "Passwords do not match";  
        }  
  
        return null;  
    } ,  
)  
,
```

```
        const SizedBox( height: 50, ) , ] , )) ] )) , Flexible( flex: 1, fit: FlexFit.loose, child: Column( children: [ Padding( padding: const EdgeInsets.only( left: 30, right: 30, bottom: 60, ), child: Center( child: SizedBox( width: double.infinity, child: ElevatedButton( onPressed: () { if (_formKey.currentState!
```

```

        .validate() ) {
            //If fields have no errors
            _createProfile();
        } else {
            null;
        }
    },
    child: const Text("Create
Profile")))))
],
))
])
);
}

Future<void> _createProfile() async {
    encryptPassword = await DataEncryption.encrypt(passwordText.text);
    var receivedServerData =
        await
    QueryServer.query("https://alleat.cpur.net/query/register.php", {
        //Send data to login.php on server with email and encrypted password
        "firstname": widget.firstname,
        "lastname": widget.lastname,
        "email": widget.email,
    });
}

```

```

"password": encryptPassword

});

if (recievedServerData["error"] == true) {

    //If error, display failed to create profile and reset password
    fields

    setState(() {

        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            content: Text(recievedServerData["message"] +
                " : Failed to create profile. Please try again")));
    });

    passwordText.text = "";
    confirmpasswordText.text = "";

} else {

    //If the email already exists, reset password fields and bring user
    back to the add user page

    if ((recievedServerData["message"])["exist"] == true) {

        passwordText.text = "";
        confirmpasswordText.text = "";

        setState(() {

            Navigator.pop(context);

            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Profile already exists')),

            );
        });

    };
}

```

```
    } else {

        try {

            List importedProfile =
(recievedServerData["message"])["profile"];

            await SQLiteLocalProfiles.createProfile(

                importedProfile[0],

                importedProfile[1],

                importedProfile[2],

                importedProfile[3],

                importedProfile[4]);
        }
    }

    bool trySelect = await SetSelected.selectProfile(
        importedProfile[0], //Try to select profile

        importedProfile[1],

        importedProfile[2],

        importedProfile[3]);

    if (trySelect == false) {
        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text("Failed to select
profile")));
        });
    }
}
```

```
//If is able to select profile, clear password fields and go  
to navigation page (defaults to homepage)  
  
    passwordText.text = "";  
  
    confirmPasswordText.text = "";  
  
    setState(() {  
  
        Navigator.pop(context);  
  
        Navigator.pop(context);  
  
        ScaffoldMessenger.of(context).showSnackBar(  
  
            const SnackBar(content: Text('Successfully created  
profile.')),  
  
        );  
  
        Navigator.push(context,  
  
            MaterialPageRoute(builder: (context) => const  
Navigation()),  
  
        );  
  
    });  
  
} catch (e) {  
  
    // If error, display error  
  
    setState(() {  
  
        ScaffoldMessenger.of(context)  
  
            .showSnackBar(SnackBar(content: Text("ERROR: $e")));  
  
    });  
  
}  
}
```

```
    }

}

}
```

Profilesetup_existing.dart

```
import 'package:alleat/screens/profilesetup/profilesetup_create.dart';

import 'package:alleat/screens/profilesetup/profilesetup_login.dart';

import 'package:flutter/material.dart';

class ProfileSetupExisting extends StatefulWidget {

  const ProfileSetupExisting({Key? key}) : super(key: key);

  @override
  State<StatefulWidget> createState() {
    return _ProfileSetupExisting();
  }
}

class _ProfileSetupExisting extends State<ProfileSetupExisting> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      //Create new screen
    );
  }
}
```

```
resizeToAvoidBottomInset: false, //Allow resize

body: Stack(children: [
    Image.asset(
        //Fullscreen image of food
        'lib/assets/images/screens/existingsetup/existingsetupscreenfood.jpg',
        fit: BoxFit.cover,
        height: double.infinity,
        width: double.infinity,
        alignment: Alignment.center,
    ),
    Column(
        children: [
            Padding(
                //Image of All Eat logo
                padding: const EdgeInsets.only(
                    top: 50, left: 30, right: 30, bottom: 200),
                child: Container(
                    width: 50,
                    height: 50,
                    decoration: BoxDecoration(
                        shape: BoxShape.circle,

```

```
        color: Theme.of(context).backgroundColor,  
    ),  
  
    child: IconButton(  
  
        color:  
Theme.of(context).textTheme.headline1?.color,  
  
        icon: const Icon(Icons.arrow_back),  
  
        onPressed: () => Navigator.of(context).pop()))))  
],  
) ,  
  
Column(  
  
crossAxisAlignment: CrossAxisAlignment.start,  
  
mainAxisAlignment: MainAxisAlignment  
.end, //Create content at the bottom of the screen  
  
children: [  
  
Padding(  
  
padding: const EdgeInsets.only(left: 10, right: 10),  
  
child: Container(  
  
//Bottom container with login and register actions  
  
decoration: BoxDecoration(  
  
color: Theme.of(context).backgroundColor,  
  
borderRadius: const BorderRadius.only(  
  
topLeft: Radius.circular(20),  
  
topRight: Radius.circular(
```

```
        20))), // Round the top corners of container
        width: double.infinity,
        padding: const EdgeInsets.only(left: 20, right: 20),
        child: Column(children: [
            const SizedBox(
                height: 40,
            ),
            Text("Add New Profile.",
                style: Theme.of(context).textTheme.headline2),
            Padding(
                padding: const EdgeInsets.only(
                    left: 50, right: 50, top: 10, bottom: 10),
                child: Text(
                    "Enhance your experience by using multiple
profiles",
                    style: Theme.of(context).textTheme.headline6,
                    textAlign: TextAlign.center,
                )),  

            const SizedBox(
                height: 20,
            ),
            Padding(
                //Button actions

```

```
padding: const EdgeInsets.only(left: 20, right: 20),  
  
child: Column(children: [  
  
    SizedBox(  
  
        width: double.infinity,  
  
        child: ElevatedButton(  
  
            style:  
  
Theme.of(context).elevatedButtonTheme.style,  
  
            onPressed: () => (Navigator.push( //go to  
profile creation page  
  
                context,  
  
                MaterialPageRoute(  
  
                    builder: (context) =>  
  
                    const  
AddProfileCreationPageName()))),  
  
            child: const Text("Create a profile"))),  
  
    SizedBox(  
  
        width: 250,  
  
        child: TextButton(  
  
            style:  
Theme.of(context).textButtonTheme.style,  
  
            onPressed: () => (Navigator.push( // go to  
profile login page  
  
                context,  
  
                MaterialPageRoute(  
))
```

```
        builder: (context) =>
          const
AddProfileLoginPage() ) ) ,  

  
          child: const Align(  

  
            alignment: Alignment.center,  

  
            child: Text(  

  
              "I already have a profile",  

  
              textAlign: TextAlign.center,  

  
            ) ) )  

  
        ] ) ,  

  
      ) ,  

  
      const SizedBox(  

  
        height: 40,  

  
      ) ,  

  
    ] ) ,  

  
  ) )  

  
],  

  
)  

  
] ) ,  

  
) ;  

  
}  

  
}
```

Profilesetup_login.dart

```
import 'package:alleat/services/dataencryption.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/queryserver.dart';

import 'package:alleat/services/setselected.dart';

import 'package:alleat/widgets/elements/elements.dart';

import 'package:alleat/widgets/navbar.dart';

import 'package:flutter/material.dart';

import 'package:email_validator/email_validator.dart';

import 'package:flutter/services.dart';

import 'dart:async';

class AddProfileLoginPage extends StatefulWidget {

  const AddProfileLoginPage({Key? key}) : super(key: key);

  @override

  State<AddProfileLoginPage> createState() => _AddProfileLoginPageState();
}

class _AddProfileLoginPageState extends State<AddProfileLoginPage> {

  final _formKey = GlobalKey<FormState>();

  static TextEditingController email =
```

```

    TextEditingController() ; //Create text controllers to allow for
dynamic variable for form fields

    static TextEditingController password = TextEditingController();

    static dynamic encryptPassword;

    late dynamic profileInfoImport;

Future<void> _loginUser() async {

    encryptPassword =

        await DataEncryption.encrypt(password.text); //Encrypt password

    var recieivedServerData =

        await QueryServer.query("https://alleat.cpur.net/query/login.php",
    {

        //Send data to login.php on server with email and encrypted
password. It checks if the credentials are correct and returns exists if
it is valid

        "email": email.text,
        "password": encryptPassword,
    });
}

if (recieivedServerData["error"] == true) {

    //If there is an error, clear password and display error from server
    setState(() {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
            content: Text(recieivedServerData["message"] +
                " : Failed to login. Please try again")));
    });
}

```

```
}) ;

password.text = "";

} else {

    if (recievedServerData["message"]["exists"] == true) {

        //If ther profile is correct and exists

        List importedProfile = recievedServerData["message"]["profile"];

        bool trySelect = await SetSelected.selectProfile(
            importedProfile[0], //Try select profile
            importedProfile[1],
            importedProfile[2],
            importedProfile[3]);

        if (trySelect == false) {

            // If the profile fails to select display error

            setState(() {
                ScaffoldMessenger.of(context).showSnackBar(
                    const SnackBar(content: Text("Failed to select
profile")));
            });
        } else {

            // If succeeds to select profile

            try {
                await SQLiteLocalProfiles.createProfile(
                    //Create profile in database

```

```
        importedProfile[0],  
  
        importedProfile[1],  
  
        importedProfile[2],  
  
        importedProfile[3],  
  
        importedProfile[4]));  
  
    email.text = ""; //Clear email and password  
  
    password.text = "";  
  
    setState(() {  
  
        ScaffoldMessenger.of(context).showSnackBar(  
  
            const SnackBar(content: Text('Successfully logged in.')),  
  
        );  
  
        Navigator.push(  
  
            context, //Go to main area  
  
            MaterialPageRoute(builder: (context) => const  
Navigation()),  
  
        );  
  
    } catch (e) {  
  
        //If there is an error, display that there was an error  
  
        setState(() {  
  
            ScaffoldMessenger.of(context).showSnackBar(  
  
                const SnackBar(  
  
                    content: Text('Failed to save profile on device.')),  
  
            );  
  
        });  
    }  
}
```

```

        } ) ;

    }

}

} else {

    // If the password or email is incorrect, display incorrect email
    // or password

    setState( () {

        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text("Incorrect email or
password")));
    });
}
}

}

Future<void> _checkDuplicate() async {

    List profileList = await SQLiteLocalProfiles.getProfiles();

    late bool alreadyLogged = false;

    for (int i = 0; i < (profileList.length - 1); i++) {

        if (profileList[i]["email"].contains(email.text)) {

            alreadyLogged = true;
        }
    }

    if (alreadyLogged == false) {

```

```
        _loginUser();

    } else {
        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text("Profile is already logged
in.")));
        });
    }
}

@Override
Widget build(BuildContext context) {
    var brightness = MediaQuery.of(context).platformBrightness;
    bool darkModeOn = brightness == Brightness.dark;
    return Scaffold(
        body: CustomScrollView(slivers: [
            SliverFillRemaining(
                hasScrollBody: false,
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.start,
                    mainAxisSize: MainAxisSize.max,
                    children: <Widget>[
                        Flex(direction: Axis.vertical, children: [

```

```
Column (  
  mainAxisAlignment: MainAxisAlignment.start,  
  children: [  
    const ScreenBackButton(),  
    Padding (  
      padding: const EdgeInsets.only(  
        left: 50, right: 50, top: 30),  
      child: Image.asset((darkModeOn)  
        ?  
          'lib/assets/images/screens/profilesetup/login-illustration-dark.png'  
        :  
          'lib/assets/images/screens/profilesetup/login-illustration-light.png')),  
    const SizedBox (  
      height: 20,  
    ),  
    Padding (  
      padding: const EdgeInsets.all(20),  
      child: Align (  
        alignment: Alignment.center,  
        child: Text("Welcome back.",  
          style: Theme.of(context)  
            .textTheme  
            .headline1)),  
    const SizedBox (
```



```

border: Theme.of(context)

    .inputDecorationTheme

    .border,
focusedBorder: Theme.of(context)

    .inputDecorationTheme

    .focusedBorder,
enabledBorder: Theme.of(context)

    .inputDecorationTheme

    .enabledBorder,
floatingLabelBehavior:

    FloatingLabelBehavior.never)),

inputFormatters: [
    //Only allows the input of letters
    a-z and A-Z and @,.-

    FilteringTextInputFormatter.allow(
        RegExp('[a-zA-Z0-9@,. -]'))
],
validator: (email) {

    //Required field and uses
    emailvalidator package to verify it is an email to simplify the code

    if (email == null || email.isEmpty)
{
    return "Required";
}

```

```
        if (EmailValidator.validate(email)
==

                false) {

            return "Please enter valid email";

        }

        return null;

    } ,

) ,

const SizedBox(
    height: 10,
) ,
ListTile(
    title: TextFormField(
        keyboardType:
TextInputType.visiblePassword,
        style:
Theme.of(context).textTheme.bodyText2,
        decoration: (InputDecoration(
            hintText: "Password",
            contentPadding:
Theme.of(context)
            .inputDecorationTheme
            .contentPadding,
        )
)
```

```
border: Theme.of(context)

    .inputDecorationTheme

    .border,
    focusedBorder: Theme.of(context)

        .inputDecorationTheme

        .focusedBorder,
    enabledBorder: Theme.of(context)

        .inputDecorationTheme

        .enabledBorder,
    floatingLabelBehavior:

FloatingLabelBehavior.never)),

    inputFormatters: [
        //Password cannot use " or ' in
order to prevent SQL injection

FilteringTextInputFormatter.allow(RegExp(
    '[a-zA-Z0-9!@#%^&*(),.?:{ }|<>]')
    ],
    obscureText: true, //Password not
visible

    controller:

        password, //Password copied and
checked by confirm password
```

```
validator: (password) {  
  
    //Must be a minimum of 8  
    characters and contain a letter and number to make sure there is variety  
    and make it harder to guess. Must be under 99 characters so that it  
    reduces processing time on the system  
  
    if (password == null ||  
  
        password.isEmpty ||  
  
        password.length < 8 ||  
  
        password.length > 99 ||  
  
        !password  
  
.contains(RegExp(r'[0-  
9]')) ||  
  
        !password  
  
.contains(RegExp(r'[a-  
z]')))  
    {  
  
        return "Required";  
  
    }  
  
    return null;  
  
},  
  
) ,  
  
) ,  
  
const SizedBox(height: 50), //Gap  
  
],  
  
) ,  
  
) ,
```

```
) ,  
)  
]) ,  
Column(  
children: [  
Padding(  
padding: const EdgeInsets.only(  
left: 30,  
right: 30,  
bottom: 60,  
),  
child: Center(  
child: SizedBox(  
width: double.infinity,  
child: ElevatedButton(  
//submit button  
style:  
Theme.of(context).elevatedButtonTheme.style,  
onPressed: () {  
if (_formKey.currentState!.validate()) {  
_checkDuplicate();  
}  
}
```

```

        } ,
        child: const Text('Login to Profile'),
    ) ,
) ) ,
] ,
)
] )
] )) ;
}

}

```

Welcomescreen.dart

```

import 'package:alleat/screens/profilesetup/profilesetup_create.dart';

import 'package:alleat/screens/profilesetup/profilesetup_login.dart';

import 'package:flutter/material.dart';

class ProfileSetupWelcome extends StatefulWidget {

  const ProfileSetupWelcome({Key? key}) : super(key: key);

  @override
  State<StatefulWidget> createState() {
    return _ProfileSetupWelcome();
  }
}

class _ProfileSetupWelcome extends State<ProfileSetupWelcome> {
  ...
}

```

```
        }

    }

}

class _ProfileSetupWelcome extends State<ProfileSetupWelcome> {

    @override

    Widget build(BuildContext context) {

        return Scaffold(
            //Create new screen

            resizeToAvoidBottomInset: false, //Allow resize

            body: Stack(children: [
                Image.asset(
                    //Fullscreen image of food
                    'lib/assets/images/screens/welcomeScreen/welcomeScreenFood.jpg',
                    fit: BoxFit.cover,
                    height: double.infinity,
                    width: double.infinity,
                    alignment: Alignment.center,
                ),
                Column(
                    children: [
                        Padding(
                            //Image of All Eat logo

```

```
padding: const EdgeInsets.only(
    top: 70, left: 40, right: 40, bottom: 200),
    child: Image.asset(
        'lib/assets/images/logo/logodark.png',
        width: 58,
        height: 58,
    )));
],
),
Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    mainAxisAlignment: MainAxisAlignment
        .end, //Create content at the bottom of the screen
    children: [
    Padding(
        padding: const EdgeInsets.only(left: 10, right: 10),
        child: Container(
            //Bottom container with login and register actions
            decoration: BoxDecoration(
                color: Theme.of(context).backgroundColor,
                borderRadius: const BorderRadius.only(
                    topLeft: Radius.circular(20),
                    topRight: Radius.circular(20),
                ),
            ),
        ),
    ),
],
```

```
        20))), // Round the top corners of container
        width: double.infinity,
        padding: const EdgeInsets.only(left: 20, right: 20),
        child: Column(children: [
            const SizedBox(
                height: 40,
            ),
            Text("Let's Get Started.",
                style: Theme.of(context).textTheme.headline2),
            Padding(
                padding: const EdgeInsets.only(
                    left: 50, right: 50, top: 10, bottom: 10),
                child: Text(
                    "A food delivery app with you in mind",
                    style: Theme.of(context).textTheme.headline6,
                    textAlign: TextAlign.center,
                )),
            const SizedBox(
                height: 20,
            ),
            Padding(
                //Button actions
                padding: const EdgeInsets.only(left: 20, right: 20),
```

```
        child: Column(children: [  
  
            SizedBox(  
  
                width: double.infinity,  
  
                child: ElevatedButton(  
  
                    style:  
  
Theme.of(context).elevatedButtonTheme.style,  
  
                    onPressed: () => (Navigator.push( //Button  
to go to profile creation page  
  
                        context,  
  
                        MaterialPageRoute(  
  
                            builder: (context) =>  
  
                                const  
AddProfileCreationPageName()))),  
  
                    child: const Text("Create a profile"))),  
  
            SizedBox(  
  
                width: 250,  
  
                child: TextButton(  
  
                    style:  
Theme.of(context).textButtonTheme.style,  
  
                    onPressed: () => (Navigator.push( //Text  
button to go to login page  
  
                        context,  
  
                        MaterialPageRoute(  
  
                            builder: (context) =>
```

```
        const AddProfileLoginPage() ) ,  
            child: const Align(  
                alignment: Alignment.center,  
                child: Text(  
                    "I already have a profile",  
                    textAlign: TextAlign.center,  
                ) ) )  
        ] ) ,  
    ) ,  
    const SizedBox(  
        height: 40,  
    ) ,  
] ) ,  
) )  
] ,  
)  
] ) ,  
) ;  
}  
}
```

Restaurant_customise.dart

```
import 'package:flutter/material.dart';

class RestaurantItemCustomisePage extends StatefulWidget {

    final String itemid;

    final String foodcategory;

    final String subfoodcategory;

    final String itemname;

    final String description;

    final String price;

    final String itemimage;

    final String reslogo;

    const RestaurantItemCustomisePage(
        {Key?

            key, //Get the items from the restaurant main page when an item
is clicked

            required this.reslogo,

            required this.itemid,

            required this.foodcategory,

            required this.subfoodcategory,

            required this.itemname,

            required this.description,

            required this.price,
```

```
        required this.itemimage))

        : super(key: key);

    }

    @override

    State<RestaurantItemCustomisePage> createState() =>

        _RestaurantItemCustomisePageState();

}

class _RestaurantItemCustomisePageState

    extends State<RestaurantItemCustomisePage> {

    @override

    Widget build(BuildContext context) {

        return Scaffold(

            body: SingleChildScrollView(


                child: Column(


                    children: [


                        Stack(children: [


                            //Display item image at bottom of stack


                            Container(


                                width: MediaQuery.of(context).size.width,


                                height: 240,


                                decoration: BoxDecoration(


                                    image: DecorationImage(
```

```
        fit: BoxFit.fitWidth,  
  
        image: NetworkImage(widget.itemimage),  
  
    ),  
),  
,  
  
Align(  
  
    //Display rounded container at bottom of image to represent  
    overhanging image  
  
    alignment: Alignment.bottomCenter,  
  
    child: Container(  
  
        margin: const EdgeInsets.only(top: 215),  
  
        width: double.infinity,  
  
        height: 30,  
  
        decoration: BoxDecoration(  
  
            color: Theme.of(context).backgroundColor,  
  
            borderRadius:  
  
                const BorderRadius.vertical(top:  
Radius.circular(20))),  
  
    ),  
  
Align(  
  
    // Display background color as outline of restaurant logo,  
    overlapping the image background  
  
    alignment: Alignment.bottomCenter,  
  
    child: Container(  

```

```
        margin: const EdgeInsets.only(top: 180),  
  
        width: 80,  
  
        height: 80,  
  
        decoration: BoxDecoration(  
  
            shape: BoxShape.circle,  
  
            color: Theme.of(context).backgroundColor,  
        )),  
  
    Align(  
  
        // Display circle restaurant logo  
  
        alignment: Alignment.bottomCenter,  
  
        child: Container(  
  
            margin: const EdgeInsets.only(top: 185),  
  
            width: 70,  
  
            height: 70,  
  
            decoration: BoxDecoration(  
  
                image: DecorationImage(  
  
                    fit: BoxFit.cover,  
  
                    image: NetworkImage(widget.reslogo.toString()),  
                ),  
  
                shape: BoxShape.circle,  
  
                color: Theme.of(context).colorScheme.onSurface,  
            )),  
  
    SafeArea(  
       
```

```
// Display back button in a circle

child: InkWell(
    onTap: () => Navigator.of(context).pop(),
    child: Container(
        margin: const EdgeInsets.only(top: 20, left: 20),
        width: 50,
        height: 50,
        decoration: BoxDecoration(
            shape: BoxShape.circle,
            color: Theme.of(context).colorScheme.onSurface,
        ),
        child: Icon(
            Icons.chevron_left,
            color: Theme.of(context).colorScheme.onBackground,
            size: 35,
        ),
    )));
]

Padding(
    // Item name
    padding:
        const EdgeInsets.only(top: 40, left: 30, right: 30,
bottom: 10),
```

```

        child: Text(
            widget.itemname,
            textAlign: TextAlign.center,
            style: Theme.of(context).textTheme.headline2,
        )),
    ),
Row(mainAxisAlignment: MainAxisAlignment.center, children: [
    //Display restuarant category(ies)
    Text(widget.foodcategory, //Must have food category
        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.headline6!.copyWith(
            color: Theme.of(context).textTheme.headline1!.color)),
    LayoutBuilder(builder: (context, constraints) {
        if (widget.subfoodcategory != "") {
            //If there is a subcategory, display it with a dot next to it
            //If there is a sub food category, show it
            return Text(" · ${widget.subfoodcategory}",
                textAlign: TextAlign.center,
                style: Theme.of(context).textTheme.headline6!.copyWith(
                    color:
                    Theme.of(context).textTheme.headline1!.color));
        } else {
            // If there is no sub-category, dont display anything
            //If there isnt a sub-food category, dont show it
        }
    })
]);

```

```

        return const Text("");

    }

} )

] ) , Padding( //Display item description padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 20) , child: Text( widget.description, textAlign: TextAlign.center, style: Theme.of(context).textTheme.bodyText1!.copyWith( color: Theme.of(context).textTheme.headline6!.color, fontWeight: FontWeight.w400), ), ), ) );
) );
}
}
}

```

Restaurant_main.dart

```
import 'package:alleat/screens/restaurant/restaurant_customise.dart';
```

```
import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

import 'package:shared_preferences/shared_preferences.dart';

class RestaurantMain extends StatefulWidget {

    final String resid;

    final String resname;

    final String reslogo;

    final String resbanner;

    const RestaurantMain(
        //Get restaurant info from the restuarant list widget (passed from
        the container)

        {Key? key,
        required this.resid,
        required this.resname,
        required this.reslogo,
        required this.resbanner})

        : super(key: key);

    @override

    State<RestaurantMain> createState() => _RestaurantMainState();
}
```

```
}

class _RestaurantMainState extends State<RestaurantMain> {

  Future<List> getMenuCategories() async {

    String phpurl =
      "https://alleat.cpur.net/query/restaurantmenucategories.php";
    //Get the list of menu categories associated with the restaurant id

    try {

      var res = await http.post(Uri.parse(phpurl), body: {
        "restaurantid": widget.resid,
      });

      if (res.statusCode == 200) {

        //If successfully sent

        var data = convert.json.decode(res.body); //Decode to array

        if (data["error"]) {

          //If failed to query

          List error = [
            {
              "error": "true",
              "restaurantcategories": "[]"
            } //Return no menu categories
          ];
        }
      }
    }
  }

  return error;
}
```

```

    } else {

        List listdata = [data];

        return listdata;

    }

} else {

    List error = [

        {"error": "true", "restaurantcategories": "[]"}

    ];

    return error;

}

} catch (e) {

    List<Map<String, String>> error = [

        {"error": "true", "restaurantcategories": "[]"}

    ];

    return error;

}

}

Future<String> favouriteRestaurant(restaurantID, action) async {

    String phpurl =

        "https://alleat.cpur.net/query/favouriterestaurant.php"; //Action
of favourite/unfavourite restaurant by id (Unused in this version)

    final prefs = await SharedPreferences.getInstance();
}

```

```
final String? email = prefs.getString('email');

try {
    var res = await http.post(Uri.parse(phiurl), body: {
        "action": action.toString(),
        "profileemail": email.toString(),
        "restaurantid": restaurantID,
    });
    if (res.statusCode == 200) {
        var data = convert.json.decode(res.body); //Decode to array
        if (data["error"]) {
            return "failed";
        } else {
            getFavourites();
            return "success";
        }
    } else {
        return "failed";
    }
} catch (e) {
    return "failed";
}
}
```

```
Future<List> getFavourites() async {

    String phpurl =
        "https://alleat.cpur.net/query/favouriterestaurantlist.php"; //Get
favourite restaurant list associated with email of profile (unused in this
version)

    final prefs = await SharedPreferences.getInstance();

    final String? email = prefs.getString('email');

    try {

        var res = await http.post(Uri.parse(phpurl), body: {
            "profileemail": email.toString(),
        });

        if (res.statusCode == 200) {

            var data = convert.json.decode(res.body); //Decode to array

            if (data["error"]) {

                List error = [
                    {"error": "true", "favouriterestaurants": "[]"}
                ];

                return error;
            } else {

                List listdata = [data];

                return listdata;
            }
        } else {
    
```

```

List<error> = [
    {"error": "true", "favouriterestaurants": "[]"}
];
return error;
}

} catch (e) {
    List<Map<String, String>> error = [
        {"error": "true", "favouriterestaurants": "[]"}
    ];
    return error;
}
}

Future<List> getMenuItems(categoryid) async {
    String phpurl =
        "https://alleat.cpur.net/query/restaurantmenuitems.php"; //Get
    list of menu items, with the inforamtion associated with it. Grabs using
    the category id it falls under

    try {
        var res = await http.post(Uri.parse(phpurl), body: {
            "categoryid": categoryid,
        });
        if (res.statusCode == 200) {
            //If fails to send data

```

```
var data = converty.json.decode(res.body); //Decode to array

if (data["error"]) {

    //If fails the query

    List error = [
        {"error": "true", "menuitems": "[]"} //Send nothing
    ];

    return error;
}

} else {

    List listdata = [
        data
    ]; //Send back the list of items associated with the category

    return listdata;
}

} else {

    List error = [
        {"error": "true", "menuitems": "[]"}
    ];

    return error;
}

} catch (e) {

    List error = [
        {"error": "true", "menuitems": "[]"}
    ];
}
```

```
        return error;
    }

}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: SingleChildScrollView(
            //Make page scrollable
            child:
                Column(crossAxisAlignment: CrossAxisAlignment.start,
            children: [
                Stack(children: [ //Display stack, with the restaurant banner at the
bottom
                    Container(
                        width: MediaQuery.of(context).size.width,
                        height: 240,
                        decoration: BoxDecoration(
                            image: DecorationImage(
                                fit: BoxFit.fitWidth,
                                image: NetworkImage(widget.resbanner),
                            ),
                        ),
                    ),
                ],
            ),
        ),
    );
}
```

```
    Align( // Display container at the bottom of the image with
rounded edges to represent image overhang

        alignment: Alignment.bottomCenter,
        child: Container(
            margin: const EdgeInsets.only(top: 215),
            width: double.infinity,
            height: 30,
            decoration: BoxDecoration(
                color: Theme.of(context).backgroundColor,
                borderRadius:
                    const BorderRadius.vertical(top:
Radius.circular(20))),
        ),
        Align( //Circle outline to represent restaurant logo outline
            alignment: Alignment.bottomCenter,
            child: Container(
                margin: const EdgeInsets.only(top: 180),
                width: 80,
                height: 80,
                decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color: Theme.of(context).backgroundColor,
                )));
        Align( // Circle restaurant logo
```

```
        alignment: Alignment.bottomCenter,  
  
        child: Container(  
  
            margin: const EdgeInsets.only(top: 185),  
  
            width: 70,  
  
            height: 70,  
  
            decoration: BoxDecoration(  
  
                image: DecorationImage(  
  
                    fit: BoxFit.cover,  
  
                    image: NetworkImage(widget.reslogo.toString()),  
  
                ),  
  
                shape: BoxShape.circle,  
  
                color: Theme.of(context).colorScheme.onSurface,  
            )),  
  
SafeArea( // Create back button as circle  
  
        child: InkWell(  
  
            onTap: () => Navigator.of(context).pop(),  
  
            child: Container(  
  
                margin: const EdgeInsets.only(top: 20, left: 20),  
  
                width: 50,  
  
                height: 50,  
  
                decoration: BoxDecoration(  
  
                    shape: BoxShape.circle,  
  
                    color: Theme.of(context).colorScheme.onSurface,  
                ),  
            ),  
        ),  
    ),  
);
```

```
        ) ,  
  
        child: Icon(  
  
            Icons.chevron_left,  
  
            color: Theme.of(context).colorScheme.onBackground,  
  
            size: 35,  
  
        ) ,  
  
    )) ,  
  
Align( // Info icon button  
  
    alignment: Alignment.topRight,  
  
    child: SafeArea(  
  
        child: InkWell(  
  
            child: Container(  
  
                margin: const EdgeInsets.only(top: 20, right: 80) ,  
  
                width: 45,  
  
                height: 45,  
  
                decoration: BoxDecoration(  
  
                    shape: BoxShape.circle,  
  
                    color:  
Theme.of(context).colorScheme.onSurface.withOpacity(0.8),  
  
                ) ,  
  
                child: Icon(  
  
                    Icons.info_outline,  
  
                    color: Theme.of(context).colorScheme.onBackground,  
            ) ,  
        ) ,  
    ) ,  
);
```

```
        size: 30,  
    ),  
    ))),  
  
FutureBuilder<List>(  
    //Get favourites list from future  
    future: getFavourites(),  
    builder: ((context, snapshot) {  
  
        if (!snapshot.hasData) {  
  
            //While no data received, show loading bar  
  
            return SafeArea( // If no data has been received, display  
loading icon instead of favourites icon  
  
                child: Align(  
  
                    alignment: Alignment.topRight,  
  
                    child: Container(  
  
                        margin: const EdgeInsets.only(top: 20, right:  
20),  
  
                        width: 45,  
  
                        height: 45,  
  
                        decoration: BoxDecoration(  
  
                            shape: BoxShape.circle,  
  
                            color: Theme.of(context)  
  
.colorScheme  
.onSurface  
.withOpacity(0.8),
```

```
        ),

        child: Icon(
            Icons.cached,
            color:
            Theme.of(context).colorScheme.onBackground,
        ))));

    } else if (snapshot.hasError) { // If there is an error,
display error icon instead of favourites icon

    return SafeArea(
        child: Align(
            alignment: Alignment.topRight,
            child: Container(
                margin: const EdgeInsets.only(top: 20, right:
20),
                width: 45,
                height: 45,
                decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color: Theme.of(context)
                        .colorScheme
                        .onSurface
                        .withOpacity(0.8),
                ),
                child: Icon(

```

```
        Icons.error,  
  
        color: Theme.of(context).colorScheme.error,  
  
    ))));  
  
} else {  
  
    //If data received  
  
    List restaurantFavourites = snapshot.data ?? []; //Set  
favourites to restaurantfavourites  
  
    if (restaurantFavourites[0]["error"] == true) {  
  
        return SafeArea(  
  
            child: Align(  
  
                alignment: Alignment.topRight,  
  
                child: Container(  
  
                    margin: const EdgeInsets.only(top: 20,  
right: 20),  
  
                    width: 45,  
  
                    height: 45,  
  
                    decoration: BoxDecoration(  
  
                        shape: BoxShape.circle,  
  
                        color: Theme.of(context)  
  
.colorScheme  
  
.onSurface  
  
.withOpacity(0.8),  
  
        ),  
  
        child: Icon(  

```

```
        Icons.error,  
  
        color:  
Theme.of(context).colorScheme.error,  
))));  
  
} else {  
  
    return SafeArea(  
  
        child: Align(  
  
            alignment: Alignment.topRight,  
  
            child: InkWell(  
  
                onTap: () {  
  
                    if  
(restaurantFavourites[0]["restaurantids"]  
  
.contains(widget.resid)) {  
  
favouriteRestaurant(  
  
                widget.resid.toString(),  
"unfavourite");  
  
                setState(() {  
  
                    getFavourites();  
  
                });  
  
} else {  
  
favouriteRestaurant(  
  
                widget.resid.toString(),  
"favourite");  
  
                setState(() {  

```

```
        getFavourites() ;  
  
    } ) ;  
  
}  
  
},  
  
child: Container(  
  
margin:  
  
    const EdgeInsets.only(top: 20,  
right: 20),  
  
width: 45,  
  
height: 45,  
  
decoration: BoxDecoration(  
  
shape: BoxShape.circle,  
  
color: Theme.of(context)  
  
.colorScheme  
  
.onSurface  
  
.withOpacity(0.8),  
  
) ,  
  
child: restaurantFavourites[0]  
  
["restaurantids"]  
  
.contains(widget.resid)  
  
? Icon(Icons.favorite,  
  
size: 30, // ? = favourited  
  
color: Theme.of(context)
```

```

        .colorScheme
        .secondary)

        : Icon(
            // : = unfavourited
            Icons.favorite_border_outlined,
            size: 30,
            color: Theme.of(context)
            .colorScheme
            .onBackground))));

    }

}

} )

] ) , Padding(
    padding: const EdgeInsets.symmetric(vertical: 20, horizontal:
30),
    child: Column(children: [
        Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
                Flexible// Restaurant name
                child: Container(
                    padding: const EdgeInsets.only(right: 13.0),

```

```
        child: Text(  
  
            widget.resname,  
  
            overflow: TextOverflow.ellipsis,  
  
            style: Theme.of(context).textTheme.headline3,  
        )),  
  
    Row( // Star rating  
  
        children: [  
  
            Icon(  
  
                Icons.star,  
  
                size: 20,  
  
                color: Theme.of(context).primaryColor,  
            ),  
  
            const SizedBox(  
  
                width: 5,  
            ),  
  
            Text(  
  
                "N/A",  
  
                style: Theme.of(context).textTheme.bodyText1,  
            )  
  
        ],  
  
    ),  
  
),
```

```
const SizedBox(height: 10),  
  
Row( //Restaurant basic info  
  
children: [  
  
Text("N/A miles away",  
  
style: Theme.of(context).textTheme.bodyText1),  
  
const SizedBox(  
  
width: 5,  
  
),  
  
Text(  
  
".",  
  
style: Theme.of(context).textTheme.bodyText1,  
  
),  
  
const SizedBox(  
  
width: 5,  
  
),  
  
Text("£N/A Delivery",  
  
style: Theme.of(context).textTheme.bodyText1),  
  
],  
) ,  
]),  
),  
FutureBuilder<List>(  
  
//Checks for updates in the restaurant menu category  
  
future: getMenuCategories(),
```

```
builder: ((context, snapshot) {

    if (!snapshot.hasData) {

        //If no data has been received, show loading bar

        return LinearProgressIndicator(
            color: Theme.of(context).primaryColor,
            backgroundColor: const Color.fromARGB(0, 235, 224,
255));

    }

    if (snapshot.hasError) {

        //If there is an error, grabbing data, show error

        return const Text("An Error occurred. Please try again");

    } else {

        //If the data has been received

        List restaurantcategories = snapshot.data ??

        [];

        //Categories data associated with
        restaurantcategories

        if (restaurantcategories[0]["error"] == true) {

            return const Text("Error getting restaurant categories");

        } else {

            return ListView.builder(
                //For each category

                physics:

                const NeverScrollableScrollPhysics(), //Disable
                scrolling. Scroll with whole page
            );
        }
    }
});
```

```

        scrollDirection: Axis.vertical,
        shrinkWrap: true,
        itemCount:
      restaurantcategories[0]["restaurantcategories"]
        .length, //For each restaurant

        itemBuilder: (context, index) {
          return LayoutBuilder(builder: (context, constraints)
{
  if
  (restaurantcategories[0]["restaurantcategories"]
    .isEmpty) {
    //If there is no categories, display menu
    unavailable
    return const Text("Menu unavailable");
  } else {
    //If there are categories
    return Column(children: [
      //Add text of category name
      Padding(
        padding: const EdgeInsets.only(
          left: 30, top: 50, right: 10, bottom:
          10),
      child: Text(
        restaurantcategories[0]
        ["restaurantcategories"][index][0],

```

```

        style:
Theme.of(context).textTheme.headline3,
),

FutureBuilder<List>(
    //For each category, use a future to get
the list of items

    future:
getMenuItems(restaurantcategories[0]

    ["restaurantcategories"][index][1]),

    builder: (context, snapshot) {

        if (!snapshot.hasData) {

            //While no data received, show loading
bar

        return LinearProgressIndicator(
            color:

Theme.of(context).primaryColor,
            backgroundColor: const
Color.fromARGB(
                0, 235, 224, 255));

        }

        if (snapshot.hasError) {

            //If there is an error, show failed to
get items

            return const Text("Failed to get
items");

        } else {

```

```

    //List of items for category

    List restaurantitems = snapshot.data
???

    [] ; //Get data from Future

    if (restaurantitems[0] ["menuitems"]

        .isEmpty) { //If there are no
items under a category, display box with title

        return Container(
            width: double.infinity,
            padding: const
EdgeInsets.symmetric(
                vertical: 30, horizontal: 20),
            decoration: BoxDecoration(
                borderRadius:
                    const BorderRadius.all(
                        Radius.circular(10)),
                color: Theme.of(context)
                    .colorScheme
                    .onSurface),
            margin: const
EdgeInsets.symmetric(
                vertical: 5, horizontal: 30),
            child: Text(

```

```
        "Oh no! There are no items found  
under this category.",  
  
        textAlign: TextAlign.center,  
  
        style: Theme.of(context)  
  
            .textTheme  
  
            .headline6,  
  
) ,  
  
) ;  
  
} else { //If there are items under a  
category  
  
    return ListView.builder(  
  
        physics:  
  
        const  
NeverScrollableScrollPhysics(),  
  
        scrollDirection: Axis.vertical,  
  
        shrinkWrap: true,  
  
        itemCount: (restaurantitems[  
  
0] //For each item,  
create a container  
  
["menuitems"]  
  
.length), //For each  
restaurant  
  
        itemBuilder: (context, index) {  
  
            return LayoutBuilder(builder:  
  
(context, constraints) {
```

```
// If there is items to
display

    return InkWell(
        //Clickable items
        onTap: () {
            Navigator.push( //On
item click, forward data over to item customisation page
            context,
            MaterialPageRoute(
                builder:
                (context) => RestaurantItemCustomisePage(
                    reslogo:
widget

.reslogo,
                    itemid:
restaurantitems[0]["menuitems"]

[index][0],
foodcategory: restaurantitems[0]

["menuitems"]

[index][1],
subfoodcategory: restaurantitems[0]
```

```
["menuitems"]

[index][2],                                         itemname:
restaurantitems[0]

["menuitems"]

[index][3],                                         description: restaurantitems[0]

["menuitems"][index][4],                           price:
restaurantitems[0]["menuitems"][index][5],           itemimage:
restaurantitems[0]["menuitems"][index][6])));

},                                                 child: Container(
                                                     //Create clickable
container

width: double.infinity,                                margin:
                                                     const
EdgeInsets.only(                                     left: 10,

```

```
        right: 10,  
  
        top: 10,  
  
        bottom: 10),  
  
        child: Row(  
  
            //Create a row  
containing item image, name and price  
  
            children: [  
  
                Container(  
  
                    width: 80,  
  
                    height: 80,  
  
                    decoration:  
BoxDecoration(  
  
borderRadius:  
  
                const  
BorderRadius  
  
.all(  
  
Radius.circular(  
  
5)),  
  
image:  
DecorationImage(  
  
fit:  
 BoxFit
```

```
.cover,  
        image:  
NetworkImage(restaurantitems[0]["menuitems"]  
  
[  
  
index] [6]  
  
.toString()))),  
        ),  
        const SizedBox(  
            width: 15),  
        Expanded(  
            child: Column(  
  
mainAxisAlignment:  
MainAxisAlignment  
  
.start,  
  
crossAxisAlignment:  
CrossAxisAlignment  
  
.start,
```

```
        children: [  
          Text(  
  
restaurantItems[0]  
  
[  
  
  "menuItems"]  
  
[  
  
  index] [3]  
  
.toString(),  
  
textAlign:  
  
TextAlign  
  
.start,  
  
style:  
Theme.of(  
  
context)  
  
.textTheme  
  
.headline6!
```

```
.copyWith(  
  
    color: Theme.of(context)  
  
    .colorScheme  
  
    .onBackground),  
    ),  
  
    Text(  
  
restaurantItems[0]  
  
[  
  
"menuitems"]  
  
[  
  
index][4]  
  
.toString(),  
  
maxLines: 3,  
  
style:  
Theme.of(  
  
context)
```

```
.textTheme  
  
.bodyText2!  
  
.copyWith(  
  
color: Theme.of(context)  
  
.textTheme  
  
.headline5!  
  
.color,  
  
fontSize:  
  
14),  
  
overflow:  
  
TextOverflow  
  
.clip,  
  
)  
  
,  
)  
,
```

```
const SizedBox( width: 15), Row( children: [ Text( "\u00a3", style: Theme.of( context).textTheme.headline6!.copyWith( color: Theme.of(context).primaryColor), ) , const SizedBox( width: 2), Text(
```

```
restaurantitems[0]["menuitems"]  
  
[  
  
index]  
  
[5]  
  
.toString(),  
  
style:  
Theme.of(  
  
context)  
  
.textTheme  
  
.headline6!  
  
.copyWith(  
  
color: Theme.of(context)  
  
.colorScheme  
  
.onBackground))  
  
],  
  
) ,
```

Setupverification.dart

```
import 'package:alleat/screens/profilesetup/welcomeScreen.dart';

import 'package:alleat/services/localprofiles_service.dart';
```

```
import 'package:alleat/widgets/genericlocading.dart';

import 'package:alleat/widgets/navigationbar.dart';

import 'package:flutter/material.dart';

class SetupWrapper extends StatefulWidget {

  const SetupWrapper({Key? key}) : super(key: key);

  @override

  State<SetupWrapper> createState() => _SetupWrapperState();
}

class _SetupWrapperState extends State<SetupWrapper> {

  //Default setup not complete if something wrong happens

  bool setup = false;

  Future<bool> isSetupComplete() async {

    List<Map> profileInfo = await SQLiteLocalProfiles

      .getFirstProfile(); //Call Database for the first entry

    if (profileInfo.isEmpty) {

      //If the first entry is empty

      //Then setup is not complete (pass to build)

      return false;
    } else {

```

```
//If the first entry exists

//Setup is complete (pass to build)

return true;

}

}

@Override

Widget build(BuildContext context) => FutureBuilder<bool> (

  future: isSetupComplete(),

  builder: (context, snapshot) {

    if (!snapshot.hasData) {

      //While loading, go to loading page

      return const GenericLoading();

    }

    if (snapshot.hasError) {

      //If the app has an error, go to error page

      return const GenericLoading();

    } else {

      bool setup = snapshot.data ?? [] as bool; //Get data from

Future

      if (setup == false) {

        //If setup is not complete

        return const ProfileSetupWelcome(); //Go to Setup page

      }

    }

  }

);
```

```

        } else if (setup == true) {

            //If setup is complete

            return const Navigation(); //Go to main page

        } else {

            //If there is an error

            return const GenericLoading(); // Go to error page

        }

    }

},
);

}

```

Dataencryption.dart

```

import 'package:encrypt/encrypt.dart' as encrypty;

import 'dart:convert' as converty;

import 'package:crypto/crypto.dart' as cryptoy;

class DataEncryption {

    static Future<String> encrpyt(plaintext) async {

        dynamic encryptPassword;

        String strkey =

            'ZC8cegCGG45d1IjIACEtrfypDXtkgJ1rA+4JABPncUE='; //Static key and
iv
    }
}

```

```

String striv = 'yvhsTTh739b2yUW9NNWrcKmHTtLTZNbjbiV3F/cSRzM=';

var iv = crypto.sha256 //Grab the substring of the key and iv

    .convert(convert.utf8.encode(striv))

    .toString()

    .substring(0, 16);

var key = crypto.sha256

    .convert(convert.utf8.encode(strkey))

    .toString()

    .substring(0, 16);

encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);

encrypty.Key keyObj = encrypty.Key.fromUtf8(key);

final encrypter =

    encrypty.Encrypter(encrypty.AES(keyObj, mode:
encrypty.AESMode.cbc));

encryptPassword = encrypter.encrypt(plaintext,

    iv: ivObj); //Use the key and iv to encrypt the plaintext password

encryptPassword = encryptPassword.base64;

return encryptPassword;

}

}

```

Localprofiles_service.dart

```
import 'dart:math';
```

```
import 'package:flutter/material.dart';

import 'package:sqflite/sqflite.dart' as sql;

class SQLiteLocalProfiles {

    // -----
    // Local Profiles Table
    // -----


    static Future<void> createTableProfile(sql.Database database) async {

        //Create localprofiles table (Used to store local logged in profiles)

        await database.execute("""CREATE TABLE localprofiles(
            id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
            profileid INT,
            firstname TEXT,
            lastname TEXT,
            email TEXT,
            password TEXT,
            selected TEXT,
            profilecolorred INT,
            profilecolorgreen INT,
            profilecolorblue INT,
            createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
        )""");
    }
}
```

```
    """);

}

static Future<sql.Database> db() async {

    //If tables don't exist, create tables

    return sql.openDatabase(
        'alleatlocal.db',
        version: 1,
        onCreate: (sql.Database database, int version) async {
            await createTableProfile(database);
        },
    );
}

//-----
// Profile Creation
//-----

// Create new profile

static Future<void> createProfile(dynamic profileid, String firstname,
    String lastname, String email, String password) async {
```

```
//Create profile using firstname, lastname, email and encrypted  
password  
  
List colors = [  
  
    [46, 41, 78],  
  
    [239, 188, 213],  
  
    [190, 151, 198],  
  
    [134, 97, 193],  
  
    [77, 104, 184],  
  
    [112, 202, 209]  
  
];  
  
Random random = Random();  
  
List randomProfileColor = colors[random.nextInt(colors.length)];  
  
final db = await SQLiteLocalProfiles.db();  
  
db.insert("localprofiles", {  
  
    //Insert data into localprofiles table  
  
    "profileid": profileid,  
  
    "firstname": firstname,  
  
    "lastname": lastname,  
  
    "email": email,  
  
    "password": password,  
  
    "profilecolorred": randomProfileColor[0],  
  
    "profilecolorgreen": randomProfileColor[1],  
  
    "profilecolorblue": randomProfileColor[2],
```

```
    });

    selectProfile(email);

}

static Future<void> selectProfile(dynamic email) async {

    final db = await SQLiteLocalProfiles.db();

    db.rawUpdate(
        'UPDATE localprofiles SET selected = false'); //Set all profiles
selected status to false

    db.rawUpdate(
        'UPDATE localprofiles SET selected = true WHERE email =
"$email"');

}

//-----
// Profile Query
//-----

// Get profiles

static Future<List<Map<String, dynamic>>> getProfiles() async {

    final db = await SQLiteLocalProfiles.db();

    return db.query('localprofiles', orderBy: "id"); //get local profiles

}
```

```
// Get first profile in profiles table

static Future<List<Map<String, dynamic>>> getFirstProfile() async {

    final db = await SQLiteLocalProfiles.db();

    return db.rawQuery('SELECT * FROM localprofiles ORDER BY id ASC LIMIT 1');

}

// Get currently selected profile color from table

static Future<List<Map<String, Object?>>> getSelectedProfileColor() async {

    final db = await SQLiteLocalProfiles.db();

    return db.rawQuery(
        'SELECT profilecolorred, profilecolorgreen, profilecolorblue FROM localprofiles ORDER BY id ASC LIMIT 1');

}

// Get all unselected profiles to be displayed

static Future<List<Map<String, Object?>>> getDisplayProfilesList() async {
    final db = await SQLiteLocalProfiles.db();
}
```

```

        return db.rawQuery(
            'SELECT id, profileid, firstname, lastname, profilecolorred,
profilecolorgreen, profilecolorblue FROM localprofiles ORDER BY selected
DESC');

    }

    // Get profile info from ID

    static Future<List<Map<String, Object?>>> getProfileFromID(@+id id) async {
        final db = await SQLiteLocalProfiles.db();
        return db.rawQuery(
            'SELECT profileid, firstname, lastname, email FROM localprofiles
WHERE id = $id');
    }

    //-----
    // Profile Modify
    //-----

    // Update an profile by id

    static Future<void> updateProfile(int id, String firstname, String
lastname,
        String email, String password) async {
        //Update profile using firstname, lastname, email and encrypted
password

```

```
final db = await SQLiteLocalProfiles.db();

final data = {
    'firstname': firstname,
    'lastname': lastname,
    'email': email,
    'password': password,
};

await db.update('localprofiles', data, where: "id = ?", whereArgs: [id]);
}

// Delete profile

static Future<void> deleteProfile(int id) async {
    //Delete profile associated with the id inputted
    final db = await SQLiteLocalProfiles.db();
    try {
        //Try to delete profile
        await db.delete("localprofiles", where: "id = ?", whereArgs: [id]);
    } catch (err) {
        // If it fails, prints the error to the console
        debugPrint("Something went wrong when deleting an item: $err");
    }
}
```

```
    }

}

}
```

Queryserver.dart

```
import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

class QueryServer {

    static Future<Map> query(phurl, body) async { //Get url and data being sent

        try { //Try to send data to server

            var res = await http.post(Uri.parse(phurl), body: body);

            if (res.statusCode != 200) { //If the server rejects the data return the error code

                return {"error": true, "message": "Error ${res.statusCode}"};

            } else { // If server successfully gets data

                var data = convert.json.decode(res.body); // Get the data sent back from the server and decode

                if (data["error"]) { //If the data has an error

                    return {"error": true, "message": "Error 500"}; //Return error 500 (server rejects data)

                } else {

                    return {"error": false, "message": data};

                }

            }

        }

    }

}
```

```

        }

    }

} catch (e) { //If there is an error, return error message

    return {"error": true, "message": "ERROR: $e"};

}

}

}

```

Setselected.dart

```

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/queryserver.dart';

import 'package:shared_preferences/shared_preferences.dart';


class SetSelected {

    static Future<bool> selectProfile(
        serverid, firstname, lastname, email) async {

        dynamic favReataurantList = await QueryServer.query( //Get favourites
from server

        'https://alleat.cpur.net/query/favouriterestaurantlist.php',
        {"profileemail": email});

        if (favReataurantList["Error"] == true) { //If getting favourites
fails, return false

        return false;
    }
}

```

```

} else {

    try { //Try to change the shared preferences to the selected profile

        final prefs = await SharedPreferences.getInstance();

        await prefs.setString('serverprofileid', serverid.toString());

        await prefs.setString('firstname', firstname);

        await prefs.setString('lastname', lastname);

        await prefs.setString('email', email);

        await prefs.setString('favrestaurants',
            ((favReataurantList["message"]) ["restaurantids"]).toString());

        SQLiteLocalProfiles.selectProfile(email); //Select profile in the
database

        return true;

    } catch (e) { //If there is an error, return false

        return false;

    }

}

}

```

Theme.dart

```

library globals;

import 'package:flutter/material.dart';

```

```
dynamic appthemetpreference = 0;

class ThemeClass {

    static Color primaryLight = const Color(0xff5806FF);

    static Color secondaryLight = const Color(0xffAD84FF);

    static Color successLight = const Color(0xff07852A);

    static Color errorLight = const Color(0xffAF0E17);

    static Color textLight = const Color(0xff1A1A1A);

    static Color bgLight = const Color(0xffFAFAFA);

    static Color bottomAppBarLight = const Color(0xffffffff);

    static Color primaryDark = const Color(0xffC1A3FF);

    static Color secondaryDark = const Color(0xffAD84FF);

    static Color successDark = const Color(0xff60F68A);

    static Color errorDark = const Color(0xFFF2555F);

    static Color textDark = const Color(0xffEAEEAE);

    static Color bgDark = const Color(0xff0C0C0C);

    static Color bottomAppBarDark = const Color(0xff161616);

    static Color mono900 = const Color(0xff0C0C0C);

    static Color mono800 = const Color(0xff161616);

    static Color mono700 = const Color(0xff292929);
```

```
static Color mono600 = const Color(0xff434343);

static Color mono500 = const Color(0xff676767);

static Color mono400 = const Color(0xff818181);

static Color mono300 = const Color(0xff979797);

static Color mono200 = const Color(0xffB6B6B6);

static Color mono100 = const Color(0xffD2D2D2);

static Color mono050 = const Color(0xffEAEAEA);

static Color mono000 = const Color(0xffffffff);

static Color primary900 = const Color(0xff060011);

static Color primary800 = const Color(0xff160041);

static Color primary700 = const Color(0xff2B0082);

static Color primary600 = const Color(0xff4100C4);

static Color primary500 = const Color(0xff5806FF);

static Color primary400 = const Color(0xff9866FF);

static Color primary300 = const Color(0xffAD84FF);

static Color primary200 = const Color(0xffC1A3FF);

static Color primary100 = const Color(0xffD6C2FF);

static Color primary050 = const Color(0xffEBE0FF);

static ThemeData lightTheme = ThemeData(

    //COLOUR
```

```
primaryColor: primaryLight,  
colorScheme: ColorScheme.fromSwatch().copyWith(  
    secondary: secondaryLight,  
    error: errorLight,  
    onBackground: textLight,  
    onSurface: mono000,  
)  
  
// MAIN APP  
  
bottomNavigationBarTheme: BottomNavigationBarThemeData(  
    backgroundColor: mono000,  
    selectedItemColor: primaryLight,  
    unselectedItemColor: mono400),  
appBarTheme: AppBarTheme(backgroundColor: primaryLight),  
backgroundColor: bgLight,  
scaffoldBackgroundColor: bgLight,  
snackBarTheme: SnackBarThemeData(  
    contentTextStyle: TextStyle(  
        color: mono900,  
        fontFamily: 'Satoshi',  
        fontWeight: FontWeight.w400,  
    )
```

```
    fontSize: 14),  
  
    backgroundColor: mono100),  
  
// TEXT  
  
textTheme: TextTheme(  
  
    headline1: TextStyle(  
  
        color: textLight,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w800,  
  
        fontSize: 32),  
  
    headline2: TextStyle(  
  
        color: mono900,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w700,  
  
        fontSize: 28),  
  
    headline3: TextStyle(  
  
        color: mono800,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w600,  
  
        fontSize: 24),  
  
    headline4: TextStyle(  
  
        color: mono800,
```

```
fontFamily: 'Satoshi',  
fontWeight: FontWeight.w500,  
fontSize: 21),  
  
headline5: TextStyle(  
  
color: mono500,  
fontFamily: 'Satoshi',  
fontWeight: FontWeight.w600,  
fontSize: 20),  
  
headline6: TextStyle(  
  
color: mono400,  
fontFamily: 'Satoshi',  
fontWeight: FontWeight.w600,  
fontSize: 16),  
  
bodyText1: TextStyle(  
  
color: textLight,  
fontWeight: FontWeight.w600,  
fontFamily: 'NotoSans',  
fontSize: 14),  
  
bodyText2: TextStyle(  
  
color: textLight,  
fontWeight: FontWeight.w400,  
fontFamily: 'NotoSans',  
fontSize: 16)),
```

```
//BUTTONS

buttonTheme: ButtonThemeData(
    shape:
        RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10.0)),
    padding:
        const EdgeInsets.only(top: 18, bottom: 18, left: 30, right:
30)),
    elevatedButtonTheme: ElevatedButtonThemeData(
        style: ElevatedButton.styleFrom(
            foregroundColor: bgLight,
            backgroundColor: primaryLight,
            padding: const EdgeInsets.only(
                top: 15, bottom: 15, left: 30, right: 30),
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(5), // <-- Radius
            ),
            textStyle: TextStyle(
                color: bgLight,
                fontFamily: 'Satoshi',
                fontWeight: FontWeight.w600,
                fontSize: 16))),

```

```
outlinedButtonTheme: OutlinedButtonThemeData(  
  
    style: OutlinedButton.styleFrom(  
  
        foregroundColor: primaryLight,  
  
        padding: const EdgeInsets.only(  
  
            top: 15, bottom: 15, left: 30, right: 30),  
  
            side: BorderSide(color: primaryLight, width: 3),  
  
            shape: RoundedRectangleBorder(  
  
                borderRadius: BorderRadius.circular(5), // <-- Radius  
            ),  
  
            textStyle: TextStyle(  
  
                color: primaryLight,  
  
                fontFamily: 'Satoshi',  
  
                fontWeight: FontWeight.w600,  
  
                fontSize: 16))),  
  
textButtonTheme: TextButtonThemeData(  
  
    style: TextButton.styleFrom(  
  
        foregroundColor: primaryLight,  
  
        padding: const EdgeInsets.only(  
  
            top: 18, bottom: 18, left: 30, right: 30),  
  
            textStyle: TextStyle(  
  
                color: primaryLight,  
  
                fontFamily: 'Satoshi',  
  
                fontWeight: FontWeight.w600,  
            ),  
        ),  
    ),
```

```
        fontSize: 16))),

// FORM FIELD

inputDecorationTheme: InputDecorationTheme(
    filled: false,
    hintStyle: TextStyle(
        color: mono300,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16),
    contentPadding: const EdgeInsets.all(10),
    border: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono100)),
    focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: primaryLight)),
    disabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono050)),
    errorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorLight)),
    focusedErrorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorDark)),
    enabledBorder: UnderlineInputBorder(
```

```
borderSide: BorderSide(width: 3, color: mono100))));  
  
static ThemeData darkTheme = ThemeData(  
  
    //COLOUR  
  
    primaryColor: primaryDark,  
  
    colorScheme: ColorScheme.fromSwatch().copyWith(  
  
        secondary: secondaryDark,  
  
        error: errorDark,  
  
        onBackground: textDark,  
  
        onSurface: mono700,  
  
) ,  
  
//MAIN APP  
  
bottomNavigationBarTheme: BottomNavigationBarThemeData(  
  
    backgroundColor: mono800, unselectedItemColor: mono600),  
  
appBarTheme: AppBarTheme(backgroundColor: mono800),  
  
backgroundColor: bgDark,  
  
scaffoldBackgroundColor: bgDark,  
  
// TEXT
```

```
textTheme: TextTheme(  
  
    headline1: TextStyle(  
  
        color: textDark,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w800,  
  
        fontSize: 32),  
  
    headline2: TextStyle(  
  
        color: mono050,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w700,  
  
        fontSize: 28),  
  
    headline3: TextStyle(  
  
        color: mono100,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w600,  
  
        fontSize: 24),  
  
    headline4: TextStyle(  
  
        color: mono100,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w500,  
  
        fontSize: 21),  
  
    headline5: TextStyle(
```

```
color: mono300,  
  
fontFamily: 'Satoshi',  
  
fontWeight: FontWeight.w500,  
  
fontSize: 20),  
  
headline6: TextStyle(  
  
color: mono400,  
  
fontFamily: 'Satoshi',  
  
fontWeight: FontWeight.w600,  
  
fontSize: 16),  
  
bodyText1: TextStyle(  
  
color: textDark,  
  
fontWeight: FontWeight.w600,  
  
fontFamily: 'NotoSans',  
  
fontSize: 14),  
  
bodyText2: TextStyle(  
  
color: mono000,  
  
fontWeight: FontWeight.w400,  
  
fontFamily: 'NotoSans',  
  
fontSize: 16)),  
  
//BUTTONS  
  
buttonTheme: ButtonThemeData(  
)
```

```
shape:

    RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10.0)),

padding:

    const EdgeInsets.only(top: 18, bottom: 18, left: 30, right:
30)),

elevatedButtonTheme: ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
        foregroundColor: bgDark,
        backgroundColor: primaryDark,
        padding: const EdgeInsets.only(
            top: 15, bottom: 15, left: 30, right: 30),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(5), // <-- Radius
        ),
        textStyle: TextStyle(
            color: mono900,
            fontFamily: 'Satoshi',
            fontWeight: FontWeight.w600,
            fontSize: 16))),
    outlinedButtonTheme: OutlinedButtonThemeData(
        style: OutlinedButton.styleFrom(
            foregroundColor: primaryDark,
            padding: const EdgeInsets.only(

```



```
// FORM FIELD

inputDecorationTheme: InputDecorationTheme(
    filled: false,
    hintStyle: TextStyle(
        color: mono400,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16),
    contentPadding: const EdgeInsets.all(10),
    border: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono700)),
    focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: primaryDark)),
    disabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono800)),
    errorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorDark)),
    focusedErrorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorLight)),
    enabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono600)));
}
```

Browse_categories.dart

```
import 'package:alleat/widgets/elements/elements.dart';

import 'package:flutter/material.dart';

List categoriesList = [ //List of categories

  [
    "Burgers",
    const Color(0xffffd8c2),
    'lib/assets/images/categories/burger-category.png'
  ],
  [
    "American",
    const Color(0xffeec4c4),
    'lib/assets/images/categories/american-category.png'
  ],
  [
    "Breakfast",
    const Color(0xffc6f0d9),
    'lib/assets/images/categories/breakfast-category.png'
  ],
  [
    "Sushi",
    const Color(0xffc6f0d9),
    'lib/assets/images/categories/sushi-category.png'
  ]
]
```

```
const Color(0xfffc2f7a9),  
'lib/assets/images/categories/sushi-category.png'  
,  
[  
  "Chinese",  
  const Color(0xfffffeec2),  
'lib/assets/images/categories/chinese-category.png'  
,  
[  
  "Thai",  
  const Color(0xffff8c6aa),  
'lib/assets/images/categories/thai-category.png'  
,  
[  
  "Coffee",  
  const Color(0xffff0c6df),  
'lib/assets/images/categories/coffee-category.png'  
,  
[  
  "Greek",  
  const Color(0xffc6e1f0),  
'lib/assets/images/categories/greek-category.png'  
,
```

```
];

class Categories extends StatefulWidget {

  const Categories({super.key});

  @override

  State<Categories> createState() => _CategoriesState();
}

class _CategoriesState extends State<Categories> {

  @override

  Widget build(BuildContext context) {

    return SizedBox(
      //Create profile selection widget with height 150px
      height: 170,
      child: ListView(scrollDirection: Axis.horizontal, children:
      <Widget>[
        Row(mainAxisAlignment: MainAxisAlignment.center, children: [
          Padding(
            padding: const EdgeInsets.only(left: 25),
            child: SizedBox(
              width: 825,
              child: ListView.builder(

```

```
        itemCount: 5,  
  
        scrollDirection: Axis.horizontal, //Make  
scrollable list  
  
        itemBuilder: (context, index) {  
  
            List category = categoriesList[index];  
  
            return Padding(  
  
                padding:  
  
                const EdgeInsets.symmetric(horizontal:  
7),  
  
                child: InkWell(  
  
                    onTap: (() {}),  
  
                    child: Stack(  
  
                        alignment:  
AlignmentDirectional.topCenter,  
  
                        children: [  
  
                            Container(  
  
                                decoration: BoxDecoration(  
  
                                    borderRadius:  
                                    const BorderRadius.all(  
Radius.circular(10)),  
  
                                    color: category[1],  
  
) ,  
  
width: 150,  
  
height: 170,
```

```
padding: const  
EdgeInsets.symmetric(  
  
    horizontal: 10),  
  
) ,  
  
Padding(  
  
padding: const  
EdgeInsets.symmetric(  
  
    vertical: 15),  
  
child: Align(  
  
alignment:  
Alignment.topCenter,  
  
child: Text(  
  
category[0],  
  
textAlign:  
TextAlign.center,  
  
style: Theme.of(context)  
  
.textTheme  
  
.headline5!  
  
.copyWith(  
  
color: const  
Color(  
  
0xff000000)),  
  
)) ),  
  
Image.asset(
```

```
//Fullscreen image of food

category[2],

alignment:

Alignment.bottomRight,

),

] )));

} )))

]

] ));

}

}

class CategoriesPage extends StatefulWidget {

const CategoriesPage({super.key});

}

@Override

State<CategoriesPage> createState() => _CategoriesPageState();

}

class _CategoriesPageState extends State<CategoriesPage> {

@Override

Widget build(BuildContext context) {

return Scaffold(
```

```
body: CustomScrollView(slivers: [  
    SliverFillRemaining(  
        hasScrollBody: false,  
        child:  
            Column(crossAxisAlignment: CrossAxisAlignment.start,  
children: [  
    const ScreenBackButton(), //Back button  
    const SizedBox(height: 20),  
    Padding(  
        padding: const EdgeInsets.symmetric(horizontal: 30),  
        child: Text(  
            "All Categories.",  
            textAlign: TextAlign.start,  
            style: Theme.of(context).textTheme.headline2,  
        )),  
    Padding(  
        padding: const EdgeInsets.only(  
            bottom: 40, top: 20, left: 20, right: 20),  
        child: SizedBox(  
            height: (categoriesList.length * 185) / 2.ceil(),  
            child: GridView.builder( //Build grid of categories  
                itemCount: categoriesList.length,  
                physics: const NeverScrollableScrollPhysics(),
```

```
gridDelegate:

    const SliverGridDelegateWithFixedCrossAxisCount (

        crossAxisCount: 2,

        crossAxisSpacing: 0.0,

        mainAxisSpacing: 10.0),

    itemBuilder: (context, index) {

        List category = categoriesList[index];

        return Padding (

            padding: const

EdgeInsets.symmetric(horizontal: 7),

            child: InkWell (

                onTap: () {},

                child: Stack (

                    alignment: AlignmentDirectional.topCenter,

                    children: [

                        Container (

                            decoration: BoxDecoration (

                                borderRadius: const

BorderRadius.all (

                                    Radius.circular(10)),

                            color: category[1],


                        ),


                        width: 150,
```

```
        height: 170,  
  
        padding: const  
EdgeInsets.symmetric(  
  
            horizontal: 10),  
  
) ,  
  
Padding(  
  
    padding: const  
EdgeInsets.symmetric(  
  
        vertical: 15),  
  
        child: Align(  
  
            alignment:  
Alignment.topCenter,  
  
            child: Text(  
  
                category[0],  
  
                textAlign:  
TextAlign.center,  
  
                style: Theme.of(context)  
  
                    .textTheme  
  
                    .headline5!  
  
                    .copyWith(  
  
                        color: const  
Color(  
  
                            0xff000000)),  
  
                    ) ),  
  
Image.asset(  
)
```

```
//Fullscreen image of food

category[2],
alignment: Alignment.bottomRight,
),
] )) );
} ) ,
)
] ) ,
)
] ;
}

}
```

Elements.dart

```
import 'package:flutter/material.dart';

class ScreenBackButton extends StatelessWidget {
final dynamic data;

const ScreenBackButton({Key? key, this.data}) : super(key: key); //Get
any data that needs to be sent to previous screen

@Override
```

```

Widget build(BuildContext context) {

  return Padding(
    padding: const EdgeInsets.only(top: 50),
    child: TextButton.icon(
      onPressed: () => Navigator.of(context).pop(data), //Remove
      current screen
      label: Text(
        "Back",
        style: Theme.of(context)
          .textTheme
          .headline6!
          .copyWith(color: Theme.of(context).primaryColor),
      ),
      icon: Icon(
        Icons.chevron_left,
        color: Theme.of(context).primaryColor,
      )));
}

}

```

Profiles_buttons.dart

```
import 'package:flutter/material.dart';
```

```
class ProfileButton extends StatelessWidget {

    final dynamic icon;

    final dynamic name;

    final dynamic action;

    const ProfileButton({Key? key, this.icon, this.name, this.action})

        : super(key: key);

    @override

    Widget build(BuildContext context) {

        return Container(
            padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 15),
            width: double.infinity,
            height: 65,
            margin: const EdgeInsets.symmetric(vertical: 7, horizontal: 25),
            decoration: BoxDecoration(
                borderRadius: const BorderRadius.all(Radius.circular(10)),
                color: Theme.of(context).colorScheme.onSurface,
                boxShadow: [
                    BoxShadow(
                        color:
                            Theme.of(context).colorScheme.onBackground.withOpacity(0.05),
                        spreadRadius: 10,
                        blurRadius: 45,
                ],
            ),
        );
    }
}
```

```
        offset: const Offset(0, 30), // changes position of shadow
    ) ,
],
),
child: InkWell(
    child: Row(
        children: [
            Icon(icon, color: Theme.of(context).colorScheme.onBackground),
            const SizedBox(width: 20),
            Text(
                name,
                style: Theme.of(context).textTheme.headline5?.copyWith(
                    color: Theme.of(context).colorScheme.onBackground,
                    fontWeight: FontWeight.w700),
            )
        ],
),
onTap: () {
    action;
},
),
);
}
}
```

```
}
```

Profiles_selection.dart

```
import 'package:alleat/screens/profilesetup/profilesetup_existing.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/setselected.dart';

import 'package:flutter/material.dart';

class ProfileList extends StatefulWidget {

  const ProfileList({super.key});

  @override

  State<ProfileList> createState() => _ProfileListState();
}

class _ProfileListState extends State<ProfileList> {

  bool edit = false;

  Future<List> getDisplayableProfiles() async {

    return await SQLiteLocalProfiles.getDisplayProfilesList();
  }
}
```

```
Future<void> selectProfile(id) async {

    var getToSelect = (await SQLiteLocalProfiles.getProfileFromID(id))[0];
    //Get profile info from id

    bool trySelect = await SetSelected.selectProfile( //Try to select
profile

        getToSelect['profileid'],
        getToSelect['firstname'],
        getToSelect['lastname'],
        getToSelect['email']);
}

if (trySelect == true) { //if it successfully selects profile

    setState(() {

        ScaffoldMessenger.of(context).showSnackBar( // Display that it
selected

        SnackBar(
            content: Text(
                'Switched profiles successfully',
                style: Theme.of(context).textTheme.headline6,
            ),
            backgroundColor: Theme.of(context).colorScheme.onSurface,
        ),
    );
}
} else { //If it fails, display that it was not selected
}
```

```
        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                    content: Text(
                        'Failed to switch profiles.',
                        style: Theme.of(context)
                            .textTheme
                            .headline6!
                            .copyWith(color: const Color(0xffffffff)),
                    ),
                    backgroundColor: Theme.of(context).colorScheme.error,
                ),
            );
        });
    }

    @override
    Widget build(BuildContext context) {
        return SizedBox(
            //Create profile selection widget with height 150px
            height: 150,
```

```
        child: ListView(scrollDirection: Axis.horizontal, children:  
<Widget>[  
  
        Row(mainAxisAlignment: MainAxisAlignment.center, children: [  
  
        FutureBuilder<List>(  
  
            // Get selected profile  
  
            future: getDisplayableProfiles(),  
  
            builder: (context, snapshot) {  
  
                if (!snapshot.hasData) {  
  
                    //While there is no information sent back  
  
                    return Column(  
  
                        //Display empty circle  
  
                        children: [  
  
                            Container(  
  
                                width: 120,  
  
                                height: 120,  
  
                                decoration: BoxDecoration(  
  
                                    shape: BoxShape.circle,  
  
                                    color: Theme.of(context)  
  
                                        .navigationBarTheme  
  
                                        .backgroundColor))  
  
                            ],  
  
                        );  
  
                } else if (snapshot.hasData) {  

```

```

    // Once there is information

    List? profileInfo = snapshot.data; // Store in
profileInfo

    if (profileInfo != null && profileInfo.isNotEmpty) {

        //If it contains a profile

        return Padding(
            padding: const EdgeInsets.only(left: 30),
            child: SizedBox(
                //Set width of data to the number of
                profiles by 120px
                width: (profileInfo.length) * 120,
                child: ListView.builder(
                    //For each profile
                    itemCount: profileInfo.length,
                    scrollDirection:
                        Axis.horizontal, //Make scrollable
list
                    itemBuilder: (context, index) {
                        Map<String, dynamic> profile =
profileInfo[
                            index]; //Store current profile
being created in profile

                        if (index == 0) {
                            return InkWell(

```

```
        onTap: (() {}),  
  
        child: Container(  
  
          padding:  
  
            const  
EdgeInsets.symmetric(  
  
              horizontal: 10),  
  
          child: Column(children: [  
  
            Stack(  
  
              alignment:  
Alignment.center,  
  
              children: [  
  
                Container(  
  
                  width: 93,  
  
                  height: 93,  
  
                  decoration:  
BoxDecoration(  
  
                    shape:  
  
BoxShape.circle,  
  
                    border:  
Border.all(  
  
                      color:  
Theme.of(  
  
context)
```

```
.primaryColor,  
    width: 3,  
    style:  
  
BorderStyle  
  
.solid)),  
) ,  
Container(  
  
    // Create a  
circle with a purple outline and and the first letter of first and last  
name  
  
    width: 80,  
    height: 80,  
    decoration:  
BoxDecoration(  
    shape:  
BoxShape  
    .circle,  
    color:  
Color.fromRGBO(  
    profile[  
  
'profilecolorred'],  
    profile[
```

```
'profilecolorgreen'] ,  
    profile[  
  
'profilecolorblue'] ,  
    1)) ,  
    child: Align(  
      alignment:  
  
Alignment  
  
.center,  
      child: Text(  
  
'${profile['firstname'][0]}${profile['lastname'][0]}' ,  
      style:  
Theme.of(  
  
context)  
  
.textTheme  
  
.headline3  
  
?.copyWith(  
  
color:
```

```
Theme.of(context).backgroundColor))))  
],  
  
const SizedBox(height:  
15),  
  
SizedBox(  
  
//Display profile name  
width: 100,  
  
child: Text(  
  
"${profile['firstname']}",  
  
textAlign:  
  
TextAlign.center,  
  
style:  
Theme.of(context)  
  
.textTheme  
.headline5  
?.copyWith(  
  
fontWeight:  
  
FontWeight  
  
.w600,  
  
color:  
Theme.of(  
)
```

```
context)

.textTheme

.headline1

?.color),

overflow:

TextOverflow.ellipsis,

)),

]) ));

} else if (index > 0) {

return InkWell(

onTap: () {

selectProfile(profile['id']);

} ,


child: Container(


padding:

const

EdgeInsets.symmetric(


horizontal: 10),


child: Column(children: [


Container(
```

```
// Create a circle
with a purple outline and and the first letter of first and last name

width: 88,
height: 88,
decoration:
BoxDecoration(
shape:
BoxShape.circle,
color:
Color.fromRGBO(
profile[

'profilecolorred'],
profile[

'profilecolorgreen'],
profile[

'profilecolorblue'],
1)),
child: Align(
alignment:
Alignment.center,
child: Text(
'${profile['firstname'][0]}${profile['lastname'][0]}',
```

```
        style:  
Theme.of(  
  
context)  
  
.textTheme  
  
.headline3  
  
?.copyWith(  
  
color:  
Theme.of(  
  
context)  
  
.backgroundColor))))) ,  
  
const SizedBox(height:  
15) ,  
  
SizedBox(  
  
//Display profile name  
width: 100,  
  
child: Text(  
  
"${profile['firstname']} ",  
  
textAlign:  
TextAlign.center,  
  
style:  
Theme.of(context)
```

```
    .textTheme  
        .headline5  
            ?.copyWith(  
                fontWeight:  
  
FontWeight  
  
.w600,  
            color:  
Theme.of(  
  
context)  
  
.textTheme  
  
.headline1  
  
?.color),  
            overflow:  
TextOverflow.ellipsis,  
        ) )  
    ] ));  

```

```
        horizontal: 10),  
  
        child: Column(children: [  
  
          Container(  
  
            // Create a circle with a  
            purple outline and the first letter of first and last name  
  
            width: 100,  
  
            height: 100,  
  
            decoration: BoxDecoration(  
  
              shape:  
              BoxShape.circle,  
  
              color:  
              Theme.of(context)  
  
                .colorScheme  
  
                .error),  
  
              child: Align(  
  
                alignment:  
                Alignment.center,  
  
                child: Text(  
  
                  "Error",  
  
                  style:  
                  Theme.of(context)  
  
                    .textTheme  
  
                    .headline5  
  
                    ?.copyWith(  
          
```

```
color: const  
Color(  
  
0xffffffff)) ,  
)  
) )  
]  
);  
}  
} ) );  
}  
} else {  
return Column(  
children: [  
Container(  
width: 100,  
height: 100,  
decoration: BoxDecoration(  
shape: BoxShape.circle,  
color: Theme.of(context).errorColor))  
],  
);  
}  
} else {  
return Column(  
children: [  
]
```

```
Container(  
    width: 100,  
    height: 100,  
    decoration: BoxDecoration(  
        shape: BoxShape.circle,  
        color: Theme.of(context).errorColor))  
,  
) ;  
}  
) ,  
Column(  
children: [  
Padding(  
padding: const EdgeInsets.symmetric(horizontal: 20),  
child: Column(children: [  
ElevatedButton(  
onPressed: () {  
Navigator.push(  
context,  
MaterialPageRoute(  
builder: (context) =>  
const ProfileSetupExisting()));  
}),
```

```
        style: ElevatedButton.styleFrom(
            shape: const CircleBorder(),
            padding: const EdgeInsets.all(22)),
        child: Icon(
            Icons.add,
            color: Theme.of(context).backgroundColor,
            size: 40,
        ),
    ) ,
    const SizedBox(height: 20),
]) )
],
)
]
)
]
);
}
}
```

Search.dart

```
import 'package:flutter/material.dart';

class SearchBar extends StatelessWidget {
```

```
const SearchBar({super.key});
```



```
@override
```

```
Widget build(BuildContext context) {
```

```
    return InkWell(
```

```
        child: Container(
```

```
            width: double.infinity,
```

```
            height: 60,
```

```
            margin: const EdgeInsets.symmetric(vertical: 10, horizontal: 30),
```

```
            padding: const EdgeInsets.symmetric(horizontal: 20),
```

```
            decoration: BoxDecoration(
```

```
                borderRadius: const BorderRadius.all(Radius.circular(10)),
```

```
                color: Theme.of(context).colorScheme.onSurface,
```

```
                boxShadow: [
```

```
                    BoxShadow(
```

```
                        color:
```

```
                        Theme.of(context).colorScheme.onBackground.withOpacity(0.05),
```

```
                        spreadRadius: 10,
```

```
                        blurRadius: 45,
```

```
                        offset: const Offset(0, 30), // changes position of shadow
```

```
                    ),
```

```
                ],
```

```
            ),
```

```
        child: Row(children: [  
  
        Icon(  
  
            Icons.search,  
  
            color: Theme.of(context).textTheme.headline1?.color,  
        ),  
  
        const SizedBox(  
  
            width: 20,  
        ),  
  
        Text(  
  
            'Try searching "Pizza"',  
  
            style: Theme.of(context).textTheme.bodyText2,  
        )  
    ]),  
));  
}  
}
```

Genericloading.dart

```
import 'package:flutter/material.dart';  
  
class GenericLoading extends StatefulWidget {  
  
    const GenericLoading({super.key});  
}
```

```

    @override

    State<GenericLoading> createState() => _GenericLoadingState();
}

class _GenericLoadingState extends State<GenericLoading> {

    @override

    Widget build(BuildContext context) {

        return MaterialApp(
            title: "Loading",
            home: Scaffold(
                resizeToAvoidBottomInset: false,
                body: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: const [CircularProgressIndicator()])),
            theme:
                Theme.of(context)); //Show blank page with circular loading circle
    }
}

```

Navigationbar.dart

```
import 'package:alleat/screens/navigationscreens/browse.dart';
```

```
import 'package:alleat/screens/navigationscreens/foryou.dart';

import 'package:alleat/screens/navigationscreens/homepage.dart';

import 'package:alleat/screens/navigationscreens/profiles.dart';

import 'package:flutter/material.dart';




class Navigation extends StatefulWidget {

  const Navigation({Key? key}) : super(key: key);

  @override

  State<Navigation> createState() => _NavigationState();
}






class _NavigationState extends State<Navigation> {

  int _selectedIndex = 0; //Start at Home page

  final List _screens = [
    {"screen": const HomePage()},
    {"screen": const ForYouPage()},
    {"screen": const BrowsePage()},
    {"screen": const ProfilePage()}
  ];

  void _onItemTapped(int index) {
```

```
//When user click button on bottom navigation bar, change to that
index

    setState(() {
        _selectedIndex = index;
    });
}

@Override
Widget build(BuildContext context) {
    return WillPopScope(
        onWillPop: () async => false,
        child: MaterialApp(
            title: "All Eat.",
            theme: Theme.of(context),
            home: Scaffold(
                body: _screens[_selectedIndex]["screen"],
                bottomNavigationBar: Theme(
                    data: Theme.of(context).copyWith(
                        canvasColor: Theme.of(context)
                            .bottomNavigationBarTheme
                            .backgroundColor),
                child: BottomNavigationBar(
                    // If button is selected, show purple. If button is
not selected show greyed out text and icon

```

```
selectedItemColor: Theme.of(context)

    .bottomNavigationBarTheme

    .selectedItemColor,

unselectedItemColor: Theme.of(context)

    .bottomNavigationBarTheme

    .unselectedItemColor,

backgroundColor: Theme.of(context)

    .bottomNavigationBarTheme

    .backgroundColor,

items: <BottomNavigationBarItem>[

    //Bottom navigation bar items

    BottomNavigationBarItem(
        icon: Container(
            padding: const EdgeInsets.only(
                bottom:
                    3), //Each have padding to separate
        from the text

        child: const Icon(Icons
            .home_outlined)), //Unselected icon is
        outlined

        label: 'Home',

        activeIcon: Container(
            padding: const EdgeInsets.only(bottom: 3),
            child: const Icon(
```

```
        Icons.home)), // Selected icon is filled  
    ),  
  
    BottomNavigationBarItem(  
        //Bottom navigation bar options  
        icon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.assistant_outlined)),  
        label: 'For You',  
        activeIcon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.assistant)),  
        ),  
  
    BottomNavigationBarItem(  
        icon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const  
Icon(Icons.manage_search_rounded)),  
        label: 'Browse',  
        activeIcon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.manage_search)),  
        ),  
  
    BottomNavigationBarItem(  
        icon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.add)),  
        label: 'Add',  
        activeIcon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.add)),  
        ),  
  
    BottomNavigationBarItem(  
        icon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.logout)),  
        label: 'Logout',  
        activeIcon: Container(  
            padding: const EdgeInsets.only(bottom: 3),  
            child: const Icon(Icons.logout)),  
        ),  
    ),  
);
```

```

        icon: Container(
          padding: const EdgeInsets.only(bottom: 3),
          child: const Icon(Icons.person_outlined)),
        label: 'Profile',
        activeIcon: Container(
          padding: const EdgeInsets.only(bottom: 3),
          child: const Icon(Icons.person)),
        ),
      ],
      onTap: _onItemTapped, //On tap, change selected option
      currentIndex:
        _selectedIndex, //Make current index the one last
      selected (defaults to home)
    ),
  ))));
}

}

```

Restaurants_list.dart

```

import 'package:alleat/screens/restaurant/restaurant_main.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

```

```
import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

class RestaurantList extends StatefulWidget {

  const RestaurantList({Key? key}) : super(key: key);

  @override

  State<RestaurantList> createState() => _RestaurantListState();
}

class _RestaurantListState extends State<RestaurantList> {

  late bool error, sending, success, serverOffline;

  late String msg;

  @override

  void initState() {

    //Default values

    error = false;

    sending = false;

    success = false;

    msg = "";

    super.initState();
  }
}
```

```
Future<List> getRestaurants() async {

    String phpurl =
        "https://alleat.cpur.net/query/restaurantlist.php"; //Get
    restaurant list

    try {

        var res = await http.post(Uri.parse(phpurl), body: {

            "sort": "id", //Send sort type (Currently permanently by id)
        });

        if (res.statusCode == 200) {

            //If sends successfully

            var data = convert.json.decode(res.body); //Decode to array

            if (data["error"]) {

                //If fails to perform query

                List error = [
                    {
                        "error": "true",
                        "restaurants": "[]"
                    } //Send blank list of restaurants
                ];
            }

            return error;
        } else {

            List listdata = [

```

```

        data

    ] ; //If success, send the list of restaurants back

    return listdata;

}

} else {

    List error = [

        {"error": "true", "restaurants": "[]"}

    ];

    return error;

}

} catch (e) {

    List<Map<String, String>> error = [

        {"error": "true", "restaurants": "[]"}

    ];

    return error;

}

}

Future<String> favouriteRestaurant(restaurantID, action) async {

    String phpurl =

        "https://alleat.cpur.net/query/favouriterestaurant.php";
//Favourite & unfavourite restaurant for sepcific profile using their
email

    final prefs = await SharedPreferences.getInstance();

```

```
final String? email = prefs.getString('email');

try {

    var res = await http.post(Uri.parse(phurl), body: {

        "action": action

            .toString(), //Action determines if it will favourite or
unfavourite depending on input

        "profileemail": email,

        "restaurantid": restaurantID,

    });

    if (res.statusCode == 200) {

        //On successfull send

        var data = convert.json.decode(res.body); //Decode to array

        if (data["error"]) {

            return "failed";

        } else {

            getFavourites(); //get favourite restaurant list

            return "success";

        }

    } else {

        return "failed";

    }

} catch (e) {

    return "failed";
}
```

```
    }

}

Future<List> getFavourites() async {

    String phpurl =

        "https://alleat.cpur.net/query/favouriterestaurantlist.php"; //Get
favourite restaurant ids using profile email

    final prefs = await SharedPreferences.getInstance();

    final String? email = prefs.getString('email');

    try {

        var res = await http.post(Uri.parse(phpurl), body: {

            "profileemail": email.toString(),
        });

        if (res.statusCode == 200) {

            //If successfull send data

            var data = convert.json.decode(res.body); //Decode to array

            if (data["error"]) {

                //If error querying data

                List error = [
                    {
                        "error": "true",
                        "favouriterestaurants": "[]"
                } //Send empty favourite restaurant ids list
            }
        }
    }
}
```

```
];

    return error;

} else {

    //If successful query

    List listdata = [data]; //Send list of restaurant ids back

    return listdata;

}

} else {

    List error = [

        {"error": "true", "favouriterestaurants": "[]"}

    ];

    return error;

}

} catch (e) {

    List<Map<String, String>> error = [

        {"error": "true", "favouriterestaurants": "[]"}

    ];

    return error;

}

}

@Override

Widget build(BuildContext context) {
```

```
return FutureBuilder<List>(

    //Get list of restaurants data from future (rebuild on change of
data)

    future: getRestaurants(),

    builder: ((context, snapshot) {

        List restaurantsdata =
            snapshot.data ?? [];
        //Data stored in restaurantsdata

        if (!snapshot.hasData) {

            //While no data received, show loading bar

            return const LinearProgressIndicator(
                color: Color(0xff4100C4),
                backgroundColor: Color.fromARGB(0, 235, 224, 255));
        } else {

            //If received data

            try {

                List restaurants = restaurantsdata[0]["restaurants"];

                return FutureBuilder<List>(
                    //Get favourites list from future

                    future: getFavourites(),

                    builder: ((context, snapshot) {
                        if (!snapshot.hasData) {

                            //While no data received, show loading bar

```

```
        return const LinearProgressIndicator(



            color: Color(0xff4100C4),



            backgroundColor: Color(0xffEBE0FF));



        } else {



            //If data received



            List restaurantFavourites = snapshot.data ?? [];



            return ListView.builder(



                //Show number of containers depending on number of



                restaurants



                physics:



                    const NeverScrollableScrollPhysics(), //Dont



                    allow scrolling (Done by main page)



                    scrollDirection: Axis.vertical,



                    shrinkWrap: true,



                    itemCount: restaurantsdata[0]["restaurants"]



                    .length, //For each restaurant



                    itemBuilder: (context, index) {



                        return Center(child:



                            LayoutBuilder(builder: (context, constraints)



{



                            if (restaurants.isEmpty) {



                                //If there are no restaurants show container



                                saying no restaurants




```

```
        return Padding(  
  
            padding: const EdgeInsets.only(  
  
                left: 20, right: 20, top: 10, bottom:  
10),  
  
            child: Container(  
  
                decoration: BoxDecoration(  
  
                    borderRadius: const  
BorderRadius.all(  
  
                        Radius.circular(20)),  
  
                    color: const Color(0xffffffff),  
  
                    boxShadow: [  
  
                        BoxShadow(  
  
                            color:  
  
Colors.black.withOpacity(0.1),  
  
                            spreadRadius: 2,  
  
                            blurRadius: 10,  
  
                            offset: const Offset(0,  
10), // changes position  
of shadow  
  
) ,  
] ),  
  
            child: Column(children: const [  
  
Padding(  
)
```

```

padding: EdgeInsets.all(30),

child: SizedBox(
width: double.infinity,
child: Center(
child: Text(
"No restaurants
found"))), //Display no restaurants text

)

] ));

} else {

// If there is restaurant

return InkWell(
//Create clickable container

onTap: () {

List restaurantinfodata =


restaurantsdata[0]["restaurants"][index];

Navigator.push(
context,
MaterialPageRoute(
builder:

(context) => //On tap, send
restaurant data associated with container to main page and go to that new
screen

```

```
        RestaurantMain(  
          resid:  
  
restaurantinfodata[0],  
          resname:  
  
restaurantinfodata[1],  
          reslogo:  
  
restaurantinfodata[2],  
          resbanner:  
  
restaurantinfodata[3],  
        )));  
},  
child: Container(  
  margin: const EdgeInsets.symmetric(  
    horizontal: 30, vertical: 20),  
  decoration: BoxDecoration(  
    borderRadius: const  
BorderRadius.all(  
    Radius.circular(10)),  
    color: Theme.of(context)  
      .colorScheme  
      .onSurface,  
  ),
```

```
) ,  
  
        child: Stack(  
  
            children: [  
  
                Container(  
  
                    //Container filled with  
restaurant banner  
  
                    width:  
  
MediaQuery.of(context).size.width,  
  
                    height: 150,  
  
                    decoration: BoxDecoration(  
  
                        borderRadius:  
  
                            const BorderRadius.only(  
  
                                topLeft:  
Radius.circular(10),  
  
                                topRight:  
Radius.circular(10),  
  
                    ) ,  
  
                    image: DecorationImage(  
  
                        fit: BoxFit.fitWidth,  
  
                        image: NetworkImage(  
  
                            restaurants[index][3]  
  
                            .toString()),  
  
                    ) ,  
            ]  
        )  
    ]  
);
```

```
) ,  
)  
),  
Align(  
    alignment:  
Alignment.topCenter,  
    child: Container(  
        margin: const  
EdgeInsets.only(  
            top: 110),  
            width: 70,  
            height: 70,  
            decoration: BoxDecoration(  
                shape: BoxShape.circle,  
                color: Theme.of(context)  
                    .colorScheme  
                    .onSurface,  
            ))),  
Align(  
    alignment:  
Alignment.topCenter,  
    child: Container(  
        margin: const  
EdgeInsets.only(  
            top: 115),
```

```
        width: 60,  
  
        height: 60,  
  
        decoration: BoxDecoration(  
  
            image: DecorationImage(  
  
                fit: BoxFit.cover,  
  
                image: NetworkImage(  
  
restaurants[index][2]  
  
.toString()),  
  
) ,  
  
shape: BoxShape.circle,  
  
color: Theme.of(context)  
  
.colorScheme  
  
.onSurface,  
  
)) ,  
  
Align(  
  
alignment: Alignment.topRight,  
  
child: InkWell(  
  
onTap: () {  
  
if  
(restaurantFavourites[0]  
  
["restaurantids"]  
  
.contains(
```

```
restaurants[index][0]

    .toString())) {
        favouriteRestaurant(
            restaurants[index][0]
                .toString(),
            "unfavourite");
    setState(() {
        getFavourites();
    });
} else {
    favouriteRestaurant(
        restaurants[index][0]
            .toString(),
            "favourite");
    setState(() {
        getFavourites();
    });
}
},
child: Container(
```

```
margin:  
        const  
EdgeInsets.only(  
    top: 10,  
    right: 10),  
    width: 45,  
    height: 45,  
    decoration:  
BoxDecoration(  
    shape:  
BoxShape.circle,  
    color:  
Theme.of(context)  
    .colorScheme  
    .onSurface  
  
.withOpacity(0.8),  
) ,  
child:  
restaurantFavourites[0]  
[  
"restaurantids"]  
.contains(  
restaurants[index]
```

```
[0]

    .toString()

    ?  
Icon(Icons.favorite,
        size:
            30, // : =
favourited

        color:
Theme.of(
    context)

    .colorScheme

    .secondary)
        : Icon(
            // : =
unfavourited

        Icons

    .favorite_border_outlined,
        size: 30,
        color:
Theme.of(
    context)
```

```
.colorScheme

.onBackground() ) ) ,
Align(
alignment:
Alignment.bottomLeft,
child: Padding(
padding: const
EdgeInsets.only(
top: 190,
left: 20,
right: 20),
child: Column(
children: [
Row(
mainAxisAlignment:
MainAxisAlignment
.spaceBetween,
children: [
Flexible(
child:
Container(
padding:
```

```
const  
EdgeInsets  
  
.only(  
  
right:  
  
13.0),  
  
child:  
Text(  
  
restaurants[  
  
index][1],  
  
overflow:  
  
TextOverflow.  
  
.ellipsis,  
  
style:  
Theme.of(  
  
context)  
  
.textTheme  
  
.headline4,  
  
))),
```

```
Row (  
  children: [  
    Icon (  
      Icons.star,  
      size: 20,  
      color:  
Theme.of(  
context)  
  
.primaryColor,  
),  
const  
SizedBox (  
  width: 5,  
) ,  
Text (  
"N/A",  
style:  
Theme.of(  
context)  
  
.textTheme  
  
.bodyText1,
```

```
)  
],  
) ,  
],  
) ,  
const SizedBox(height:  
10),  
Row(  
children: [  
Text("N/A miles  
away",  
style:  
Theme.of(  
context)  
.textTheme  
.bodyText1!  
.copyWith(  
fontSize:  
12)),  
const SizedBox(  
width: 5,  
) ,
```

```
Text(  
    "...",  
    style:  
  
Theme.of(context)  
  
.textTheme  
  
.bodyText1!  
  
.copyWith(  
  
fontSize:  
  
12),  
  
) ,  
  
const SizedBox(  
    width: 5,  
),  
  
Text("£N/A  
Delivery",  
    style:  
Theme.of(  
  
context)  
  
.textTheme
```

```
.bodyText1!  
  
.copyWith(  
  
fontSize:  
  
12)),  
  
],  
  
) ,  
  
const SizedBox(height:  
20),  
  
],  
  
) ) )  
  
],  
  
) ) ;  
  
}  
  
} ) ) ;  
  
} ) ;  
  
}  
  
} ) ,  
  
) ;  
  
} catch (e) {  
  
return Padding(  
  
padding: const EdgeInsets.only(  
top: 10, bottom: 10),  
child: Text(e.toString(),  
style: TextStyle(color: Colors.red)),  
);  
}  
  
}  
  
}
```

```
        left: 20, right: 20, top: 10, bottom: 10),  
  
        child: Container(  
  
            decoration: BoxDecoration(  
  
                borderRadius:  
  
                    const BorderRadius.all(Radius.circular(20)),  
  
                color: const Color(0xffffffff),  
  
                boxShadow: [  
  
                    BoxShadow(  
  
                        color: Colors.black.withOpacity(0.1),  
  
                        spreadRadius: 2,  
  
                        blurRadius: 10,  
  
                        offset: const Offset(  
  
                            0, 10), // changes position of shadow  
  
                    ),  
  
                ],  
  
                child: Column(children: [  
  
                    Padding(  
  
                        padding: const EdgeInsets.all(30),  
  
                        child: SizedBox(  
  
                            width: double.infinity,  
  
                            child: Center(  
  
                                child: Column(children: [  
  
                                    const Text("Error 400: Connection failed"),  
  
                                ],  
  
                            ),  
  
                        ),  
  
                    ),  
  
                ],  
  
            ),  
  
        ),  
  
    ),  
  
);
```

```
ElevatedButton(  
    onPressed: () {  
        setState(() {  
            getRestaurants();  
        });  
    },  
    child: const Text("Retry"))  
],),  
)  
];);  
}  
}  
},  
);  
}  
}  
}
```

Topbar.dart

```
import 'package:flutter/material.dart';  
  
class MainAppBar extends StatelessWidget implements PreferredSizeWidget {  
    final double height;
```

```
const MainAppBar({Key? key, required this.height}) : super(key: key);

@override

Widget build(BuildContext context) {

    return Container( //Container of height 120px

        height: 120,

        color: Theme.of(context).bottomNavigationBarTheme.backgroundColor,

        child: SafeArea(


            child: Row(


                mainAxisAlignment: MainAxisAlignment.spaceAround,


                crossAxisAlignment: CrossAxisAlignment.center,


                children: [


                    Icon( //Location icon

                        Icons.location_on_outlined,


                        color: Theme.of(context).textTheme.headline1?.color,


                    ),


                    Column(mainAxisAlignment: MainAxisAlignment.center,


children: [ //Column that will display the destination the food will be delivered to


                    Text(


                        "Location",


                        style: Theme.of(context)


                            .textTheme


                                .headline6

```

```
? .copyWith(fontWeight: FontWeight.w500),  
) ,  
  
const SizedBox(  
height: 2,  
) ,  
  
Text(  
"54 Main Street",  
style: Theme.of(context)  
    .textTheme  
    .bodyText2  
? .copyWith(fontWeight: FontWeight.w600),  
)  
]) ,  
  
Icon( //Cart icon  
Icons.shopping_cart_outlined,  
color: Theme.of(context).textTheme.headline1?.color,  
) ,  
]) ,  
);  
  
}  
  
@override  
Size get preferredSize => Size.fromHeight(height);
```

```
}
```

Main.dart

```
import 'package:alleat/screens/setupverification.dart';

import 'package:flutter/services.dart';

import 'package:alleat/theme/theme.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:alleat/theme/theme.dart' as globals;

void main() {

    //Start App

    runApp(const MyApp());
}

class MyApp extends StatelessWidget {

    const MyApp({super.key});

    Future getTheme() async {

        final prefs = await SharedPreferences.getInstance();

        final int? appTheme = prefs.getInt('appTheme');

        globals.appthemepreference = appTheme;

    }
}
```

```
@override

Widget build(BuildContext context) {
    SystemChrome.setPreferredOrientations([
        //Force portrait mode
        DeviceOrientation.portraitUp,
        DeviceOrientation.portraitDown,
    ]);
    WidgetsFlutterBinding.ensureInitialized();
    getTheme();
    switch (globals.appthemepreference) {
        case 1:
            return MaterialApp(
                //Create main app
                title: 'AllEat.',
                themeMode: ThemeMode.light,
                theme: ThemeClass.lightTheme,
                home:
                    const SetupWrapper(), //SetupWrapper(), //Check if setup is
                    complete for app (reference)
            );
        case 2:
            return MaterialApp(
```

```
//Create main app

    title: 'AllEat.',

    themeMode: ThemeMode.dark,

    theme: ThemeClass.darkTheme,

    home:

        const SetupWrapper(), //SetupWrapper(), //Check if setup is
complete for app (reference)

    );

}

return MaterialApp(
    //Create main app

    title: 'AllEat.',

    themeMode: ThemeMode.system,

    theme: ThemeClass.lightTheme,

    darkTheme: ThemeClass.darkTheme,

    home:

        const SetupWrapper(), //SetupWrapper(), //Check if setup is
complete for app (reference)

    );

}
```

}

}

Files - Server

Favouriterestaurant.php

```
<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "FO7&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "success" ] = false;

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["action"]) && ($_POST["profileemail"]) &&
isset($_POST["restaurantid"]);

if($val){
    $email = $_POST["profileemail"]; //grabing the data from headers
    $restaurantid = $_POST["restaurantid"];
    $action = $_POST["action"];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT
1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row[ 'ProfileID' ];
            }
            // Free result set
            mysqli_free_result($result);
            if ($action == "unfavourite"){
                $sqlunfavourite = "DELETE FROM favrestaurant WHERE profileid
= $ProfileID AND restaurantid = $restaurantid";
                if ($link->query($sqlunfavourite) === TRUE) {
                    $return[ "success" ] = true;
                } else{
                    $return[ "error" ] = true;
                }
            }
        }
    }
}
```

```

    }
    else if ($action == "favourite"){
        $sqlfavourite = "INSERT INTO favrestaurant (profileid,
restaurantid) VALUES ($ProfileID, $restaurantid)";
        if ($link->query($sqlfavourite) === TRUE) {
            $return["success"] = true;
        } else{
            $return["error"] = true;
        }

    }
    else{
        $return["error"] = true;
        $return["message"] = "action unspecified";
    }
}

else{
    $return["error"] = true;
    $return["message"] = "duplicate found";
}
else{
    $return["error"] = true;
    $return["message"] = "failed";
}
}
else{$return["error"] = true;
    $return["message"] = "data not specified;"}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

Favouriterestaurantdata.php

```
<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurants" ] = array();
$return[ "temp" ] = array();
$restaurantids = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileemail"]);

if($val){
    $email = $_POST[ "profileemail" ];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row[ 'ProfileID' ];
            }
            // Free result set
            mysqli_free_result($result);
        }
    } else{
        $return[ "error" ] = true;
        $return[ "message" ] = "duplicate found";
    }
    $sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
profileid = $ProfileID";
    if($result2 = mysqli_query($link, $sqlfavrestaurant)){
        while($row2 = mysqli_fetch_assoc($result2)) {
            array_push($restaurantids, $row2[ "restaurantid" ]);
        }
        mysqli_free_result($result2);
    }
}
```

```

    }

    foreach ($restaurantids as &$value) {
        $sqlrestaurantinfo = "SELECT res.restaurantid,
res.restaurantname, res.restaurantlogo, res.restaurantbanner FROM
restaurant res WHERE res.restaurantid = $value";
        $return["temp"] = $sqlrestaurantinfo;
        if($result3 = mysqli_query($link, $sqlrestaurantinfo)){
            while($row3 = mysqli_fetch_assoc($result3)) {
                array_push($return["restaurants"],
[$row3["restaurantid"], $row3["restaurantname"], $row3["restaurantlogo"],
$row3["restaurantbanner"]]);
            }
            mysqli_free_result($result3);
        }
    }

}

else{
    $return["error"] = true;
    $return["message"] = "failed";
}

}

else{
    $return["error"] = true;
    $return["message"] = "data not specified 2";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

Favouriterestaurantlist.php

```
<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurantids" ] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileemail"]);

if($val){
    $email = $_POST[ "profileemail" ];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row[ 'ProfileID' ];
            }
            // Free result set
            mysqli_free_result($result);
        }
    } else{
        $return[ "error" ] = true;
        $return[ "message" ] = "duplicate found";
    }
}

else{
    $return[ "error" ] = true;
    $return[ "message" ] = "failed";
}

$sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
```

```

profileid = $ProfileID";
        if($result2 = mysqli_query($link, $sqlfavrestaurant)){
            while($row2 = mysqli_fetch_assoc($result2)) {
                array_push($return["restaurantids"],
$row2["restaurantid"]);
            }
            mysqli_free_result($result2);
        }

    }
else{
    $return["error"] = true;
    $return["message"] = "data not specified";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

Login.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";
$return["exists"] = false;
$return["profile"] = "";
$return["temp1"] = "";
$return["temp2"] = "";

```

```

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);
//connecting to database server

$val = isset($_POST["email"]) && isset($_POST["password"]);

if($val){
    //checking if there is POST data
    $email = $_POST["email"];
    $password = $_POST["password"];
    $return["temp1"] = $email;
    $return["temp2"] = $password;

    $secret_iv = '5fd0b31f582beb1f';
    $secret_key = 'e16a3f356b2366c1';
    $cipher = "AES-128-CBC";
    $decryptedString = openssl_decrypt ($password, $cipher,
$secret_key, 0, $secret_iv);

    $sql = "SELECT ProfileID, FirstName, LastName, Email, Password
FROM profile WHERE Email = '$email'";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileIDR = $row['ProfileID'];
                $FirstNameR = $row['FirstName'];
                $LastNameR = $row['LastName'];
                $EmailR = $row['Email'];
                $PasswordR = $row['Password'];
            }
            // Free result set
            mysqli_free_result($result);
            $ispasswordsame = password_verify($decryptedString,
$PasswordR);
            if ($ispasswordsame == true){
                $PasswordREncrypted = openssl_encrypt($PasswordR,
$cipher, $secret_key, 0, $secret_iv);
                $return["exists"] = true;
                $return["profile"] = array($ProfileIDR,
$FirstNameR, $LastNameR, $EmailR, $PasswordREncrypted);
            }
        }
    }
}

```

```

        $return["error"] = false;
        $return["message"] = "Email or password incorrect";
        $return["exists"] = false;
        $return["profile"] = null;
    }
} else{
    $return["error"] = false;
    $return["message"] = "No matching profiles";
    $return["exists"] = false;
    $return["profile"] = null;
}
} else{
    $return["error"] = true;
    $return["message"] = "Failed profile search";
}

}else{
    $return["error"] = true;
    $return["message"] = 'Send all parameters.';
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string

?>

```

Register.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";

```

```

$return[ "hash" ] = "";
$return[ "profile" ] = "";
$return[ "exist" ] = false;
$return[ "temp" ] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);
//connecting to database server

$val = isset($_POST["firstname"]) && isset($_POST["lastname"])
      && isset($_POST["email"]) && isset($_POST["password"]);

if($val){
    //checking if there is POST data
    $firstname = $_POST[ "firstname" ]; //grabing the data from headers
    $lastname = $_POST[ "lastname" ];
    $email = $_POST[ "email" ];
    $password = $_POST[ "password" ];

    $secret_iv = '5fd0b31f582beb1f';
    $secret_key = 'e16a3f356b2366c1';
    $cipher = "AES-128-CBC";
    $decryptedString = openssl_decrypt ($password, $cipher,
$secret_key, 0, $secret_iv);
    $hashpassword = password_hash($decryptedString, PASSWORD_DEFAULT);
    $base64hash = base64_encode($hashpassword);
    $encryptedhashedpassword = openssl_encrypt ($base64hash, $cipher,
$secret_key, 0, $secret_iv);
    $return[ "hash" ] = $encryptedhashedpassword; //Encrypt hashed
password to be sent back

    if($return[ "error" ] == false){
        $firstname = mysqli_real_escape_string($link, $firstname);
        $lastname = mysqli_real_escape_string($link, $lastname);
        $email = mysqli_real_escape_string($link, $email);
        $hashpassword = mysqli_real_escape_string($link,
$hashpassword);
        //escape inverted comma query conflict from string

        $sqlverify = "SELECT * FROM profile WHERE Email = '$email'";
        $res = mysqli_query($link, $sqlverify);
        if (mysqli_num_rows($result) > 0) {
            $return[ "exist" ] = true;
        }
    }
}

```

```

    }
else{
    $sql = "INSERT INTO profile SET
        FirstName = '$firstname',
        LastName = '$lastname',
        Email = '$email',
        Password = '$hashpassword'";
    $res = mysqli_query($link, $sql);

    $sqlget = "SELECT ProfileID, FirstName, LastName, Email,
    Password FROM profile WHERE Email = '$email'";
    if($result = mysqli_query($link, $sqlget)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileIDR = $row['ProfileID'];
                $FirstNameR = $row['FirstName'];
                $LastNameR = $row['LastName'];
                $EmailR = $row['Email'];
                $PasswordR = $row['Password'];
            }
            // Free result set
            mysqli_free_result($result);
            $ispasswordsame = password_verify($decryptedString,
$PasswordR);
            if ($ispasswordsame == true){
                $PasswordREncrypted = openssl_encrypt($PasswordR,
$cipher, $secret_key, 0, $secret_iv);
                $return["exists"] = true;
                $return["profile"] = array($ProfileIDR,
$FirstNameR, $LastNameR, $EmailR, $PasswordREncrypted);
            }
            else{
                $return["error"] = false;
                $return["message"] = "Email or password incorrect";
                $return["exists"] = false;
                $return["profile"] = null;
            }
        } else{
            $return["error"] = false;
            $return["message"] = "No matching profiles";
            $return["exists"] = false;
            $return["profile"] = null;
        }
    }
}

```

```

        }
    } else{
        $return["error"] = true;
        $return["message"] = "Failed profile search";
    }

    if($res){
        //write success
    }else{
        $return["error"] = true;
        $return["message"] = "Database error";
    }
}
}

else{
    $return["error"] = true;
    $return["message"] = 'Send all parameters.';
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

Restaurantlist.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";
$return["restaurants"] = array();

```

```

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["sort"]);

if($val){
    $sort = $_POST["sort"];

    if ($sort == "id"){

        $sql = "SELECT res.restaurantid, res.restaurantname,
res.restaurantlogo, res.restaurantbanner FROM restaurant res LIMIT 20";
        if($result = mysqli_query($link, $sql)){
            while($row = mysqli_fetch_assoc($result)) {
                array_push($return["restaurants"],
[$row["restaurantid"], $row["restaurantname"], $row["restaurantlogo"],
$row["restaurantbanner"]]);
            }
            mysqli_free_result($result);
        }
    }
    else{
        $return["error"] = true;
        $return["message"] = "sort method not specified";
    }
}
else{
    $return["error"] = true;
    $return["message"] = "data not specified";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

Restaurantmenucategories.php

```
<?php
```

```

$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "FO7&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurantcategories" ] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["restaurantid"]);

if($val){
    $resid = $_POST["restaurantid"];

    $sql = "SELECT categoryname, categoryid FROM menucategories WHERE
restaurantid = '$resid' ORDER BY viewingorder ASC";
    if($result = mysqli_query($link, $sql)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($return[ "restaurantcategories" ],
[$row[ "categoryname" ], $row[ "categoryid" ]]);
        }
        mysqli_free_result($result);
    }
    else{
        $return[ "error" ] = true;
        $return[ "message" ] = "restaurant not sepcified";
    }
}
else{
    $return[ "error" ] = true;
    $return[ "message" ] = "data not specified";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string

```

```
?>
```

Restaurantmenuitems.php

```
<?php  
$db = "u352759660_m3uFj"; //database name  
$dbuser = "u352759660_GDfIr"; //database username  
$dbpassword = "F07&Ruvr;0G"; //database password  
$dbhost = "localhost"; //database host  
  
$return[ "error" ] = false;  
$return[ "message" ] = "";  
$return[ "menuitems" ] = array();  
  
$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);  
  
$val = isset($_POST["categoryid"]);  
  
if($val){  
    $rescategory = $_POST["categoryid"];  
  
    $sql = "SELECT itemid, foodcategory, subfoodcategory, itemname,  
description, price, itemimage FROM item WHERE categoryid = '$rescategory'  
ORDER BY itemid ASC";  
    if($result = mysqli_query($link, $sql)){  
        while($row = mysqli_fetch_assoc($result)) {  
            array_push($return[ "menuitems" ], [$row[ "itemid" ],  
$row[ "foodcategory" ], $row[ "subfoodcategory" ], $row[ "itemname" ],  
$row[ "description" ], $row[ "price" ], $row[ "itemimage" ]]);  
        }  
        mysqli_free_result($result);  
    }  
    else{  
        $return[ "error" ] = true;  
        $return[ "message" ] = "category not sepcified";  
    }  
}  
else{  
    $return[ "error" ] = true;  
    $return[ "message" ] = "data not specified";  
}
```

```
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
    // tell browser that its a json data
echo json_encode($return);
    //converting array to JSON string
?>
```

Summary

Prototype 1B redesigned the entire app using a more user-friendly experience by both simplifying the number of different widgets and reducing the complexity of the code. It was done by using repeating code blocks in a polymorphic form.

Evaluation

Over the course of Prototype 1B I found many issues including data management sending data from one file to another and many issues with formatting. With many of the problems encountered, the data was being overflowed into other elements causing visual inaccuracies and incorrect button placements but by using the overflow function and the expanded function, I was able to mitigate these errors.

Prototype 2

Design

Prototype 2 is an extension of prototype 1, building off the main structure, and getting it closer to concept 3 shown previously in concept art 4. Previously I found it difficult to conceptualise how prototype 1 would look, only having the final concept to base it off.

Success Criteria

For this prototype, I aim to fulfil the following success criteria

- 1.3* - The database should be in third-normal form and normalised
- 3.1 - The user should be able to sort restaurants using a button
- 3.2* - When sorting restaurants, it should allow the user to sort by recommended, rating or distance
- 3.4 - The user should be able to filter restaurants using a button
- 3.5* - When filtering restaurants, it should allow the user to filter by price range, delivery, pickup, delivery price, new
- 3.7 - Before filtering, there should be a button to confirm that they would like to filter the restaurant.
- 3.8 - When the user has selected some restaurant filters, the user should have a button they can click to clear the filters and have it show all the restaurants
- 3.9 - When the user has selected a restaurant filter, they should be able to unselect a filter and change the settings

**Will be revisited in the future for when more features are added*

Concept Art

Filters & Sorting

When a user clicks on a filter icon, they will be brought to this page where they can select the options they wish. On going back, it will save the options and the list of restaurants will search using the options selected.

< Back

Sort



Recommended (Default)



Top Rated



Popular



Distance

Filter

Clear



Show only Favourites

Price Range

£

££

£££

££££

Max. Delivery Fee

£1

£2

£3

£4+



Minimum Order Price

None

£10

£20

£30

£40+



Item Customisation

A user should be able to select how they would like their food served. There are various ways it can be customised, including select, add and remove.



PAPA JOHN'S

Cheese & Tomato

Pizza • American

The classic. Full of flavour, with our tomato sauce and mozzarella. Enjoy it pure and simple or add extra toppings to create your own masterpiece.

£ 14.99 🕒 N/A

Select your crust. Required

Original

Authentic Thin Crust

Stuffed Crust +£2.99

Defaults Optional

- Base Pizza Sauce
- Mozzarella Cheese
- Dip Special Garlic

Optionals Optional

- + Anchovies +£1.60
- + Bacon +£1.60
- + Base Pizza Sauce +£1.60
- + Barbecue Drizzle +£1.60
- + Black Olives +£1.60
- + Chargrilled Chicken +£1.60
- + Fresh Tomatoes +£1.60

1

Add to Order £16.99

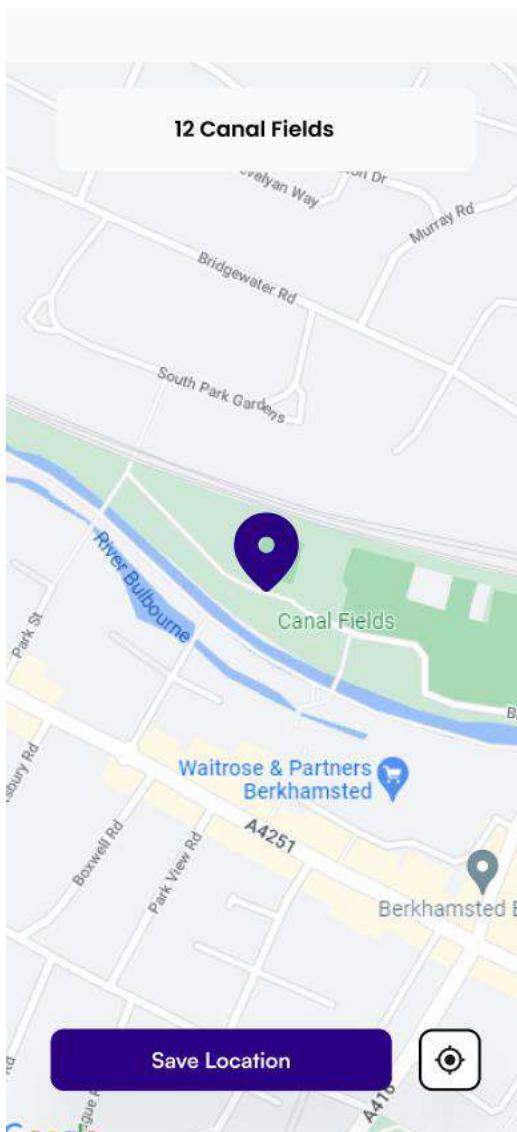
Home Chat Search Profile

Location

The location is very important to get correct so that the food gets delivered to the correct location so instead of doing a field based location, a map and pin will be used where the map will translate the location into both longitude and latitude values and translate that to a street address when showing it in text form. While Prototype 1 had formatting requirements, it could still be filled in incorrectly.

I will try to use the following package to build this:

https://pub.dev/packages/map_picker



App Flowchart

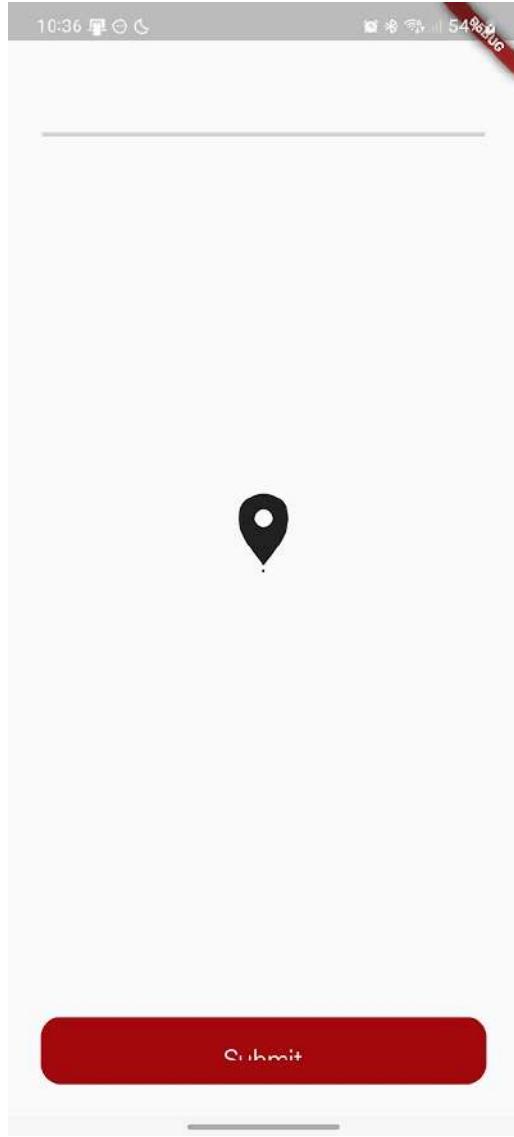
FLOWCHART TOO LARGE - FILE IN FOLDER - "*Prototype 2 flowchat.png*"

Development

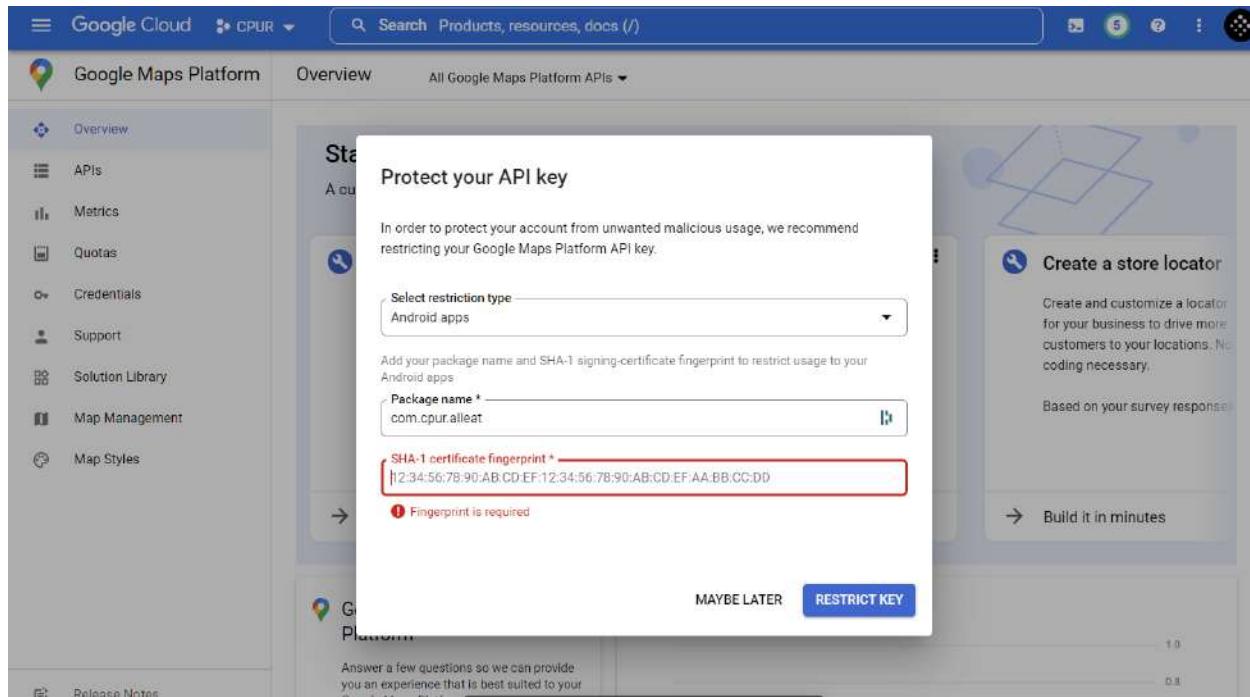
Location

To build the location selection, I used the map_picker package and started by using the example code provided as the building block for the screen.

After trying to do this I found that an API key was needed in order for it to work.



What I then found was that a fingerprint was needed but I was unable to do this since it was not linked to the play store so this was ignored. If I had time in the future, I would need to add this to prevent malicious use of the Google Maps API



After getting the API Key and putting it into the Android XML file, the map was working but when the user went to a location not defined by a street name, it would crash due to it not having a name. To counter this, I used a try statement to attempt to get the location on the map and if it fails, display Unknown Location.

```
//get address name from camera position
try {
    List<Placemark> placemarks = await placemarkFromCoordinates(
        cameraPosition.target.latitude,
        cameraPosition.target.longitude,
    );

    // update the ui with the address
    textController.text =
        '${placemark.first.name}, ${placemark.first.administrativeArea}, ${placemark.firs
    } catch (e) {
        textController.text = "Unknown Location.";
    }
},
```



The next issue to solve was getting to the default location to place the user. What I will do is first check the location saved in the shared preferences and if there is none, use the current location but if something goes wrong or the user denies using their location, go to London, England.

To do this I used:

```
Future<CameraPosition> getCameraPosition() async {  
  
    try {  
  
        final prefs = await SharedPreferences.getInstance();  
  
        final double? savedLocationLat = prefs.getDouble('locationlat');  
  
        final double? savedLocationLng = prefs.getDouble('locationlat');
```

```
if (savedLocationLat != null && savedLocationLng != null) {  
  
    print("Saved location not null");  
  
    return CameraPosition(  
  
        target: LatLng(savedLocationLat, savedLocationLng), zoom: 20);  
  
} else {  
  
    print("Getting current location");  
  
    bool serviceEnabled;  
  
    LocationPermission permission;  
  
  
    // Test if location services are enabled.  
  
    serviceEnabled = await Geolocator.isLocationServiceEnabled();  
  
    if (!serviceEnabled) {  
  
        return Future.error('Location services are disabled.');  
  
    }  
  
  
    // Check if the location is denied  
  
    permission = await Geolocator.checkPermission();  
  
    if (permission == LocationPermission.denied) {  
  
        permission = await Geolocator.requestPermission();  
  
        if (permission == LocationPermission.denied) {  
  
            return Future.error('Location permissions are denied');  
  
        }  
    }  
}
```

```
}

if (permission == LocationPermission.deniedForever) {

    // Permissions are denied forever, handle appropriately.

    return Future.error('Location permissions are permanently denied.');
}

// Access the current position of the device

Position locationDevice = await Geolocator.getCurrentPosition();

return CameraPosition(
    target: LatLng(locationDevice.latitude, locationDevice.longitude),
    zoom: 20);

}

} catch (e) {

    print("Caught error");

    return const CameraPosition(
        target: LatLng(51.509865, -0.118092),
        zoom: 18,
    );
}

}
```

While it did catch any errors if they occurred and cause it not to crash, it didn't display what was going wrong so I had it print the error.

```
    catch (e) {
      print("'$e");
    }
  }
}
```

```
I/ViewRootImpl@e/d289f[MainActivity]( 6818): ViewPostIme pointer 1
I/flutter ( 6818): Getting current location
I/flutter ( 6818): No location permissions are defined in the manifest. Make sure at least ACCESS_FINE_LOCATION or ACCESS_COARSE_LOCATION
I/flutter ( 6818): Caught error
D/MapsInitializer( 6818): preferredRenderer: null
D/zccb   ( 6818): preferredRenderer: null
I/Google Maps Android API( 6818): Google Play services package version: 224312045
I/Google Maps Android API( 6818): Google Play services maps renderer version(legacy): 203115000
I/PlatformViewController( 6818): Using hybrid composition for platform view: 0
E/GoogleMapController( 6818): Cannot enable Mylocation layer as location permissions are not granted
E/Surface ( 6818): freeAllBuffers: 1 buffers were freed while being dequeued!
④ I/Counters( 6818): exceeded sample count in FrameTime
```

This error meant I forgot to add the permissions to the app so they had to be added.

After adding this code, I got another error

```
W/zzcb   (14615): preferredRenderer: null
I/zccb   (14615): Making Creator dynamically
② W/ziparchive(14615): Unable to open '/data/user_de/0/com.google.android.gms/app_chimera/m/0000006f/DynamiteLoader.dm': No such file or directory
I/DynamiteModule(14615): Considering local module com.google.android.gms.maps_dynamite:0 and remote module com.google.android.gms.maps_dynamite:203115000
I/DynamiteModule(14615): Selected remote version of com.google.android.gms.maps_dynamite, version >= 203115000
V/DynamiteModule(14615): Dynamite loader version >= z, using loadModuleNoCrashUtils
W/System  (14615): Classloader referenced unknown path:
② W/ziparchive(14615): unable to open '/data/user_de/0/com.google.android.gms/app_chimera/m/00000073/MapsDynamite.dm': No such file or directory
I/Google Maps Android API(14615): Google Play services client version: 12451000
D/CompatibilityChangeReporter(14615): Compat change id reported: 183155436; UID 10471; state: DISABLED
I/Google Maps Android API(14615): Google Play services package version: 224312045
I/Google Maps Android API(14615): Google Play services maps renderer version(legacy): 203115000
D/MapsInitializer(14615): loadedRenderer: LEGACY
D/zccb   (14615): preferredRenderer: null
I/Google Maps Android API(14615): Google Play services package version: 224312045
I/Google Maps Android API(14615): Google Play services maps renderer version(legacy): 203115000
D/CompatibilityChangeReporter(14615): Compat change id reported: 210923482; UID 10471; state: DISABLED
D/CompatibilityChangeReporter(14615): Compat change id reported: 37756858; UID 10471; state: ENABLED
D/CompatibilityChangeReporter(14615): Compat change id reported: 171228096; UID 10471; state: ENABLED
I/PlatformViewController(14615): using hybrid composition for platform view: 0
I/bb    (14615): Successfully registered with Phenotype.
E/Surface (14615): freeAllBuffers: 1 buffers were freed while being dequeued!
D/TrafficStats(14615): tagSocket(123) with statsTag-0x30001101, statsUid-1
W/DynamiteModule(14615): Local module descriptor class for com.google.android.gms.googlecertificates not found.
I/DynamiteModule(14615): Considering local module com.google.android.gms.googlecertificates:0 and remote module com.google.android.gms.googlecertificates:7
I/DynamiteModule(14615): Selected remote version of com.google.android.gms.googlecertificates, version >= 7
② W/ziparchive(14615): unable to open '/data/user_de/0/com.google.android.gms/app_chimera/m/00000072/googlecertificates.dm': No such file or directory
```

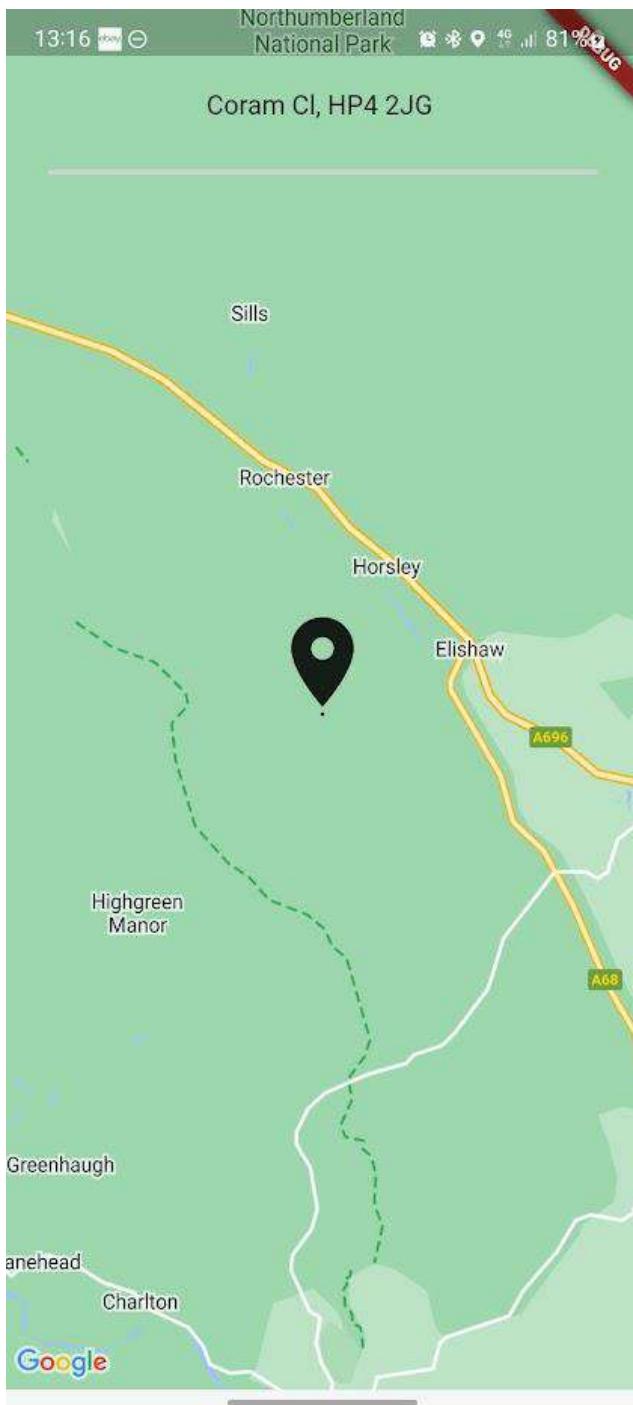
This was happening because the data was not compiled correctly and after an IDE reload, it was working.

While testing the map, I encountered this error:

```
invalid pointerIndex 1 for MotionEvent { action=POINTER_UP(1), id[0]=0, x[0]=624.111,
y[0]=2059.57, eventTime=43256835000000, downTime=43256541000000, deviceId=5,
source=TOUCHSCREEN, displayId=0, eventId=-268711367}
```

To fix this, I had to set a default position to put the map while it is determining the new position.

After implementing the new code to have an adaptive location depending on if a location was saved, another bug arose, around the placemark not changing.



To get it to update, I used a Future event to update the text

The next thing to fix was to remove the user's ability to rotate and tilt the camera, which by using some variables, fixed it.

```
mapType: mapType.normal,  
rotateGesturesEnabled: false,  
tiltGesturesEnabled: false,  
// camera position
```

Since the basic layout was in place, to get the look in the concept art, I used containers positioned at the bottom with InkWells to make them clickable.



To make the current location button work, I grabbed the location of the user and replaced the location that the user is currently looking at with the coordinates of the phone.

While trying to update the location, although I updated the location of the camera, it would not update it visually with only the name of where it is located changing. The reason it was not working was because I was going through the Futurebuilder twice instead of once causing there to be an error where the data was already final. To fix the code, I used the getCurrentLocation Future and returned back the data to replace the previous location. This would prevent full rebuilds.

```
    width: 20,
),
// SizedBox
InkWell(
  onTap: () async {
    // Get the current location of the device
    List currentLocation = await getCurrentLocation();

    // Move the camera to the current location
    final GoogleMapController controller =
      await _controller.future;
    controller
      .animateCamera(CameraUpdate.newCameraPosition(
        CameraPosition(
          target: LatLng(
            currentLocation[0], currentLocation[1]), // LatLng
            zoom: 18,
          ),
        ), // CameraPosition
      )));
},
child: Container(
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(10),
```

In order to save the location, it will save both the address plaintext and the latitude and longitude. This will mean that extra processing doesn't need to happen when displaying the current location. When storing the location in databases, they will be saved as latitude and longitude coordinates since it both takes less space and is more accurate.

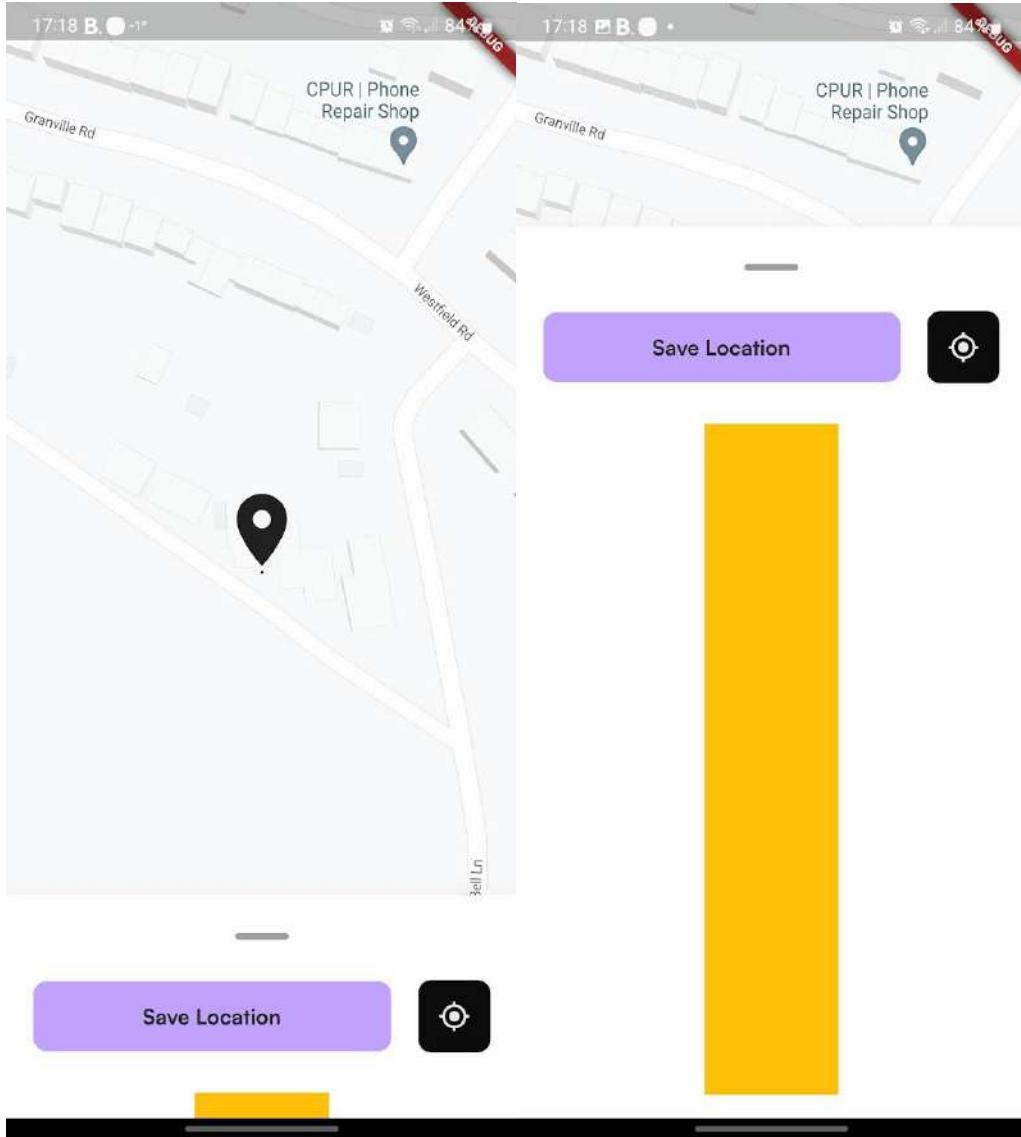
These details are stored in the following variables

```
onTap: () {
  print(cameraPosition.target.latitude);
  print(cameraPosition.target.longitude);
  print(textController.text);
},
```

While testing moving around on the map I found that some remote places or private streets do not display the correct information so to fix this, I enabled editing of the location.

As well, I found that the design of having the name at the top was unnecessary since now it could be edited so I added a rounded background overlay to the bottom which had all the buttons and text together that could be edited. To do this, i used a DraggableScrollableSheet which I passed the scrolling to. Since I had never done this before I used the following tutorial to help me: <https://www.youtube.com/watch?v=JiSsS1Xj5uQ>

To start with, I moved the bottom buttons to the bottom draggable container to ensure that it was working. I added a max height to be 0.8 to ensure that it would be easy to know where to get back to the map.



(Yellow container to test content height)

```
DraggableScrollableSheet(  
  
    initialChildSize: 0.2,  
  
    minChildSize: 0.2,  
  
    maxChildSize: 0.8,  
  
    builder: ((context, scrollController) {  
  
        return Container(  
    }  
}
```

```
decoration: const BoxDecoration(  
    color: Colors.white,  
    boxShadow: [  
        BoxShadow(  
            spreadRadius: 2,  
            blurRadius: 10,  
            color: Color.fromARGB(8, 0, 0, 0))  
    ],  
    child: ListView.builder(  
        controller: scrollController,  
        itemCount: 1,  
        itemBuilder: ((context, index) {  
            return Column(  
                children: [  
                    Container(  
                        width: 40,  
                        height: 5,  
                        decoration: BoxDecoration(  
                            color: Colors.grey,  
                            borderRadius:  
                                BorderRadius.circular(5)),  
                        Padding(  
                    ),  
                ],  
            );  
        }),  
    ),  
);
```

```
padding: const EdgeInsets.symmetric(
    vertical: 30, horizontal: 20),
child: Row(
  children: [
    Expanded(
      child: InkWell(
        onTap: () {
          print(cameraPosition
            .target.latitude);
          print(cameraPosition
            .target.longitude);
          print(textController.text);
        },
        child: Container(
          decoration: BoxDecoration(
            borderRadius:
              BorderRadius.circular(10),
            color: Theme.of(context)
              .primaryColor),
          padding: const EdgeInsets
            .symmetric(
```

```
        vertical: 15,  
  
        horizontal: 30),  
  
        child: Text(  
  
            "Save Location",  
  
            textAlign:  
  
                TextAlign.center,  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline6!  
  
                .copyWith(  
  
                    color: Theme.of(  
  
                        context)  
  
                    .colorScheme  
  
                    .onSurface),  
  
(  
  
    ))),  
  
const SizedBox(  
  
    width: 20,  
  
(  
  
InkWell(  
  
    onTap: () async {  
  
        // Get the current location of the  
device
```

```
        List currentLocation =  
  
            await getCurrentLocation();  
  
            // Move the camera to the current  
location  
  
            final GoogleMapController  
  
            controller =  
  
            await _controller.future;  
  
            controller.animateCamera(  
  
                CameraUpdate  
  
                .newCameraPosition(  
  
                    CameraPosition(  
  
                        target: LatLng(  
  
                            currentLocation[0],  
  
                            currentLocation[1]),  
  
                        zoom: 18,  
  
                    ),  
  
                )),  
  
            );  
  
        },  
  
        child: Container(  
  
            decoration: BoxDecoration(  
  
                borderRadius:  
  
                BorderRadius.circular(  
  
                    10),
```

```
        color: Theme.of(context)

        .backgroundColor),

padding:

const EdgeInsets.all(15),

child: Icon(
Icons.my_location,
color: Theme.of(context)

.textTheme

.headline1!

.color,

)),

)

],

)),

Container(
height: 500,
color: Colors.amber,
width: 100,
)

],
);

}));
```

}),),

The next thing I added was a text container which was editable. I made it display not just the house number, street and town but now by having multiple text boxes, I added a Postcode and a place to add apartment building or other information.

```
Future<void> getPlacemark() async {
try {
List<Placemark> placemarks = await placemarkFromCoordinates(
    cameraPosition.target.latitude,
    cameraPosition.target.longitude,
);
addresslineone = TextEditingController(text: placemarks.first.name);
addresslinetwo = TextEditingController(text: placemarks.first.street);
postcode = TextEditingController(text: placemarks.first.postalCode);
city = TextEditingController(
    text:
        "${placemarks.first.subAdministrativeArea}, ${placemarks.first.administrativeArea}");
} catch (e) {
textController.text = "";
}
}
```



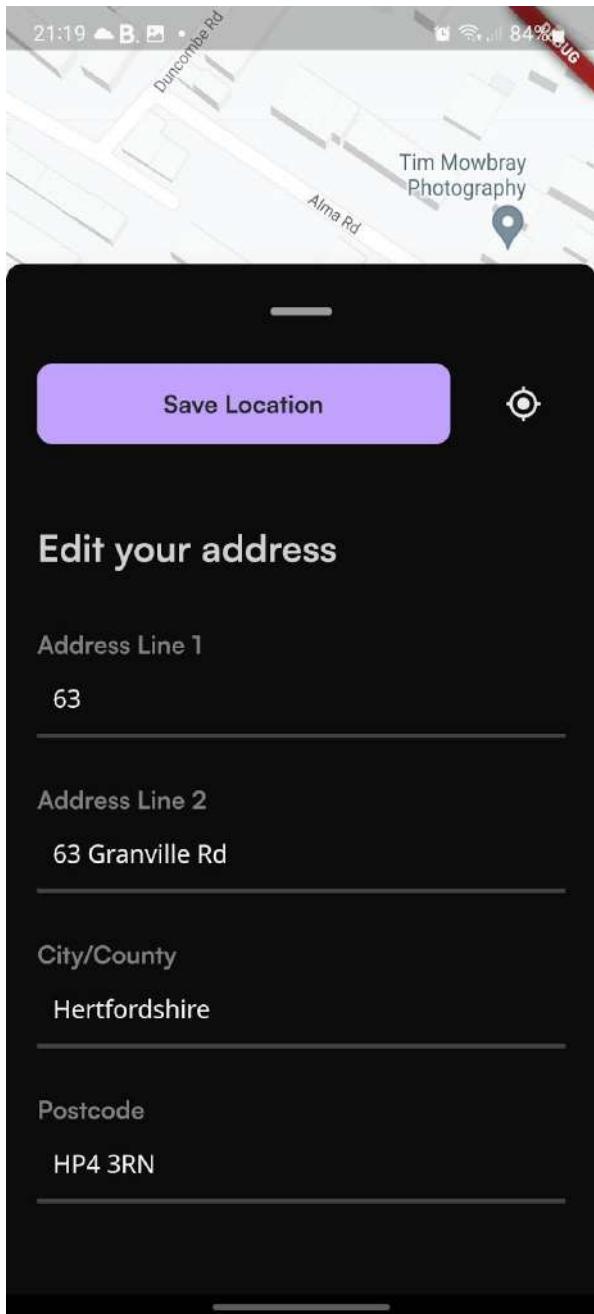
61

61 Granville Rd

Hertfordshire, England

HP4 3RN

After adding some decoration to it. I got the following:



```
DraggableScrollableSheet(  
  initialChildSize: 0.2,  
  snap: true,  
  minChildSize: 0.2,  
  maxChildSize: 0.8,
```

```
builder: ((context, scrollController) {  
  return Container(  
    decoration: BoxDecoration(  
      borderRadius: const BorderRadius.vertical(  
        top: Radius.circular(10)),  
      color: Theme.of(context).backgroundColor,  
      boxShadow: const [  
        BoxShadow(  
          spreadRadius: 2,  
          blurRadius: 10,  
          color: Color.fromARGB(8, 0, 0, 0))  
      ]),  
    child: ListView.builder(  
      controller: scrollController,  
      itemCount: 1,  
      itemBuilder: ((context, index) {  
        return Column(  
          children: [  
            Container(  
              width: 40,  
              height: 5,  
              decoration: BoxDecoration(  
               
```

```
        color: Colors.grey,  
        borderRadius:  
            BorderRadius.circular(5)),  
        Padding(  
            padding: const EdgeInsets.symmetric(  
                vertical: 30, horizontal: 20),  
            child: Row(  
                children: [  
                    Expanded(  
                        child: InkWell(  
                            onTap: (() {  
                                print(cameraPosition  
                                    .target.latitude);  
                                print(cameraPosition  
                                    .target.longitude);  
                                print(textController.text);  
                            }),  
                            child: Container(  
                                decoration: BoxDecoration(  
                                    borderRadius:  
                                        BorderRadius  
                                            .circular(10),
```

```
        color: Theme.of(context)

                .primaryColor),

        padding: const EdgeInsets

                .symmetric(
                    vertical: 15,
                    horizontal: 30),

        child: Text(
            "Save Location",
            textAlign:
                TextAlign.center,
            style: Theme.of(context)

                .textTheme
                .headline6!
                .copyWith(
                    color: Theme.of(
                        context)
                    .colorScheme
                    .onSurface),
            ),
        ))),
    const SizedBox(
        width: 20,
```

```
        ),  
  
        InkWell(  
  
            onTap: () async {  
  
                // Get the current location of the  
device  
  
                List currentLocation =  
  
                    await getCurrentLocation();  
  
                // Move the camera to the current  
location  
  
                final GoogleMapController  
  
                controller =  
  
                    await _controller.future;  
  
                controller.animateCamera(  
  
                    CameraUpdate  
  
                        .newCameraPosition(  
  
                            CameraPosition(  
  
                                target: LatLng(  
  
                                    currentLocation[0],  
  
                                    currentLocation[1]),  
  
                                zoom: 18,  
  
                            ),  
  
                ));  
  
            },  
        ),
```

```
        child: Container(  
  
            decoration: BoxDecoration(  
  
                borderRadius:  
  
                    BorderRadius.circular(  
  
                        10),  
  
                color: Theme.of(context)  
  
                    .backgroundColor),  
  
            padding:  
  
                const EdgeInsets.all(15),  
  
            child: Icon(  
  
                Icons.my_location,  
  
                color: Theme.of(context)  
  
                    .textTheme  
  
                    .headline1!  
  
                    .color,  
  
            )),  
  
        ),  
  
    ],  
  
)),  
  
Container(  
  
    padding: const EdgeInsets.all(20),  
  
    alignment: Alignment.bottomLeft,
```

```
        child: Column(  
  
            mainAxisAlignment:  
                MainAxisAlignment.start,  
  
            children: [  
  
                Text("Edit your address",  
  
                    style: Theme.of(context)  
  
                        .textTheme  
  
                        .headline3),  
  
                const SizedBox(height: 20),  
  
                Padding(  
  
                    padding:  
  
                    const EdgeInsets.symmetric(  
  
                        vertical: 15),  
  
                    child: Column(  
  
                        mainAxisAlignment:  
                            MainAxisAlignment  
                                .start,  
  
                        children: [  
  
                            Text(  
  
                                "Address Line 1",  
  
                                style: Theme.of(context)  
  
                                    .textTheme
```

```
        .headline6,  
    ),  
    TextFormField(  
        controller:  
            addresslineone,  
        style: Theme.of(context)  
            .textTheme  
            .bodyText2,  
    )  
]),  
Padding(  
    padding:  
        const EdgeInsets.symmetric(  
            vertical: 15),  
    child: Column(  
        crossAxisAlignment:  
            CrossAxisAlignment  
            .start,  
    children: [  
        Text(  
            "Address Line 2",  
            style: Theme.of(context)
```

```
        .textTheme  
        .headline6,  
    ),  
    TextFormField(  
        controller:  
        addresslinetwo,  
        style: Theme.of(context)  
            .textTheme  
            .bodyText2,  
    )  
]),  
Padding(  
    padding:  
    const EdgeInsets.symmetric(  
        vertical: 15),  
    child: Column(  
        crossAxisAlignment:  
        CrossAxisAlignment.start,  
        children: [  
            Text(  
                "City/County",
```

```
        style: Theme.of(context)

            .textTheme

            .headline6,

        ),

TextField(
    controller: city,
    style: Theme.of(context)

        .textTheme

        .bodyText2,
    )
]),

Padding(
    padding:
        const EdgeInsets.symmetric(
            vertical: 15),
    child: Column(
        crossAxisAlignment:
            CrossAxisAlignmentAlignment
            .start,
    children: [
        Text(
            "Postcode",

```

```
        style: Theme.of(context)

            .textTheme

            .headline6,

        ),

TextField(
    controller: postcode,
    style: Theme.of(context)

        .textTheme

        .bodyText2,
    )
]),

]))]

],
);

}));}

}),
```

With the address being editable it means that I am unable to use geolocation so instead primarily, I will only be using the lat and long to find out if the restaurant is within the service distance.

While relooking at the code I realised that I forgot to add input formatters so those were added.

The next thing to do was to store the data on the device. To do this, I used shared preferences to store the location.

Latitude - double (float)

Longitude - double (float)

Address line 1, Address line 2, city/county, postcode - String list

To reduce the number of stored variables, the lat and longitude are stored as integers. In contrast, the text-based location is stored as a string array.

```
Future<bool> saveLocation() async {  
    try {  
  
        final prefs = await SharedPreferences.getInstance();  
  
        await prefs.setDouble('locationLatitude', cameraPosition.target.latitude);  
  
        await prefs.setDouble(  
            'locationLatitude', cameraPosition.target.longitude);  
  
        await prefs.setStringList('locationPlacemark', <String>[  
            addresslineone.text,  
  
            addresslinetwo.text,  
  
            city.text,  
  
            postcode.text  
        ]);  
  
        return true;  
    } catch (e) {  
        return false;  
    }  
}
```

```
onTap: () async {

    bool hasSavedLocation =
        await saveLocation();

    if (hasSavedLocation ==
        true) {

        setState(() {
            Navigator.of(context)
                .pop();
        });
    } else {
        setState(() {
            ScaffoldMessenger.of(
                context)
                .showSnackBar(
                    const SnackBar(
                        content:
Text(
                'Failed
to update location.')));
        });
    }
}),
```

After adding the code to the program and adding the onTap function to the button it would go back to the main screen but would not save since when I reopened the location setter, it would try to get my current location instead of the saved location. To get an understanding of what was breaking I printed the result of the check if it was saved which returned true. This means that the check is wrong or the function that is calling the data back is broken.

```
I/flutter ( 5558): true
```

As it turned out it was being saved correctly which means that the retrieval was not working correctly.

```
I/flutter ( 5558): Here
I/flutter ( 5558): [Coram Close, Coram Cl, Hertfordshire, HP4 2JG]
D/SurfaceView@d201f04( 5558): setAlpha: mUseAlpha = false alpha=1.0
D/SurfaceView@d201f04( 5558): updateSurfaceAlphaInternal(alpha) is not
```

When trying to get the latitude and longitude from shared preferences, they were not being received

```
I/flutter ( 5558): >> null
Reloaded 1 of 1216 libraries in 58
```

After doing testing, I found that there was a late variable which was resetting the lat and long stored when it was being saved, preventing it from saving. Since the lat and long was 0, 0, it would then return null. To fix this, I removed it from being assigned early.

The next thing to fix was that whenever the map was opening and getting the current location, it would not look for the placemark so if the user wanted to save instantly without moving the map, it would not work since the text controllers were null. To fix this, I made the getPlacemark Future require a latitude and longitude and whenever there was a call, it would put in the lat and long directly into the Future meaning that it could be called before the map is created since it normally does a placemark check after the map has stopped moving

```
//get address name from camera position
getPlacemark(cameraPosition.target.latitude,
    cameraPosition.target.longitude);
} catch (e) {
```

After doing the change, it did the update at map creation but when moving it would not update it. This seems to happen because the lat and long are not updating. As I found out, the

Futurebuilder was being remade with the current location every time. To fix this, I will need to remake the app to use a setstate for the initial creation.

Fixing the location setting final code:

```
import 'dart:async';

import 'package:alleat/widgets/genericlocading.dart';

import 'package:flutter/services.dart';

import 'package:google_maps_flutter/google_maps_flutter.dart';

import 'package:flutter/material.dart';

import 'package:map_picker/map_picker.dart';

import 'package:geocoding/geocoding.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:geolocator/geolocator.dart';

class SelectLocation extends StatefulWidget {

  const SelectLocation({super.key});

  @override

  State<SelectLocation> createState() => _SelectLocationState();
}

class _SelectLocationState extends State<SelectLocation> {

  final _controller = Completer<GoogleMapController>();

  static TextEditingController addresslineone = TextEditingController();
```

```
static TextEditingController addresslinetwo = TextEditingController();

static TextEditingController postcode = TextEditingController();

static TextEditingController city = TextEditingController();

MapPickerController mapPickerController = MapPickerController();

late var cameraPosition =

const CameraPosition(target: LatLng(0, 0), zoom: 20);

Future<List> getSavedPosition() async {

//Try to get saved location and current location

final prefs = await SharedPreferences

.getInstance(); // Get saved location from shared preferences

final double? savedLocationLat = prefs.getDouble('locationLatitude');

final double? savedLocationLng = prefs.getDouble('locationLongitude');

final List<String>? savedLocationText =

prefs.getStringList('locationPlacemark');

if (savedLocationLat != null &&

savedLocationLng != null &&

savedLocationText != null) {

//If not null returned, return the value

addresslineone = TextEditingController(text: savedLocationText[0]);
```

```
addresslinetwo = TextEditingController(text: savedLocationText[1]);  
  
city = TextEditingController(text: savedLocationText[2]);  
  
postcode = TextEditingController(text: savedLocationText[3]);  
  
return [savedLocationLat, savedLocationLng];  
  
} else {  
  
    //If null returned from saved location, get the approximate location  
  
    return getCurrentLocation();  
  
}  
  
}  
  
  
  
Future<bool> saveLocation() async {  
  
    try {  
  
        final prefs = await SharedPreferences.getInstance();  
  
        await prefs.setDouble('locationLatitude', cameraPosition.target.latitude);  
  
        await prefs.setDouble(  
  
            'locationLongitude', cameraPosition.target.longitude);  
  
        await prefs.setStringList('locationPlacemark', <String>[  
  
            addresslineone.text,  
  
            addresslinetwo.text,  
  
            city.text,  
  
            postcode.text  
        ]);  
    }  
}
```

```
        return true;

    } catch (e) {

        return false;
    }
}

Future<List> getCurrentLocation() async {

    bool serviceEnabled;

    LocationPermission permission;

    // Test if location services are enabled.

    serviceEnabled = await Geolocator.isLocationServiceEnabled();

    if (!serviceEnabled) {

        await getPlacemark([false, 51.509865, -0.118092]);

        return [51.509865, -0.118092];
    }

    // Check if the location is denied

    permission = await Geolocator.checkPermission();

    if (permission == LocationPermission.denied) {

        permission = await Geolocator.requestPermission();
    }
}
```

```

    if (permission == LocationPermission.denied) {

        await getPlacemark([false, 51.509865, -0.118092]);

        return [51.509865, -0.118092];

    }

}

if (permission == LocationPermission.deniedForever) {

    // Permissions are denied forever, handle appropriately.

    await getPlacemark([false, 51.509865, -0.118092]);

    return [51.509865, -0.118092];

}

// Access the current possition of the device

Position locationDevice = await Geolocator.getCurrentPosition();

await getPlacemark(

    [false, locationDevice.latitude, locationDevice.longitude]);

return [locationDevice.latitude, locationDevice.longitude];

}

Future<void> getPlacemark(placemarkLocation) async {

    try {

        switch (placemarkLocation[0]) {

```

```
case true:

    List<Placemark> placemarks = await placemarkFromCoordinates(
        cameraPosition.target.latitude,
        cameraPosition.target.longitude,
    );

    addresslineone = TextEditingController(text: placemarks.first.name);
    addresslinetwo = TextEditingController(text: placemarks.first.street);
    postcode = TextEditingController(text: placemarks.first.postalCode);

    if ((placemarks.first.locality) == "") {

        city = TextEditingController(
            text: placemarks.first.subAdministrativeArea);
    } else {

        city = TextEditingController(text: placemarks.first.locality);
    }

    break;

case false:

    List<Placemark> placemarks = await placemarkFromCoordinates(
        placemarkLocation[1], placemarkLocation[2]);
    addresslineone = TextEditingController(text: placemarks.first.name);
    addresslinetwo = TextEditingController(text: placemarks.first.street);
    postcode = TextEditingController(text: placemarks.first.postalCode);

    if ((placemarks.first.locality) == "") {
```

```
        city = TextEditingController(  
  
            text: placemarks.first.subAdministrativeArea);  
  
    } else {  
  
        city = TextEditingController(text: placemarks.first.locality);  
  
    }  
  
    break;  
}  
  
}  
  
} catch (e) {  
  
    textController.text = "";  
  
}  
  
}  
  
  
  
var textController = TextEditingController();  
  
@override  
  
Widget build(BuildContext context) {  
  
    return Scaffold(  
  
        body: FutureBuilder<List>(  
  
            future: getSavedPosition(),  
  
            builder: (context, snapshot) {  
  
                if (!snapshot.hasData) {  
  
                    return const GenericLoading();  
  
                }  
  
            },  
  
        ),  
  
    );  
}
```

```
if (snapshot.hasData) {  
  
    var savedPosition = snapshot.data ?? [];  
  
    CameraPosition cameraPosition = CameraPosition(  
        target: LatLng(savedPosition[0], savedPosition[1]), zoom: 19);  
  
    return Stack(  
        alignment: Alignment.topCenter,  
        children: [  
            LayoutBuilder(  
                builder: (BuildContext context, BoxConstraints constraints) {  
  
                    return MapPicker(  
                        // pass icon widget  
                        iconWidget: const Icon(  
                            Icons.location_on,  
                            size: 65,  
                        ),  
                        //add map picker controller  
                        mapPickerController: mapPickerController,  
                        child: GoogleMap(  
                            myLocationEnabled: false,  
  
                            zoomControlsEnabled: false,  
                        ),  
                    );  
                },  
            ),  
        ],  
    );  
}  
  
else {  
    return Center(  
        child: CircularProgressIndicator(),  
    );  
}
```

```
// hide location button

myLocationButtonEnabled: true,

mapType: MapType.normal,


rotateGesturesEnabled: false,
tiltGesturesEnabled: false,
// camera position

initialCameraPosition: cameraPosition,
onMapCreated: (GoogleMapController controller) {
    _controller.complete(controller);
},
onCameraMoveStarted: () {
    // notify map is moving
    try {
        mapPickerController.mapMoving!();
    } catch (e) {
        Navigator.pop(context);
    }
},
onCameraMove: (cameraPosition) {
    try {
        //If the pointer position is invalid, dont try to move
the map.
    }
}
```

```
        this.cameraPosition = cameraPosition;

    } catch (e) {

        cameraPosition = cameraPosition;

    }

    },
onCameraIdle: () async {

    // notify map stopped moving

    try {

        // if there is an error getting the placemark return
null

        mapPickerController.mapFinishedMoving!();

        //get address name from camera position

        getPlacemark([true]);

    } catch (e) {

        return;

    }

},
));
}),
DraggableScrollableSheet( //Dragable bottom bar

initialChildSize: 0.2,

snap: true,

minChildSize: 0.2,
```



```
decoration: BoxDecoration(  
    color: Colors.grey,  
    borderRadius:  
        BorderRadius.circular(5)),  
  
Padding(  
    padding: const EdgeInsets.symmetric(  
        vertical: 30, horizontal: 20),  
  
    child: Row(  
        children: [  
            Expanded(  
                child: InkWell( //Save Location  
button  
                    onTap: (() async {  
                        bool hasSavedLocation =  
                            await saveLocation();  
//Try to save  
  
                        if (hasSavedLocation ==  
                            true) {  
                            setState(() {  
                                Navigator.of(context)  
                                    .pop();  
                            });  
                    }));
```

```
        } else {

            setState(() { //Display
failed to update

            ScaffoldMessenger.of(
                context)
                .showSnackBar(
                    const SnackBar(
                        content:
Text(
                'Failed
to update location.')));
            });
        }
    },
    child: Container(
        decoration: BoxDecoration(
            borderRadius:
                BorderRadius
                    .circular(10),
            color: Theme.of(context)
                .primaryColor),
        padding: const EdgeInsets
            .symmetric(
                vertical: 15,
```

```
        horizontal: 30),  
  
        child: Text(  
  
            "Save Location",  
  
            textAlign:  
  
                TextAlign.center,  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline6!  
  
            .copyWith(  
  
                color: Theme.of(  
  
                    context)  
  
                .colorScheme  
  
                .onSurface),  
  
(,),  
  
        ))),  
  
        const SizedBox(  
  
            width: 20,  
  
(,),  
  
        InkWell( //Current location button  
  
            onTap: () async {  
  
                // Get the current location of the  
device  
  
                List currentLocation =
```

```
        await getCurrentLocation();

        // Move the camera to the current
location

        final GoogleMapController

        controller =
        await _controller.future;

        controller.animateCamera(
        CameraUpdate
        .newCameraPosition(
        CameraPosition(
        target: LatLng(
        currentLocation[0],
        currentLocation[1]),
        zoom: 18,
        ),
        ),
        );
        },
        child: Container(
        decoration: BoxDecoration(
        borderRadius:
        BorderRadius.circular(
        10),
        color: Theme.of(context)
```

```
        .backgroundColor),  
  
        padding:  
          const EdgeInsets.all(15),  
  
        child: Icon(  
          Icons.my_location,  
          color: Theme.of(context)  
            .textTheme  
            .headline1!  
            .color,  
        )),  
      ),  
    ],  
  )),  
Container(  
  padding: const EdgeInsets.all(20),  
  alignment: Alignment.bottomLeft,  
  child: Column(  
    mainAxisAlignment:  
      MainAxisAlignment.start,  
    children: [  
      Text("Edit your address",  
        style: Theme.of(context)
```

```
        .textTheme  
        .headline3),  
  
    Padding(  
  
        padding: const EdgeInsets.only(  
  
            top: 5,  
  
            right: 20,  
  
            bottom: 30),  
  
        child: Text(  
  
            "Set your exact address on  
the map and edit it below.",  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline6)),  
  
    Padding(  
  
        padding:  
        const EdgeInsets.symmetric(  
  
            vertical: 15),  
  
        child: Column(  
  
            crossAxisAlignment:  
            CrossAxisAlignment.start,  
  
            children: [  
  
                Text(  
                    "Address:  
                       
                    
```

```
        "Address Line 1",  
  
        style: Theme.of(context)  
            .textTheme  
            .headline6,  
        ),  
  
        TextFormField(  
            controller:  
  
                addresslineone,  
  
            style: Theme.of(context)  
                .textTheme  
                .bodyText2,  
  
            inputFormatters: [  
  
                //Only allows the input  
of letters a-z and A-Z and @,.-
```

FilteringTextInputFormatter

```
        .allow(RegExp(  
            r'[a-zA-Z0-9,.  
]+|\s'))  
  
        ],  
    )  
]),  
  
Padding(  
    padding:
```

```
const EdgeInsets.symmetric(
    vertical: 15),
child: Column(
    mainAxisAlignment:
        MainAxisAlignment
            .start,
children: [
    Text(
        "Address Line 2",
        style: Theme.of(context)
            .textTheme
            .headline6,
    ),
    TextFormField(
        controller:
            addresslinetwo,
        style: Theme.of(context)
            .textTheme
            .bodyText2,
        inputFormatters: [
            //Only allows the input
of letters a-z and A-Z and ,.- and space
```

```
FilteringTextInputFormatter

    .allow(RegExp(
        r'[a-zA-Z0-9,-]+|\s'))
    ],
)
]),

Padding(
    padding:
        const EdgeInsets.symmetric(
            vertical: 15),
    child: Column(
        crossAxisAlignment:
            CrossAxisAlignment.start,
        children: [
            Text(
                "City/County",
                style: Theme.of(context)
                    .textTheme
                    .headline6,
            ),
        ],
    ),
)
```

```
        TextFormField(  
  
            controller: city,  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .bodyText2,  
  
            inputFormatters: [  
  
                //Only allows the input  
of letters a-z and A-Z and ,.- and space
```

FilteringTextInputFormatter

```
.allow(RegExp(  
  
    r'[a-zA-Z0-9,.-  
]+|\s'))  
  
],  
  
)  
  
]),  
  
Padding(  
  
padding:  
  
const EdgeInsets.symmetric(  
  
    vertical: 15),  
  
child: Column(  
  
crossAxisAlignment:  
  
CrossAxisAlignment  
  
.start,
```

```
children: [  
  Text(  
    "Postcode",  
    style: Theme.of(context)  
      .textTheme  
      .headline6,  
  ),  
  TextFormField(  
    controller: postcode,  
    style: Theme.of(context)  
      .textTheme  
      .bodyText2,  
    inputFormatters: [  
      //Only allows the input  
      of letters a-z and A-Z and ,.- and space
```

FilteringTextInputFormatter

```
.allow(RegExp(  
  r'[a-zA-Z0-9,.-]  
  ]+|\s'))  
,  
)  
]),  
]))
```

```
        ],
      );
    })));
  })),
],
);
} else {
  return const GenericLoading();
}
},
));
}
}
```

Top Bar Update 1

To display the current location, I used a FutureBuilder so that it would update every time the location is changed.

I also made it so that the InkWell for the location button had a bigger hit target so that you would not need to click on the button but now also the area around it.

```
import 'package:alleat/screens/locationselection.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';
```

```
class MainAppBar extends StatelessWidget implements PreferredSizeWidget {

    final double height;

    const MainAppBar({Key? key, required this.height}) : super(key: key);

    Future<String> getSavedLocation() async {

        final prefs = await SharedPreferences.getInstance();

        final List<String>? savedLocationText =
            prefs.getStringList('locationPlacemark');

        if (savedLocationText != null) {

            return savedLocationText[1];

        } else {

            return ("No Location Set");

        }

    }

}

@Override

Widget build(BuildContext context) {

    return Container(

        //Container of height 120px

        height: 120,

        color: Theme.of(context).bottomNavigationBarTheme.backgroundColor,

        child: SafeArea(
```

```
child: Row(  
  
    mainAxisAlignment: MainAxisAlignment.spaceAround,  
  
    crossAxisAlignment: CrossAxisAlignment.center,  
  
    children: [  
  
        InkWell(  
  
            onTap: () {  
  
                Navigator.push(  
  
                    //go to profile creation page  
  
                    context,  
  
                    MaterialPageRoute(  
  
                        builder: (context) => const SelectLocation()));  
  
            },  
  
            child: Padding(  
  
                padding: const EdgeInsets.all(10),  
  
                child: Icon(  
  
                    //Location icon  
  
                    Icons.location_on_outlined,  
  
                    color: Theme.of(context).textTheme.headline1?.color,  
  
                )),  
  
        Column(mainAxisAlignment: MainAxisAlignment.center, children: [  
  
            //Column that will display the destination the food will be  
delivered to  
  
            Text(  
        
```

```
    "Location",  
  
    style: Theme.of(context)  
  
        .textTheme  
  
        .headline6  
  
        ?.copyWith(fontWeight: FontWeight.w500),  
,  
  
const SizedBox(  
  
    height: 2,  
,  
  
FutureBuilder<String>(  
  
    future: getSavedLocation(),  
  
    builder: (context, snapshot) {  
  
        if (!snapshot.hasData) {  
  
            return const Text("Location loading");  
        }  
  
        if (snapshot.hasData) {  
  
            var location = snapshot.data ?? [];  
  
            return Text(location.toString());  
        } else {  
  
            return const Text("No Location Set");  
        }  
    })
```

```

        ],
        Icon(
            //Cart icon
            Icons.shopping_cart_outlined,
            color: Theme.of(context).textTheme.headline1?.color,
        ),
    ],
));
}

@Override
Size get preferredSize => Size.fromHeight(height);
}

```

Restaurant Distance

The next thing to do was to update the server's database to include the latitude and longitude of the restaurant and in this way, I can use the hypotenuse of the two locations to determine the distance between the destination and the restaurant.

$$\sqrt{(destinationLat - restaurantLat)^2 + (destinationLng - restaurantLng)^2} = distance$$

After doing some research on this topic, I found that this conversion is more complicated than expected, using the following calculation instead.

$$12742 * \text{asin}(\text{sqrt}(0.5 - \cos((\text{lat2} - \text{lat1}) * \pi/2) + \cos(\text{lat1} * \pi) * \cos(\text{lat2} * \pi) * (1 - \cos((\text{lon2} - \text{lon1}) * \pi)/2)))$$

I used Google Maps to get the latitude and longitude in degrees and saved them to new columns in the database

	restaurantid	restaurantname	address	latitude	longitude	website	phonenumber	thrid	des
<input type="checkbox"/>	1	Papa John's Pizza	211b High Street, Berkhamsted, HP4 1AD	51.7609709052447	-0.5670057130957606	https://www.papajohns.co.uk/	1442862900	202320	The inte add Joh
<input type="checkbox"/>	2	PizzaExpress	300 High Street, Berkhamsted, HP4 1ZZ	51.76234546801358	-0.569736766167508	https://www.pizzaexpress.com/	1442750510	1285623	
<input type="checkbox"/>	3	The Fat Buddha	378 High Street, Berkhamsted, HP4 1HU	51.76336426903882	-0.5731856301529226	https://www.thefatbuddha.co.uk	1442879995	1262835	
<input type="checkbox"/>	4	The Meating Room	307 High Street, Berkhamsted, HP4 1AL	51.76255448298398	-0.5714537515869422	https://www.meating-room.co.uk	1442879994	1150863	Our th... is q... foc... ...
<input type="checkbox"/>	5	Starbucks	200 High Street, Berkhamsted, HP4 3AP	51.76074040128775	-0.5650503034146545	https://www.starbucks.co.uk/	1422777800	1452823	
<input type="checkbox"/>	6	Zero Sushi	8-12 Lower Kings Road, Berkhamsted, HP4 2AE	51.76096669941346	-0.5653389643538229	https://zerosushi.co.uk/	1442877982	720477	Zer... deli... qua... t...

From the restaurantlist.php file, I added the code to import the longitude and latitude to each restaurant

```
if($val){
    $sort = $_POST["sort"];
    if ($sort == "id"){
        $sql = "SELECT res.restaurantid, res.restaurantname, res.restaurantlogo, res.restaurantbanner, res.latitude, res.longitude FROM restaurant res LIMIT 20";
        if($result = mysqli_query($link, $sql)){
            while($row = mysqli_fetch_assoc($result)) {
                array_push($return["restaurants"], [$row["restaurantid"], $row["restaurantname"], $row["restaurantlogo"], $row["restaurantbanner"]]);
            }
            mysqli_free_result($result);
        }
    }
}
```

From there, I created a Future which used the output from the getRestataurus Future to update the list to contain the distance from the destination.

```
Future<void> getDistance(restaurantdata) async {

    var p = 0.017453292519943295; //Convert constant from degrees to radians

    final prefs = await SharedPreferences.getInstance();

    final double? savedLocationLat = prefs.getDouble('locationLatitude'); //lat2

    final double? savedLocationLng =
        prefs.getDouble('locationLongitude'); //lng2
```

```

for (var i = 0; i < restaurantdata["restaurants"].length; i++) {

    double latRestaurant =
        double.parse(restaurantdata["restaurants"][i][4]); //lat1

    double lngRestaurant =
        double.parse(restaurantdata["restaurants"][i][5]); //lng1

    double distance = 12742 *
        asin(sqrt(0.5 -
            cos((savedLocationLat! - latRestaurant) * p) / 2 +
            cos(latRestaurant * p) *
            cos(savedLocationLat * p) *
            (1 - cos((savedLocationLng! - lngRestaurant) * p)) /
            2));
}

}

```

To incorporate it into the main app and ensure that it cannot be used with null, I made it return true if a location has been set and false if not. It is then returned back to the original getRestaurant Future where the app rounds the result shown to 1dp.

```

Text(
    "$${(restaurantsdata[0]["restaurants"][index][6]).toStringAsFixed(1)} km away",
    style: Theme.of(
        context)
    .textTheme

```

```
.bodyText1!  
  
.copyWith(  
  
    fontSize:  
  
        12)),  
  
    const SizedBox(  
  
        width: 5,  
  
) ,
```

To then also get the distance in the restaurant pages, I forwarded over the variables when the new screen was called.

```
class RestaurantMain extends StatefulWidget {  
  
    final String resid;  
  
    final String resname;  
  
    final String reslogo;  
  
    final String resbanner;  
  
    final String resdistance;  
  
  
    const RestaurantMain(  
  
        //Get restaurant info from the restaurant list widget (passed from the  
        container)  
  
        {Key? key,  
  
        required this.resid,  
  
        required this.resname,
```

```

        required this.reslogo,
        required this.resbanner,
        required this.resdistance
    })
    : super(key: key);

}

@Override
State<RestaurantMain> createState() => _RestaurantMainState();
}

```

While attempting to get the data to show, I had a few encounters where the error shown to the user said it was a connection error. To fix this, I will use the message forwarded over catch or else statements.

While testing, I got this error where it was bypassing the checks.

The screenshot shows a code editor with several lines of Dart code. A tooltip is displayed, indicating a type error: '_TypeError (type 'String' is not a subtype of type 'List<dynamic>')'. The code includes a try block with a print statement and a list assignment, followed by a return statement that triggers the error. The error message is highlighted in red.

```

235 //try {
236   print(restaurantsdata[0]);
237   List restaurants = restaurantsdata[0]["restaurants"];
Exception has occurred. ×
_TypeError (type 'String' is not a subtype of type 'List<dynamic>')
238   return FutureBuilder<List>(
239     //Get favourites list from future
240     future: getFavourites(),
{error: true, message: Failed to get location. Try changing your destination address, restaurants: []}

```

To fix this, I had to change it from being a boolean check to being a string check

```

if [restaurantsdata[0]["error"] == "true"] {
    return Padding(

```

In order to help future prototypes, I made all boolean outputs booleans. After this fix the errors were being forwarded to the error message container correctly.

Sorting & Filtering

To do the sorting and filtering, I first have to make the page to change these settings.

Using the concept art shown below, we see it uses the prebuilt back button, a button stack, checkbox, button row and two sliders.

< Back

Sort

 Recommended (Default)

 Top Rated

 Popular

 Distance

Filter

[Clear](#)

Show only Favourites

Price Range

Max. Delivery Fee

£1 £2 £3 £4+



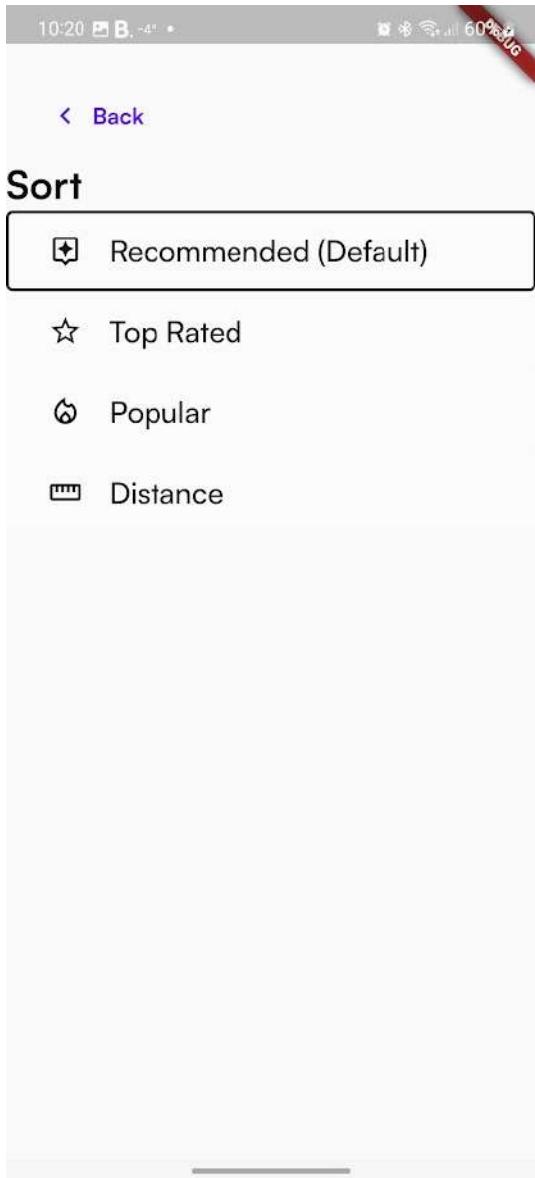
Minimum Order Price

None £10 £20 £30 £40+



Sorting

I first created the basic sort options:



```
import 'package:alleat/widgets/elements/elements.dart';

import 'package:flutter/material.dart';

class FilterSort extends StatefulWidget {
```



```
        "Recommended (Default)",  
        style: Theme.of(context).textTheme.headline4,  
    )  
],  
)),  
ElevatedButton(  
    onPressed: null,  
    child: Row(  
        children: [  
            Icon(Icons.star_border_outlined,  
                color: Theme.of(context).textTheme.headline4?.color),  
            const SizedBox(width: 20),  
            Text(  
                "Top Rated",  
                style: Theme.of(context).textTheme.headline4,  
            )  
        ],  
    )),  
ElevatedButton(  
    onPressed: null,  
    child: Row(  
        children: [  
           
```



```
        ],
      )),
    ],
  );
}
```

In order to have a selection based system that would forward over the options selected, I needed it to save the data changed in a variable so I used a map with two indexes, filter and sort. With sort, if a user were to select an option, it would overwrite the previous selected option. With filter, once a user presses a button, it would toggle on and off

In order to simplify the program, I used a ListView which looked at an array and used its index to create a list of buttons and their options. This also decreased the programming time since I only had to do it once.

To have a border show for each option selected, I used the same system that I used for the favourite icon on the browse page where it uses an if statement to check if a statement is true and used the result to change the border. With this, it checks the saved result and if it is the same result as the index and creates a border if true.

```
import 'package:alleat/widgets/elements/elements.dart';

import 'package:flutter/material.dart';

class FilterSort extends StatefulWidget {

  const FilterSort({super.key});

  @override
  State<FilterSort> createState() => _FilterSortState();
}
```

```
}
```

```
class _FilterSortState extends State<FilterSort> {

  Map customiseSelected = {"sort": "default", "filters": []};

  List sortOptions = [
    ["Recommended", Icons.assistant_outlined, "default"],
    ["Top Rated", Icons.star_border_outlined, "top"],
    ["Popular", Icons.local_fire_department_outlined, "hot"],
    ["Distance", Icons.straighten_outlined, "distance"]
  ];

  @override

  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
        const ScreenBackButton(),
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 30),
          child:
            Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
              const SizedBox(
                height: 20,
              ),
            ],

```

```
Text("Sort", style: Theme.of(context).textTheme.headline2),  
  
ListView.builder(  
  
    physics: const NeverScrollableScrollPhysics(),  
  
    scrollDirection: Axis.vertical,  
  
    shrinkWrap: true,  
  
    itemCount: sortOptions.length, //For each restaurant  
  
    itemBuilder: (context, index) {  
  
        return Padding(  
  
            padding: const EdgeInsets.symmetric(vertical: 7),  
  
            child: ElevatedButton(  
  
                style: ElevatedButton.styleFrom(  
  
                    side: customiseSelected["sort"] ==  
  
                        sortOptions[index][2]  
  
                    ? BorderSide(  
  
                        color: Theme.of(context).primaryColor,  
  
                        width: 2)  
  
                    : BorderSide.none,  
  
                    backgroundColor:  
  
                        Theme.of(context).colorScheme.onSurface,  
  
                    padding: const EdgeInsets.symmetric(  
  
                        vertical: 15, horizontal: 25)),  
  
                    onPressed: () {
```

```
        setState(() {
            customiseSelected["sort"] = sortOptions[index][2];
        });
    },
    child: Row(
        children: [
            Icon(
                sortOptions[index][1],
                color: customiseSelected["sort"] ==
                    sortOptions[index][2]
                    ? Theme.of(context)
                        .textTheme
                        .headline1
                        ?.color
                    : Theme.of(context)
                        .textTheme
                        .headline5
                        ?.color,
            ),
            const SizedBox(width: 20),
            Flexible(
                child: Text(

```

```
        sortOptions[index][0],  
  
        style: Theme.of(context)  
  
            .textTheme  
  
            .headline5!  
  
            .copyWith(  
  
                color: customiseSelected["sort"] ==  
  
                    sortOptions[index][2]  
  
                ? Theme.of(context)  
  
                    .textTheme  
  
                    .headline1  
  
                    ?.color  
  
                : Theme.of(context)  
  
                    .textTheme  
  
                    .headline5  
  
                    ?.color,  
  
            ),  
  
        ))  
  
    ],  
  
    )));  
  
},  
  
const SizedBox(  
  
    height: 20,
```

```
        ),  
  
        Text("Filter", style: Theme.of(context).textTheme.headline2),  
    ])),  
]);  
}  
}
```

Filtering

Favourites

The next thing I did was the filter options. To start with, I created a checkbox. I was able to do this because flutter has a built-in function called checkbox.

I decided to make a row containing a checkbox and the text. Since I had never used a checkbox before, I used <https://api.flutter.dev/flutter/material/Checkbox-class.html> to make the checkbox initially. After testing it out though, I realised that the hitbox was only the checkbox itself which is usually not how people use a checkbox. After some further research I found that CheckboxListTile allowed for text as well as the checkbox so I added that instead.

An error I encountered was an infinite size error where the tile doesn't have any size constraints. To fix this, I used a Flexible widget which means that it tries to go to the edge of the widget/screen and moves text to the next line if it goes over.

Another error I encountered was screen scrolling and resolutions where if the screen was small enough, it would overflow. To fix this, I added a SingleChildScrollView

```
Row(  
    children: [  
        Flexible(  
            child: CheckboxListTile(  
                title: Text("Favourites"),  
                value: _isFavouritesSelected,  
                onChanged: (value) {  
                    setState(() {  
                        _isFavouritesSelected = value;  
                    });  
                },  
            ),  
        ),  
    ],  
)
```

```
        title: Text(  
            "Show only favourites",  
            style: Theme.of(context).textTheme.headline6,  
        ),  
  
        checkColor: Colors.white,  
  
        value: isChecked,  
  
        controlAffinity: ListTileControlAffinity.leading,  
  
        onChanged: (newValue) {  
  
            setState(() {  
  
                isChecked = !isChecked;  
  
            });  
  
            if (isChecked) {  
  
                customiseSelected["filters"].add("favourite");  
  
            } else {  
  
                customiseSelected["filters"].remove("favourite");  
  
            }  
  
        },  
  
    )),  
],  
) ,
```

Price Range

To do the price range, I would need to have it by default be stored as 4 values for the different price ranges. Each price range can be added or removed.

After realising this, I decided to remake the stored Map to have separate indexes for each filter instead of having it being stored in a list.

```
Map customiseSelected = {"sort": "default", "favourite": false, "price":  
[1,2,3,4], "maxDelivery": 4, "minOrder": 40};
```

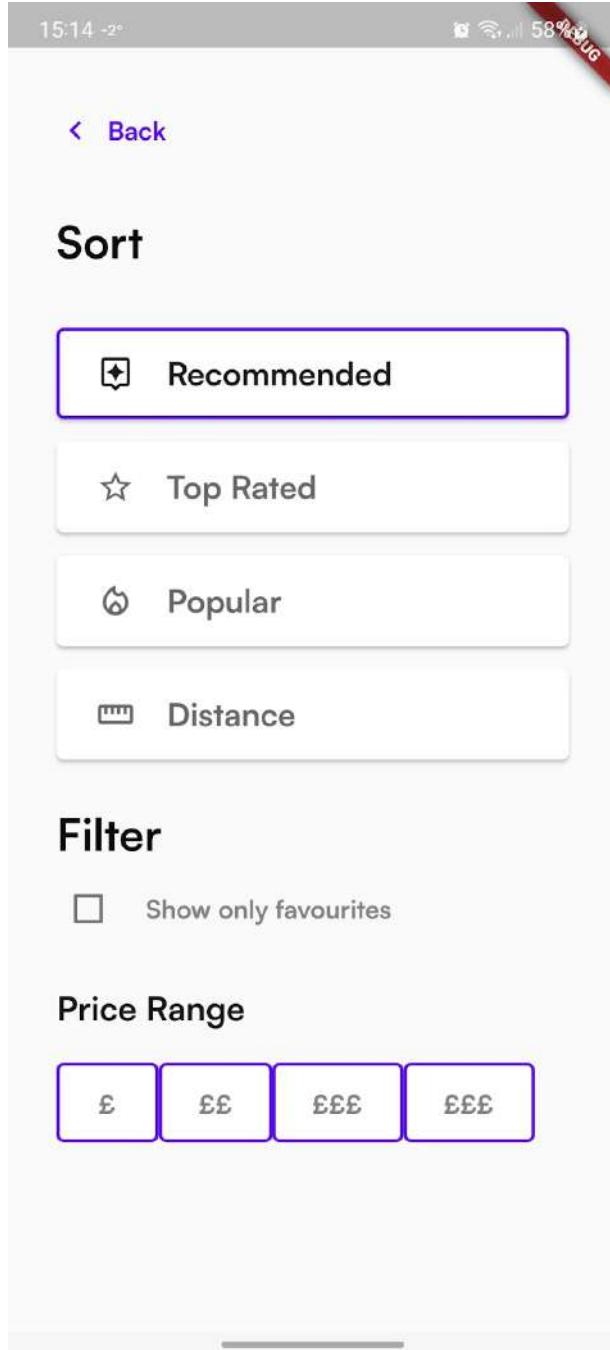
This allows for different types of filters and for there to always be a filter even when not iterated.

After doing this change, it was much more apparent how to do the price range. By using a container with a row of 4 buttons which then checks the Map, it could then do a task associated with it either being active or not.

```
Container(  
  
    color: Theme.of(context).colorScheme.onSurface,  
  
    height: 50,  
  
    child: ListView.builder(  
  
        scrollDirection: Axis.horizontal,  
  
        shrinkWrap: true,  
  
        itemCount: priceRangeOptions.length, //For each restaurant  
  
        itemBuilder: (context, index) {  
  
            return ElevatedButton(  
  
                style: ElevatedButton.styleFrom(  
  
                    side: customiseSelected["price"]  
  
                    .contains(priceRangeOptions[index][1]))
```

```
? BorderSide(  
    color: Theme.of(context).primaryColor,  
    width: 2)  
  
: BorderSide.none,  
  
backgroundColor:  
    Theme.of(context).colorScheme.onSurface,  
  
padding: const EdgeInsets.symmetric(  
    vertical: 15, horizontal: 25)),  
  
onPressed: null,  
  
child: Text(  
    priceRangeOptions[index][0],  
    style: Theme.of(context).textTheme.headline6,  
,  
);  
});  
]),  
]),
```

An issue I found was that the buttons were of varying sized dependent on the width of the text



To fix this, I used an expanded widget and a sizedbox to set the max width. I also added padding to the sides to ensure that there was a gap between each items.

After doing this though, I encountered an error where it did not like me added the expanded to a sizedbox

```
===== Exception caught by widgets library =====
Incorrect use of ParentDataWidget.
```

I decided to remove the expanded and it seemed to work the same but with the width being 100px, it consistently overflowed so had to be scrolled.

To fix this, I simply reduced the padding and set the width to 70px instead

Finally, to add the styling to the buttons where the buttons update, the exact same code was used from the sorting buttons.

```
Container(  
    color: Theme.of(context).colorScheme.onSurface,  
    height: 50,  
    child: ListView.builder(  
        scrollDirection: Axis.horizontal,  
        shrinkWrap: true,  
        itemCount: priceRangeOptions.length, //For each restaurant  
        itemBuilder: (context, index) {  
            return Padding(  
                padding: const EdgeInsets.symmetric(horizontal: 5),  
                child: SizedBox(  
                    width: 70,  
                    child: ElevatedButton(  
                        style: ElevatedButton.styleFrom(  
                            side: customiseSelected["price"].contains(
```

```
    priceRangeOptions[index][1])

    ? BorderSide(
        color:
            Theme.of(context).primaryColor,
        width: 2)

    : BorderSide.none,
backgroundColor:
    Theme.of(context).colorScheme.onSurface,
padding: const EdgeInsets.symmetric(
    vertical: 15, horizontal: 10)),

onPressed: () {
    if (customiseSelected["price"]
        .contains(priceRangeOptions[index][1])) {
        setState(() {
            customiseSelected["price"]
                .remove(priceRangeOptions[index][1]);
        });
    } else {
        setState(() {
            customiseSelected["price"]
                .add(priceRangeOptions[index][1]);
        });
    }
}
```

```
        },

        print(customiseSelected);

    },
}

child: Text(
    priceRangeOptions[index][0],
    style: Theme.of(context)

        .textTheme
        .headline6!
        .copyWith(
            color: customiseSelected["price"]
            .contains(
                priceRangeOptions[index][1]
                ? Theme.of(context).primaryColor
                : Theme.of(context)
                    .textTheme
                    .headline6
                    ?.color,
            ),
        ),
    ))),
});,
```

Max Delivery Fee

To do the max delivery fee, I will be using a slider instead of a button since it is a linear scale that has correlation.

In order to make this, I will be using the built in slider function within Flutter:

<https://api.flutter.dev/flutter/material/Slider-class.html>

I started by making the main slider.

```
Slider(  
    value: _currentMaxDeliveryFeeValue,  
    max: 4,  
    min: 0,  
    thumbColor: Theme.of(context).primaryColor,  
    activeColor: Theme.of(context).primaryColor,  
    inactiveColor:  
        Theme.of(context).colorScheme.onBackground.withOpacity(0.2),  
    divisions: 4,  
    label: _currentMaxDeliveryFeeValue.toString(),  
    onChanged: (double value) {  
        setState(() {  
            _currentMaxDeliveryFeeValue = value;  
        });  
    },  
,
```

After doing some research, I found that in order to get labels to show like the one in the concept art I would need to use a package. I decided that the best thing to do was to take the value obtained and output the result as text.

```
Row(mainAxisAlignment: MainAxisAlignment.spaceBetween, children: [  
    Text("Max Delivery Fee",  
        style: Theme.of(context).textTheme.headline3),  
    LayoutBuilder(builder: (context, constraints) {  
        if (_currentMaxDeliveryFeeValue == 0) {  
            return const Text("FREE");  
        } else if (_currentMaxDeliveryFeeValue == 4) {  
            return const Text("£4.00+");  
        } else {  
            return Text("£${_currentMaxDeliveryFeeValue}0");  
        }  
    })  
]),
```

To change the value stored, the value in the Map is replaced with the stored in the slider.

```
onChanged: (double value) {  
    setState(() {  
        _currentMaxDeliveryFeeValue = value;  
    });  
    customiseSelected["maxDelivery"] = _currentMaxDeliveryFeeValue;
```

```
 },
```

Min Order Price

A copy of max delivery fee was copied and altered to make the min order price

```
Slider(  
    value: _currentMinOrderPriceValue,  
    max: 40,  
    min: 0,  
    thumbColor: Theme.of(context).primaryColor,  
    activeColor: Theme.of(context).primaryColor,  
    inactiveColor:  
        Theme.of(context).colorScheme.onBackground.withOpacity(0.2),  
    divisions: 4,  
    onChanged: (double value) {  
        setState(() {  
            _currentMinOrderPriceValue = value;  
        });  
        customiseSelected["minOrder"] = _currentMinOrderPriceValue;  
    },  
,
```

This completed the different filters with all of them iterating on the customiseSelected Map.

Clear Filters

If the user wants to reset the filters, they can click a reset button. This will change the `customiseSelected` Map back to the original and as a result will update the different buttons.

```
LayoutBuilder(builder: (context, constraints) {  
  
    if (customiseSelected["sort"] != "default" ||  
  
        customiseSelected["favourite"] != false ||  
  
        !customiseSelected["price"].contains(1) ||  
  
        !customiseSelected["price"].contains(2) ||  
  
        !customiseSelected["price"].contains(3) ||  
  
        !customiseSelected["price"].contains(4) ||  
  
        customiseSelected["maxDelivery"] != 4.0 ||  
  
        customiseSelected["minOrder"] != 40.0) {  
  
    return InkWell(  
  
        onTap: () {  
  
            setState(() {  
  
                customiseSelected = {  
  
                    "sort": "default",  
  
                    "favourite": false,  
  
                    "price": [1, 2, 3, 4],  
  
                    "maxDelivery": 4.0,  
  
                    "minOrder": 40.0  
  
                };  
  
                _currentMaxDeliveryFeeValue = 4;  
            });  
        }  
    );  
};
```

```
        _currentMinOrderPriceValue = 40;

    });

},
child: Padding(
padding: const EdgeInsets.symmetric(
horizontal: 40, vertical: 20),
child: Text(
"Clear all",
style: Theme.of(context)
.textTheme
.headline6!
.copyWith(
color: Theme.of(context).colorScheme.error),
)));
} else {
return InkWell(
onTap: () {
setState(() {
customiseSelected = {
"sort": "default",
" favourite": false,
"price": [1, 2, 3, 4],
```

```
        "maxDelivery": 4.0,  
  
        "minOrder": 40.0  
  
    };  
  
    _currentMaxDeliveryFeeValue = 4;  
  
    _currentMinOrderPriceValue = 40;  
  
});  
  
},  
  
child: Padding(  
  
    padding: const EdgeInsets.symmetric(  
  
        horizontal: 40, vertical: 20),  
  
    child: Text(  
  
        "Clear all",  
  
        style: Theme.of(context)  
  
            .textTheme  
  
            .headline6!  
  
            .copyWith(  
  
                color: Theme.of(context)  
  
                    .colorScheme  
  
                    .error  
  
                    .withOpacity(0.1)),  
  
    )));  
  
}
```

```
})
```

Saving Filters and Sorts

In order to save the filters, there should be a button where the user can save the changes. In order to save the details in shared preferences, I will need to create a new save button. To do this, I first removed the back button and made a button below the filters, forcing the user to scroll through the options before saving. The filter options can then be saved within the shared preferences. While adding the code, I realised that shared preferences do not have an option to add Maps to it. Doing some research I found that json encoding and decoding can solve this problem.

<https://stackoverflow.com/questions/63964927/how-can-i-save-data-type-map-by-using-sharedpreferences>

Recalling Filters and Sorts

In order to get the data back from shared preferences, it must be initiated first before the screen is created so an initState was made

```
@override  
  
void initState() {  
  
    super.initState();  
  
    _loadFilter();  
  
}  
  
  
_loadFilter() async {  
  
    SharedPreferences prefs = await SharedPreferences.getInstance();  
  
    setState(() {  
  
        encodedCustomiseSelected = prefs.getString('filtersort');  
    });  
}
```

```

});
```

```

if (encodedCustomiseSelected != null) {
```

```

    customiseSelected = json.decode(encodedCustomiseSelected.toString());
```

```

} else {
```

```

    customiseSelected = {
```

```

        "sort": "default",
```

```

        "favourite": false,
```

```

        "price": [1, 2, 3, 4],
```

```

        "maxDelivery": 4.0,
```

```

        "minOrder": 40.0
```

```

    };
```

```

    _currentMaxDeliveryFeeValue = 4;
```

```

    _currentMinOrderPriceValue = 40;
```

```

}
```

```
}
```

Sending filters/Sort to Server

To send the data to the server, it first needs to be retrieved before it is sent to the server. While previously the sort method within the `restarauntList` file was by id, prototype 2 allows the user to set their sort and filter methods. In order to get the data, a future method was created which attempts to grab the data from `sharedPreferences`. Since the user may want the default sort and filter methods, if there is nothing in `sharedPreferences`, the sort and filter methods are set to default. If there is data, it is copied.

Since the user will want to see the restaurants in the local area, the longitude and latitude are also grabbed. If there is no location, an error is sent back to the main program.

```
Future<Map> getUserMetadata() async {

    final prefs = await SharedPreferences.getInstance();

    final String? filterSortEncoded = prefs.getString('filtersort');

    final double? locationLatitude = prefs.getDouble('locationLatitude');

    final double? locationLongitude = prefs.getDouble('locationLongitude');

    if (filterSortEncoded != null) {

        Map filterSort = json.decode(filterSortEncoded);

        metadataTemp["sort"] = filterSort["sort"];

        metadataTemp["favourite"] = filterSort["favourite"];

        metadataTemp["price"] = filterSort["price"];

        metadataTemp["maxDelivery"] = filterSort["maxDelivery"];

        metadataTemp["minOrder"] = filterSort["minOrder"];

    }

    if (locationLatitude == null ||
        locationLatitude == 0 ||
        locationLongitude == null ||
        locationLongitude == 0) {

        metadataTemp["error"] = true;

        return metadataTemp;

    } else {

        metadataTemp["latitude"] = locationLatitude;

        metadataTemp["longitude"] = locationLongitude;
    }
}
```

```
        return metadataTemp;  
    }  
}
```

```
Future<List> getRestaurants() async {  
  
    Map metadata = await getUserMetadata();  
  
    if (metadata["error"] != true) {  
  
        String phpurl =  
            "https://alleat.cpur.net/query/restaurantlist.php"; //Get restaurant  
list  
  
        try {  
  
            var res = await http.post(Uri.parse(phpurl), body: metadata);  
  
            if (res.statusCode == 200) {  
  
                //If sends successfully  
  
                var data = json.decode(res.body); //Decode to array  
  
                if (data["error"]) {  
  
                    //If fails to perform query  
  
                    List error = [  
  
                        {  
  
                            "error": true,  
  
                            "message": "Error: ${data["error"]}",  
  
                            "restaurants": "[]"  
                        } //Send blank list of restaurants  
                    ]  
                } else {  
                    List restaurants = data["restaurants"];  
                }  
            } else {  
                print("Error: ${res.statusCode}");  
            }  
        } catch (e) {  
            print("Error: $e");  
        }  
    }  
}
```

```
];

return error;

} else {

List listdata = await getDistance(data);

if (listdata[0] == false) {

List error = [

{

"error": true,

"message": 

"Failed to get Location. \nTry changing your destination address",

"restaurants": "[]"

} //Send blank list of restaurants

];

return error;

} else {

return [listdata[1]];

}

//If success, send the list of restaurants back

}

} else {

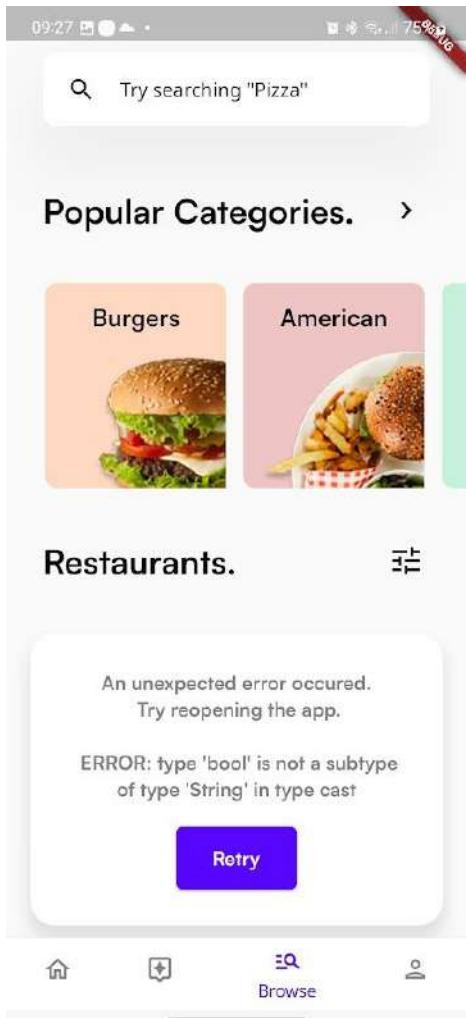
List error = [

{
```

```
        "error": true,  
        "message":  
            "Error ${res.statusCode}: Failed to connect to server.",  
        "restaurants": "[]"  
    }  
];  
  
return error;  
}  
}  
} catch (e) {  
  
List error = [  
  
{  
    "error": true,  
    "message": "An unexpected error occurred.\n Try reopening the app.",  
    "restaurants": "[]"  
}  
];  
  
return error;  
}  
}  
}  
} else {  
  
List error = [  
  
{  
    "error": true,  
}
```

```
    "message":  
        "Failed to get Location. \nTry changing your destination address",  
    "restaurants": "[]"  
} //Send blank list of restaurants  
];  
  
return error;  
}  
}  
}
```

While testing this code, the app was giving me an unknown error. To test this error, the caught error was output to the user.



The issue seems to be that HTTP does not support boolean values to be sent. To fix this, what I will do is encode the data at the client and decode it at the server.

While also doing this, I have removed the error status in the Map since it is unneeded for the server.

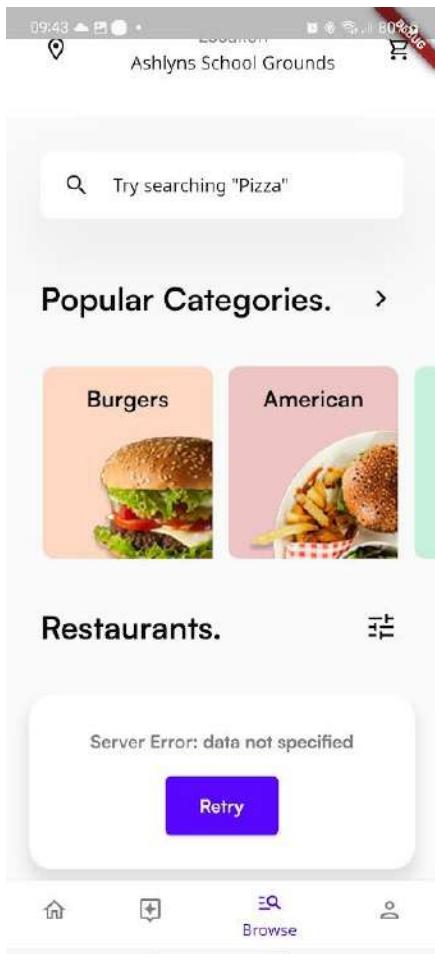
```
if (metadata["error"] != true) {  
    metadata.remove("error");  
  
    var metadataEncoded = json.encode(metadata);  
  
    String phurl =  
        "https://alleat.cpur.net/query/restaurantlist.php"; //Get restaurant  
list
```

```
try {  
  
    var res = await http.post(Uri.parse(phiurl), body: metadataEncoded);
```

This sends the data but sends back an error since it is unspecified.

Processing Filters/Sorts

While outputting what the server receives, it seems that there was no data.

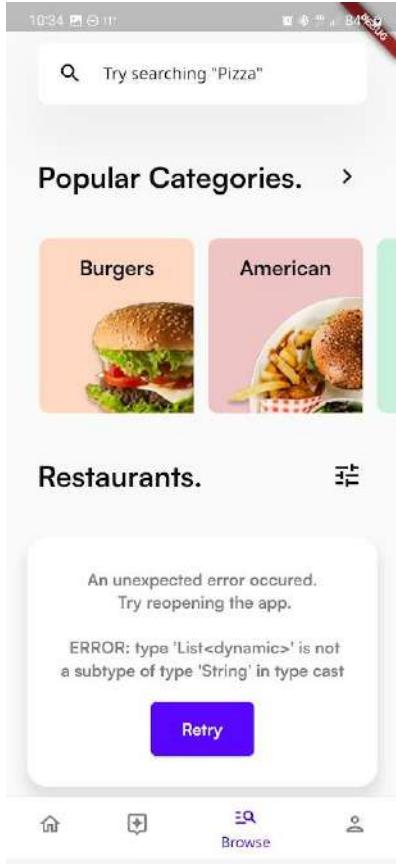


```
{error: true, message: data not specified, restaurants: [], test: []}
```

To overcome this issue, I decided that I would convert the boolean into a string instead of encoding it.

```
switch (metadata["favourite"]) {  
  
    case (true):  
  
        metadata.remove("favourite");  
  
        metadata["favourite"] = "true";  
  
        break;  
  
    case (false):  
  
        metadata.remove("favourite");  
  
        metadata["favourite"] = "false";  
  
        break;  
  
}
```

The next issue I faced was with the list of price ranges. Since it was not a string, this caused an error. To fix this, I converted it to a string and joined the values.



```
metadata["price"] = metadata["price"].join(",");
```

Once again, there was another error where it needed the double to be a string so i Converted them all to strings

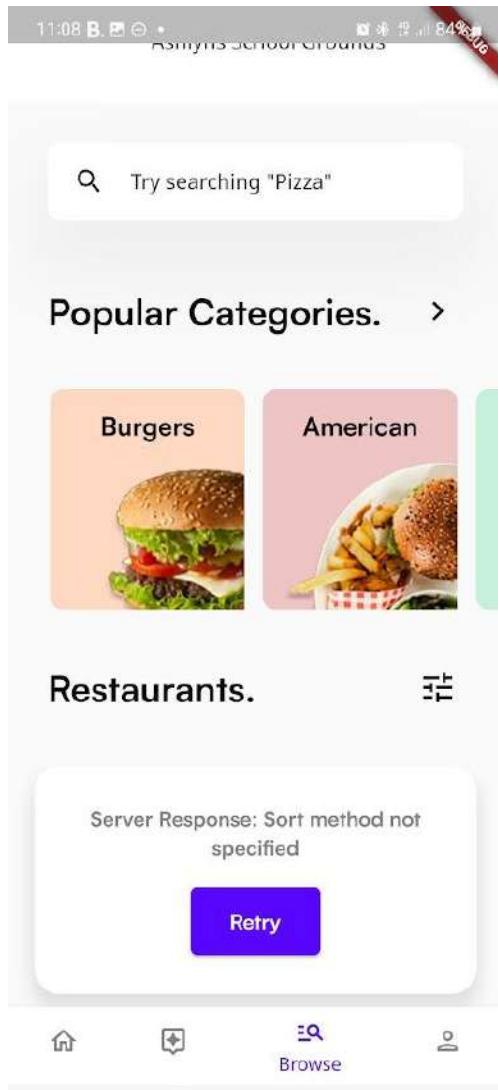
```
metadata["maxDelivery"] = metadata["maxDelivery"].toString();

metadata["minOrder"] = metadata["minOrder"].toString();

metadata["latitude"] = metadata["latitude"].toString();

metadata["longitude"] = metadata["longitude"].toString();
```

After finally getting them all to be Strings, I finally got a valid error.



This error occurred because it passed the validation that it had all the values but did not have a sort method that was defined.

To test to see if the different functions work, I replaced the check for id to distance. This seemed to work as with other sort methods, I would give the above error but with distance it would output the restaurants.

The next thing to do was to convert the strings back into decimals and the list into a series of values.

I used the following resource to help me with that: <https://www.geeksforgeeks.org/how-to-convert-a-string-into-number-in-php/>

```

$sort = $_POST["sort"];
$favourite = $_POST["favourite"];
$maxDelivery = (double)$_POST["maxDelivery"];
$minOrder = (double)$_POST["minOrder"];
$destinationLatitude = (double)$_POST["latitude"];
$destinationLongitude = (double)$_POST["longitude"];
$priceTarget = explode(",", $_POST["price"]); //Convert string of values to list
for($i = 0; $i < $priceTarget; $i++){
    $priceTarget[$i] = (int)$priceTarget[$i]; //Convert List of strings to list of integers
}

```

I will first use if statements to create a list of different commands to perform on the database.

After trying to get the favourites from the server, I realised, it needs the profile id from the app so that was added.

While trying to test favourites, I found that the favourites checkmark kept being reset when it was reopened. I found that the variable was different from the rest where it checked a different variable.

```

if (encodedCustomiseSelected != null) {

    customiseSelected = json.decode(encodedCustomiseSelected.toString());

    _currentMaxDeliveryFeeValue = customiseSelected["maxDelivery"];

    _currentMinOrderPriceValue = customiseSelected["minOrder"];

    isChecked = customiseSelected["favourite"];
}

```

After fixing this, I encountered an Error 500 where after debugging I found that it was related to converting the String to a list of integers. I found this out because when I removed the code for it, it would return data back instead of causing an error.

As it turned out, the issue was because I had forgotten to add .length when doing the for statement so it was causing an error. After fixing this, it started working.

To do the filters, I used the PHP file to run a check on the favourite field and if true, it checks the profile's favourites and filters the result by only showing those.

```

if ($favourite == "true"){
    $sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
}

```

```

profileid = $profileid";
        if($result = mysqli_query($link, $sqlfavrestaurant)){
            while($row = mysqli_fetch_assoc($result)) {
                array_push($favrestaurantslist, $row["restaurantid"]);
            }
        }
        array_push($filterquery, "WHERE res.restaurant IN
$favrestaurantslist");
    }

```

```

array_push($filterquery, "WHERE res.pricebracket IN (" . implode(',', $priceTarget) . ")",
"WHERE res.delivery <= $maxDelivery", "WHERE
res.ordermin <= $minOrder");

```

```
{error: false, message: , restaurants: [], test: [WHERE res.restaurant IN Array, WHERE res.pricebracket IN Array, WHERE res.delivery <= 4, WHERE res.ordermin <= 40]}
```

After returning back the result given, I found that it was just returning the name Array instead of the array. After some research, an article specified that it needs to be imploded before it can be displayed

<https://stackoverflow.com/questions/9618277/how-to-use-php-array-with-sql-in-operator>

After this change, it was fixed:

```
{error: false, message: , restaurants: [], test: [WHERE res.restaurant IN (1,7), WHERE res.pricebracket IN (1,2,3,4), WHERE res.delivery <= 4, WHERE res.ordermin <= 40]}
```

The next part was the sorting. Since there were 4 sorting methods, I first started with the distance.

To do distance, I need to grab all the restaurants, get the distance between the restaurant and the destination. I would then run a query to get only the restaurants that are in the service distance for it. I would then sort the distance in ascending order.

I will use the haversine formula to get the distance between the two points

```

if ($sort == "distance"){
    $sqldistance = "SELECT res.restaurantid, res.restaurantname,
res.restaurantlogo, res.restaurantbanner, res.servicedistance,
res.latitude, res.longitude, 6371 * acos( cos( radians($latitude) ) * cos(

```

```

radians( res.latitude ) ) * cos( radians( res.longitude ) -
radians($longitude) ) + sin( radians($latitude) ) * sin( radians(
res.latitude ) ) ) ) AS distance FROM restaurant res HAVING
res.servicedistance < $distance WHERE (" . implode(' AND ', $filterquery) .
") ORDER BY distance ASC";
    if($result = mysqli_query($link, $sqldistance)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($return["restaurants"],
[$row["restaurantid"], $row["restaurantname"], $row["restaurantlogo"],
$row["restaurantbanner"], $row["latitude"], $row["longitude"]]);
        }
        mysqli_free_result($result);
    }
}

```

After attempting to do a query, the SQL failed. The issue was that the variable names for the latitude and longitude were incorrect. As well, the implode text did not work as intended which resulted in an error. To test for this, I removed the code and made it output in a test return variable.

To fix this, I removed the where part of each statement and made it only add it to the end of the final WHERE and made it have AND instead.

Another issue I found was that the favourites were not working for the filter. This was because I had specified the restaurant, not the restaurant id.

Displaying Delivery Fee on Restaurant List

Another issue which I had to fix was to return the minimum order and delivery price. This meant that I needed to add it in a similar way to the distance but without the calculations. Since only delivery is showing, I first set out to add that. I had to add an if statement to the display where if the price was 0, it would say free delivery instead of £0.00 Delivery

```

Text(
(double.parseDouble(restaurantsdata[0]["restaurants"])

```

[

```
index]

    [
        7]) ==

        0)

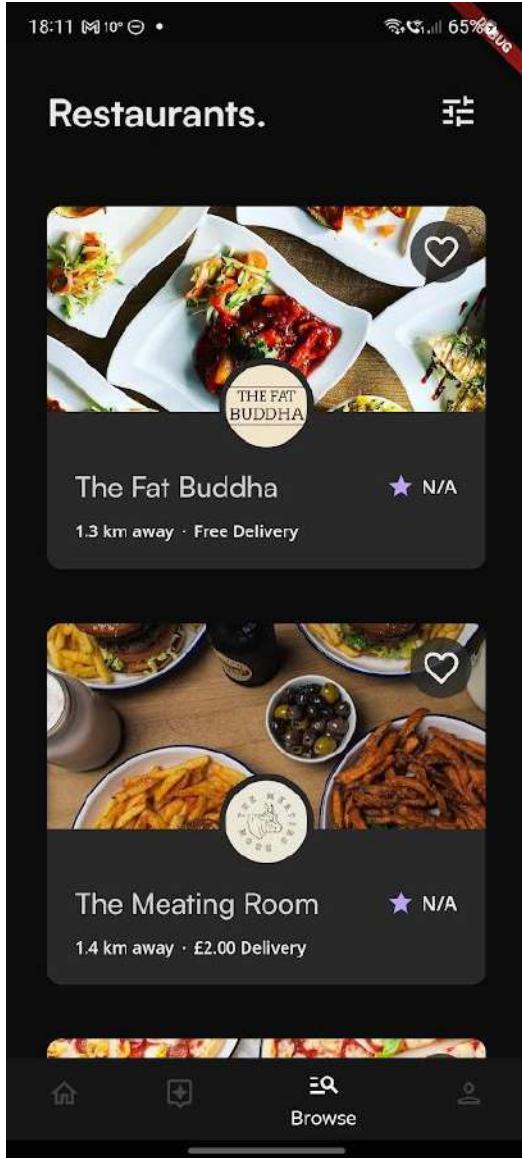
    ? "Free
Delivery"

    :

"${(restaurantsdata[0]["restaurants"][index][7])} Delivery",

        style: Theme.of(
            context)

        .textTheme
        .bodyText1!
        .copyWith(
            fontSize:
            12)),
```



Displaying Basic Data on Restaurant Page

To get the basic data including delivery fee and order minimum, it must be moved over from the restaurant list. To do this, I added them to the stateful widget on the restaurant page, requiring the data to be imported when being called.

```
class RestaurantMain extends StatefulWidget {  
  
    final String resid;  
  
    final String resname;
```

```
final String reslogo;

final String resbanner;

final String resdistance;

final String resdelivery;

final String resordermin;

const RestaurantMain(


    //Get restaurant info from the restaurant list widget (passed from the
    container)

    {

        Key? key,

        required this.resid,

        required this.resname,

        required this.reslogo,

        required this.resbanner,

        required this.resdistance,

        required this.resdelivery,

        required this.resordermin,

    ) : super(key: key);

    @override

    State<RestaurantMain> createState() => _RestaurantMainState();

}
```

I then imported them from the restaurantinfodata from the server query when getting the restaurant info.

```
Navigator.push(  
  context,  
  MaterialPageRoute(  
    builder:  
      (context) => //On tap, send  
      restaurant data associated with container to main page and go to that new screen  
      RestaurantMain(  
        resid:  
          restaurantinfodata[0],  
        resname:  
          restaurantinfodata[1],  
        reslogo:  
          restaurantinfodata[2],  
        resbanner:  
          restaurantinfodata[3],  
        resdistance:  
        (restaurantsdata[  
          0][  
            "restaurants"]  
          [index][8])  
          .toStringAsFixed(1),
```

```
    resdelivery:  
        restaurantinfodata[6],  
  
    resordermin:  
        restaurantinfodata[  
            7]),  
    );
```

Fix to Location Save Button

I found that when the user saves their location, it would not refresh the homepage, to fix this, I made it push the Navigation page again which causes the homepage to open and as a result, everything checks with the updated location.

```
if (hasSavedLocation ==  
    true) {  
  
    setState(() {  
  
        Navigator.of(context)  
            .push(  
                MaterialPageRoute(  
                    builder: (_) =>  
                    const  
Navigation()),  
            );  
    });
```

Fix to Filter & Sort

I found that when the user saves the filters, it does not refresh the previous page, so I made it push the browse page after it closes so that it reopens it.

```
setState(() {  
    Navigator.of(context).push(  
        MaterialPageRoute(builder: (_) => BrowsePage()),  
    );  
});
```

Customise Item Page

Getting Customise Data

To make the customise item page, I start with the restaurant_customise.dart file. Since all the previous data about basic information has already been imported, the next thing to do is create a PHP on the server to get the customised data from the server.

customiseitem

	<input type="checkbox"/>	customised	itemid	customisename	description	viewingorder	optiontype	required	parentid	maxquantity
	<input type="checkbox"/>	Edit	1	Please Choose a Pizza Type		1	SELECT	1	0	
	<input type="checkbox"/>	Edit	2	1 Defaults	Select which what you would like removed	1	REMOVE	0	1	
	<input type="checkbox"/>	Edit	3	1 Optional	Select which what you would like added	2	ADD	0	1 20	
	<input type="checkbox"/>	Edit	4	2 Please Choose a Pizza Type		1	SELECT	1	0	
	<input type="checkbox"/>	Edit	5	2 Defaults	Select which what you would like removed	1	REMOVE	0	4	
	<input type="checkbox"/>	Edit	6	2 Optional	Select which what you would like added	2	ADD	0	4 20	
	<input type="checkbox"/>	Edit	7	3 Please Choose a Pizza Type		1	SELECT	1	0	
	<input type="checkbox"/>	Edit	8	3 Defaults	Select which what you would like removed	1	REMOVE	0	7	
	<input type="checkbox"/>	Edit	9	3 Optional	Select which what you would like added	2	ADD	0	7 20	
	<input type="checkbox"/>	Edit	10	4 Please Choose a Pizza Type		1	SELECT	1	0	
	<input type="checkbox"/>	Edit	11	4 Defaults	Select which what you would like removed	1	REMOVE	0	10	
	<input type="checkbox"/>	Edit	12	4 Optional	Select which what you would like added	2	ADD	0	10 20	
	<input type="checkbox"/>	Edit	13	5 Please Choose a Pizza Type		1	SELECT	1	0	
	<input type="checkbox"/>	Edit	14	5 Defaults	Select which what you would like removed	1	REMOVE	0	13	
	<input type="checkbox"/>	Edit	15	5 Optional	Select which what you would like added	2	ADD	0	13 20	
	<input type="checkbox"/>	Edit	16	9 Defaults	Select which what you would like removed	1	REMOVE	0	0	
	<input type="checkbox"/>	Edit	17	9 Optional	Select which what you would like added	2	ADD	0	0	
	<input type="checkbox"/>	Edit	18	10 Additions	Select which what you would like added	1	ADD	0	0	
	<input type="checkbox"/>	Edit	19	10 Optional	Select which what you would like added	2	ADD	0	0	
	<input type="checkbox"/>	Console	20	Please add your French Fries		1	SELECT	1	0	

customiseitemselection

customiseid	optionnumber	name	Image	pricechange
	1	1 Small Original		0.00
	1	2 Medium Original		2.00
	1	3 Large Original		4.00
	1	4 Medium Authentic Thin Crust		2.00
	1	5 Large Authentic Thin Crust		4.00
	1	6 Medium Stuffed Crust		4.99
	1	7 Large Suffed Crust		6.99
	2	1 Bacon		0.00
	2	2 Base Pizza Sauce		0.00
	2	3 Ham		0.00
	2	4 Mozzarella Cheese		0.00
	2	5 Pepperoni		0.00
	2	6 Pork Sausage		0.00
	2	7 Spicy Beef		0.00
	2	8 Dip Special Garlic		0.00
	3	1 Anchovies		2.00
	3	2 Bacon		2.00
	3	3 Base Pizza Sauce		2.00
	3	4 Barbecue Drizzle		2.00
	3	5 Black Olives		2.00
	3	6 Chargrilled Chicken		2.00
	3	7 Fresh Tomatoes		2.00
	3	8 Green Peppers		3.00
	3	9 Ham		2.00

Using the flowchart made in the design phase can be used for the PHP file where in the flowchart, it shows the layout which can be used.

```
1 {CustomiseTitle1: {
2   Info: ["multi", "optional", true, 0, 20, 1],
3   Options: [Option1, Option2, Option3, Option4],
4   Values: [1,1,0,2]
5 },
6 CustomiseTitle2: {
7   Info: ["add", "required", true, 5, 5, 2],
8   Options: [Option1, Option2, Option3, Option4]
9   Values: [0,2,0,3]
10  Prices: [1.05, 0, 0.75, 1.5, 0]
11 }}
```

After some research, I found that php does not have dictionaries but arrays can be used in a similar way, having an index and key.

<https://stackoverflow.com/questions/6490482/are-there-dictionaries-in-php>

I first start it like any other php file, validating the data exists and creating a connection to the database

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";
$return["customiseitem"] = [];

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["itemid"]);

if($val){
    $itemid = $_POST["itemid"];
}
else{
    $return["error"] = true;
    $return["message"] = "Unknown itemid";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

I then created an sql statement to get the details from the customiseitem table.

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";
$return["customiseitem"] = [];
$return["test"] = [];

$customisetitles = [];

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

```

```

$val = isset($_POST["itemid"]);

if($val){
    $itemid = $_POST["itemid"];
    $sqlcustomiseids = "SELECT * FROM customiseitem WHERE itemid =
$itemid";
    if($result = mysqli_query($link, $sqlcustomiseids)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($return["test"], [$row["customiseid"],
$row["customisename"], $row["description"], $row["viewingorder"],
$row["optiontype"], $row["required"], $row["parentid"],
$row["maxquantity"]]);
        }
        mysqli_free_result($result);
    }
}
else{
    $return["error"] = true;
    $return["message"] = "Unknown itemid";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

To test this code, I create a Future in the customiseitem screen that could be activated by clicking a button.

```

class _RestaurantItemCustomisePageState

extends State<RestaurantItemCustomisePage> {

Future<void> getCustomiseOptions() async {

String phpurl = "https://alleat.cpur.net/query/itemcustomise.php";

```

```

var res =
    await http.post(Uri.parse(phiurl), body: {"itemid": widget.itemid});

if (res.statusCode == 200) {
    //If sends successfully

    var data = json.decode(res.body); //Decode to array

    if (data["error"]) {
        //If fails to perform query

        print("Error: ${data["message"]}");

    } else {
        print("Data: ${data["test"]}");

    }
} else {
    print("SERVER ERROR ${res.statusCode}");

}
}

```

The result is the following:

```

ræd@MainActivity:~(19395)? ViewPostime pointer`1
Data: [[26, Which Base?, , 1, SELECT, 1, 0, ], [27, Add a Side?, , 2, ADD, 0, 0, 1], [35, Would you like to add any extra toppings?, , 1, ADD, 0, 26, 3]]

```

To then get the options for each, you create a for statement for each title and then query the server for the options under each customiseitemid held in index 0.

```

for ($i = 0; $i < count($return["customiseitem"]); $i++){
    $sqlcustomiseoptions = "SELECT * FROM customiseitemselection WHERE
    customiseid = {$return['customiseitem'][$i][0]}";
    if($result2 = mysqli_query($link, $sqlcustomiseoptions)){

```

```

        while($row2 = mysqli_fetch_assoc($result2)) {
            array_push($return[ "customiseitem"][$i][8], [$row2[ "name" ],
$row2[ "image"], $row2[ "pricechange"]]);
        }
    }
}

```

There was a few issues in this process where I did not know that it needed curly brackets so null was being sent back for the value. After this fix though, it worked flawlessly.

```
[[26, Which Base?, , 1, SELECT, 1, 0, , [[Romana Margherita, null, 1.95], [Classic Margherita, null, 0.00], [Gluten-Free Margherita, null, 0.00], [Roma
```

Instead of using a dictionary, I found that the best solution was to use a 4D array since PHP can send it over more easily.

```
[[itemid, title, description, optiontype, required, parentitemid, maxquantity,
[optionname, image, pricechange]]]]
```

Displaying Customise Data

Since an item can have no customise options, it must have another option for if there is an error or not. This is sorted by the error key since it will not fail the SQL query, only resulting in null.

The data is then imported and tested if it is imported

```
Future<List> getCustomiseOptions() async {

    String phpurl = "https://alleat.cpur.net/query/itemcustomise.php";

    var res =
        await http.post(Uri.parse(phpurl), body: {"itemid": widget.itemid});

    if (res.statusCode == 200) {
        //If sends successfully
        var data = json.decode(res.body); //Decode to array
        print(data);
    }
}
```

```

if (data["error"]) {

    List error = [
        {
            "error": true,
            "message": "Server Error: ${data["message"]}",
            "customise": "[]"
        } //Send blank list of customise data
    ];
    return error;
    //If fails to perform query

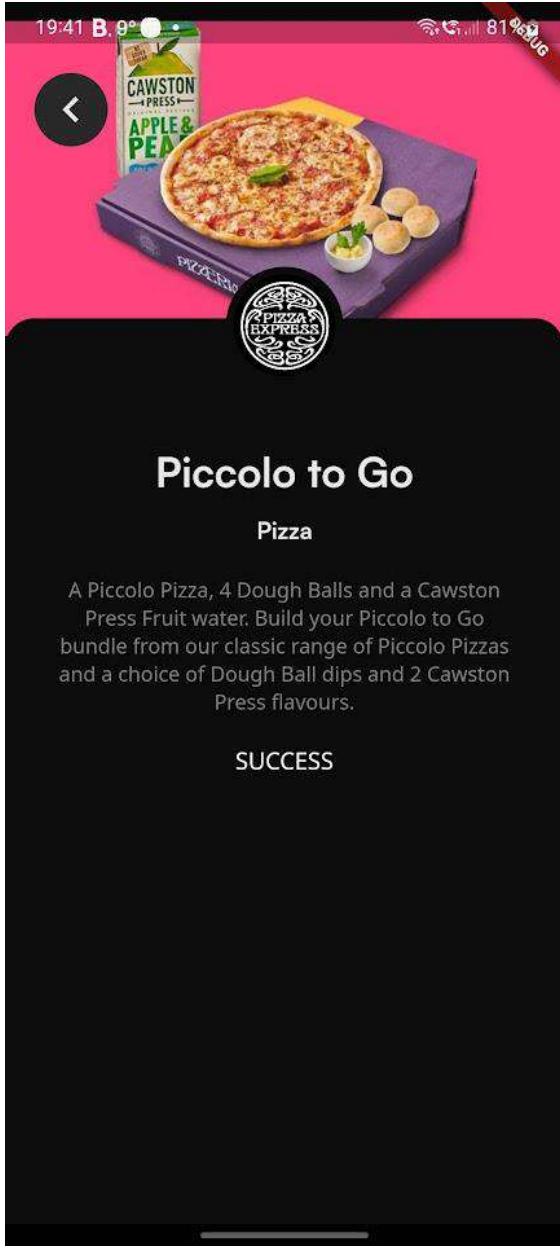
} else {
    List success = [
        {
            "error": false,
            "message": "",
            "customise": data
        } //Send back returned data
    ];
    return success;
}

} else {
    List error = [

```

```
{  
  "error": true,  
  "message": "Error $e: Please try again",  
  "customise": "[]"  
} //Send blank list of customise data  
];  
return error;  
}  
}
```

```
FutureBuilder(  
  future: getCustomiseOptions(),  
  builder: (context, snapshot) {  
    if (!snapshot.hasData) {  
      return const LinearProgressIndicator(  
        color: Color(0xff4100C4),  
        backgroundColor: Color(0xffEBE0FF));  
    } else {  
      return Text("SUCCESS");  
    }  
  })
```



Add to Cart Button

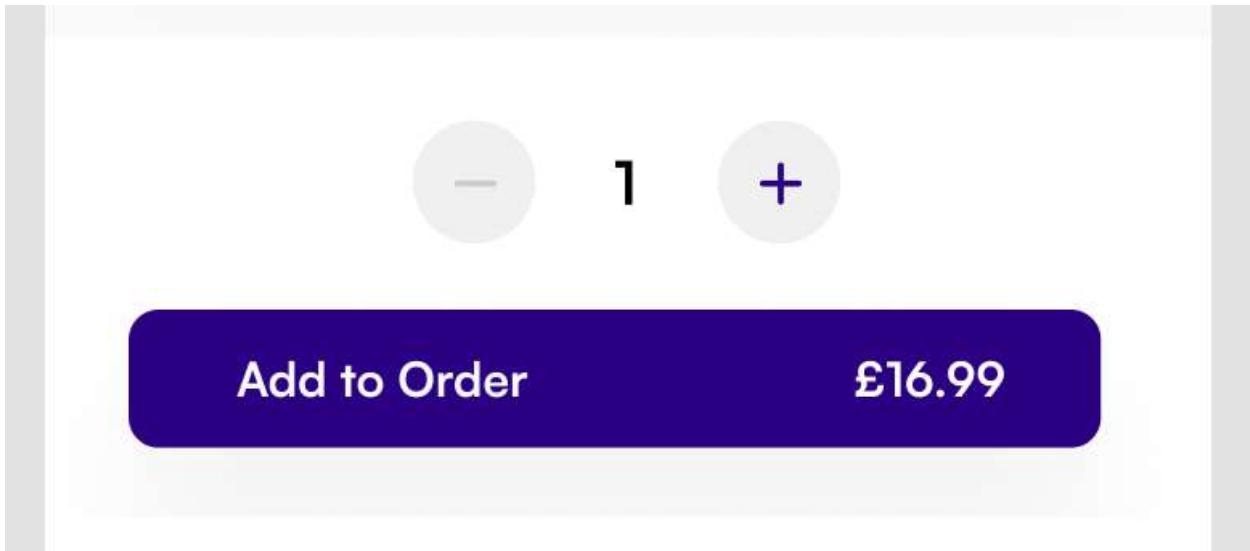
If there is no data received, this means there is nothing to be returned so only add to cart is added.

```
if (customiseData[0]["customise"].length == 0) {  
  
    return Padding(  
  
        padding: const EdgeInsets.all(20),
```

```
child: Row(children: [  
    Expanded(  
        child: ElevatedButton(  
            onPressed: () {  
                print("Success");  
            },  
            child: const Text("Add to Cart"))  
    )),
```

To do quantity, I will first start with a class bound variable called quantity. The quantity will then be displayed above the add to cart, similar to the concept art, with the + and - sign changing the variable by one.

Concept Art



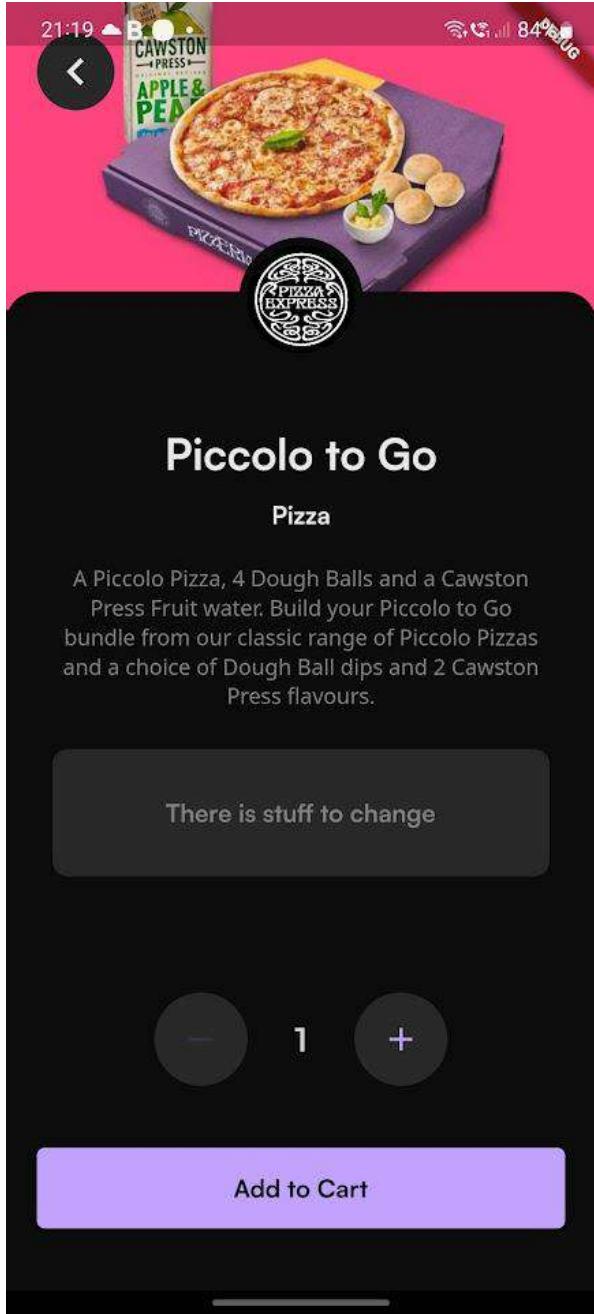
Since the price relies on the customised data, I first created the basic format, using IconButton.

```
Column(crossAxisAlignment: CrossAxisAlignment.center, children: [
```

```
const SizedBox(height: 50),  
  
Padding(  
  
    padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 20),  
  
    child: Row(  
  
        mainAxisAlignment: MainAxisAlignment.center,  
  
        children: [  
  
            CircleAvatar(  
  
                radius: 30,  
  
                backgroundColor: Theme.of(context).colorScheme.onSurface,  
  
                child: IconButton(  
  
                    icon: const Icon(Icons.remove),  
  
                    iconSize: 25,  
  
                    color: (quantity == 1)  
  
                        ? Theme.of(context).primaryColor.withOpacity(0.1)  
  
                        : Theme.of(context).primaryColor,  
  
                    onPressed: () {  
  
                        if (quantity != 1) {  
  
                            setState(() {  
  
                                quantity -= 1;  
  
                            });  
  
                        }  
  
                    },  
  
            ),  
  
            Text("x " + quantity.toString()),  
  
            IconButton(  
  
                icon: const Icon(Icons.add),  
  
                iconSize: 25,  
  
                color: Theme.of(context).primaryColor,  
  
                onPressed: () {  
  
                    setState(() {  
  
                        quantity += 1;  
  
                    });  
  
                },  
  
            ),  
  
        ],  
  
    ),  
  
),
```

```
    )),  
  
    const SizedBox(  
  
        width: 30,  
  
    ),  
  
    Text(  
  
        quantity.toString(),  
  
        style: Theme.of(context).textTheme.headline3,  
  
    ),  
  
    const SizedBox(  
  
        width: 30,  
  
    ),  
  
    CircleAvatar(  
  
        radius: 30,  
  
        backgroundColor: Theme.of(context).colorScheme.onSurface,  
  
        child: IconButton(  
  
            icon: const Icon(Icons.add),  
  
            iconSize: 25,  
  
            color: (quantity == 9)  
  
                ? Theme.of(context).primaryColor.withOpacity(0.1)  
  
                : Theme.of(context).primaryColor,  
  
            onPressed: () {  
  
                if (quantity != 9) {
```

```
        setState(() {
            quantity += 1;
        });
    }
},
),
],
)),
Padding(
padding: const EdgeInsets.all(20),
child: Row(children: [
Expanded(
child: ElevatedButton(
 onPressed: () {
print("Success");
},
child: const Text("Add to Cart")))
]),
const SizedBox(height: 20)
]),
```



Determining Customise Type

To determine the customise type, I use an if statement to look at index 3 and return a different display option for each option.

In order to cut down the amount of code within a single file, it will use a separate file for each option and pass each customise to it.

I used a ListView to build each widget separately.

```
return ListView.builder(  
  
    physics:  
  
        const NeverScrollableScrollPhysics(), //Disable  
scrolling. Scroll with whole page  
  
    scrollDirection: Axis.vertical,  
  
    shrinkWrap: true,  
  
    itemCount: (customiseData[  
  
        "customise"] //For each item, create a container  
  
.length),  
  
    itemBuilder: ((context, index) {  
  
        print(index);  
  
        if (customiseData["customise"][index][3] ==  
  
            "SELECT") {  
  
            return customiseSelectBuild(customiseSelectData: [  
  
                customiseData["customise"][index]  
  
            ]);  
  
        } else if (customiseData["customise"][index][3] ==  
  
            "ADD") {  
  
            return customiseAddBuild(customiseAddData: [  
  
                customiseData["customise"][index]  
  
            ]);  
  
        } else if (customiseData["customise"][index][3] ==
```

```
    "REMOVE") {  
  
    return customiseRemoveBuild(customiseRemoveData: [  
  
        customiseData["customise"][index]  
    ]);  
  
} else {  
  
    return Container(  
  
        width: double.infinity,  
  
        padding: const EdgeInsets.symmetric(  
  
            vertical: 30, horizontal: 20),  
  
        decoration: BoxDecoration(  
  
            borderRadius: const BorderRadius.all(  
  
                Radius.circular(10)),  
  
            color:  
  
                Theme.of(context).colorScheme.onSurface),  
  
        margin: const EdgeInsets.symmetric(  
  
            vertical: 5, horizontal: 30),  
  
        child: Text(  
  
            "Unknown Customise Option", //Return that there  
was an error  
  
            textAlign: TextAlign.center,  
  
            style: Theme.of(context).textTheme.headline6,  
        ),  
    );  
}
```

```
}
```

```
});
```

```
import 'package:flutter/material.dart';

class customiseSelectBuild extends StatefulWidget {

  final List customiseSelectData;

  const customiseSelectBuild({
    Key? key,
    required this.customiseSelectData,
  }) : super(key: key);

  @override
  State<customiseSelectBuild> createState() => _customiseSelectBuildState();
}

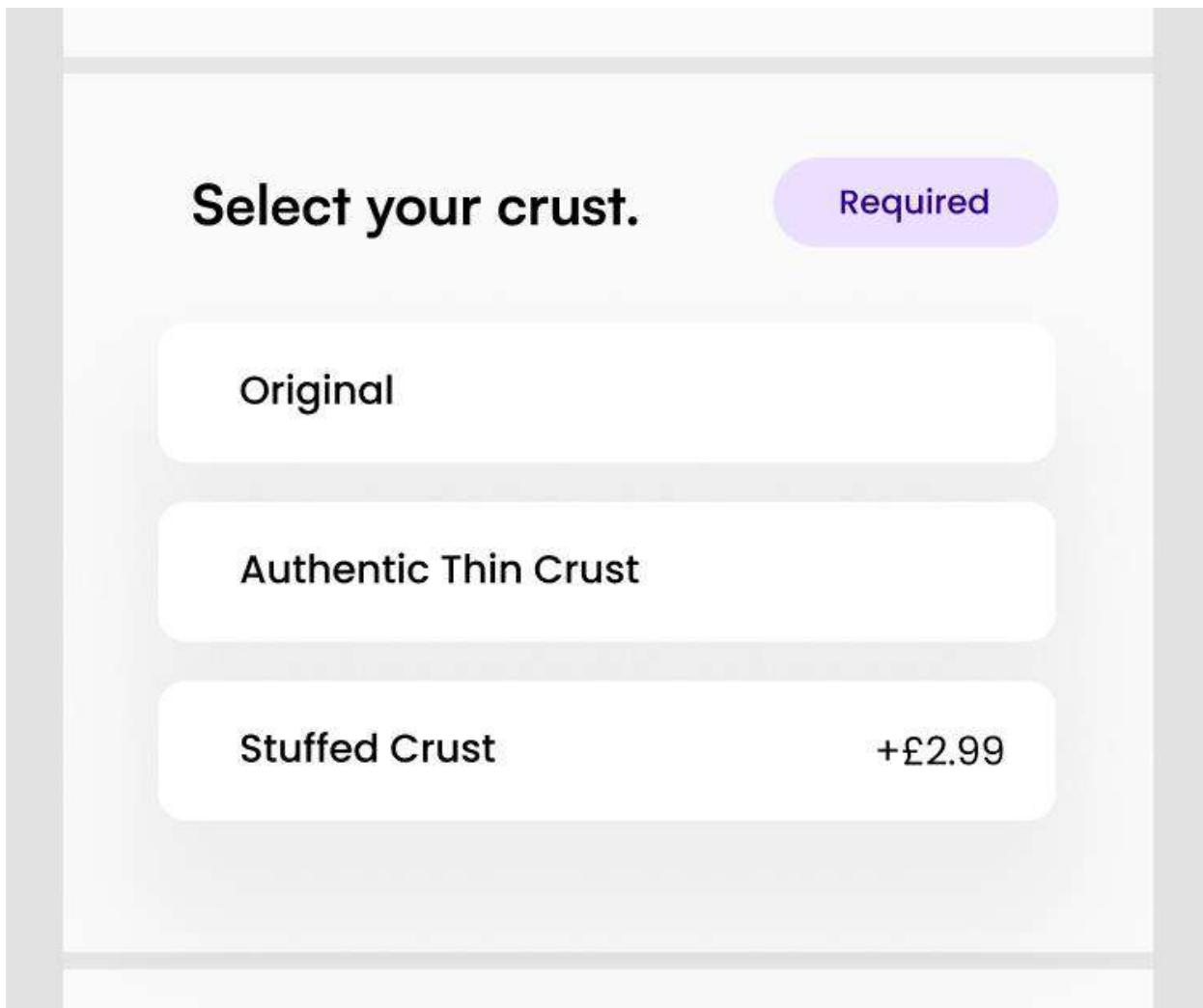
class _customiseSelectBuildState extends State<customiseSelectBuild> {

  @override
  Widget build(BuildContext context) {
    return Container(
      child: Text("Select widget"),
    );
}
```

```
    }  
}  
}
```

Customise Select Widget

Taking a look at the concept art, we see that it is built up with a column, having two horizontal lines above and below the widget. It then has a row with the name and the type of option it is (required or optional). It will then contain a ListView with a padding around each button with an if statement for each ElevatedButton for if it is selected or not.



```
import 'package:flutter/material.dart';
```

```
class CustomiseSelectBuild extends StatefulWidget {

  final List customiseSelectData;

  const CustomiseSelectBuild({
    Key? key,
    required this.customiseSelectData,
  }) : super(key: key);

  @override
  State<CustomiseSelectBuild> createState() => _CustomiseSelectBuildState();
}

class _CustomiseSelectBuildState extends State<CustomiseSelectBuild> {

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Padding(
          padding: const EdgeInsets.only(left: 20, right: 20, top: 30, bottom: 10),
          child: Row(crossAxisAlignment: CrossAxisAlignment.start, children: [
            Expanded(
```

```
        child: Column(children: [
          Text(
            widget.customiseSelectData[0][1].toString(),
            style: Theme.of(context).textTheme.headline5,
            overflow: TextOverflow.visible,
          ),
          Text(
            widget.customiseSelectData[0][2].toString(),
            style: Theme.of(context).textTheme.bodyText1,
            overflow: TextOverflow.visible,
          ),
        ])),
      const SizedBox(
        width: 20,
      ),
    ],
  ),
LayoutBuilder(builder: ((context, constraints) {
  if (widget.customiseSelectData[0][4] == "0") {
    return Container(
      decoration: BoxDecoration(
        color: Theme.of(context)
          .colorScheme
          .tertiary
    );
  }
}))
```

```
.withOpacity(0.2),  
  
borderRadius: BorderRadius.circular(20)),  
  
padding:  
  
    const EdgeInsets.symmetric(horizontal: 20, vertical: 5),  
  
child: Text(  
  
    "Optional",  
  
    style: Theme.of(context).textTheme.headline6?.copyWith(  
  
        color: Theme.of(context).colorScheme.tertiary),  
  
(),  
  
);  
  
} else if (widget.customiseSelectData[0][4] == "1") {  
  
    return Container(  
  
        decoration: BoxDecoration(  
  
            color: Theme.of(context).primaryColor.withOpacity(0.2),  
  
            borderRadius: BorderRadius.circular(20)),  
  
        padding:  
  
            const EdgeInsets.symmetric(horizontal: 20, vertical: 5),  
  
        child: Text(  
  
            "Required",  
  
            style: Theme.of(context)  
  
.textTheme  
  
.headline6
```

```
    ?.copyWith(color: Theme.of(context).primaryColor),  
 ),  
 );  
 } else {  
 return Container(  
 decoration: BoxDecoration(  
 color:  
     Theme.of(context).colorScheme.error.withOpacity(0.2),  
 borderRadius: BorderRadius.circular(20)),  
 padding:  
     const EdgeInsets.symmetric(horizontal: 20, vertical: 5),  
 child: Text(  
     "ERROR",  
 style: Theme.of(context)  
         .textTheme  
         .headline6  
     ?.copyWith(color: Theme.of(context).colorScheme.error),  
 ),  
 );  
 }  
 }));  
 ],
```

```
),

ListView.builder(
    physics:
        const NeverScrollableScrollPhysics(), //Disable scrolling. Scroll
with whole page

    scrollDirection: Axis.vertical,
    shrinkWrap: true,
    itemCount: (widget
        .customiseSelectData[0][7] //For each item, create a container
        .length),
    itemBuilder: ((context, index) {
        return Padding(
            padding:
                const EdgeInsets.symmetric(horizontal: 20, vertical: 5),
            child: ElevatedButton(
                style: ButtonStyle(
                    alignment: Alignment.centerLeft,
                    padding: MaterialStateProperty.all(
                        const EdgeInsets.symmetric(
                            horizontal: 30, vertical: 20)),
                    textStyle: MaterialStateProperty.all(
                        Theme.of(context).textTheme.headline6),
                    backgroundColor: MaterialStateProperty.all(

```

```
        Theme.of(context).colorScheme.onSurface),  
  
        onPressed: () {  
  
            print("Success");  
  
        },  
  
        child: Text(widget.customiseSelectData[0][7][index][0],  
  
                    style: Theme.of(context).textTheme.headline6),  
  
    )),  
  
    const SizedBox(  
  
        height: 50,  
  
    )  
  
],  
  
);  
  
}  
}
```

17:28 B! 7% our classic range of Piccolo to Go Dough Balls with 2 Cawston Press dips and 2 Cawston Press flavours.

Please pick your
Dough Balls

Required

Piccolo Dough Balls with Garlic
Butter

Piccolo Dough Balls with Houmous

Please pick your
Piccolo to Go Pizza

Required

Piccolo American

Piccolo La Reine

Piccolo Margherita

Piccolo Pollo

In order to build the selection part of it, I have to make the main customise file save the currently selected part.

NOTE: After doing some thinking, I realised that in order to save customisation options correctly, I would need to create a proper primary key for the options so that they can be saved uniquely since they currently do not have that. This will allow for inheritance

To do multi-item selection, we first have an empty map on the main customise page. For each selection widget, we add the customise option id as the key to the map with an empty list as the value.

```
import 'dart:math';

import 'package:alleat/widgets/elements/customise_add.dart';

import 'package:alleat/widgets/elements/customise_remove.dart';

import 'package:alleat/widgets/elements/customise_select.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

import 'package:shared_preferences/shared_preferences.dart';

class RestaurantItemCustomisePage extends StatefulWidget {

    final String itemid;

    final String foodcategory;

    final String subfoodcategory;

    final String itemname;

    final String description;

    final String price;

    final String itemimage;

    final String reslogo;
```

```
const RestaurantItemCustomisePage( {Key? key, //Get the items from the restaurant main page when an item is clicked required this.reslogo, required this.itemid, required this.foodcategory, required this.subfoodcategory, required this.itemname, required this.description, required this.price, required this.itemimage}) : super(key: key);

@override State<RestaurantItemCustomisePage> createState() => _RestaurantItemCustomisePageState(); }

class _RestaurantItemCustomisePageState extends State<RestaurantItemCustomisePage> {
int quantity = 1;
```

```

Future<Map> getCustomiseOptions() async {

    try {

        String phpurl = "https://alleat.cpur.net/query/itemcustomise.php";

        var res =

            await http.post(Uri.parse(phpurl), body: {"itemid": widget.itemid});

        if (res.statusCode == 200) {

            //If sends successfully

            var data = json.decode(res.body); //Decode to array

            if (data["error"]) {

                Map error = {

                    "error": true,

                    "message": "Server Error: ${data["message"]}",

                    "customise": "[]"

                } //Send blank list of customise data

                ;

            }

            return error;

            //If fails to perform query

        } else {

            Map success = {

                "error": false,

                "message": "",

                "customise": data["customiseitem"]

            }

        }

    }

}

```

```
        } //Send back returned data

        ;

        return success;

    }

} else {

    Map error = {

        "error": true,

        "message": "Error $e: Please try again",

        "customise": "[]"

    } //Send blank list of customise data

    ;

    return error;

}

} catch (e) {

    Map error = {

        "error": true,

        "message": "Unexpected Error: $e",

        "customise": "[]"

    } //Send blank list of customise data

    ;

    return error;

}
```

```
}

Future<void> storeTempCustomise() async {

    final prefs = await SharedPreferences.getInstance();

    Map customisedOptions = {};

    var encodedCustomisedOptions = json.encode(customisedOptions);

    await prefs.setString('tempCustomisedOptions', encodedCustomisedOptions);

}

@Override

Widget build(BuildContext context) {

    return Scaffold(

        body: SingleChildScrollView(

            child: Column(

                children: [

                    Stack(children: [

                        //Display item image at bottom of stack

                        Container(

                            width: MediaQuery.of(context).size.width,

                            height: 240,

                            decoration: BoxDecoration(

                                image: DecorationImage(
```

```
        fit: BoxFit.fitWidth,  
  
        image: NetworkImage(widget.itemimage),  
  
    ),  
  
    ),  
  
    ),  
  
Align(  
  
    //Display rounded container at bottom of image to respresent  
overhanging image  
  
    alignment: Alignment.bottomCenter,  
  
    child: Container(  
  
        margin: const EdgeInsets.only(top: 215),  
  
        width: double.infinity,  
  
        height: 30,  
  
        decoration: BoxDecoration(  
  
            color: Theme.of(context).backgroundColor,  
  
            borderRadius:  
  
                const BorderRadius.vertical(top: Radius.circular(20))),  
  
    )),  
  
Align(  
  
    // Display background color as outline of restaurant logo,  
overlapping the image background  
  
    alignment: Alignment.bottomCenter,  
  
    child: Container(  

```

```
margin: const EdgeInsets.only(top: 180),  
  
width: 80,  
  
height: 80,  
  
decoration: BoxDecoration(  
  
shape: BoxShape.circle,  
  
color: Theme.of(context).backgroundColor,  
)),  
  
Align(  
  
// Display circle restaurant logo  
  
alignment: Alignment.bottomCenter,  
  
child: Container(  
  
margin: const EdgeInsets.only(top: 185),  
  
width: 70,  
  
height: 70,  
  
decoration: BoxDecoration(  
  
image: DecorationImage(  
  
fit: BoxFit.cover,  
  
image: NetworkImage(widget.reslogo.toString()),  
),  
  
shape: BoxShape.circle,  
  
color: Theme.of(context).colorScheme.onSurface,  
)),
```

```
SafeArea(  
    // Display back button in a circle  
    child: InkWell(  
        onTap: () => Navigator.of(context).pop(),  
        child: Container(  
            margin: const EdgeInsets.only(top: 20, left: 20),  
            width: 50,  
            height: 50,  
            decoration: BoxDecoration(  
                shape: BoxShape.circle,  
                color: Theme.of(context).colorScheme.onSurface,  
            ),  
            child: Icon(  
                Icons chevron_left,  
                color: Theme.of(context).colorScheme.onBackground,  
                size: 35,  
            ),  
        )),  
    ]),  
    Padding(  
        // Item name  
        padding:
```

```

        const EdgeInsets.only(top: 40, left: 30, right: 30, bottom: 10),

        child: Text(
            widget.itemname,
            textAlign: TextAlign.center,
            style: Theme.of(context).textTheme.headline2,
        )),

Row(mainAxisAlignment: MainAxisAlignment.center, children: [
    //Display restuarant category(ies)

    Text(widget.foodcategory, //Must have food category
        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.headline6!.copyWith(
            color: Theme.of(context).textTheme.headline1!.color)),
    LayoutBuilder(builder: (context, constraints) {
        if (widget.subfoodcategory != "") {
            //If there is a subcategory, display it with a dot next to it
            //If there is a sub food category, show it
            return Text(" · ${widget.subfoodcategory}",
                textAlign: TextAlign.center,
                style: Theme.of(context).textTheme.headline6!.copyWith(
                    color: Theme.of(context).textTheme.headline1!.color));
        } else {
            // If there is no sub-category, dont display anything

```

```

        //If there isn't a sub-food category, don't show it

        return const Text("");
    }

})

]),

Padding(
    //Display item description

    padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 20),

    child: Text(
        widget.description,
        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.bodyText1!.copyWith(
            color: Theme.of(context).textTheme.headline6!.color,
            fontWeight: FontWeight.w400),
    ),
),

FutureBuilder<Map>(
    future: getCustomiseOptions(),
    builder: (context, snapshot) {
        if (snapshot.hasData) {
            Map customiseData = snapshot.data ?? [] as Map;
            if (customiseData["error"] == true) {

```

```

//Check if there was an error getting the data from the server

return Container(
    width: double.infinity,
    padding: const EdgeInsets.symmetric(
        vertical: 30, horizontal: 20),
    decoration: BoxDecoration(
        borderRadius:
            const BorderRadius.all(Radius.circular(10)),
        color: Theme.of(context).colorScheme.onSurface),
    margin:
        const EdgeInsets.symmetric(vertical: 5, horizontal: 30),
    child: Text(
        customiseData["message"], //Return that there was an error
        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.headline6,
    ),
);

} else {
    Map customisedOptions = {};
}

//If there was not an error getting the data

if (customiseData["customise"].length == 0) {

```

```
//if there is no customise options, return nothing

return const SizedBox(
  height: 1,
);

} else {

  //If there are customise options, return customise options
  return ListView.builder(
    physics:
      const NeverScrollableScrollPhysics(), //Disable
scrolling. Scroll with whole page

    scrollDirection: Axis.vertical,
    shrinkWrap: true,
    itemCount: (customiseData[
      "customise"] //For each item, create a container
      .length),
    itemBuilder: ((context, index) {
      return Column(
        children: [
          Container(
            width: double.infinity,
            height: 10,
            color: Theme.of(context)
              .colorScheme
        ],
      );
    })
  );
}
```

```
.onBackground  
.withOpacity(0.1),  
,  
  
LayoutBuilder(builder: ((context, constraints) {  
  
if (customiseData["customise"][index][3] ==  
"SELECT") {  
  
return CustomiseSelectBuild(  
  
customiseSelectData: [  
  
customiseData["customise"][index]  
, customisedOptions: customisedOptions);  
  
} else if (customiseData["customise"][index]  
[3] ==  
"ADD") {  
  
return CustomiseAddBuild(customiseAddData: [  
  
customiseData["customise"][index]  
]);  
  
} else if (customiseData["customise"][index]  
[3] ==  
"REMOVE") {  
  
return CustomiseRemoveBuild(  
  
customiseRemoveData: [
```

```
        customiseData["customise"][index]

    ]);

} else {

    return Container(
        width: double.infinity,
        padding: const EdgeInsets.symmetric(
            vertical: 30, horizontal: 20),
        decoration: BoxDecoration(
            borderRadius: const BorderRadius.all(
                Radius.circular(10)),
            color: Theme.of(context)
                .colorScheme
                .onSurface),
        margin: const EdgeInsets.symmetric(
            vertical: 5, horizontal: 30),
        child: Text(
            "Unknown Customise Option", //Return that
there was an error

            textAlign: TextAlign.center,
            style:
                Theme.of(context).textTheme.headline6,
        ),
    );
}
```

```
        }

    }))

    ],
);

}));
```

}

}

} else {

//While loading, return progress indicator

```
return const LinearProgressIndicator(
    color: Color(0xff4100C4),
    backgroundColor: Color(0xffEBE0FF));
}
```

}),

```
Column(crossAxisAlignment: CrossAxisAlignment.center, children: [
    //Add to cart and quantity
    Padding(
        padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 20),
        child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                CircleAvatar(

```

```
        radius: 30,  
  
        backgroundColor: Theme.of(context).colorScheme.onSurface,  
  
        child: IconButton(  
  
            icon: const Icon(Icons.remove),  
  
            iconSize: 25,  
  
            color: (quantity == 1)  
  
                ? Theme.of(context).primaryColor.withOpacity(0.1)  
  
                : Theme.of(context).primaryColor,  
  
            onPressed: () {  
  
                if (quantity != 1) {  
  
                    setState(() {  
  
                        quantity -= 1;  
  
                    });  
  
                }  
  
            },  
  
        )),  
  
        const SizedBox(  
  
            width: 30,  
  
        ),  
  
        SizedBox(  
  
            width: 30,  
  
            child: Text(  
    
```

```
        quantity.toString(),

        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.headline3,
      )),
    const SizedBox(
      width: 30,
    ),
    CircleAvatar(
      radius: 30,
      backgroundColor: Theme.of(context).colorScheme.onSurface,
      child: IconButton(
        icon: const Icon(Icons.add),
        iconSize: 25,
        color: (quantity == 99)
          ? Theme.of(context).primaryColor.withOpacity(0.1)
          : Theme.of(context).primaryColor,
        onPressed: () {
          if (quantity != 99) {
            setState(() {
              quantity += 1;
            });
          }
        }
      )
    );
  }
}
```

```
        },
      )),
    ],
  )),
Padding(
  padding: const EdgeInsets.all(20),
  child: Row(children: [
    Expanded(
      child: ElevatedButton(
        onPressed: () {}, child: const Text("Add to Cart")))
  ]),
const SizedBox(height: 20)
]),
],
));
}
}
```

```
import 'package:flutter/material.dart';

class CustomiseSelectBuild extends StatefulWidget {
```

```
final List customiseSelectData;

final Map customisedOptions;

const CustomiseSelectBuild(
    {Key? key,
     required this.customiseSelectData,
     required this.customisedOptions})
    : super(key: key);

}

@Override
State<CustomiseSelectBuild> createState() => _CustomiseSelectBuildState();
```

}

```
class _CustomiseSelectBuildState extends State<CustomiseSelectBuild> {

    @override
    Widget build(BuildContext context) {
        widget.customisedOptions[widget.customiseSelectData[0][0]] = [];
        return Column(
            children: [
                Padding(
                    padding:
                        const EdgeInsets.only(left: 20, right: 20, top: 30, bottom: 10),
                    child: Row(crossAxisAlignment: CrossAxisAlignment.start, children: [
```

```
Expanded(  
    child: Column(children: [  
        Text(  
            widget.customiseSelectData[0][1].toString(),  
            style: Theme.of(context).textTheme.headline5,  
            overflow: TextOverflow.visible,  
        ),  
        Text(  
            widget.customiseSelectData[0][2].toString(),  
            style: Theme.of(context).textTheme.bodyText1,  
            overflow: TextOverflow.visible,  
        ),  
    ])),  
    const SizedBox(  
        width: 20,  
    ),  
    LayoutBuilder(builder: ((context, constraints) {  
        if (widget.customiseSelectData[0][4] == "0") {  
            return Container(  
                decoration: BoxDecoration(  
                    color: Theme.of(context)  
                        .colorScheme  
                ),  
            );  
        } else {  
            return Container(  
                decoration: BoxDecoration(  
                    color: Theme.of(context).  
                        colorScheme  
                ),  
            );  
        }  
    })),  
);
```

```
.tertiary

.withOpacity(0.2),

borderRadius: BorderRadius.circular(20)),

padding:

const EdgeInsets.symmetric(horizontal: 20, vertical: 5),

child: Text(

"Optional",

style: Theme.of(context).textTheme.headline6?.copyWith(

color: Theme.of(context).colorScheme.tertiary),


),


);

} else if (widget.customiseSelectData[0][4] == "1") {

return Container(

decoration: BoxDecoration(

color: Theme.of(context).primaryColor.withOpacity(0.2),


borderRadius: BorderRadius.circular(20)),


padding:

const EdgeInsets.symmetric(horizontal: 20, vertical: 5),


child: Text(


"Required",

style: Theme.of(context)

.textTheme
```

```
        .headline6

        ?.copyWith(color: Theme.of(context).primaryColor),

    ),

};

} else {

    return Container(
        decoration: BoxDecoration(
            color:

                Theme.of(context).colorScheme.error.withOpacity(0.2),
            borderRadius: BorderRadius.circular(20)),
        padding:
            const EdgeInsets.symmetric(horizontal: 20, vertical: 5),
        child: Text(
            "ERROR",
            style: Theme.of(context)
                .textTheme
                .headline6
                ?.copyWith(color: Theme.of(context).colorScheme.error),
        ),
    );
}

}))
```

```
]),  
)  
  
ListView.builder(  
  
    physics:  
  
        const NeverScrollableScrollPhysics(), //Disable scrolling. Scroll  
with whole page  
  
    scrollDirection: Axis.vertical,  
  
    shrinkWrap: true,  
  
    itemCount: (widget.customiseSelectData[0][7].length),  
  
    itemBuilder: ((context, index) {  
  
        return Padding(  
  
            padding:  
  
                const EdgeInsets.symmetric(horizontal: 20, vertical: 5),  
  
            child: ElevatedButton(  
  
                style: ButtonStyle(  
  
                    side: (widget.customisedOptions[  
  
                        widget.customiseSelectData[0][0]]  
  
.contains(  
  
                        widget.customiseSelectData[0][7][index]))  
  
                    ? MaterialStateProperty.all(BorderSide(  
  
                        width: 2, color: Theme.of(context).primaryColor))  
  
                    : null,  
  
                    alignment: Alignment.centerLeft,  
    
```

```
padding: MaterialStateProperty.all(  
  
    const EdgeInsets.symmetric(  
  
        horizontal: 30, vertical: 20)),  
  
    textStyle: MaterialStateProperty.all(  
  
        Theme.of(context).textTheme.headline6),  
  
    backgroundColor: MaterialStateProperty.all(  
  
        Theme.of(context).colorScheme.onSurface),  
  
>,  
  
onPressed: () {  
  
    setState(() {  
  
        if (widget  
  
            .customisedOptions[  
  
                widget.customiseSelectData[0][0]]  
  
.length <  
  
int.parse(widget.customiseSelectData[0][6])) &&  
  
!widget.customisedOptions[  
  
    widget.customiseSelectData[0][0]]  
  
.contains(  
  
    widget.customiseSelectData[0][7][index])) {  
  
widget.customisedOptions[widget.customiseSelectData[0]  
  
[0]]  
  
.add(widget.customiseSelectData[0][7][index]);  
});  
},
```

```
        print(widget.customisedOptions[  
            widget.customiseSelectData[0][0]]);  
  
        print(widget.customiseSelectData[0][7][index]);  
    } else {  
  
        widget.customisedOptions[widget.customiseSelectData[0]  
            [0]]  
  
        .remove(widget.customiseSelectData[0][7][index]);  
    }  
});  
},  
  
child: Text(widget.customiseSelectData[0][7][index][1],  
  
style: Theme.of(context).textTheme.headline6),  
));  
}}),  
  
const SizedBox(  
  
height: 50,  
)  
],  
);  
}  
}  
}
```

After a day of testing, I was not successful at making the data persist so instead, I moved the data directly to the page.

```
LayoutBuilder(builder: ((context, constraints) {  
  
    if (customiseData["customise"][index][3] ==  
        "SELECT") {  
  
        return Column(  
            children: [  
  
                Padding(  
                    padding: const EdgeInsets.only(  
                        left: 20,  
                        right: 20,  
                        top: 30,  
                        bottom: 10),  
  
                    child: Row(  
                        crossAxisAlignment:  
                            CrossAxisAlignment.start,  
                        children: [  
                            Expanded(  
                                child: Column(children: [  
                                    Text(  
                                        customiseData["customise"]  
                                            [index][1]  
                                ]  
                            )  
                        ]  
                    )  
                )  
            ]  
        );  
    }  
});
```

```
        .toString(),  
  
        style: Theme.of(context)  
  
            .textTheme  
  
            .headline5,  
  
        overflow:  
  
            TextOverflow.visible,  
  
) ,  
  
Text(  
  
    customiseData["customise"]  
  
    [index][2],  
  
    style: Theme.of(context)  
  
        .textTheme  
  
        .headline5,  
  
    overflow:  
  
        TextOverflow.visible,  
  
)  
]),  
  
const SizedBox(width: 20),  
  
LayoutBuilder(builder:  
  
    ((context, constraints) {  
  
        if (customiseData["customise"]  
  
            [index][4] ==
```

```
"0") {  
  
    return Container(  
  
        decoration: BoxDecoration(  
  
            color: Theme.of(context)  
  
                .colorScheme  
  
                .tertiary  
  
                .withOpacity(0.2),  
  
            borderRadius:  
  
                BorderRadius  
  
                .circular(20)),  
  
        padding: const EdgeInsets  
  
            .symmetric(  
  
                horizontal: 20,  
  
                vertical: 5),  
  
        child: Text(  
  
            "Optional",  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline6  
  
                ?.copyWith(  
  
                    color: Theme.of(  
  
                        context)
```

```
.colorScheme  
    .tertiary),  
),  
);  
} else if (customiseData[  
    "customise"][[index]  
[4] ==  
"1") {  
return Container(  
    decoration: BoxDecoration(  
        color: Theme.of(context)  
            .primaryColor  
            .withOpacity(0.2),  
        borderRadius:  
            BorderRadius  
            .circular(20)),  
    padding: const EdgeInsets  
        .symmetric(  
            horizontal: 20,  
            vertical: 5),  
    child: Text(  
        "Required",
```

```
        style: Theme.of(context)

            .textTheme

            .headline6

            ?.copyWith(
                color: Theme.of(
                    context)
                    .primaryColor),
            ),
        );
    } else {
        return Container(
            decoration: BoxDecoration(
                color: Theme.of(context)
                    .colorScheme
                    .error
                    .withOpacity(0.2),
            borderRadius:
                BorderRadius
                    .circular(20)),
            padding: const EdgeInsets
                    .symmetric(
                        horizontal: 20,
```

```
        vertical: 5),  
  
        child: Text(  
  
            "ERROR",  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline6  
  
                ?.copyWith(  
  
                    color: Theme.of(  
  
                        context)  
  
                        .colorScheme  
  
                        .error),  
  
                ),  
  
            );  
  
        }  
  
    }));  
  
],  
  
,  
  
ListView.builder(  
  
physics:  
  
const  
NeverScrollableScrollPhysics(), //Disable scrolling. Scroll with whole page  
  
scrollDirection: Axis.vertical,  
  
shrinkWrap: true,
```

```
itemCount: customiseData["customise"]  
[index][7]  
.length,  
itemBuilder: ((context, index2) {  
    return Padding(  
        padding:  
            const EdgeInsets.symmetric(  
                horizontal: 20,  
                vertical: 5),  
        child: ElevatedButton(  
            style: ButtonStyle(  
                side:  
(customisedOptions[customiseData["customise"]][index][0]).contains(  
customiseData["customise"]  
[index][7]  
[index2]  
[0]))  
?  
MaterialStateProperty.all(BorderSide(  
width: 2,  
color:  
Theme.of(context)
```

```
.primaryColor))

        : null,

        alignment:
Alignment.centerLeft,

        padding:
MaterialStateProperty.all(const EdgeInsets.symmetric(horizontal: 30, vertical:
20)),

        textStyle:
(customisedOptions[customiseData["customise"][index][0]].contains(customiseData["
customise"][index][7][index2][0])) ?
MaterialStateProperty.all(Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).textTheme.headline1?.color)) :
MaterialStateProperty.all(Theme.of(context).textTheme.headline6),

        backgroundColor:
MaterialStateProperty.all(Theme.of(context).colorScheme.onSurface)),

        onPressed: () {

        if
(customisedOptions[customiseData["customise"][index][0]]


.int.parse(customiseData[

"customise"]

        [index][6]) &&

        !customisedOptions[

customiseData["customise"]]
```

```
[index]  
[0]]  
  
.contains(customiseData["customise"]  
[index][7]  
[index2])) {  
    setState(() {  
        customisedOptions[  
            customiseData[  
"customise"]  
[  
    index][0]]  
.add(customiseData[  
"customise"]  
[index][7]  
[index2][0]);  
});  
} else {  
    setState(() {  
        customisedOptions[  
            customiseData[
```

```
"customise"]  
[  
    index][0]]  
  
.remove(customiseData[  
  
"customise"]  
[  
  
index][7]  
[  
    index2][0]  
.toString());  
});  
}  
},  
child: Text(  
    customiseData["customise"]  
    [index][7][index2][1],  
    style: (customisedOptions[  
  
customiseData["customise"]  
[index]
```

```
[0]]  
  
.contains(customiseData["customise"]  
[index][7]  
[index2][0]))  
? Theme.of(context)  
.textTheme  
.headline6  
?.copyWith(  
color:  
Theme.of(context)  
.textTheme  
.headline1  
?.color)  
: Theme.of(context)  
.textTheme  
.headline6,  
));  
}),  
const SizedBox(height: 50)  
],  
);  
}
```

Customise Add Widget

To do the add widget, I first copied over the select widget and added a count to get the number of values.

```
LayoutBuilder(builder:  
    ((context,  
        constraints) {  
        int count = 0;  
        for (int i = 0;  
            i <  
            customisedOptions[  
                customiseData["customise"]  
                    [  
                        index][0]]  
                            .length;  
                            i++) {  
                            if (customisedOptions[  
                                customiseData["customise"]  
                                    [  
                                        [
```

```

        index]

[0]][i] ==

customiseData[

"customise"]

[index][7]

[index2][0])) {

count += 1;

}

}

return Text(
    count.toString(),
    style:
        Theme.of(context)

        .textTheme

        .headline6,

);

}))
```

What I found was that it would add up to 2 but would go back down. To fix this, I would need to disable removals from button clicking and make it part of a button. I used the basis of the quantity add button to create the remove button and moved the remove algorithm to the button.

```
IconButton(
```

```
        icon: const Icon(  
          Icons.delete),  
        iconSize: 25,  
        color: Theme.of(context)  
          .colorScheme  
          .error,  
        onPressed: () {  
          setState(() {  
  
customisedOptions[customiseData[  
  "customise"]  
    [index][0]]  
  
.remove(customiseData["customise"]  
    [  
      index][7]  
    [  
      index2][0]  
        .toString());  
      });  
    },
```

```
 ),
```

The next thing to do is to make the add widget show up like an add widget to ensure that it is easy to understand that it is adding not removing. To do this, when an option is not selected, it will show an add button instead of the quantity. I used the count function which was used to display the changing quantity, to instead change the whole function if it was 0.

```
child: LayoutBuilder(builder:  
    ((context, constraints) {  
        int count = 0;  
        for (int i = 0;  
             i < customisedOptions[  
                customiseData[  
                    "customise"]  
                [  
                    index][0]]  
                .length;  
             i++) {  
            if (customisedOptions[  
                customiseData[  
                    "customise"]]
```

```
[index]

[0]][i] ==

customiseData[

"customise"]

[index][7]

[index2][0]) {

count += 1;

}

if (count == 0) {

return Row(


mainAxisAlignment:

MainAxisAlignment

.spaceBetween,


children: [


Icon(


Icons


.add_box_outlined,


color: Theme.of(


context)

.colorScheme
```

```
        .tertiary,  
        ),  
        const SizedBox(  
            width: 30),  
        Expanded(  
            child: Text(  
                customiseData[  
  
"customise"]  
                [index][7]  
                [index2][1],  
                style:  
(customisedOptions[  
  
customiseData["customise"][index][  
                    0]]  
  
.contains(customiseData["customise"][index][7]  
[index2]  
                    [0]))  
?  
Theme.of(context)  
        .textTheme  
        .headline6
```

```
? .copyWith(  
    color:  
Theme.of(context)  
  
.textTheme  
  
.headline1  
  
? .color)  
  
:  
Theme.of(context)  
  

```

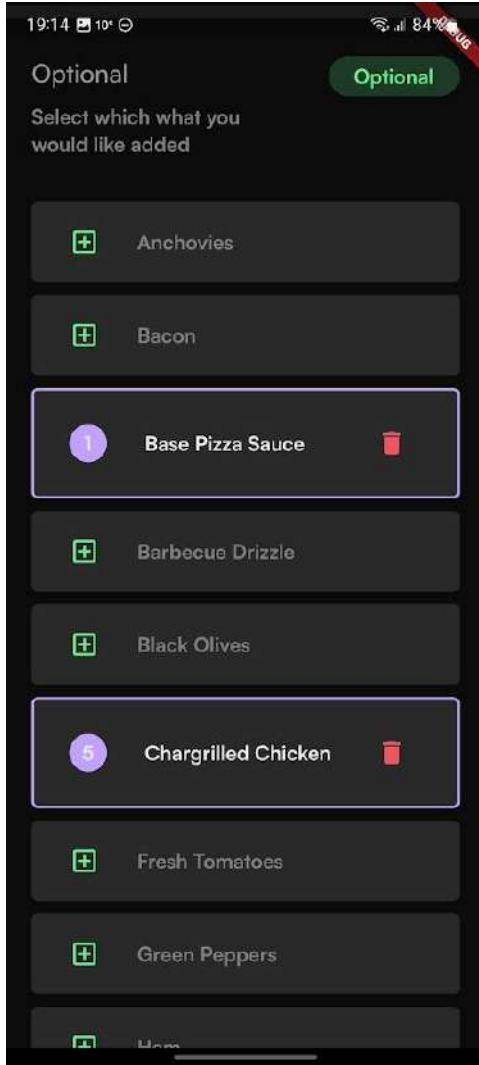
```
        radius: 15,  
  
        backgroundColor:  
  
        Theme.of(  
  
context)  
  
.primaryColor,  
  
        child: Text(  
  
        count  
  
.toString(),  
  
        style:  
Theme.of(  
  
context)  
  
.textTheme  
  
.headline6  
  
? .copyWith(  
  
        color:  
Theme.of(context)  
  

```

```
        ),  
  
        const SizedBox(  
            width: 30,  
        ),  
  
        Expanded(  
            child: Text(  
                customiseData[  
  
"customise"]  
  
                [index][7]  
  
                [index2][1],  
  
                style:  
                (customisedOptions[  
  
                    customiseData["customise"][index][  
  
                        0]]  
  
.contains(customiseData["customise"][index][7]  
  
[index2]  
  
[0]))  
?  
Theme.of(context)  
  
.textTheme  
  
.headline6
```

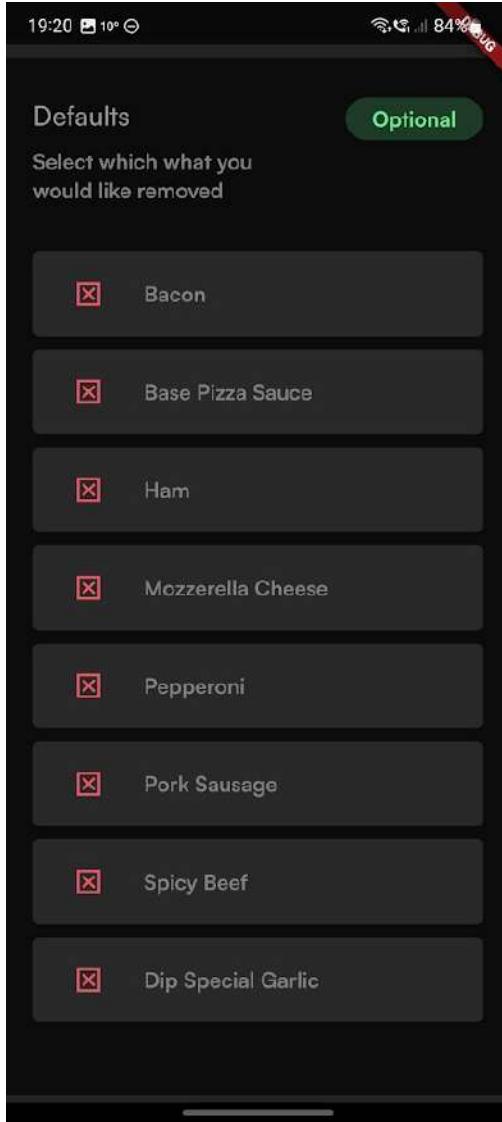
```
? .copyWith(  
    color:  
Theme.of(context)  
  
.textTheme  
  
.headline1  
  
?.color)  
  
:  
Theme.of(context)  
  
.textTheme  
  
.headline6,  
()),  
const SizedBox(  
    width: 20),  
    IconButton(  
        icon: const  
Icon(Icons  
    .delete),  
    splashRadius: 15,  
    color: Theme.of(  
        context)  
    .colorScheme  
    .error,
```

```
        onPressed: () {
            setState(() {
                customisedOptions[
                    customiseData["customise"][index]
                        [
                            0]];
                .remove(customiseData["customise"][index][7][index2]
                    [
                        0]);
                .toString());
            });
        },
    ],
]);
}
})),
));
})),
});
```



Customise Remove Widget

To do this function, I copied the exact same function used by the add function but replaced the icons



Price Changes

To do price changes we first create a LayoutBuilder to create the price at the bottom of the screen, by default, it will show the original price. For every select it will add by the value selected and for every remove it will remove by that much and for every add, it will add to the price.

```
LayoutBuilder(builder: ((p0, p1) {  
  
    double finalSinglePrice = double.parse(widget.price);  
  
    return Text("£${finalSinglePrice.toString()}",  
  
        style: Theme.of(context).textTheme.headline4);  
})
```

```
 })),
```

In order to not cause an error when calculating the price, I will create an double variable that by default starts with 0 and is added onto the original price. When a user clicks on a price, it is added or removed from the variable. This means there is no extra calculations needed.

With selection, in order to stop repeated removals, instead of using a plain else statement, I will use a check, to check if it is in the list to start off with.

Trying to do the following did not work, failing to pass, so I investigated on why that was true

```
else if (customisedOptions[  
    "customise"]  
        .contains(customiseData[  
            "customise"]  
                [index][0])) {
```

As it turns out, it was checking for the wrong value (first value is the list, second is the value being checked)

```
[258]  
[258, Romana Margherita, null, 1.95]
```

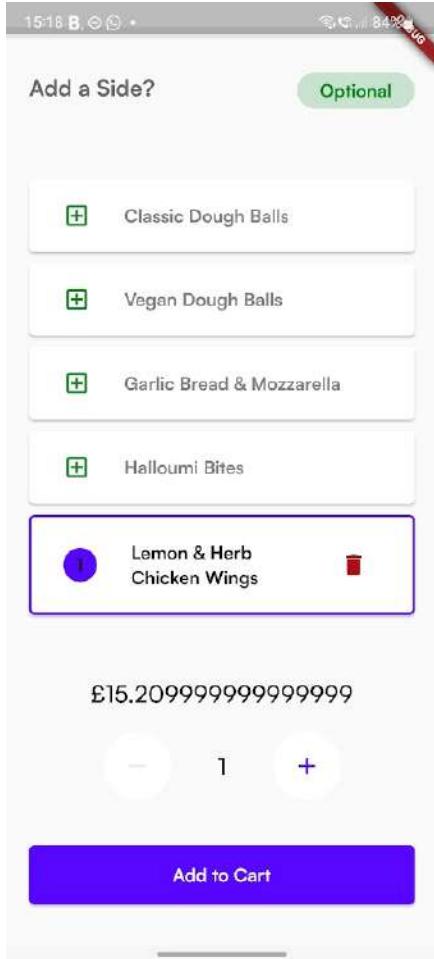
After some more testing and trial and error, the fixed code is as follows

```
customisedOptions[  
      
        customiseData[  
              
                "customise"]  
                  
                [index][0]]  
              
            .contains(customiseData[  
                  
                "customise"]  
                  
                [index][7]  
              
            [index2][0])
```

To add to the price, I used this:

```
changingPrice += double  
      
.parse(customiseData[  
      
        "customise"]  
          
        [index][7]  
      
    [index2][3]);
```

An issue I had was that for some of the prices, it would display as a long decimal:



This occurred when the value was not a whole number. What did not make sense was that displaying the parsed value would still display the value as a normal value.

According to this article, they say that the best way is to multiply them by 100 and add them then divide them by 100 again in order to fix it.

<https://stackoverflow.com/questions/69858616/why-cant-dart-calculate-simple-double-value>

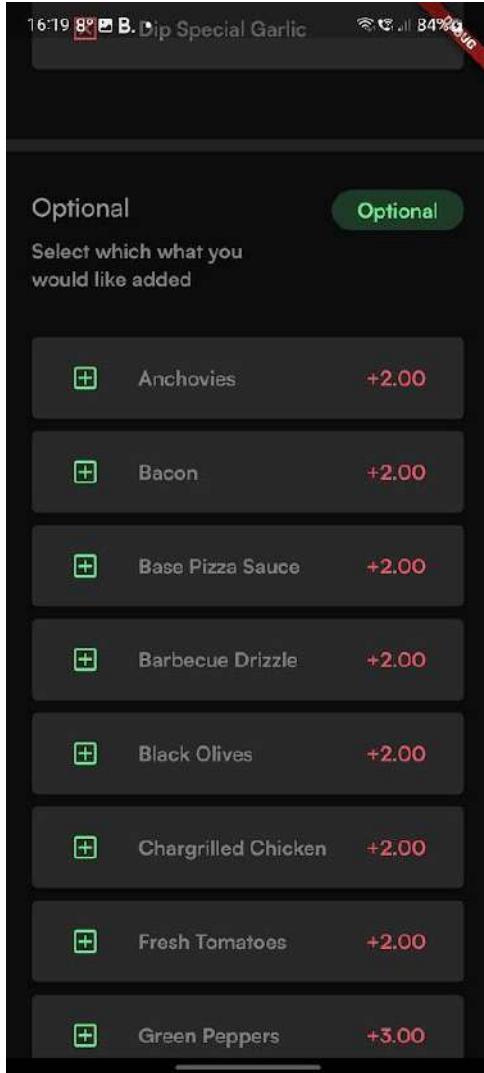
Another problem to fix is that it will display to the number of decimal places necessary, so sometimes the price will display 1 decimal or 0 decimal places. To fix this, I will round the value to 2 decimal places

```
Text("£${finalSinglePrice.toStringAsFixed(2)}",
      style: Theme.of(context).textTheme.headline4);
```

In order to show the user when the prices are going to change, I will display the price change on the right.

```
LayoutBuilder(builder: ((p0, p1) {  
  
    if (customiseData[  
        "customise"]  
        [index][7]  
        [index2][3] !=  
        "0.00"){  
  
        return Text(  
            "+${customiseData[  
                "customise"]  
                [index][7]  
                [index2][3]}",  
            style: Theme.of(context)  
                .textTheme  
                .headline6  
                ?.copyWith(  
                    color: Theme.of(  
                        context)  
                        .colorScheme  
                        .error),  
            );}  
    else{  
        return Text(  
    }  
});
```

```
"-",  
    style: Theme.of(context)  
        .textTheme  
        .headline6  
    ?.copyWith(  
        color: Theme.of(  
            context)  
        .colorScheme  
  
.onBackground.withOpacity(0.5)),);  
}  
}  
}
```

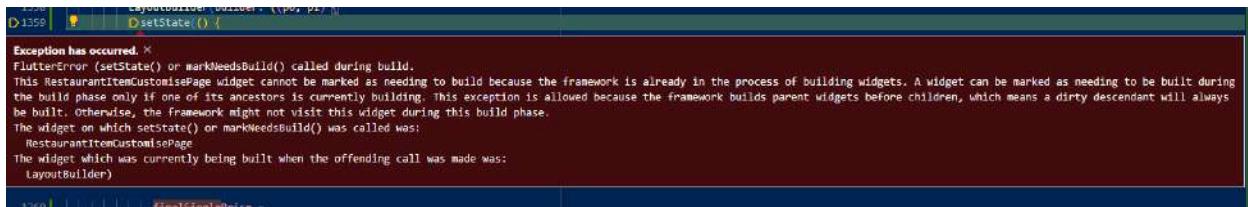


In order to help the user know the price of the number of items they are adding to the cart, the quantity must be multiplied by the final price.

```
Text(  
    "£${(finalSinglePrice *  
quantity).toStringAsFixed(2)}")
```

While testing, I found that when the item is opened, the price would be shown as 0 since the price is first set as 0. To fix this, I first made the default value, the price and also made a setState for when the finalSinglePrice updates.

After doing that, I got this error:



I decided that it would be easier to move the entire section within the LayoutBuilder instead. This fixed my code.

While testing, I found that I forgot to add an Expanded widget so if the text was too long, the text would overlap. The Expanded widget was added.

Required Fields

Required fields are those that are required before an item can be added to the cart and are stored in `customiseData["customise"][index][4]`

To do a check for if the required fields have been filled, for each item in the list, if it has `customiseData["customise"][index][4]` equal to 1, it will check the `customisedOption` for the dictionary id and see if it is not null.

While testing with required fields, it would consistently display that it has all the correct required fields filled. This was incorrect and as it turned out, it was not putting the required fields in the list

```
onPressed: () {  
  
    bool tempCheckRequiredFilled = true;  
  
    print("${requiredFields}");  
  
    for (int i = 0; i < requiredFields.length; i++) {  
  
        if (customisedOptions[requiredFields[i]].isEmpty) {  
  
            tempCheckRequiredFilled = false;  
  
        }  
  
    }  
}
```

```
if (tempCheckRequiredFilled == true) {  
  
    ScaffoldMessenger.of(context).showSnackBar(  
  
        const SnackBar(content: Text("Success")));  
  
} else {  
  
    ScaffoldMessenger.of(context).showSnackBar(  
  
        const SnackBar(  
  
            content: Text("Required not filled")));  
  
}  
  
,
```

As it turned out, I tested for an integer instead of a string

```
if (customiseData["customise"][index][4] == "1") {  
    requiredFields  
        .add(customiseData["customise"][index][0]);  
}  
};
```

One issue I found was that for every rebuild, it would add the values again.

To fix this, I also have it check if the value is already in the list. This fixed the issue

```
if (customiseData["customise"][index][4] == "1" &&
!requiredFields.contains(customiseData["customise"][index][0])) {
    requiredFields
        .add(customiseData["customise"][index][0]);
}
```

```
        }
```

```
3ffae4[MainActivity
): [20, 21, 22]
```

To make a visual difference for when the required fields are complete, I used a LayoutBuilder to do the same check that was within the button but instead, have it display different buttons dependent on if the required fields were complete.

```
LayoutBuilder(builder: ((context, constraints) {

    bool tempCheckRequiredFilled = true;

    print("${requiredFields}");

    for (int i = 0; i < requiredFields.length; i++) {

        if (customisedOptions[requiredFields[i]].isEmpty) {

            tempCheckRequiredFilled = false;

        }

    }

    if (tempCheckRequiredFilled == true) {

        return ElevatedButton(

            onPressed: () {

                ScaffoldMessenger.of(context).showSnackBar(
                    const SnackBar(content: Text("Success")));

            },

            child: Row(
                mainAxisSize: MainAxisSize.spaceBetween,
```

```
        children: [  
  
            const Text("Add to Cart"),  
  
            Text("£${quantityPrice.toStringAsFixed(2)}")  
        ]));  
  
    } else {  
  
        return Opacity(  
  
            opacity: 0.2,  
  
            child: ElevatedButton(  
  
                style: ButtonStyle(  
  
                    backgroundColor: MaterialStatePropertyAll(  
  
                        Theme.of(context)  
  
                            .colorScheme  
  
                            .onBackground)),  
  
                onPressed: () {},  
  
                child: Row(  
  
                    mainAxisAlignment:  
  
                        MainAxisAlignment.spaceBetween,  
  
                    children: [  
  
                        const Text("Add to Cart"),  
  
                        Text("£${quantityPrice.toStringAsFixed(2)}")  
                    ]));  
  
    }  

```

```
 }))
```

Saving Customised Item

To save the customised item we need to store the data on the device permanently since the user may want to close the app before finishing the order.

The following needs to be saved:

- Cart ID (Primary key)
- Item ID
- List with the customised options
- Quantity

When the cart is loaded, it should get the data from the server to ensure that it gets the most accurate information with the correct prices and it ensures that if the

```
import 'package:sqflite/sqflite.dart' as sql;

class SQLiteCartItems {

    // -----
    // Cart Items Table
    // -----



    static Future<void> createTableCartItems(sql.Database database) async {

        //Create cart table (Used to store items and their customised details)
        await database.execute("""
            CREATE TABLE cartItems(
                cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
                itemid INT,
                """)
```



```
static Future<void> addToCart(  
    int itemid, dynamic customised, int quantity) async {  
  
    final db = await SQLiteCartItems.db();  
  
    db.insert("cartItems", {  
  
        "itemid": itemid,  
  
        "customised": customised,  
  
        "quantity": quantity,  
  
    });  
  
}  
}
```

```
onPressed: () async {  
  
    try {  
  
        String customisedOptionsEncoded =  
            json.encode(customisedOptions);  
  
        await SQLiteCartItems.addToCart(  
  
            int.parse(widget.itemid),  
  
            customisedOptionsEncoded,  
  
            quantity);  
  
    } catch (e) {
```

```
ScaffoldMessenger.of(context).showSnackBar(  
    SnackBar(content: Text("Error: $e")));  
}  
},
```

When I try to add the item, it says the table is unavailable.

```
77[MainActivity](17227): ViewPostIME pointer 1  
(1) no such table: cartItems in "INSERT INTO cartItems (itemid, customised, quantity) VALUES (?, ?, ?)"
```

To fix this, I made both the LocalProfileDatabase call the creation of the database and the cart to be called so that depending on which is called on app creation, it creates both of them. This fixed my issue:

```
static Future<sql.Database> cartdb() async {  
  
    //If tables don't exist, create tables  
  
    return sql.openDatabase(  
  
        'alleatlocal.db',  
  
        version: 1,  
  
        onCreate: (sql.Database database, int version) async {  
  
            await SQLiteLocalProfiles.createTableProfile(database);  
  
            await createTableCartItems(database);  
  
        },  
  
    );  
}
```

Post-Development

Testing

Testing Summary

Test Number	Success Criteria	What it is	Expected	Actual	Pass/Fail
1	1.3	The database should be in third-normal form and normalised	The database should have no repeated data	The data has no repeated data	PASS
2	3.1	The user should be able to sort restaurants using a button	The user should have options to sort.	The user has the ability to sort with different options although it is currently only sorting by distance	PASS
3	3.2	When sorting restaurants, it should allow the user to sort by recommended, rating or distance	There should be options to sort by recommended, rating or distance	There should be options to sort by recommended, rating or distance	PASS
4	3.4	The user should be able to filter restaurants using a button	There should be different sections for each filter with the ability to change each	There are filters for price, minimum order price, maximum delivery fee and favourites which are all functioning. There is no	PASS
5	3.5	When filtering restaurants, it should allow	There should be filters for price range, delivery,		PASS

		the user to filter by price range, delivery, pickup, delivery price, new	pickup, delivery price, new	pickup option or filter by only new	
6	3.7	Before filtering, there should be a button to confirm that they would like to filter the restaurant.	There should be a button where the user can confirm their options.	The user must press the save the option or if they want to revert, they must put the filters and sorts in the same combination as they had previously	PASS
7	3.8	When the user has selected some restaurant filters, the user should have a button they can click to clear the filters and have it show all the restaurants	There should be a clear filters button which resets the filters and sorts back to default	There is a button within the filter and sort screen which resets the filters and sort methods back to the default settings	PASS
8	3.9	When the user has selected a restaurant filter, they should be	The filters should be able to be changed	The filters can be changed and it will update when the page is refreshed	PASS

		able to unselect a filter and change the settings		
--	--	---	--	--

Testing Criteria (1.3)

The database should be in third-normal form and normalised

Justification

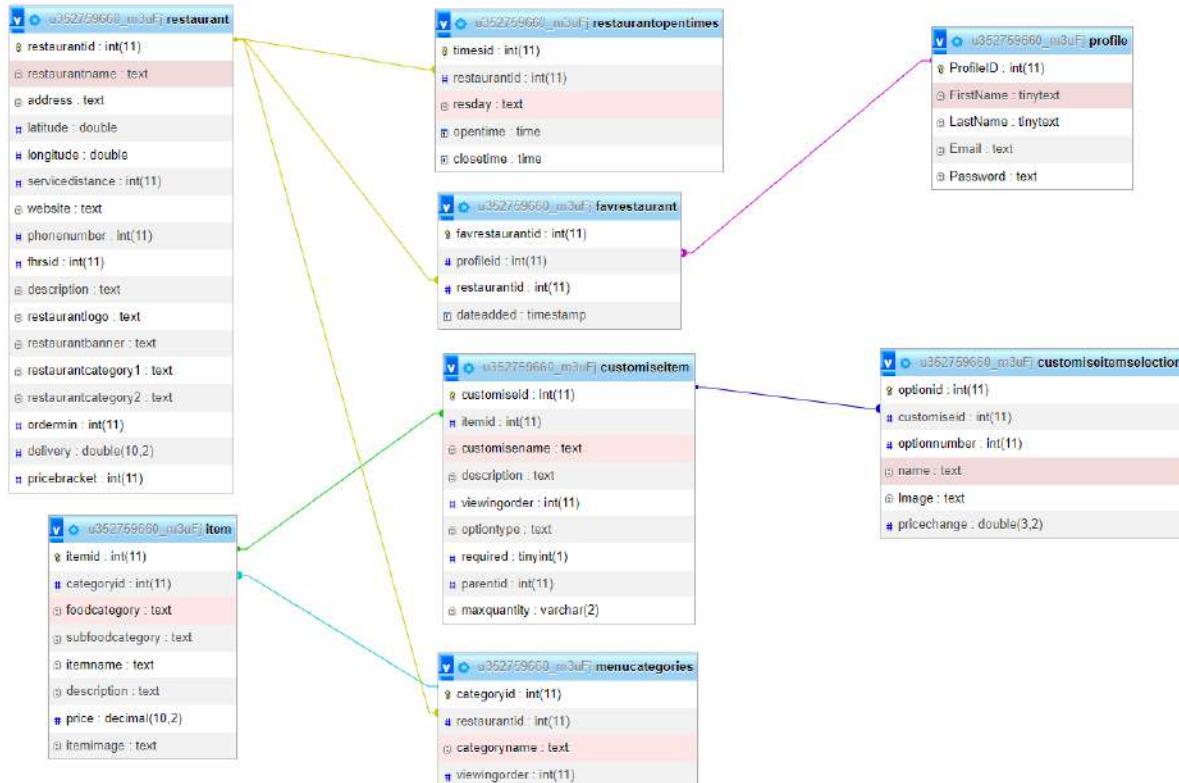
This makes the database not have repeated data so that query time is reduced

Expected result

The data is third-normal form

Actual result

The database uses third-normal, using foreign keys to create relationships between the tables



Pass or Fail?

— PASS — (Revisit in future)

Testing Criteria (3.1)

The user should be able to sort restaurants using a button

Justification

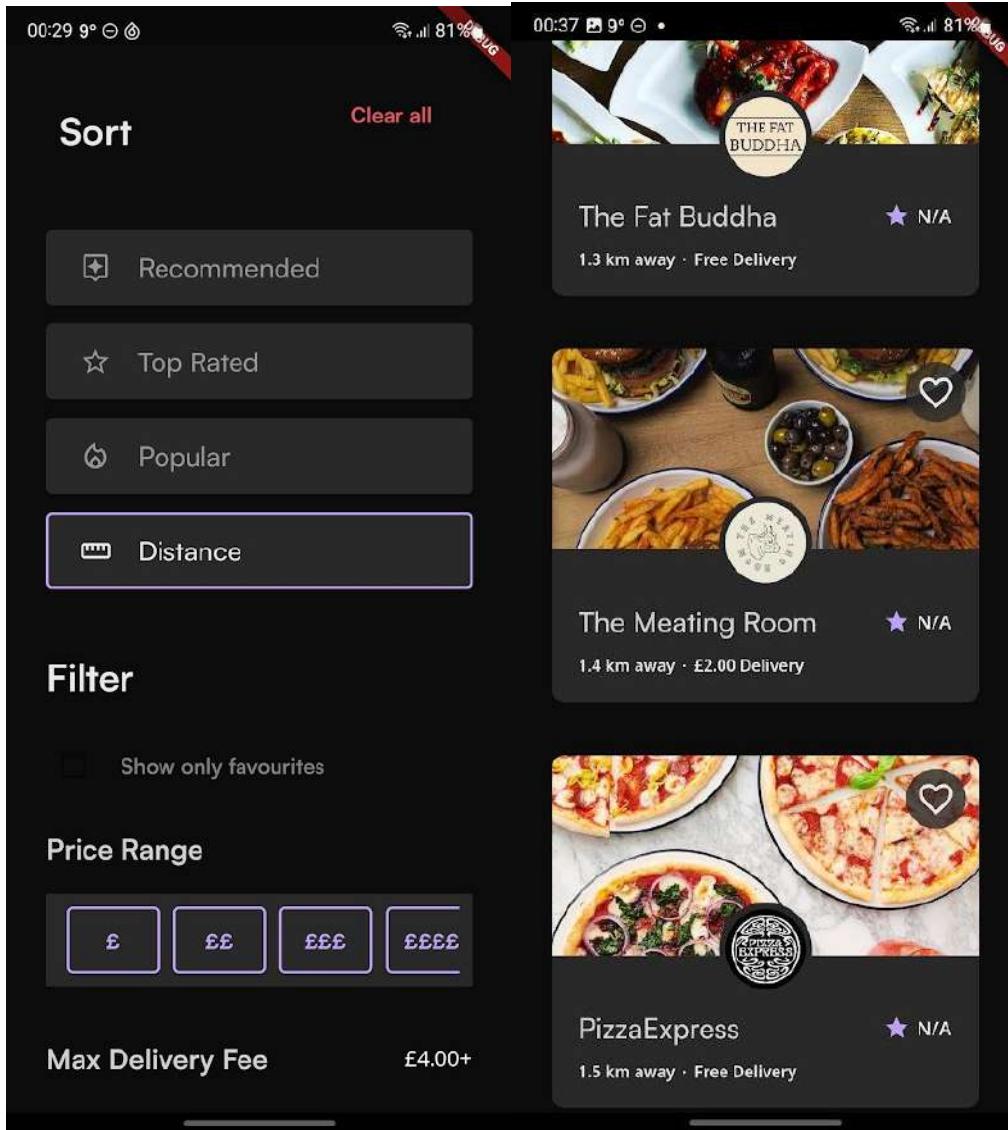
This will allow the user to change how the many restaurants are shown so that they can see the best option first that they are looking for

Expected Result

The user should have options to sort.

Actual Result

The user is able to sort using a button. While the code is ready for the sort of methods recommended, popular and hot, without doing the cart system, they are pointless since it relies on statistics.



```

if (sort == "distance") {
    // If the sort type is distance
    $sql = "SELECT res.id AS id, res.restaurantname AS restaurantname, res.restaurantbanner AS restaurantbanner, res.latitude AS latitude, res.longitude AS longitude, res.servicedistance AS servicedistance, res.ordermin AS ordermin, res.delivery AS delivery, ( 4326 * acos( cos( radians($lat) ) * cos( radians( latitude ) ) + sin( radians( $lat ) ) * sin( radians( latitude ) ) * cos( radians( longitude ) - radians( $long ) ) ) ) AS distance FROM restaurant res ORDER BY distance ASC LIMIT 30";
    $result = mysqli_query($connection, $sql);
    while ($row = mysqli_fetch_assoc($result)) {
        array_push($array, $row);
    }
    mysqli_free_result($result);
}
else {
    // If the sort type is not distance, return error
    $error["error"] = true;
    $error["message"] = "Sort method unavailable";
}

```

Pass or Fail?

— PASS — (Revisit in future)

Testing Criteria (3.2)

When sorting restaurants, it should allow the user to sort by recommended, rating or distance

Justification

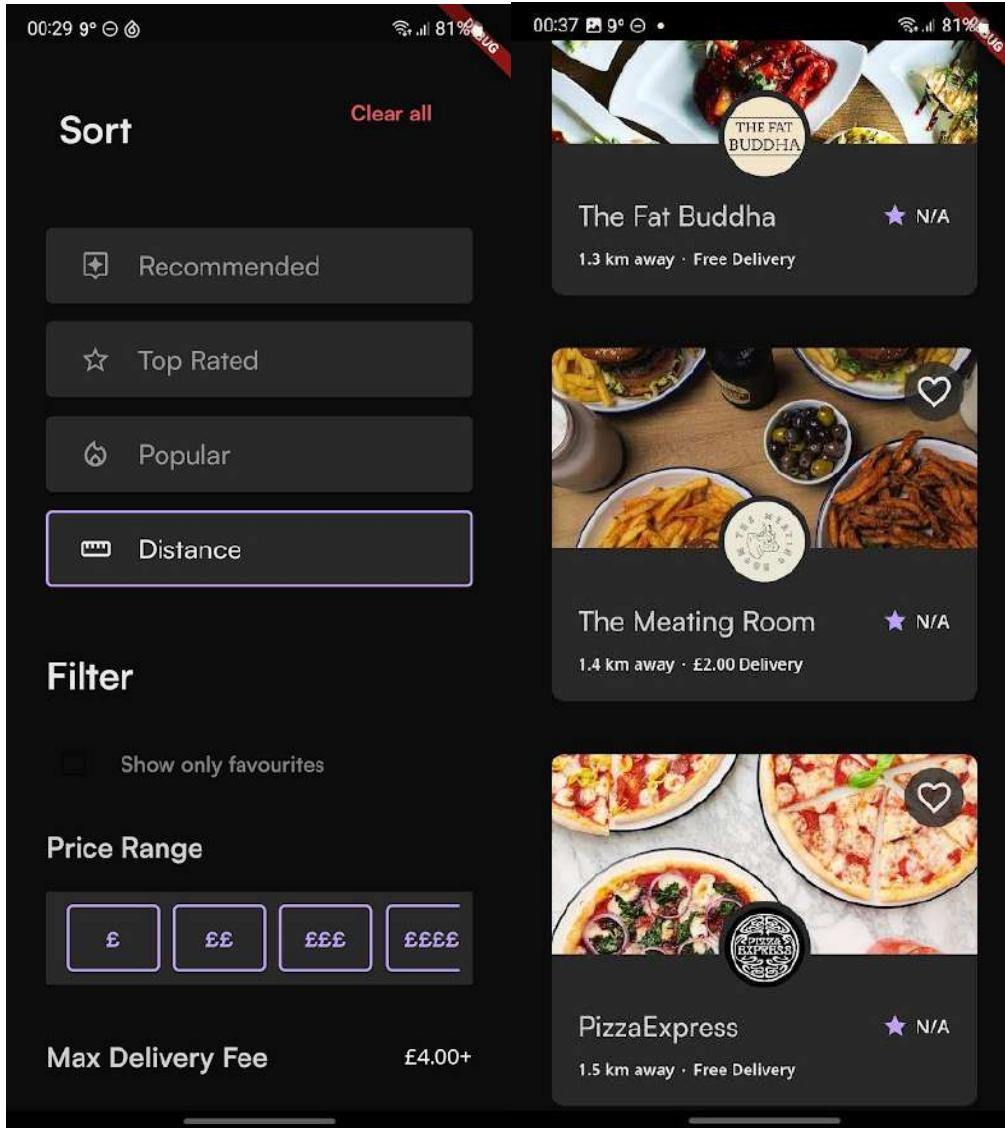
This will allow the user to change how the many restaurants are shown so that they can see the best option first that they are looking for

Expected Result

There should be options to sort by recommended, rating or distance

Actual Result

The user is able to sort using a button. While the code is ready for the sort of methods recommended, popular and hot, without doing the cart system, they are pointless since it relies on statistics.



```

if (sort == "distance") {
    if (the sort type is distance)
        $sql = "SELECT res.restaurantid, res.restaurantname, res.restaurantlogo, res.longitude, res.latitude, res.servicedistance, res.ordermin, res.delivery, ( 4375 * acos( cos( radians($latitutine) ) * cos( radians( latitude ) ) + sin( radians( latitude ) ) * sin( radians( longitude ) ) * cos( radians( longitude ) ) ) ) AS distance FROM restaurant res ORDER BY distance ASC LIMIT 20";
    else ($res = mysqli_query($con, $sql));
    while($row = mysqli_fetch_array($res)) {
        array_push($array, array("restaurantid", $row["restaurantname"], $row["restaurantlogo"], $row["restaurantbanner"], $row["latitude"], $row["longitude"], $row["ordermin"], $row["delivery"]));
    }
    mysqli_free_result($res);
}
else {
    // If the sort type is not distance, return error
    die("Error! - true");
    //return["message"] = "Sort method unavailable";
}

```

Pass or Fail?

— PASS — (Revisit in future)

Testing Criteria (3.4)

The user should be able to filter restaurants using a button

Justification

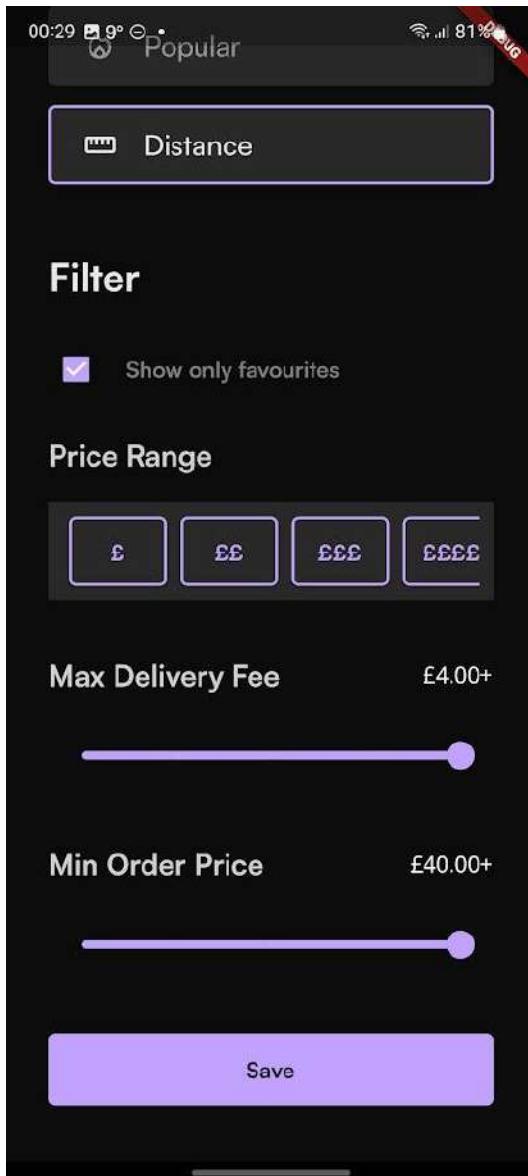
This will allow the user to reduce the amount of options and only see the relevant restaurants they are looking for

Expected Result

There should be different sections for each filter with the ability to change each

Actual Result

There are options to change the filter type for the following; toggling showing only favourites, the ability to choose which price ranges you want to show (£ - ££££), choosing the maximum delivery fee you want to pay, choosing the maximum minimum order price. These work together and can be combined together to get a result



```
if ($favorite == "true"){
    $sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE profileid = $profileid";
    if($result = mysqli_query($link, $sqlfavrestaurant)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($favrestaurantslist, $row["restaurantid"]);
        }
    }
    array_push($filterquery, "res.restaurantid IN (" . implode(',', $favrestaurantslist) . ")");
}
array_push($filterquery, "res.pricebracket IN (" . implode(',', $priceTarget) . ")", "res.delivery <= $maxDelivery", "res.ordermin <= $minOrder");
$fullfilter = "WHERE " . implode(" AND ", $filterquery); //Compile the where statements into one
```

Pass or Fail?

— PASS —

Testing Criteria (3.5)

When filtering restaurants, it should allow the user to filter by price range, delivery, pickup, delivery price, new

Justification

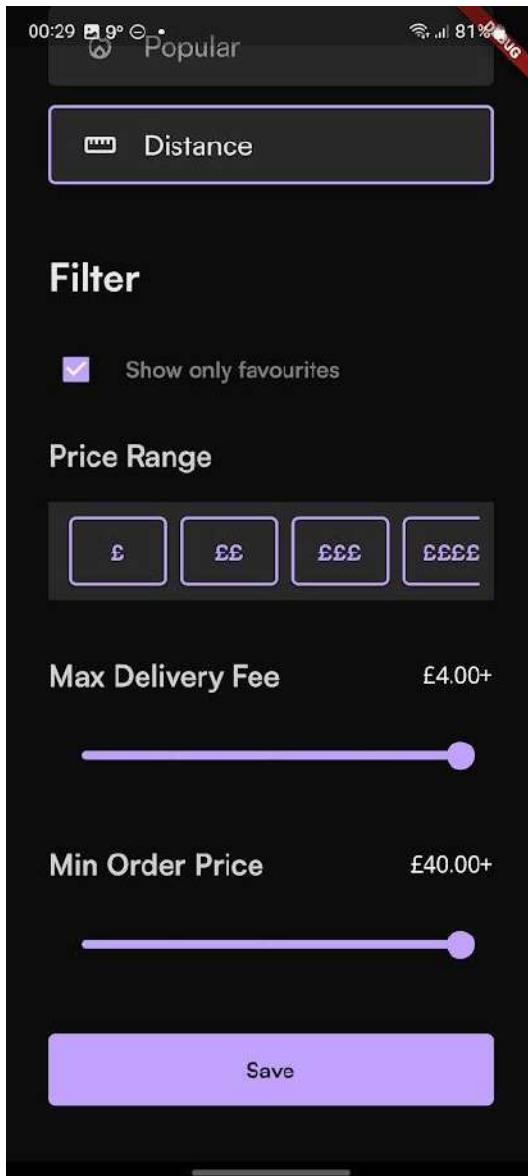
This will allow the user to reduce the amount of options and only see the relevant restaurants they are looking for

Expected Result

There should be filters for price range, delivery, pickup, delivery price, new

Actual Result

There are options to change the filter type for the following; toggling showing only favourites, the ability to choose which price ranges you want to show (£ - ££££), choosing the maximum delivery fee you want to pay, choosing the maximum minimum order price. These work together and can be combined together to get a result



```
if ($favorite == "true"){
    $sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE profileid = $profileid";
    if($result = mysqli_query($link, $sqlfavrestaurant)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($favrestaurantslist, $row["restaurantid"]);
        }
    }
    array_push($filterquery, "res.restaurantid IN (" . implode(',', $favrestaurantslist) . ")");
}
array_push($filterquery, "res.pricebracket IN (" . implode(',', $priceTarget) . ")", "res.delivery <= $maxDelivery", "res.ordermin <= $minOrder");
$fullfilter = "WHERE " . implode(" AND ", $filterquery); //Compile the where statements into one
```

Pass or Fail?

— PASS —

Testing Criteria (3.7)

Before filtering, there should be a button to confirm that they would like to filter the restaurant.

Justification

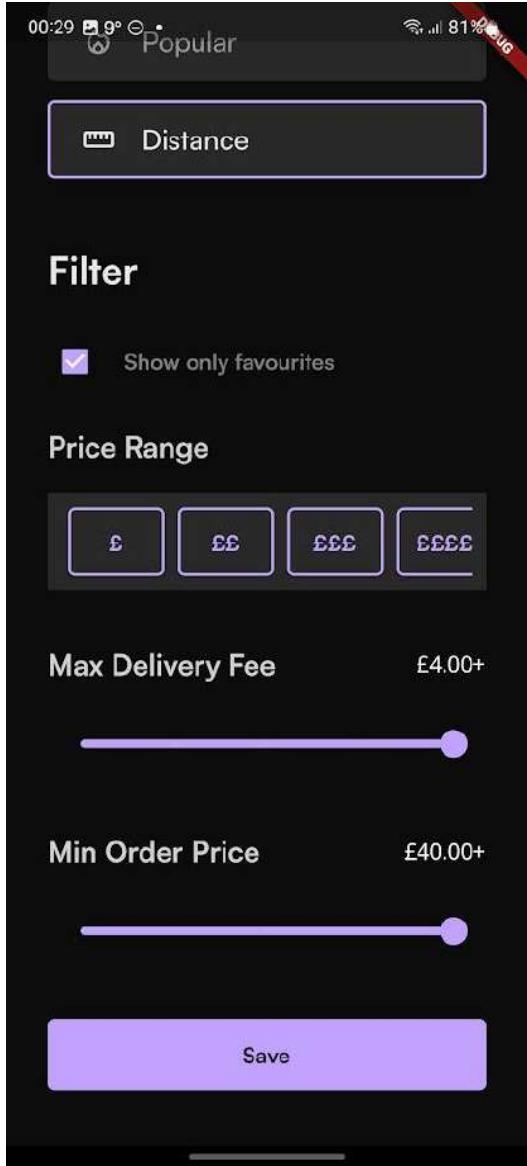
This ensures that the user would like to filter the options and shows how many restaurants will be shown after the filter so the user is not surprised if there are a lot or only a few restaurants remaining

Expected Result

There should be a button where the user can confirm their options.

Actual Result

At the bottom of the filter and sort screen, there is a save button which is used to save the filter and sort terms and causes the restaurant list to refresh.



Pass or Fail?

— PASS —

Testing Criteria (3.8)

When the user has selected some restaurant filters, the user should have a button they can click to clear the filters and have it show all the restaurants

Justification

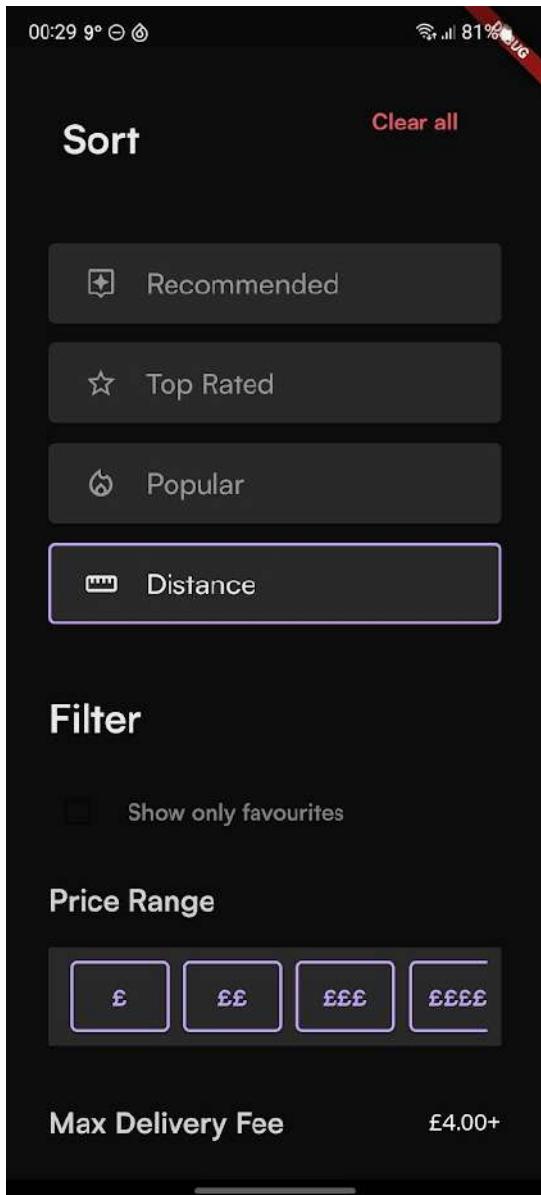
This decreases the amount of time it takes to unselect the filters, with a single button to undo any changes

Expected Result

There should be a clear filters button which resets the filters and sorts back to default

Actual Result

At the top of the filter and sort screen there is a button called "Clear all" which resets the filter and sort options back to the default settings in order to see all options



Pass or Fail?

— PASS —

Testing Criteria (3.9)

When the user has selected a restaurant filter, they should be able to unselect a filter and change the settings

Justification

This means if the user wants to remove a filter because it is too refined, they can do it

Expected Result

The filters should be able to be changed

Actual Result

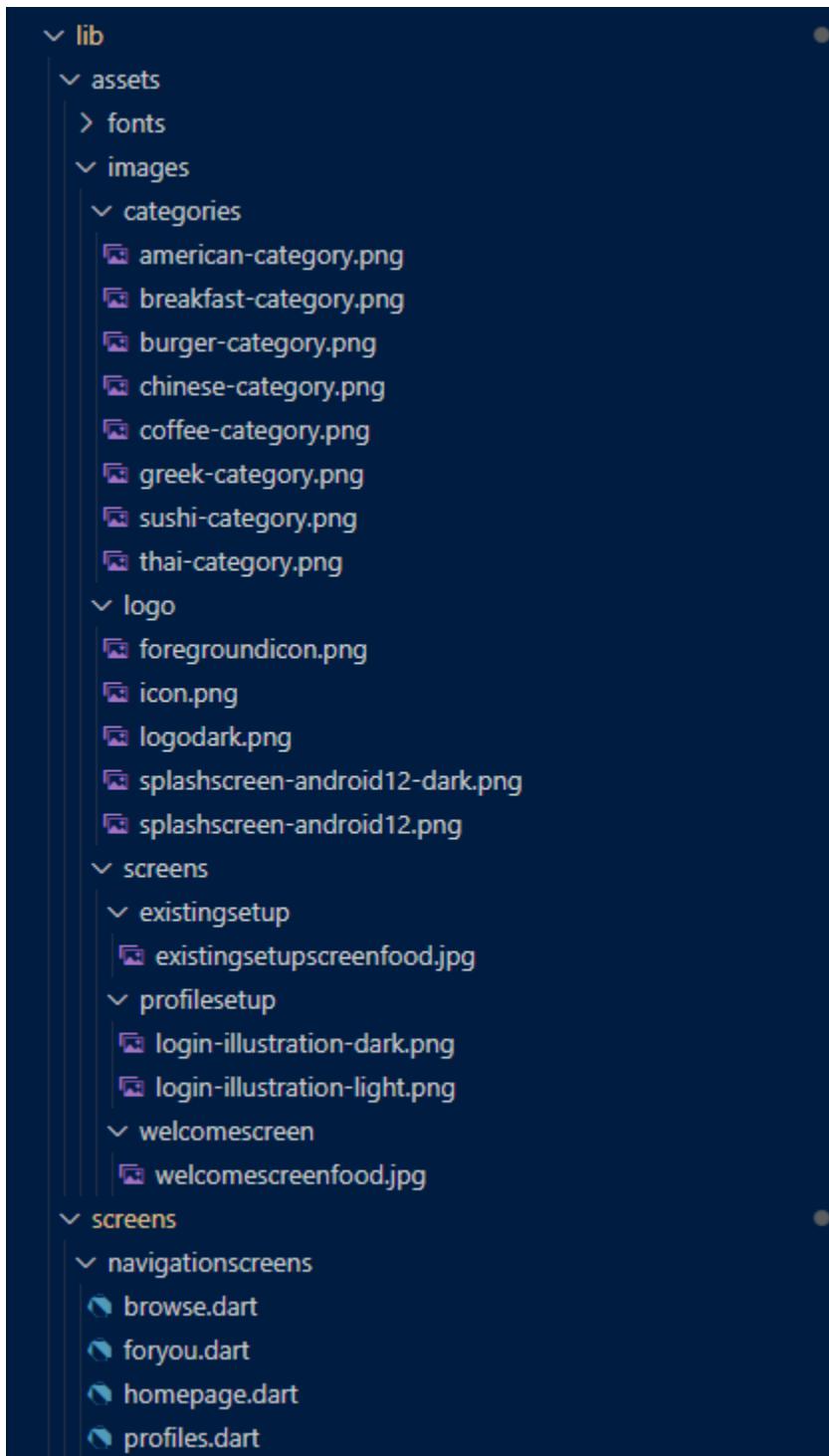
The filters can be changed and results in a customised restaurant list.

Pass or Fail?

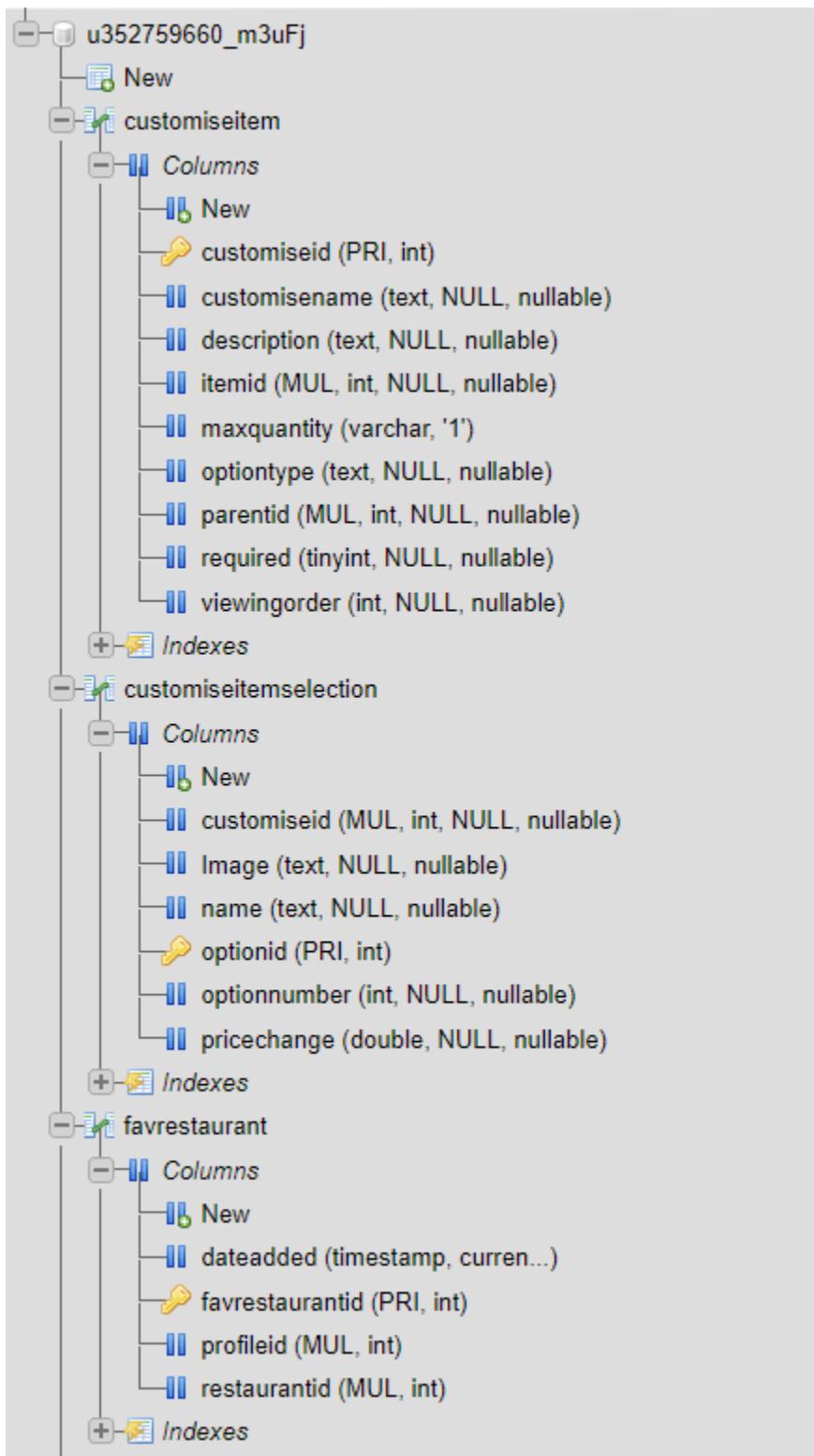
— PASS —

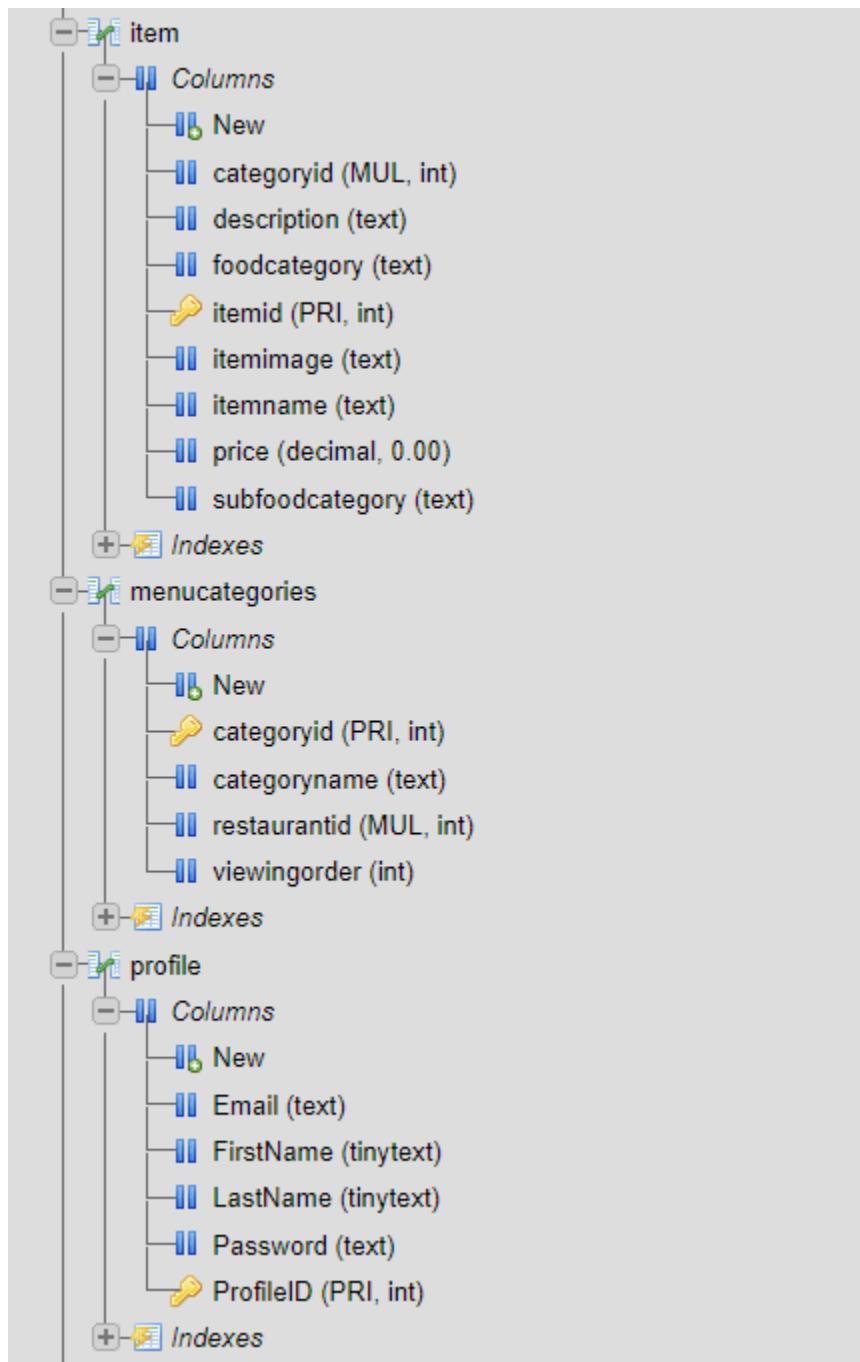
Prototype 2 Final Code

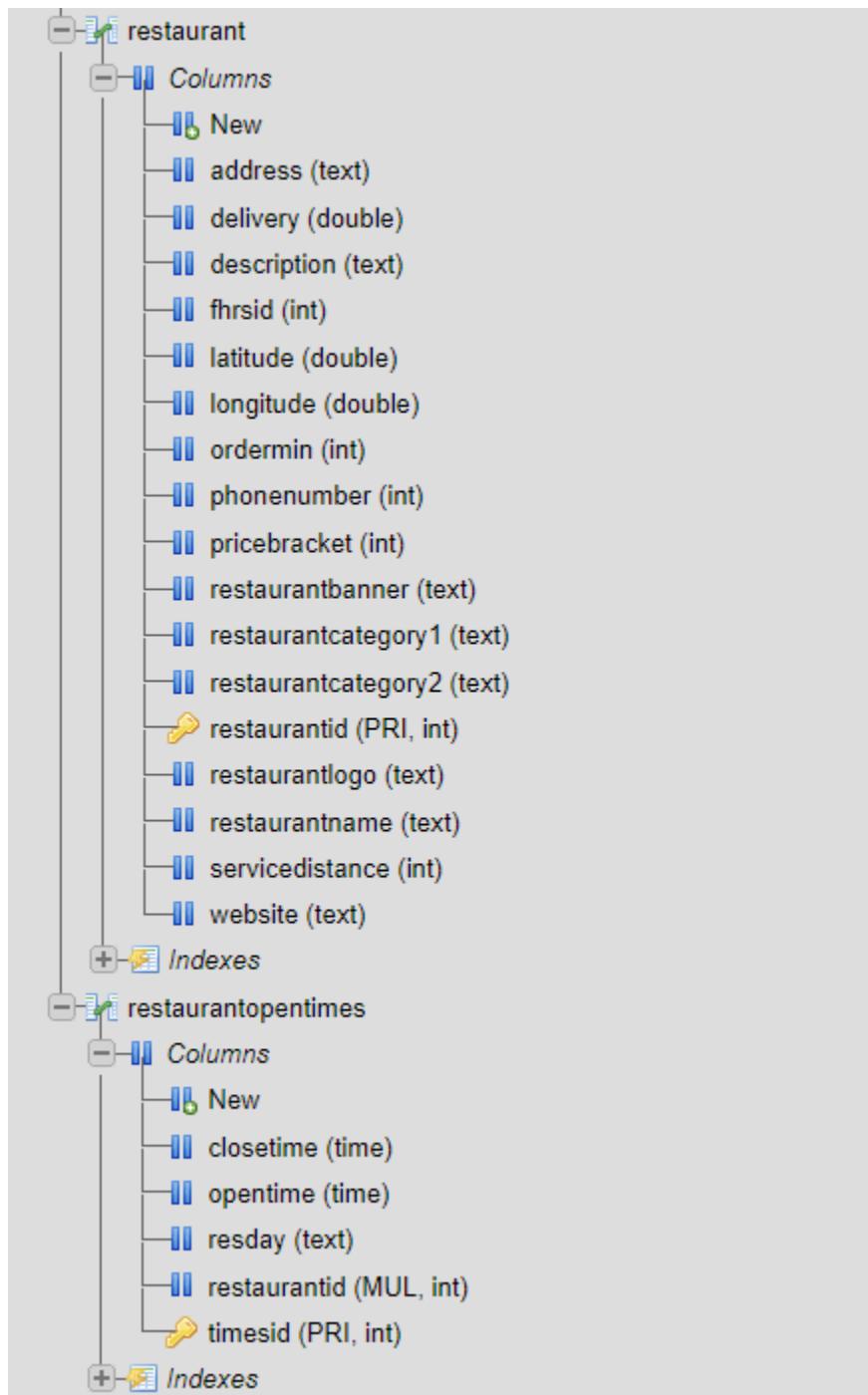
File Structure



```
✓ profilesetup
  ⚒ profilesetup_create.dart
  ⚒ profilesetup_existing.dart
  ⚒ profilesetup_login.dart
  ⚒ welcomescreen.dart
✓ restaurant
  ⚒ restaurant_customise.dart
  ⚒ restaurant_main.dart
  ⚒ filtersort.dart
  ⚒ locationselection.dart
  ⚒ setupverification.dart
✓ services
  ⚒ cart_service.dart
  ⚒ dataencryption.dart
  ⚒ localprofiles_service.dart
  ⚒ queryserver.dart
  ⚒ setselected.dart
✓ theme
  ⚒ theme.dart
✓ widgets
  ✓ elements
    ⚒ browse_categories.dart
    ⚒ elements.dart
    ⚒ profiles_buttons.dart
    ⚒ profiles_selection.dart
    ⚒ search.dart
    ⚒ genericlocadning.dart
    ⚒ navigationbar.dart
    ⚒ restaurants_list.dart
    ⚒ topbar.dart
  ⚒ main.dart
```







Name ↑	Size	Last modified	
< > favouriterestaurant.php	2.38 KB	4 months ago	-rw-r--r--x
< > favouriterestaurantdata.php	2.49 KB	4 months ago	-rw-r--r--x
< > favouriterestaurantlist.php	1.81 KB	2 months ago	-rw-r--r--x
< > itemcustomise.php	1.7 KB	4 days ago	-rw-r--r--x
< > login.php	3.05 KB	2 months ago	-rw-r--r--x
< > register.php	4.63 KB	2 months ago	-rw-r--r--x
< > restaurantlist.php	3.38 KB	6 days ago	-rw-r--r--x
< > restaurantmenucategories.php	1.25 KB	4 months ago	-rw-r--r--x
< > restaurantmenuitems.php	1.36 KB	4 months ago	-rw-r--r--x

Files - Application

browse.dart

```

import 'package:alleat/screens/filtersort.dart';

import 'package:alleat/widgets/elements/browse_categories.dart';

import 'package:alleat/widgets/elements/search.dart';

import 'package:alleat/widgets/restaurants_list.dart';

import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

class BrowsePage extends StatefulWidget {

  const BrowsePage({Key? key}) : super(key: key);

  @override
  State<BrowsePage> createState() => _BrowsePageState();
}

class _BrowsePageState extends State<BrowsePage> {

```

```
@override

Widget build(BuildContext context) {

  return SafeArea(
    //Keep within screen area

    child: Scaffold(
      body: SingleChildScrollView(
        child: Column(children: [
          const MainAppBar(
            height: 150,
          ),
          const SizedBox(height: 20),
          const SearchBar(), //Search bar
          const SizedBox(height: 40),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 30),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              crossAxisAlignment: CrossAxisAlignment.end,
              children: [
                //Display row with the title popular categories and an arrow
                //pointing right to go to all the categories
                Text(
                  "Popular Categories.",
                )
              ],
            ),
          ),
        ],
      ),
    ),
  );
}
```

```
        style: Theme.of(context).textTheme.headline2,  
    ),  
  
    InkWell(  
  
        child: Padding(  
  
            padding: const EdgeInsets.all(5),  
  
            child: Icon(  
  
                Icons.chevron_right,  
  
                size: 30,  
  
                color: Theme.of(context).colorScheme.onBackground,  
            )),  
  
        onTap: () => (Navigator.push( //Go to Categories Page  
  
            context,  
  
            MaterialPageRoute(  
  
                builder: (context) => const CategoriesPage(),  
            ))))  
    ],  
),  
  
const SizedBox(height: 40),  
  
const Categories(), // Display horizontally scrolling categories  
  
const SizedBox(height: 40),  
  
Padding(  
  
    padding: const EdgeInsets.symmetric(horizontal: 30),
```

```
child: Row(  
  mainAxisAlignment: MainAxisAlignment.spaceBetween,  
  children: [  
    // Display title restaurants with a filter icon to go to filtering  
    options  
  
    Text(  
      "Restaurants.",  
      style: Theme.of(context).textTheme.headline2,  
    ),  
    InkWell(  
      child: Padding(  
        padding: const EdgeInsets.all(5),  
        child: Icon(  
          Icons.tune,  
          size: 30,  
          color: Theme.of(context).colorScheme.onBackground,  
        )),  
      onTap: () => (Navigator.push(  
        context,  
        MaterialPageRoute(  
          builder: (context) => const FilterSort(),  
        ))))
```

```
        ],
        )),
        const SizedBox(height: 30),
        const RestaurantList(),
    ])));
}

}
```

foryou.dart

```
import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

class ForYouPage extends StatelessWidget {

    const ForYouPage({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        return Column(children: [
            const MainAppBar(
                height: 150,
            ),
            Center(

```

```
        child: Text(  
            'For You',  
            style: Theme.of(context).textTheme.headline1,  
        ),  
    ),  
);  
  
}  
}
```

homepage.dart

```
import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

class HomePage extends StatelessWidget {

  const HomePage({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    Future<List> getName() async {

      // Get the name of the profile that is selected from shared preferences

      final prefs = await SharedPreferences.getInstance();
```

```
final String? firstname = prefs.getString('firstname');

final String? lastname = prefs.getString('lastname');

return [firstname, lastname];

}

return Column(children: [

const MainAppBar(


//Display main app bar at the top


height: 150,


),


Padding(


padding:


const EdgeInsets.only(left: 40, top: 40, right: 60, bottom: 80),


child: Column(


crossAxisAlignment: CrossAxisAlignment.start,


children: [


Text(


// Display welcome back


'Welcome back',


style: Theme.of(context).textTheme.headline6,


),


FutureBuilder<List>(
```

```

// run future to get the name from shared preferences

future: getName(),

builder: (context, snapshot) {

  if (snapshot.hasData) {

    //If the data is received

    List? name = snapshot.data; //Assign received data to name

    if (name![0] != null) {

      // If the data is not null

      return Text(

        // Display firstname and lastname

        ('${name[0]} ${name[1]}'),

        style: Theme.of(context).textTheme.headline1,

      );

    } else {

      // If the data is null or anything else

      return Text(

        // Display Profile Unknown

        'Profile Unknown.',

        style: Theme.of(context).textTheme.headline1,

      );

    }

  } else {

```

```
// While waiting to get data or there is an error while
getting the data, display loading profile

    return Text(
        'Loading Profile...',

        style: Theme.of(context).textTheme.headline1,
    );
}

),
],
))

]);
}

}
```

profiles.dart

```
import 'package:alleat/widgets/elements/profiles_buttons.dart';

import 'package:alleat/widgets/elements/profiles_selection.dart';

import 'package:flutter/material.dart';

class ProfilePage extends StatefulWidget {

    const ProfilePage({super.key});
```

```
    @override

    State<ProfilePage> createState() => _ProfilePageState();

}

class _ProfilePageState extends State<ProfilePage> {

    @override

    Widget build(BuildContext context) {

        return SafeArea(
            child: Scaffold(
                body: CustomScrollView(slivers: [
                    SliverFillRemaining(
                        hasScrollBody: false,
                        child: Padding(
                            padding: const EdgeInsets.only(top: 50, bottom: 50),
                            child: Column(children: [
                                Padding(
                                    padding: const EdgeInsets.only(left: 40, right: 40),
                                    child: Row(
                                        //Display text with title Profiles
                                        mainAxisAlignment: MainAxisAlignment.spaceBetween,
                                        children: [

```

```
        Text(
            "Profiles",
            style: Theme.of(context).textTheme.headline1,
        ),
    ],
)),
const SizedBox(
    height: 50,
),
const ProfileList(), //Display slidable row of profiles
const SizedBox(
    height: 35,
),
const ProfileButton(
    // Display buttons for favourites, orders, settings and profile
settings with an icon using the widget ProfileButton
    icon: (Icons.favorite),
    name: "Favourites",
    action: null,
),
const ProfileButton(
    icon: (Icons.local_mall),
    name: "Orders",
)
```

```
        action: null,  
    ),  
  
    const ProfileButton(  
        icon: (Icons.settings),  
        name: "Settings",  
        action: null,  
    ),  
  
    const ProfileButton(  
        icon: (Icons.person),  
        name: "Profile Settings",  
        action: null,  
    ),  
])))  
])));  
}  
}
```

profilesetup_create.dart

```
import 'package:alleat/services/dataencryption.dart';  
  
import 'package:alleat/services/localprofiles_service.dart';  
  
import 'package:alleat/services/queryserver.dart';
```

```
import 'package:alleat/services/setselected.dart';

import 'package:alleat/widgets/elements/elements.dart';

import 'package:alleat/widgets/navigationbar.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:email_validator/email_validator.dart';



class AddProfileCreationPageName extends StatefulWidget {

  const AddProfileCreationPageName({Key? key}) : super(key: key);

  @override

  State<AddProfileCreationPageName> createState() =>

    _AddProfileCreationPageNameState();

}

class _AddProfileCreationPageNameState

  extends State<AddProfileCreationPageName> {

  final _formKey = GlobalKey<FormState>();

  static TextEditingController firstnameText = TextEditingController();

  static TextEditingController lastnameText = TextEditingController();

  final data = [

    firstnameText.text = "",
```

```
lastnameText.text = ""

]; //Store the data to be reset if back button pressed

@Override

Widget build(BuildContext context) {

    return Scaffold(

        body: CustomScrollView(slivers: [

            SliverFillRemaining(

                hasScrollBody: false,

                child: Column(

                    mainAxisAlignment: MainAxisAlignment.start,

                    mainAxisSize: MainAxisSize.spaceBetween,

                    children: <Widget>[

                        Flexible(


                            //Create flexible widgets to be able to resize with keyboard

                            flex: 2,


                            fit: FlexFit.loose,


                            child: Column(


                                mainAxisAlignment: MainAxisAlignment.start,


                                children: [


                                    const ScreenBackButton(), //Back button to go to starting


screen


                                    Padding(


                                        padding: const EdgeInsets.only(
```

```
        top: 20, left: 20, right: 20, bottom: 5),  
  
        child: Align(  
  
            alignment: Alignment.center,  
  
            child: Text(  
  
                "Step 1 of 3",  
  
                style: Theme.of(context).textTheme.headline6,  
  
                textAlign: TextAlign.center,  
  
            )),  
  
        Padding(  
  
            padding: const EdgeInsets.only(  
  
                bottom: 20, left: 20, right: 20),  
  
            child: Align(  
  
                alignment: Alignment.center,  
  
                child: Text(  
  
                    "Let's Get Started.",  
  
                    style: Theme.of(context).textTheme.headline1,  
  
                    textAlign: TextAlign.center,  
  
                )),  
  
        const SizedBox(  
  
            height: 20,  
  
        )  
  
    ],
```

```
  )),  
  
  Flexible(  
  
    flex: 2,  
  
    fit: FlexFit.loose,  
  
    child: Column(  
  
      mainAxisAlignment: CrossAxisAlignment.start,  
  
      children: [  
  
        Form(  
  
          key: _formKey,  
  
          child: Column(  
  
            children: [  
  
              ListTile(  
  
                title: TextFormField(  
  
                  controller:  
  
                  firstnameText, //Form data lastname  
collected and sent to database  
  
                  keyboardType: TextInputType.name,  
  
                  style:  
  
                  Theme.of(context).textTheme.bodyText2,  
  
                  decoration: (InputDecoration(  
  
                    hintText: "Forename",  
  
                    contentPadding: Theme.of(context)  
  
                      .inputDecorationTheme
```

```
        .contentPadding,  
  
        border: Theme.of(context)  
  
            .inputDecorationTheme  
  
            .border,  
  
        focusedBorder: Theme.of(context)  
  
            .inputDecorationTheme  
  
            .focusedBorder,  
  
        enabledBorder: Theme.of(context)  
  
            .inputDecorationTheme  
  
            .enabledBorder,  
  
        floatingLabelBehavior:  
  
            FloatingLabelBehavior.never)),  
  
        inputFormatters: [  
  
            //Only allows the input of letters a-z and  
A-Z and @,.-  
  
            FilteringTextInputFormatter.allow(  
  
                RegExp('[a-zA-Z0-9@,.]'))  
  
        ],  
  
        validator: (firstname) {  
  
            //Required field and uses emailvalidator  
package to verify it is an email to simplify the code  
  
            if (firstname == null ||  
  
                firstname.isEmpty) {
```

```
        return "Required";  
  
    }  
  
    return null;  
},  
)),  
const SizedBox(  
    height: 10,  
,  
ListTile(  
    title: TextFormField(  
        controller:  
            lastnameText, //Form data lastname  
collected and sent to database  
        keyboardType: TextInputType.name,  
        style:  
            Theme.of(context).textTheme.bodyText2,  
        decoration: (InputDecoration(  
            hintText: "Surname",  
            contentPadding: Theme.of(context)  
                .inputDecorationTheme  
                .contentPadding,  
            border: Theme.of(context)
```

```
.inputDecorationTheme

    .border,
    focusedBorder: Theme.of(context)

        .inputDecorationTheme

        .focusedBorder,
        enabledBorder: Theme.of(context)

            .inputDecorationTheme

            .enabledBorder,
            floatingLabelBehavior:

                FloatingLabelBehavior.never)),

        inputFormatters: [
            //Only allows the input of letters a-z and
            A-Z and @, .-
            FilteringTextInputFormatter.allow(
                RegExp('[a-zA-Z0-9@,.]'))
        ],
        validator: (lastname) {
            //Required field and uses emailvalidator
            package to verify it is an email to simplify the code
            if (lastname == null ||
                lastname.isEmpty) {
                return "Required";
            }
        }
    
```

```
        return null;  
  
    },  
  
    )),  
  
    const SizedBox(  
  
        height: 50,  
  
    ),  
  
    ],  
  
))  
  
]),  
  
Flexible(  
  
    flex: 1,  
  
    fit: FlexFit.loose,  
  
    child: Column(  
  
        children: [  
  
            Padding(  
  
                padding: const EdgeInsets.only(  
  
                    left: 30,  
  
                    right: 30,  
  
                    bottom: 60,  
  
                ),  
  
                child: Center(  
  
                    child: SizedBox(  

```

```
width: double.infinity,  
  
        child: ElevatedButton(  
  
            onPressed: () {  
  
                if (_formKey.currentState!  
  
                    .validate()) {  
  
                    //If fields have no errors  
  
                    Navigator.pop(context);  
  
                    Navigator.push(  
  
                        context,  
  
                        MaterialPageRoute(  
  
                            builder: (context) =>  
  
AddProfileCreationPageEmail(  
  
                firstname:  
  
                    firstnameText  
  
                    .text,  
  
                lastname:  
  
                    lastnameText.text,  
  
                )));  
  
            } else {  
  
                null;  
  
            }  
  
        },
```



```
_AddProfileCreationPageEmailState();  
}  
  
class _AddProfileCreationPageEmailState  
    extends State<AddProfileCreationPageEmail> {  
  
    final GlobalKey<FormState> _formKey = GlobalKey<FormState>();  
  
    static TextEditingController emailText = TextEditingController();  
  
    static TextEditingController confirmemailText = TextEditingController();  
  
    dynamic data = [  
  
        emailText.text = "",  
  
        confirmemailText.text = ""  
    ]; // Store email and confirm email to be reset if back button pressed  
  
    @override  
  
    Widget build(BuildContext context) {  
  
        return Scaffold(  
  
            body: CustomScrollView(slivers: [  
  
                SliverFillRemaining(  
  
                    hasScrollBody: false,  
  
                    child: Column(  
  
                        crossAxisAlignment: CrossAxisAlignment.start,  
  
                        mainAxisAlignment: MainAxisAlignment.spaceBetween,  
  
                        children: <Widget>[
```

```
Flexible(  
    flex: 2,  
    fit: FlexFit.loose,  
    child: Column(  
        mainAxisAlignment: MainAxisAlignment.start,  
        children: [  
            ScreenBackButton(  
                data:  
                    data), //If back button pressed, reset inputted  
data  
            Padding(  
                padding: const EdgeInsets.only(  
                    top: 20, left: 20, right: 20, bottom: 5),  
                child: Align(  
                    alignment: Alignment.center,  
                    child: Text(  
                        "Step 2 of 3",  
                        style: Theme.of(context).textTheme.headline6,  
                        textAlign: TextAlign.center,  
                    )),  
            Padding(  
                padding: const EdgeInsets.only(  
                    bottom: 20, left: 20, right: 20),  
            ),  
        ],  
    ),  
);
```

```
        child: Align(  
  
            alignment: Alignment.center,  
  
            child: Text(  
  
                "Profile Setup.",  
  
                style: Theme.of(context).textTheme.headline1,  
  
                textAlign: TextAlign.center,  
  
            )),  
  
        const SizedBox(  
  
            height: 20,  
  
        )  
  
    ],  
  
)),  
  
Flexible(  
  
    flex: 2,  
  
    fit: FlexFit.loose,  
  
    child: Column(  
  
        crossAxisAlignment: CrossAxisAlignment.start,  
  
        children: [  
  
            Form(  
  
                key: _formKey,  
  
                child: Column(  
  
                    children: [  
  
            ]  
        ]  
    ]  
);
```

```
ListTile(  
    title: TextFormField(  
        controller:  
            emailText, //Form data lastname collected  
and sent to database  
        keyboardType: TextInputType.emailAddress,  
        style:  
            Theme.of(context).textTheme.bodyText2,  
        decoration: (InputDecoration(  
            hintText: "Email",  
            contentPadding: Theme.of(context)  
                .inputDecorationTheme  
                .contentPadding,  
            border: Theme.of(context)  
                .inputDecorationTheme  
                .border,  
            focusedBorder: Theme.of(context)  
                .inputDecorationTheme  
                .focusedBorder,  
            enabledBorder: Theme.of(context)  
                .inputDecorationTheme  
                .enabledBorder,  
            floatingLabelBehavior:
```

```
        FloatingLabelBehavior.never)),  
  
        inputFormatters: [  
  
            //Only allows the input of letters a-z and  
            A-Z and @, .-  
  
            FilteringTextInputFormatter.allow(  
  
                RegExp('[a-zA-Z0-9@,.]'))  
  
,  
  
        validator: (email) {  
  
            //Required field and uses emailvalidator  
            package to verify it is an email to simplify the code  
  
            if (email == null || email.isEmpty) {  
  
                return "Required";  
  
            }  
  
            if (EmailValidator.validate(email) ==  
  
                false) {  
  
                return "Please enter valid email";  
  
            }  
  
            return null;  
  
        },  
  
    )),  
  
    const SizedBox(  
  
        height: 10,  
  
    ),
```

```
ListTile(  
    title: TextFormField(  
        controller:  
            confirmEmailText, //Form data lastname  
collected and sent to database  
        keyboardType: TextInputType.emailAddress,  
        style:  
            Theme.of(context).textTheme.bodyText2,  
        decoration: (InputDecoration(  
            hintText: "Confirm email",  
            contentPadding: Theme.of(context)  
                .inputDecorationTheme  
                .contentPadding,  
            border: Theme.of(context)  
                .inputDecorationTheme  
                .border,  
            focusedBorder: Theme.of(context)  
                .inputDecorationTheme  
                .focusedBorder,  
            enabledBorder: Theme.of(context)  
                .inputDecorationTheme  
                .enabledBorder,  
            floatingLabelBehavior:
```

```
        FloatingLabelBehavior.never)),  
  
        inputFormatters: [  
  
            //Only allows the input of letters a-z and  
            A-Z and @, .-  
  
            FilteringTextInputFormatter.allow(  
  
                RegExp('[a-zA-Z0-9@,.]'))  
  
,  
  
        validator: (confirmemail) {  
  
            //Required field and uses emailvalidator  
            package to verify it is an email to simplify the code  
  
            if (confirmemail == null ||  
  
                confirmemail.isEmpty) {  
  
                return "Required";  
  
            } else if (confirmemail !=  
  
                emailText.text) {  
  
                return "Emails do not match";  
  
            }  
  
            return null;  
  
        },  
  
    )),  
  
    const SizedBox(  
  
        height: 50,  
  
    ),
```



```
        Navigator.pop(context);

        Navigator.push(
            // On continue, export inputted
            fields to the final screen

            context,
            MaterialPageRoute(
                builder: (context) =>

AddProfileCreationPagePassword(
            email: emailText.text,
            confirmemail:
            confirmemailText
            .text,
            firstname:
            widget.firstname,
            lastname:
            widget.lastname,
        )));
    } else {
        null;
    }
},
child: const Text("Continue")))))
```

```
        ],
      )),
    )));
  });

}

class AddProfileCreationPagePassword extends StatefulWidget {

  const AddProfileCreationPagePassword(
    {Key? key, this.email, this.confirmemail, this.firstname, this.lastname})
    : super(key: key);

  final dynamic email;
  final dynamic confirmemail;
  final dynamic firstname;
  final dynamic lastname;

  @override
  State<AddProfileCreationPagePassword> createState() =>
    _AddProfileCreationPagePasswordState();
}

class _AddProfileCreationPagePasswordState
```

```
extends State<AddProfileCreationPagePassword> {

final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

static TextEditingController passwordText = TextEditingController();

static TextEditingController confirmPasswordText = TextEditingController();

static dynamic encryptPassword;

dynamic data = [
    passwordText.text = "",
    confirmPasswordText.text = ""
]; //If back button pressed, reset data

bool _passwordVisible = false;

@Override

Widget build(BuildContext context) {
    return Scaffold(
        body: CustomScrollView(slivers: [
            SliverFillRemaining(
                hasScrollBody: false,
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: <Widget>[
                        Flexible(
                            flex: 2,
```

```
fit: FlexFit.loose,  
  
child: Column(  
  
    mainAxisAlignment: MainAxisAlignment.start,  
  
    children: [  
  
        ScreenBackButton(data: data),  
  
        Padding(  
  
            padding: const EdgeInsets.only(  
  
                top: 20, left: 20, right: 20, bottom: 5),  
  
            child: Align(  
  
                alignment: Alignment.center,  
  
                child: Text(  
  
                    "Step 3 of 3",  
  
                    style: Theme.of(context).textTheme.headline6,  
  
                    textAlign: TextAlign.center,  
  
                )),  
  
        Padding(  
  
            padding: const EdgeInsets.only(  
  
                bottom: 20, left: 20, right: 20),  
  
            child: Align(  
  
                alignment: Alignment.center,  
  
                child: Text(  
  
                    "Secure Password.",  
  
                ),  
  
            ),  
  
        ),  
  
    ),  
  
),  
  
);
```

```
        style: Theme.of(context).textTheme.headline1,  
        textAlign: TextAlign.center,  
      )),  
    const SizedBox(  
      height: 20,  
    )  
  ],  
()),  
Flexible(  
  flex: 2,  
  fit: FlexFit.loose,  
  child: Column(  
    mainAxisAlignment: MainAxisAlignment.start,  
    children: [  
      Form(  
        key: _formKey,  
        child: Column(  
          children: [  
            ListTile(  
              title: TextFormField(  
                controller:  
                  passwordText, //Form data lastname  
collected and sent to database
```

```
        keyboardType: TextInputType.visiblePassword,  
        obscureText: !_passwordVisible,  
        style:  
          Theme.of(context).textTheme.bodyText2,  
        decoration: (InputDecoration(  
          hintText: "Password",  
          contentPadding: Theme.of(context)  
            .inputDecorationTheme  
            .contentPadding,  
          border: Theme.of(context)  
            .inputDecorationTheme  
            .border,  
          focusedBorder: Theme.of(context)  
            .inputDecorationTheme  
            .focusedBorder,  
          enabledBorder: Theme.of(context)  
            .inputDecorationTheme  
            .enabledBorder,  
          floatingLabelBehavior:  
            FloatingLabelBehavior.never,  
          suffixIcon: IconButton(  
            //Button to toggle password visibility
```



```

        RegExp(r'[a-zA-Z0-9@#$!#?]+'))
    ],
    validator: (password) {
        //Required field and uses emailvalidator
        package to verify it is an email to simplify the code

        if (password == null ||
            password.isEmpty) {

            return "Required";

        } else if (!password
            .contains(RegExp(r'[0-9]')) ||
            !password
            .contains(RegExp(r'[a-z]')))) {

            return "Password must contain at least 1
number and 1 letter";

        } else if (password.length < 8) {

            return "Password must be a minimum of 8
characters";

        } else if (password.length > 99) {

            return "Password must be a maximum of 99
characters";

        }

        return null;
    },
}),

```

```
const SizedBox( height: 10, ), ListTile( title: TextFormField( controller: confirmPasswordText, //Form data lastname collected and sent to database keyboardType: TextInputType.visiblePassword, style: Theme.of(context).textTheme.bodyText2, decoration: InputDecoration( hintText: "Confirm password", contentPadding: Theme.of(context) .inputDecorationTheme .contentPadding, border: Theme.of(context) .inputDecorationTheme .border, focusedBorder: Theme.of(context) .inputDecorationTheme .focusedBorder, .focusedBorder, 
```

```
        enabledBorder: Theme.of(context)

            .inputDecorationTheme

            .enabledBorder,

floatingLabelBehavior:

    FloatingLabelBehavior.never,

)),

obscureText: true,

autofillHints: const [

    AutofillHints.newPassword

],

autovalidateMode:

    AutovalidateMode.onUserInteraction,

inputFormatters: [

    //Only allows the input of letters a-z and
A-Z and @,.-

    FilteringTextInputFormatter.allow(

        RegExp('[a-zA-Z0-9@,.]'))

],

validator: (confirmpassword) {

    //Required field and uses emailvalidator
package to verify it is an email to simplify the code

    if (confirmpassword == null ||
        confirmpassword.isEmpty) {
```

```
        return "Required";

    } else if (confirmPassword !=

        passwordText.text) {

        return "Passwords do not match";

    }

    return null;

},

)),
const SizedBox(
    height: 50,
),
],
))
]),
Flexible(
    flex: 1,
    fit: FlexFit.loose,
    child: Column(
        children: [
            Padding(
                padding: const EdgeInsets.only(
                    left: 30,
```

```
        right: 30,  
  
        bottom: 60,  
  
) ,  
  
child: Center(  
  
    child: SizedBox(  
  
        width: double.infinity,  
  
        child: ElevatedButton(  
  
            onPressed: () {  
  
                if (_formKey.currentState!  
  
                    .validate()) {  
  
                    //If fields have no errors  
  
                    _createProfile();  
  
                } else {  
  
                    null;  
  
                }  
  
            },  
  
            child: const Text("Create Profile"))))  
  
        ],  
  
    ))  
  
]))  
  
]);  
}
```

```

Future<void> _createProfile() async {

    encryptPassword = await DataEncryption.encrypt(passwordText.text);

    var recievedServerData =
        await QueryServer.query("https://alleat.cpur.net/query/register.php", {

        //Send data to login.php on server with email and encrypted password

        "firstname": widget.firstname,
        "lastname": widget.lastname,
        "email": widget.email,
        "password": encryptPassword
    });

    if (recievedServerData["error"] == true) {
        //If error, display failed to create profile and reset password fields
        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                content: Text(recievedServerData["message"] +
                    " : Failed to create profile. Please try again")));
        });
        passwordText.text = "";
        confirmPasswordText.text = "";
    } else {
        //If the email already exists, reset password fields and bring user back to
        the add user page
    }
}

```

```
if ((recievedServerData["message"])["exist"] == true) {  
  
    passwordText.text = "";  
  
    confirmPasswordText.text = "";  
  
    setState(() {  
  
        Navigator.pop(context);  
  
        ScaffoldMessenger.of(context).showSnackBar(  
  
            const SnackBar(content: Text('Profile already exists')),  
  
        );  
  
    });  
} else {  
  
    try {  
  
        List importedProfile = (recievedServerData["message"])["profile"];  
  
        await SQLiteLocalProfiles.createProfile(  
  
            importedProfile[0],  
  
            importedProfile[1],  
  
            importedProfile[2],  
  
            importedProfile[3],  
  
            importedProfile[4]);  
  
  
        bool trySelect = await SetSelected.selectProfile(  
  
            importedProfile[0], //Try to select profile  
  
            importedProfile[1],
```

```
    importedProfile[2],  
  
    importedProfile[3]);  
  
    if (trySelect == false) {  
  
        setState(() {  
  
            ScaffoldMessenger.of(context).showSnackBar(  
  
                const SnackBar(content: Text("Failed to select profile")));  
  
        });  
  
    } else {  
  
        //If is able to select profile, clear password fields and go to  
navigation page (defaults to homepage)  
  
        passwordText.text = "";  
  
        confirmPasswordText.text = "";  
  
        setState(() {  
  
            Navigator.pop(context);  
  
            Navigator.pop(context);  
  
            ScaffoldMessenger.of(context).showSnackBar(  
  
                const SnackBar(content: Text('Successfullly created profile.')),  
  
            );  
  
            Navigator.push(context,  
  
                MaterialPageRoute(builder: (context) => const Navigation()));  
  
        });  
  
    }  
  
} catch (e) {
```

```
// If error, display error

setState(() {

    ScaffoldMessenger.of(context)

        .showSnackBar(SnackBar(content: Text("ERROR: $e")));

});

}

}

}

}
```

profilesetup_existing.dart

```
import 'package:alleat/screens/profilesetup/profilesetup_create.dart';

import 'package:alleat/screens/profilesetup/profilesetup_login.dart';

import 'package:flutter/material.dart';

class ProfileSetupExisting extends StatefulWidget {

  const ProfileSetupExisting({Key? key}) : super(key: key);

  @override

  State<StatefulWidget> createState() {
```

```
        return _ProfileSetupExisting();  
    }  
}  
  
class _ProfileSetupExisting extends State<ProfileSetupExisting> {  
  
    @override  
    Widget build(BuildContext context) {  
  
        return Scaffold(  
  
            //Create new screen  
  
            resizeToAvoidBottomInset: false, //Allow resize  
  
            body: Stack(children: [  
  
                Image.asset(  
  
                    //Fullscreen image of food  
  
                    'lib/assets/images/screens/existingsetup/existingsetupscreenfood.jpg',  
  
                    fit: BoxFit.cover,  
  
                    height: double.infinity,  
  
                    width: double.infinity,  
  
                    alignment: Alignment.center,  
  
                ),  
  
                Column(  
  
                    children: [  
  
                ],  
            ),  
        );  
    }  
}
```

```
Padding(  
    //Image of All Eat logo  
    padding: const EdgeInsets.only(  
        top: 50, left: 30, right: 30, bottom: 200),  
    child: Container(  
        width: 50,  
        height: 50,  
        decoration: BoxDecoration(  
            shape: BoxShape.circle,  
            color: Theme.of(context).backgroundColor,  
        ),  
        child: IconButton(  
            color: Theme.of(context).textTheme.headline1?.color,  
            icon: const Icon(Icons.arrow_back),  
            onPressed: () => Navigator.of(context).pop()))  
    ],  
),  
Column(  
    mainAxisAlignment: MainAxisAlignment.start,  
    mainAxisSize: MainAxisSize  
        .end, //Create content at the bottom of the screen  
    children: [
```

```
Padding(  
  
    padding: const EdgeInsets.only(left: 10, right: 10),  
  
    child: Container(  
  
        //Bottom container with login and register actions  
  
        decoration: BoxDecoration(  
  
            color: Theme.of(context).backgroundColor,  
  
            borderRadius: const BorderRadius.only(  
  
                topLeft: Radius.circular(20),  
  
                topRight: Radius.circular(  
  
                    20))), // Round the top corners of container  
  
        width: double.infinity,  
  
        padding: const EdgeInsets.only(left: 20, right: 20),  
  
        child: Column(children: [  
  
            const SizedBox(  
  
                height: 40,  
  
) ,  
  
            Text("Add New Profile.",  
  
                style: Theme.of(context).textTheme.headline2),  
  
            Padding(  
  
                padding: const EdgeInsets.only(  
  
                    left: 50, right: 50, top: 10, bottom: 10),  
  
                child: Text(  

```

```
    "Enhance your experience by using multiple profiles",

    style: Theme.of(context).textTheme.headline6,

    textAlign: TextAlign.center,

)),

const SizedBox(
    height: 20,
),

Padding(
    //Button actions

    padding: const EdgeInsets.only(left: 20, right: 20),

    child: Column(children: [
        SizedBox(
            width: double.infinity,
            child: ElevatedButton(
                style:
                    Theme.of(context).elevatedButtonTheme.style,
                onPressed: () => (Navigator.push( //go to profile
creation page
                    context,
                    MaterialPageRoute(
                        builder: (context) =>
                            const
AddProfileCreationPageName()))),
    
```

```
        child: const Text("Create a profile"))),  
  
        SizedBox(  
  
            width: 250,  
  
            child: TextButton(  
  
                style: Theme.of(context).textButtonTheme.style,  
  
                onPressed: () => (Navigator.push( // go to  
profile login page  
  
                    context,  
  
                    MaterialPageRoute(  
  
                        builder: (context) =>  
  
                            const AddProfileLoginPage()))),  
  
                child: const Align(  
  
                    alignment: Alignment.center,  
  
                    child: Text(  
  
                        "I already have a profile",  
  
                        textAlign: TextAlign.center,  
  
                    ))))  
  
            ]),  
  
        ),  
  
        const SizedBox(  
  
            height: 40,  
  
        ),  
  
    ]),
```

```
)  
],  
)  
]),  
);  
}  
}
```

profilesetup_login.dart

```
import 'package:alleat/services/dataencryption.dart';  
  
import 'package:alleat/services/localprofiles_service.dart';  
  
import 'package:alleat/services/queryserver.dart';  
  
import 'package:alleat/services/setselected.dart';  
  
import 'package:alleat/widgets/elements/elements.dart';  
  
import 'package:alleat/widgets/navbar.dart';  
  
import 'package:flutter/material.dart';  
  
import 'package:email_validator/email_validator.dart';  
  
import 'package:flutter/services.dart';  
  
import 'dart:async';  
  
  
class AddProfileLoginPage extends StatefulWidget {
```

```

const AddProfileLoginPage({Key? key}) : super(key: key);

}

@Override
State<AddProfileLoginPage> createState() => _AddProfileLoginPageState();

}

class _AddProfileLoginPageState extends State<AddProfileLoginPage> {

final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

static TextEditingController email =
    TextEditingController(); //Create text controllers to allow for dynamic
variable for form fields

static TextEditingController password = TextEditingController();

static dynamic encryptPassword;

late dynamic profileInfoImport;

Future<void> _loginUser() async {
    encryptPassword =
        await DataEncryption.encrypt(password.text); //Encrypt password

    var recieivedServerData =
        await QueryServer.query("https://alleat.cpur.net/query/login.php", {
            //Send data to login.php on server with email and encrypted password. It
            checks if the credentials are correct and returns exists if it is valid
            "email": email.text,

```

```
"password": encryptPassword,  
});  
  
if (recievedServerData["error"] == true) {  
  
    //If there is an error, clear password and display error from server  
  
    setState(() {  
  
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(  
  
            content: Text(recievedServerData["message"] +  
  
                " : Failed to login. Please try again")));  
  
    });  
  
    password.text = "";  
  
} else {  
  
    if (recievedServerData["message"]["exists"] == true) {  
  
        //If ther profile is correct and exists  
  
        List importedProfile = recievedServerData["message"]["profile"];  
  
        bool trySelect = await SetSelected.selectProfile(  
  
            importedProfile[0], //Try select profile  
  
            importedProfile[1],  
  
            importedProfile[2],  
  
            importedProfile[3]);  
  
        if (trySelect == false) {  
  
            // If the profile fails to select display error  
  
            setState(() {  
  
            });  
    }  
}
```

```
ScaffoldMessenger.of(context).showSnackBar(  
    const SnackBar(content: Text("Failed to select profile")));  
});  
  
} else {  
  
    // If succeeds to select profile  
  
    try {  
  
        await SQLiteLocalProfiles.createProfile(  
            //Create profile in database  
            importedProfile[0],  
            importedProfile[1],  
            importedProfile[2],  
            importedProfile[3],  
            importedProfile[4]);  
  
        email.text = ""; //Clear email and password  
        password.text = "";  
  
        setState(() {  
  
            ScaffoldMessenger.of(context).showSnackBar(  
                const SnackBar(content: Text('Successfully logged in.')),  
            );  
  
            Navigator.push(  
                context, //Go to main area  
                MaterialPageRoute(builder: (context) => const Navigation()));  
        });  
    } catch (e) {  
        ScaffoldMessenger.of(context).showSnackBar(  
            const SnackBar(content: Text("Failed to select profile")));  
    }  
}
```

```
});  
  
} catch (e) {  
  
    //If there is an error, display that there was an error  
  
    setState(() {  
  
        ScaffoldMessenger.of(context).showSnackBar(  
  
            const SnackBar(  
  
                content: Text('Failed to save profile on device.')),  
  
        );  
  
    });  
  
}  
  
}  
  
} else {  
  
    // If the password or email is incorrect, display incorrect email or  
password  
  
    setState(() {  
  
        ScaffoldMessenger.of(context).showSnackBar(  
  
            const SnackBar(content: Text("Incorrect email or password")));  
  
    });  
  
}  
  
}  
  
}  
  
Future<void> _checkDuplicate() async {
```

```
    List profileList = await SQLiteLocalProfiles.getProfiles();

    bool alreadyLogged = false;

    for (int i = 0; i < (profileList.length - 1); i++) {

        if (profileList[i]["email"].contains(email.text)) {

            alreadyLogged = true;

        }

    }

    if (alreadyLogged == false) {

        _loginUser();

    } else {

        setState(() {

            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text("Profile is already logged in.")));
        });

    }

}

@Override

Widget build(BuildContext context) {

    var brightness = MediaQuery.of(context).platformBrightness;

    bool darkModeOn = brightness == Brightness.dark;

    return Scaffold(
```

```
body: CustomScrollView(slivers: [  
    SliverFillRemaining(  
        hasScrollBody: false,  
        child: SingleChildScrollView(  
            child: Column(  
                crossAxisAlignment: CrossAxisAlignment.start,  
                mainAxisAlignment: MainAxisAlignment.spaceBetween,  
                children: <Widget>[  
                    Column(crossAxisAlignment: CrossAxisAlignment.start, children: [  
                        const ScreenBackButton(),  
                        Padding(  
                            padding:  
                                const EdgeInsets.only(left: 50, right: 50, top: 30),  
                            child: Image.asset((darkModeOn)  
                                ? 'lib/assets/images/screens/profilesetup/login-  
                                illustration-dark.png'  
                                : 'lib/assets/images/screens/profilesetup/login-  
                                illustration-light.png')),  
                        const SizedBox(  
                            height: 20,  
                        ),  
                        Padding(  
                            padding: const EdgeInsets.all(20),  
                        ),  
                ],  
            ),  
        ),  
    ],  
),
```

```
        child: Align(


            alignment: Alignment.center,


            child: Text("Welcome back.",


                style: Theme.of(context).textTheme.headline1)),


        const SizedBox(


            height: 20,


        ),


        Padding(


            padding: const EdgeInsets.all(15),


            child: Form(


                key: _formKey,


                child: SingleChildScrollView(


                    child: Column(


                        children: [


                            ListTile(


                                title: TextFormField(


                                    controller:


                                        email, //Form data lastname collected and sent to
database


                                    keyboardType: TextInputType.emailAddress,


                                    style: Theme.of(context).textTheme.bodyText2,


                                    decoration: (InputDecoration(


                                        hintText: "Email",
```

```
contentPadding: Theme.of(context)

    .inputDecorationTheme

    .contentPadding,

border: Theme.of(context)

    .inputDecorationTheme

    .border,

focusedBorder: Theme.of(context)

    .inputDecorationTheme

    .focusedBorder,

enabledBorder: Theme.of(context)

    .inputDecorationTheme

    .enabledBorder,

floatingLabelBehavior:

    FloatingLabelBehavior.never)),

inputFormatters: [
    //Only allows the input of letters a-z and A-Z and
@, . -
    FilteringTextInputFormatter.allow(
        RegExp('[a-zA-Z0-9@,. -]'))
],
validator: (email) {
    //Required field and uses emailvalidator package to
verify it is an email to simplify the code
```

```
        if (email == null || email.isEmpty) {

            return "Required";

        }

        if (EmailValidator.validate(email) == false) {

            return "Please enter valid email";

        }

        return null;

    },

)),

const SizedBox(

    height: 10,

),

ListTile(

    title: TextFormField(

        keyboardType: TextInputType.visiblePassword,

        style: Theme.of(context).textTheme.bodyText2,

        decoration: (InputDecoration(

            hintText: "Password",

            contentPadding: Theme.of(context)

                .inputDecorationTheme

                .contentPadding,

            border: Theme.of(context)


```

```
.inputDecorationTheme  
    .border,  
  
focusedBorder: Theme.of(context)  
    .inputDecorationTheme  
    .focusedBorder,  
  
enabledBorder: Theme.of(context)  
    .inputDecorationTheme  
    .enabledBorder,  
  
floatingLabelBehavior:  
    FloatingLabelBehavior.never)),  
  
inputFormatters: [  
  
    //Password cannot use " or ' in order to prevent  
    SQL injection  
  
    FilteringTextInputFormatter.allow(  
        RegExp('[a-zA-Z0-9!@#%^&*(),.?:{}}|<>]'))  
],  
  
obscureText: true, //Password not visible  
  
controller:  
  
    password, //Password copied and checked by  
confirm password  
  
validator: (password) {  
  
    //Must be a minimum of 8 characters and contain a  
letter and number to make sure there is variety and make it harder to guess. Must  
be under 99 characters so that it reduces processing time on the system
```

```
        if (password == null ||

            password.isEmpty ||

            password.length < 8 ||

            password.length > 99 ||

            !password.contains(RegExp(r'[0-9]')) ||

            !password.contains(RegExp(r'[a-z]')))

        return "Required";

    }

    return null;

},

),

),

const SizedBox(height: 50), //Gap

],

),

),

),

),

)

]),

Padding(



padding: const EdgeInsets.only(
```

```
        right: 30,  
  
        bottom: 60,  
  
(),  
  
child: Center(  
  
    child: SizedBox(  
  
        width: double.infinity,  
  
        height: 52,  
  
        child: ElevatedButton(  
  
            //submit button  
  
            style: Theme.of(context).elevatedButtonTheme.style,  
  
            onPressed: () {  
  
                if (_formKey.currentState!.validate()) {  
  
                    _checkDuplicate();  
  
                }  
  
            },  
  
            child: const Text('Login to Profile'),  
  
        ),  
  
    ))),  
  
],  
  
)))  
  
));  
}  
}
```

```
}
```

welcomescreen.dart

```
import 'package:alleat/screens/profilesetup/profilesetup_create.dart';

import 'package:alleat/screens/profilesetup/profilesetup_login.dart';

import 'package:flutter/material.dart';

class ProfileSetupWelcome extends StatefulWidget {

  const ProfileSetupWelcome({Key? key}) : super(key: key);

  @override
  State<StatefulWidget> createState() {
    return _ProfileSetupWelcome();
  }
}

class _ProfileSetupWelcome extends State<ProfileSetupWelcome> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      //Create new screen
      resizeToAvoidBottomInset: false, //Allow resize
```

```
body: Stack(children: [  
    Image.asset(  
        //Fullscreen image of food  
        'lib/assets/images/screens/welcomeScreen/welcomeScreenFood.jpg',  
        fit: BoxFit.cover,  
        height: double.infinity,  
        width: double.infinity,  
        alignment: Alignment.center,  
    ),  
    Column(  
        children: [  
            Padding(  
                //Image of All Eat logo  
                padding: const EdgeInsets.only(  
                    top: 70, left: 40, right: 40, bottom: 200),  
                child: Image.asset(  
                    'lib/assets/images/logo/logoDark.png',  
                    width: 58,  
                    height: 58,  
                ))  
        ],  
    ),  
],
```

```
),
Column(
  mainAxisAlignment: MainAxisAlignment.start,
  mainAxisSize: MainAxisSize.min,
  crossAxisAlignment: CrossAxisAlignment.end, //Create content at the bottom of the screen
  children: [
    Padding(
      padding: const EdgeInsets.only(left: 10, right: 10),
      child: Container(
        //Bottom container with login and register actions
        decoration: BoxDecoration(
          color: Theme.of(context).backgroundColor,
          borderRadius: const BorderRadius.only(
            topLeft: Radius.circular(20),
            topRight: Radius.circular(20)), // Round the top corners of container
        ),
        width: double.infinity,
        padding: const EdgeInsets.only(left: 20, right: 20),
        child: Column(children: [
          const SizedBox(
            height: 40,
          ),
        ],
      ),
    ),
  ],
);
```

```
Text("Let's Get Started.",  
    style: Theme.of(context).textTheme.headline2),  
  
Padding(  
    padding: const EdgeInsets.only(  
        left: 50, right: 50, top: 10, bottom: 10),  
  
    child: Text(  
        "A food delivery app with you in mind",  
        style: Theme.of(context).textTheme.headline6,  
        textAlign: TextAlign.center,  
    )),  
  
const SizedBox(  
    height: 20,  
,  
    Padding(  
        //Button actions  
        padding: const EdgeInsets.only(left: 20, right: 20),  
        child: Column(children: [  
            SizedBox(  
                width: double.infinity,  
                child: ElevatedButton(  
                    style:  
                        Theme.of(context).elevatedButtonTheme.style,
```

```
        onPressed: () => (Navigator.push( //Button to go  
to profile creation page  
  
            context,  
  
            MaterialPageRoute(  
  
                builder: (context) =>  
  
                    const  
AddProfileCreationPageName()))),  
  
                child: const Text("Create a profile"))),  
  
SizedBox(  
  
    width: 250,  
  
    child: TextButton(  
  
        style: Theme.of(context).textTheme.button,  
  
        onPressed: () => (Navigator.push( //Text button  
to go to login page  
  
            context,  
  
            MaterialPageRoute(  
  
                builder: (context) =>  
  
                    const AddProfileLoginPage()))),  
  
                child: const Align(  
  
                    alignment: Alignment.center,  
  
                    child: Text(  
  
                        "I already have a profile",  
  
                        textAlign: TextAlign.center,  
                    ))))
```

```
        ],
        ),
        const SizedBox(
            height: 40,
        ),
    ],
))
],
)
]),
);
}
}
```

restaurant_customise.dart

```
import 'dart:math';

import 'package:alleat/services/cart_service.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

class RestaurantItemCustomisePage extends StatefulWidget {
```

```
final String itemid;  
  
final String foodcategory;  
  
final String subfoodcategory;  
  
final String itemname;  
  
final String description;  
  
final String price;  
  
final String itemimage;  
  
final String reslogo;  
  
const RestaurantItemCustomisePage(  
  
{Key?  
  
key, //Get the items from the restaurant main page when an item is  
clicked  
  
required this.reslogo,  
  
required this.itemid,  
  
required this.foodcategory,  
  
required this.subfoodcategory,  
  
required this.itemname,  
  
required this.description,  
  
required this.price,  
  
required this.itemimage})  
  
: super(key: key);  
  
  
@override
```

```

State<RestaurantItemCustomisePage> createState() =>
    _RestaurantItemCustomisePageState();

}

class _RestaurantItemCustomisePageState
    extends State<RestaurantItemCustomisePage> {
    int quantity = 1; //default quantity 1

    Map customisedOptions = {};//changes to the item go in here in a dictionary

    bool beenMade = false; //Used to only build customisedOptions once

    double changingPrice = 0; //How much is the original price affected by
customised options

    List requiredFields = [];//Which customsie options are required to be filled

    late double quantityPrice = double.parse(widget.price);

    late double finalSinglePrice = double.parse(widget.price);

Future<bool> addToCart() async { //Add item with cutomsie options to db
    try {
        String customisedOptionsEncoded = json.encode(customisedOptions); //Encode
Dictionary into a string

        await SQLiteCartItems.addToCart(
            int.parse(widget.itemid), customisedOptionsEncoded, quantity);

        return true; //If no error, return true
    } catch (e) {

```

```

        return false; //If error, return false
    }

}

Future<Map> getCustomiseOptions() async {

    try {

        String phpurl = "https://alleat.cpur.net/query/itemcustomise.php";

        var res =

            await http.post(Uri.parse(phpurl), body: {"itemid": widget.itemid});

        if (res.statusCode == 200) {

            //If sends successfully

            var data = json.decode(res.body); //Decode to array

            if (data["error"]) {

                Map error = {

                    "error": true,

                    "message": "Server Error: ${data["message"]}",

                    "customise": "[]"

                } //Send blank list of customise data

                ;

            }

            return error;

            //If fails to perform query

        } else {
    
```

```
Map success = {  
    "error": false,  
    "message": "",  
    "customise": data["customiseitem"]  
} //Send back returned data  
  
;  
  
return success;  
}  
  
}  
else {  
    Map error = {  
        "error": true,  
        "message": "Error $e: Please try again",  
        "customise": "[]"  
} //Send blank list of customise data  
  
;  
  
return error;  
}  
}  
}  
catch (e) {  
    Map error = {  
        "error": true,  
        "message": "Unexpected Error: $e",  
        "customise": "[]"  
}
```

```
        } //Send blank list of customise data

        ;

        return error;

    }

}

@Override

Widget build(BuildContext context) {

    return Scaffold(

        body: SingleChildScrollView(

            child: Column(children: [

                Stack(children: [

                    //Display item image at bottom of stack

                    Container(

                        width: MediaQuery.of(context).size.width,

                        height: 240,

                        decoration: BoxDecoration(

                            image: DecorationImage(

                                fit: BoxFit.fitWidth,

                                image: NetworkImage(widget.itemimage),

                            ),

                        ),

                    ),


                ],


            ),


        ),


    );


}
```

```
Align(  
    //Display rounded container at bottom of image to respresent  
    overhanging image  
  
    alignment: Alignment.bottomCenter,  
  
    child: Container(  
  
        margin: const EdgeInsets.only(top: 215),  
  
        width: double.infinity,  
  
        height: 30,  
  
        decoration: BoxDecoration(  
  
            color: Theme.of(context).backgroundColor,  
  
            borderRadius:  
  
                const BorderRadius.vertical(top: Radius.circular(20))),  
  
    )),  
  
Align(  
    // Display background color as outline of restaurant logo,  
    overlapping the image background  
  
    alignment: Alignment.bottomCenter,  
  
    child: Container(  
  
        margin: const EdgeInsets.only(top: 180),  
  
        width: 80,  
  
        height: 80,  
  
        decoration: BoxDecoration(  
  
            shape: BoxShape.circle,
```

```
        color: Theme.of(context).backgroundColor,  
    )),  
  
Align(  
  
    // Display circle restaurant logo  
  
    alignment: Alignment.bottomCenter,  
  
    child: Container(  
  
        margin: const EdgeInsets.only(top: 185),  
  
        width: 70,  
  
        height: 70,  
  
        decoration: BoxDecoration(  
  
            image: DecorationImage(  
  
                fit: BoxFit.cover,  
  
                image: NetworkImage(widget.reslogo.toString()),  
  
            ),  
  
            shape: BoxShape.circle,  
  
            color: Theme.of(context).colorScheme.onSurface,  
        )),  
  
SafeArea(  
  
    // Display back button in a circle  
  
    child: InkWell(  
  
        onTap: () => Navigator.of(context).pop(),  
  
        child: Container(  
    
```

```
        margin: const EdgeInsets.only(top: 20, left: 20),  
  
        width: 50,  
  
        height: 50,  
  
        decoration: BoxDecoration(  
  
            shape: BoxShape.circle,  
  
            color: Theme.of(context).colorScheme.onSurface,  
  
  
        child: Icon(  
  
            Icons.chevron_left,  
  
            color: Theme.of(context).colorScheme.onBackground,  
  
            size: 35,  
  
        ),  
  
    ))),  
  
]),  
  
Padding(  
  
    // Item name  
  
    padding:  
  
    const EdgeInsets.only(top: 40, left: 30, right: 30, bottom: 10),  
  
    child: Text(  
  
        widget.itemname,  
  
        textAlign: TextAlign.center,  
  
        style: Theme.of(context).textTheme.headline2,
```

```

    )),

Row(mainAxisAlignment: MainAxisAlignment.center, children: [
    //Display restuarant category(ies)

    Text(widget.foodcategory, //Must have food category

        textAlign: TextAlign.center,
        style: Theme.of(context)

            .textTheme
            .headline6!
            .copyWith(color: Theme.of(context).textTheme.headline1!.color)),

LayoutBuilder(builder: (context, constraints) {

    if (widget.subfoodcategory != "") {

        //If there is a subcategory, display it with a dot next to it

        //If there is a sub food category, show it

        return Text(" · ${widget.subfoodcategory}",
            textAlign: TextAlign.center,
            style: Theme.of(context).textTheme.headline6!.copyWith(
                color: Theme.of(context).textTheme.headline1!.color));

    } else {

        // If there is no sub-category, dont display anything

        //If there isnt a sub-food category, dont show it

        return const Text("");
    }
}

```

```
    }),

    Padding(
        //Display item description
        padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 20),
        child: Text(
            widget.description,
            textAlign: TextAlign.center,
            style: Theme.of(context).textTheme.bodyText1!.copyWith(
                color: Theme.of(context).textTheme.headline6!.color,
                fontWeight: FontWeight.w400),
        ),
    ),
    FutureBuilder<Map>(
        future: getCustomiseOptions(), //Get customise options from server
        builder: (context, snapshot) {
            if (snapshot.hasData) {
                Map customiseData = snapshot.data ?? [] as Map;
                if (customiseData["error"] == true) {
                    //Check if there was an error getting the data from the server
                    return Container(
                        width: double.infinity,
```

```

padding:

    const EdgeInsets.symmetric(vertical: 30, horizontal: 20),

decoration: BoxDecoration(
    borderRadius: const BorderRadius.all(Radius.circular(10)),

    color: Theme.of(context).colorScheme.onSurface),

margin:

    const EdgeInsets.symmetric(vertical: 5, horizontal: 30),

child: Text(
    customiseData["message"], //Return that there was an error

    textAlign: TextAlign.center,
    style: Theme.of(context).textTheme.headline6,
),

);

} else {

    if (beenMade == false) { //If the customised dictionary has not
been made

        for (int i = 0; i < customiseData["customise"].length; i++) {
// For each customise option, add it as key with an empty list as the value

        customisedOptions[customiseData["customise"][[i][0]]] = [];

    }

    beenMade = true; //Set has been built to true

}

```

```

//If there was not an error getting the data

if (customiseData["customise"].length == 0) {

    //if there is no customise options, return nothing

    return const SizedBox(
        height: 1,
    );
}

} else {

    //If there are customise options, return customise options

    return ListView.builder(
        physics:
            const NeverScrollableScrollPhysics(), //Disable
scrolling. Scroll with whole page

        scrollDirection: Axis.vertical,
        shrinkWrap: true,
        itemCount: (customiseData[
            "customise"] //For each customise section, build
.length),
        itemBuilder: ((context, index) {
            if (customiseData["customise"][index][4] == "1" &&
                !requiredFields.contains(
                    customiseData["customise"][index][0])) { //For
each section marked as 1 (required), add customise id to required list. Check if
it has not been added before.

                requiredFields

```

```
        .add(customiseData["customise"][index][0]));

    }

    return Column( //Customise section

        children: [

            Container( //Thick border above each section to
separate each section.

                width: double.infinity,

                height: 10,

                color: Theme.of(context)

                    .colorScheme

                    .onBackground

                    .withOpacity(0.1),

            ),

            LayoutBuilder(builder: ((context, constraints) {

                if (customiseData["customise"][index][3] ==

                    "SELECT") { //If the section is marked as a
select widget

                    return Column(
                        children: [
                            Padding(
                                padding: const EdgeInsets.only(
                                    left: 20,
                                    right: 20,
```

```
        top: 30,  
  
        bottom: 10),  
  
        child: Row( //Top row containing the title,  
description and container indicating if it is required or optional  
  
        mainAxisAlignment:  
  
            CrossAxisAlignment.start,  
  
        children: [  
  
            Expanded( //Go to next line if too  
long  
  
            child: Column(  
  
            mainAxisAlignment:  
  
                CrossAxisAlignment  
  
                    .start,  
  
            children: [  
  
                Text( //Customise section title  
customiseData["customise"]  
  
                    [index][1]  
  
                    .toString(),  
  
                style: Theme.of(context)  
  
                    .textTheme  
  
                        .headline5,  
  
                overflow:  
  
                    TextOverflow.visible,  
]
```

```
        ),  
  
        const SizedBox(  
  
            height: 10,  
  
) , //Customise section  
  
description  
  
Text(  
  
    customiseData["customise"]  
  
    [index][2],  
  
    style: Theme.of(context)  
  
.textTheme  
  
.headline5,  
  
overflow:  
  
    TextOverflow.visible,  
  
)  
  
]),  
  
const SizedBox(width: 30),  
  
LayoutBuilder(builder:  
  
(context, constraints) {  
  
if (customiseData["customise"]  
  
[index][4] ==  
  
"0") { //If the customise  
section is not required, display container in green with text "optional"  
  
return Container(  
  
)
```

```
decoration: BoxDecoration(  
    color: Theme.of(context)  
        .colorScheme  
        .tertiary  
        .withOpacity(0.2),  
    borderRadius:  
        BorderRadius.circular(  
            20)),  
    padding: const EdgeInsets  
        .symmetric(  
            horizontal: 20,  
            vertical: 5),  
    child: Text(  
        "Optional",  
        style: Theme.of(context)  
            .textTheme  
            .headline6  
            ?.copyWith(  
                color: Theme.of(  
                    context)  
                    .colorScheme  
                    .tertiary),
```

```
        ),  
    );  
  
} else if (customiseData[  
    "customise"][[index][4] ==  
    "1"]) { //If the customise  
section is marked as required, display container in primary colour (purple) with  
text "required"  
  
    return Container(  
        decoration: BoxDecoration(  
            color: Theme.of(context)  
                .primaryColor  
                .withOpacity(0.2),  
            borderRadius:  
                BorderRadius.circular(  
                    20)),  
        padding: const EdgeInsets  
            .symmetric(  
                horizontal: 20,  
                vertical: 5),  
        child: Text(  
            "Required",  
            style: Theme.of(context)  
                .textTheme
```

```
        .headline6

        ?.copyWith(
            color: Theme.of(
                context)

            .primaryColor),
        ),
    );
}

} else { //If the section is not
marked as 0 or 1, display it as and error with a red container and text "ERROR"

    return Container(
        decoration: BoxDecoration(
            color: Theme.of(context)
                .colorScheme
                .error
                .withOpacity(0.2),
        borderRadius:
            BorderRadius.circular(
                20)),
        padding: const EdgeInsets
            .symmetric(
                horizontal: 20,
                vertical: 5),
        child: Text(

```

```
        "ERROR",

        style: Theme.of(context)

            .textTheme

            .headline6

            ?.copyWith(

                color: Theme.of(

                    context)

                .colorScheme

                .error),

            ),

        );

    }

}

}())

]),

),

ListView.builder( //For each option on the
customise section

physics:

const NeverScrollableScrollPhysics(),

//Disable scrolling. Scroll with whole page

scrollDirection: Axis.vertical,

shrinkWrap: true,

itemCount: customiseData["customise"]
```

```

        [index][7]

        .length,

        itemBuilder: ((context, index2) {

            return Padding(
                padding: const
                EdgeInsets.symmetric(
                    horizontal: 20, vertical: 5),
                child: ElevatedButton( //Create a
button for the option
                    style: ButtonStyle(
                        side:
(customisedOptions[customiseData["customise"]][index][0]).contains(
customiseData["customise"]
                    [index][7]
                        [index2][0])) //If the option is in the customisedOptions (selected) then display with purple
border otherwise don't have a border
                    ?
MaterialStateProperty.all(BorderSide(
                        width: 2,
                        color:
Theme.of(context)
                            .primaryColor))
                    : null,
                    alignment:

```

```

        Alignment.centerLeft,
        padding:
MaterialStateProperty.all(const EdgeInsets.symmetric(horizontal: 30, vertical:
20)),
        textStyle:
(customisedOptions[customiseData["customise"][[index][0]]].contains(customiseData["
customise"][[index][7][index2][0]])) //If the option is in the customisedOptions
(selected), display with brighter text
        ?
MaterialStateProperty.all(Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).textTheme.headline1?.color))
        :
MaterialStateProperty.all(Theme.of(context).textTheme.headline6),
        backgroundColor:
MaterialStateProperty.all(Theme.of(context).colorScheme.onSurface)),
        onPressed: () {
        if
(customisedOptions[customiseData["customise"][[index][0]]]
        .length <
        int.parse(
customiseData["customise"]
        [index]
        [6]) &&
        !customisedOptions[
customiseData["customise"]

```

```
[index][0]]  
.contains(  
  
customiseData["customise"]  
  
[index][7]  
[index2][0]))  
{ //If the length is less than the max amount selected and it is not in the  
customisedoptions list add it to the list and add to the changed price  
  
        setState(() {  
  
        customisedOptions[  
  
        customiseData[  
  
"customise"]  
[index][0]]  
.add(customiseData[  
  
"customise"]  
[index][7]  
[index2][0]);  
changingPrice += double  
.parse(customiseData[  
  
"customise"]  
[index][7]
```

```

        [index2][3]);

    });

} else if (customisedOptions[
    customiseData[
        "customise"]
    [index][0]]
.contains(customiseData[
    "customise"]
    [index][7]
    [index2][0])) { //If
the option is in the customisedoptions list, remove from the list and remove to
the changed price

    setState(() {
        customisedOptions[
            customiseData[
                "customise"]
            [index][0]]
        .remove(customiseData[
            "customise"]
            [index][
                7][index2][0]

```

```

        .toString());

        changingPrice -= double

        .parse(customiseData[

"customise"]

        [index][7]

        [index2][3]);

    });

}

},
child: Row(mainAxisAlignment:
MainAxisAlignment.spaceBetween, children: [ //Within button, display a row
containing the option text and the price change

Expanded(
    child: Text(
        customiseData["customise"]

        [index][7][index2][1],

        style: (customisedOptions[

customiseData["customise"]

        [index]

        [0]]]

.contains(customiseData["customise"]

```

```
[index][7]

        [index2][0)))
//If the option is in the customisedOptions (selected), display with brighter
text

        ? Theme.of(context)

        .textTheme

        .headline6

        ?.copyWith(
            color:

Theme.of(context)

        .textTheme

        .headline1

        ?.color)

        : Theme.of(context)

        .textTheme

        .headline6,

        )),
LayoutBuilder(
    builder: ((context,
constraints) {

        if (customiseData[

"customise"]

        [index][7]
```

```

        [index2][3] !=

                "0.00") { //If the
price change is not nothing, display the price change in red

            return Text(


"+E${customiseData["customise"][[index][7][index2][3]]}",

            style: Theme.of(
                context)

                .textTheme
                    .headline6

                    ?.copyWith(
                        color:

Theme.of(


context)

.colorScheme

                    .error),

                );
            } else { //If the price
change in nothing, display dash in grey

            return Text(
                "-",
                style: Theme.of(
                    context)

```

```
.textTheme  
    .headline6  
    ?.copyWith(  
        color:  
Theme.of(  
  
context)  
  
.colorScheme  
  
.onBackground  
  
.withOpacity(  
    0.5)),  
);  
}  
});  
]);  
});  
const SizedBox(height: 50)  
],  
);  
} else if (customiseData["customise"][index][3] ==  
"ADD") {
```

```

        return Column(
            children: [
                Padding(
                    padding: const EdgeInsets.only(
                        left: 20,
                        right: 20,
                        top: 30,
                        bottom: 10),
                    child: Row( //Top row containing the title,
description and container indicating if it is required or optional
                        mainAxisAlignment:
                            MainAxisAlignment.start,
                        children: [
                            Expanded( //Go to next line if too
long
                                child: Column(
                                    mainAxisAlignment:
                                        MainAxisAlignment
                                            .start,
                                    children: [
                                        Text( //Customise section title
                                            customiseData["customise"]
                                                [index][1]

```

```
        .toString(),

        style: Theme.of(context)

            .textTheme

            .headline5,

            overflow:

                TextOverflow.visible,

            ),

            const SizedBox(

                height: 10,

            ), //Customise section

description

Text(


    customiseData["customise"]

        [index][2],


        style: Theme.of(context)

            .textTheme

            .headline6,


            overflow:

                TextOverflow.visible,


            ),


            const SizedBox(


                height: 20,


            ),
```

```
]),

const SizedBox(width: 30),

LayoutBuilder(builder:

((context, constraints) {

if (customiseData["customise"]

[index][4] ==

"0") { //If the customise
section is not required, display container in green with text "optional"

return Container(
decoration: BoxDecoration(
color: Theme.of(context)

.colorScheme

.tertiary

.withOpacity(0.2),

borderRadius:
BorderRadius.circular(
20)),

padding: const EdgeInsets

.symmetric(
horizontal: 20,

vertical: 5),

child: Text(
"Optional",
```

```
        style: Theme.of(context)

        .textTheme

        .headline6

        ?.copyWith(
            color: Theme.of(
                context)
            .colorScheme
            .tertiary),
        ),
    );
}

} else if (customiseData[
    "customise"][index][4] ==
    "1") { //If the customise
section is marked as required, display container in primary colour (purple) with
text "required"

    return Container(
        decoration: BoxDecoration(
            color: Theme.of(context)
            .primaryColor
            .withOpacity(0.2),
        borderRadius:
            BorderRadius.circular(
                20)),
}
```

```
padding: const EdgeInsets  
        .symmetric(  
            horizontal: 20,  
            vertical: 5),  
  
        child: Text(  
            "Required",  
            style: Theme.of(context)  
                .textTheme  
                .headline6  
                ?.copyWith(  
                    color: Theme.of(  
                        context)  
                        .primaryColor),  
            ),  
        );  
  
    } else { //If the section is not  
marked as 0 or 1, display it as and error with a red container and text "ERROR"  
  
        return Container(  
            decoration: BoxDecoration(  
                color: Theme.of(context)  
                    .colorScheme  
                    .error  
                    .withOpacity(0.2),
```

```
borderRadius:  
    BorderRadius.circular(  
        20)),  
    padding: const EdgeInsets  
        .symmetric(  
            horizontal: 20,  
            vertical: 5),  
    child: Text(  
        "ERROR",  
        style: Theme.of(context)  
            .textTheme  
            .headline6  
            ?.copyWith(  
                color: Theme.of(  
                    context)  
                    .colorScheme  
                    .error),  
            ),  
        );  
    }  
});  
],
```

```
        ),  
  
        ListView.builder( //For each option on the  
        customise section  
  
            physics:  
  
                const NeverScrollableScrollPhysics(),  
//Disable scrolling. Scroll with whole page  
  
            scrollDirection: Axis.vertical,  
  
            shrinkWrap: true,  
  
            itemCount: customiseData["customise"]  
  
                [index][7]  
  
            .length,  
  
            itemBuilder: ((context, index2) {  
  
                return Padding(  
  
                    padding:  
  
                        const EdgeInsets.symmetric(  
  
                            horizontal: 20,  
  
                            vertical: 5),  
  
                    child: ElevatedButton( //Create a  
button for the option  
  
                        style: ButtonStyle(  
  
                            side:  
  
(customisedOptions[customiseData["customise"]][index][0]).contains(customiseData["  
customise"][index][7][index2][0])) //If the option is in the customisedOptions  
(selected) then display with purple border otherwise don't have a border
```

```

MaterialStateProperty.all(BorderSide(
    width: 2,
    color:
    Theme.of(context)
        .primaryColor))
    : null,
    alignment:
        Alignment.centerLeft,
    padding:
MaterialStateProperty.all(
    const
EdgeInsets.symmetric(
    horizontal: 30,
    vertical: 20)),
    textStyle:
(customisedOptions[customiseData["customise"][index][0]].contains(customiseData["customise"][index][7][index2][0])) //If the option is in the customisedOptions (selected), display with brighter text
    ?
MaterialStateProperty.all(Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).textTheme.headline1?.color))
    :
MaterialStateProperty.all(Theme.of(context).textTheme.headline6),
    backgroundColor:
MaterialStateProperty.all(Theme.of(context).colorScheme.onSurface)),
    onPressed: () {

```



```
"customise"]  
[index][7]  
[index2][0]);  
changingPrice += double  
.parse(customiseData[  
  
"customise"]  
[index][7]  
[index2][3]);  
});  
}  
},  
child: LayoutBuilder(builder:  
((context, constraints) {  
int count = 0;  
for (int i = 0;  
i <  
customisedOptions[  
customiseData[  
"customise"]  
[  
index][0]]
```

```

.length;

    i++) { //Count the number
of times, the customise option is mentioned in the customised list. This will get
the quantity.

        if (customisedOptions[
            customiseData[
                "customise"]
            [
                index][0]][i] ==
            customiseData[
                "customise"]
            [index][7]
            [index2][0])) {
                count += 1;
            }
        }

        if (count == 0) { //If the
option is not in the list (not selected)

            return Row(
                mainAxisAlignment:
                    MainAxisAlignment
                    .spaceBetween,
                children: [ //Display row
with icon, customise option text and price

```

```
Icon(  
    Icons  
  
.add_box_outlined,  
    color: Theme.of(  
        context)  
        .colorScheme  
        .tertiary,  
,  
const SizedBox(  
    width: 30),  
Expanded(  
    child: Text(  
        customiseData[  
"customise"]  
        [index][7]  
        [index2][1],  
        style:  
(customisedOptions[  
        customiseData["customise"][index]  
        [0]]
```

```
.contains(customiseData["customise"][index][7]

[index2]

[0])))

//If the option is in the list, display it in a brighter text

?

Theme.of(context)

    .textTheme

    .headline6

    ?.copyWith(

        color:

Theme.of(context)

    .textTheme

    .headline1

    ?.color)

    :

Theme.of(context)

    .textTheme

    .headline6,

    )),

const SizedBox(
    width: 20),
```

```

LayoutBuilder(builder:

constraints) {

"customise"]

[

index][7]

[

index2][3] !=

"0.00") { //If

the price change is not nothing, display the price change in red

return Text(


"+${customiseData["customise"][[index][7][index2][3]]}",

style: Theme.of(


context)

.textTheme

.headline6

?.copyWith(


color:

Theme.of(context)

.colorScheme

```

```
.error),  
);  
  
} else { //If the  
price change is nothing, display dash in grey  
  
    return Text(  
  
        "-",  
        style: Theme.of(  
            context)  
  
            .textTheme  
            .headline6  
            ?.copyWith(  
                color:  
Theme.of(context)  
  
.colorScheme  
  
.onBackground  
  
.withOpacity(0.5)),  
);  
}  
})  
]);
```

```
        } else { //If the option is in  
the list  
  
            return Row( //Display row  
with the quantity, text and a delete button  
  
                mainAxisAlignment:  
  
                    MainAxisAlignment  
  
                    .spaceBetween,  
  
                children: [  
  
                    CircleAvatar(  
  
                        radius: 15,  
  
                        backgroundColor:  
  
                            Theme.of(  
  
                                context)  
  
                                .colorScheme  
  
                                .tertiary  
  
                                .withOpacity(  
  
                                    0.5),  
  
                            child: Text(  
  
                                count  
  
                                .toString(),  
  
                            style: Theme.of(  
  
                                context)  
  
                                .textTheme  

```

```
        .headline6

        ?.copyWith(
            color:
Theme.of(context)

        .textTheme

        .headline1

        ?.color)),
        ),
        const SizedBox(
            width: 30,
        ),
        Expanded(
            child: Text(
                customiseData[
"customise"]

                [index][7]
                [index2][1],
                style:
(customisedOptions[
customiseData["customise"][index]
```

```
[0]]
```

```
.contains(customiseData["customise"][index][7]

[index2]

[0]))

//If the option is in the customisedOptions list, display the text brighter

?

Theme.of(context)

.textTheme

.headline6

?.copyWith(

color:

Theme.of(context)

.textTheme

.headline1

?.color)

:

Theme.of(context)

.textTheme

.headline6,

)),

const SizedBox(
```

```
        width: 20),  
  
        IconButton( //Delete  
button  
  
            icon: const Icon(  
  
                Icons.delete),  
  
            splashRadius: 15,  
  
            color: Theme.of(  
  
                context)  
  
.colorScheme  
  
.error,  
  
        onPressed: () { //On  
remove button pressed, remove from the list and remove from changed price.  
  
            setState(() {  
  
customisedOptions[  
  
customiseData["customise"][index]  
  
[0]]  
  
.remove(customiseData["customise"][index][7]  
  
[  
  
index2][0]  
  
.toString());  
    }  
}
```

```
changingPrice -=  
  
double.parseDouble(customiseData["customise"]  
[  
  
index][7]  
[  
  
index2][3]);  
  
});  
  
},  
),  
]);  
}  
}),  
));  
});  
});  
const SizedBox(height: 50)  
],  
);  
}  
else if (customiseData["customise"][index][3] ==  
"REMOVE") {  
return Column(  
children: [  
]  
};
```

```
Padding(  
  padding: const EdgeInsets.only(  
    left: 20,  
    right: 20,  
    top: 30,  
    bottom: 10),  
  
  child: Row( //Top row containing the title,  
description and container indicating if it is required or optional  
    mainAxisAlignment:  
      MainAxisAlignment.start,  
    children: [  
      Expanded( //Go to next line if too  
long  
        child: Column(  
          mainAxisAlignment:  
            MainAxisAlignment  
              .start,  
          children: [  
            Text( //Customise section title  
              customiseData["customise"]  
                [index][1]  
                .toString(),  
            style: Theme.of(context)
```

```
        .textTheme  
        .headline5,  
  
        overflow:  
          TextOverflow.visible,  
        ),  
        const SizedBox(  
          height: 10,  
        ), //Customise section  
description  
  
        Text(  
          customiseData["customise"]  
          [index][2],  
          style: Theme.of(context)  
            .textTheme  
            .headline6,  
          overflow:  
            TextOverflow.visible,  
        ),  
        const SizedBox(  
          height: 20,  
        ),  
      ])),  
      const SizedBox(width: 30),
```

```
LayoutBuilder(builder:  
  
    ((context, constraints) {  
  
        if (customiseData["customise"]  
  
            [index][4] ==  
  
                "0") { //If the customise  
section is not required, display container in green with text "optional"  
  
            return Container(  
  
                decoration: BoxDecoration(  
  
                    color: Theme.of(context)  
  
                        .colorScheme  
  
                        .tertiary  
  
                        .withOpacity(0.2),  
  
                borderRadius:  
  
                    BorderRadius.circular(  
  
                        20)),  
  
                padding: const EdgeInsets  
  
                    .symmetric(  
  
                        horizontal: 20,  
  
                        vertical: 5),  
  
                child: Text(  
  
                    "Optional",  
  
                    style: Theme.of(context)  
  
                        .textTheme
```

```

        .headline6

        ?.copyWith(
            color: Theme.of(
                context)

            .colorScheme
            .tertiary),
    ),
    );
}

} else if (customiseData[
    "customise"][index][4] ==
    "1") { //If the customise
section is marked as required, display container in primary colour (purple) with
text "required"

    return Container(
        decoration: BoxDecoration(
            color: Theme.of(context)
            .primaryColor
            .withOpacity(0.2),
        borderRadius:
            BorderRadius.circular(
                20)),
        padding: const EdgeInsets
            .symmetric(

```

```
        horizontal: 20,  
  
        vertical: 5),  
  
        child: Text(  
  
            "Required",  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline6  
  
                ?.copyWith(  
  
                    color: Theme.of(  
  
                        context)  
  
                    .primaryColor),  
  
(  
  
    ),  
  
);  
  
} else { //If the section is not  
marked as 0 or 1, display it as and error with a red container and text "ERROR"  
  
    return Container(  
  
        decoration: BoxDecoration(  
  
            color: Theme.of(context)  
  
                .colorScheme  
  
                .error  
  
                .withOpacity(0.2),  
  
        borderRadius:  
  
            BorderRadius.circular(  

```

```
        20)),  
  
        padding: const EdgeInsets  
            .symmetric(  
                horizontal: 20,  
                vertical: 5),  
  
        child: Text(  
            "ERROR",  
            style: Theme.of(context)  
                .textTheme  
                .headline6  
                ?.copyWith(  
                    color: Theme.of(  
                        context)  
                        .colorScheme  
                        .error),  
            ),  
        );  
    }  
});  
],  
,  
),  
ListView.builder( //For each option on the  
customise section
```

```
physics:  
    const NeverScrollableScrollPhysics(),  
//Disable scrolling. Scroll with whole page  
  
    scrollDirection: Axis.vertical,  
  
    shrinkWrap: true,  
  
    itemCount: customiseData["customise"]  
  
        [index][7]  
  
        .length,  
  
    itemBuilder: ((context, index2) {  
  
        return Padding(  
  
            padding:  
  
                const EdgeInsets.symmetric(  
  
                    horizontal: 20,  
  
                    vertical: 5),  
  
            child: ElevatedButton( //Create a  
button for the option  
  
                style: ButtonStyle(  
  
                    side:  
(customisedOptions[customiseData["customise"][index][0]].contains(customiseData["  
customise"][index][7][index2][0])) //If the option is in the customisedOptions  
(selected) then display with purple border otherwise don't have a border  
  
                    ?  
MaterialStateProperty.all(BorderSide(  
  
                        width: 2,
```

```

        color:

Theme.of(context)

        .primaryColor))

        : null,

        alignment:

Alignment.centerLeft,

padding:

MaterialStateProperty.all(


        const

EdgeInsets.symmetric(


        horizontal: 30,


        vertical: 20)),


        textStyle:

(customisedOptions[customiseData["customise"][[index][0]].contains(customiseData["customise"][[index][7][index2][0]])) //If the option is in the customisedOptions (selected), display with brighter text


        ?


MaterialStateProperty.all(Theme.of(context).textTheme.headline6?.copyWith(color:


Theme.of(context).textTheme.headline1?.color))



        :



MaterialStateProperty.all(Theme.of(context).textTheme.headline6),


        backgroundColor:


MaterialStateProperty.all(Theme.of(context).colorScheme.onSurface)),


        onPressed: () {




        if


(customisedOptions[customiseData["customise"][[index][0]]


.length <

```

```
int.parse(customiseData[  
    "customise"]  
[index][6]) &&  
!customisedOptions[  
  
customiseData["customise"]  
[index][0]]  
.contains(  
  
customiseData["customise"]  
[index][7]  
[index2])) {  
//If the length is less than the max amount selected and it is not in the  
customisedoptions list add it to the list and add to the changed price  
setState(() {  
    customisedOptions[  
        customiseData[  
    "customise"]  
[index][0]]  
.add(customiseData[  
    "customise"]  
[index][7]
```

```
[index2][0]);  
  
changingPrice -= double  
.parse(customiseData[  
  
"customise"]  
  
[index][7]  
  
[index2][3]));  
  
});  
}  
  
},  
  
child: LayoutBuilder(builder:  
  
((context, constraints) {  
  
int count = 0;  
  
for (int i = 0;  
  
i <  
  
customisedOptions[  
  
customiseData[  
  
"customise"]  
  
[  
  
index][0]]  
  
.length;
```

```

        i++) { //Count the number
of times, the customise option is mentioned in the customised list. This will get
the quantity.

        if (customisedOptions[
            customiseData[
                "customise"]
            [
                index][0]][i] ==
            customiseData[
                "customise"]
            [index][7]
            [index2][0])) {

            count += 1;
        }
    }

    if (count == 0) { //If the
option is not in the list (not selected)

        return Row(
            mainAxisAlignment:
            MainAxisAlignment
            .spaceBetween,
            children: [ //Display row
with icon, customise option text and price
Icon(

```

Icons

```
.disabled_by_default_outlined,  
    color: Theme.of(  
        context)  
            .colorScheme  
            .error,  
(  
    const SizedBox(  
        width: 30),  
    Expanded(  
        child: Text(  
            customiseData[  
"customise"]  
                [index][7]  
                [index2][1],  
                style:  
(customisedOptions[  
    customiseData["customise"][index]  
        [0]]  
.contains(customiseData["customise"][index][7]
```

```
[index2]

[0])))

//If the option is in the list, display it in a brighter text

?

Theme.of(context)

    .textTheme

    .headline6

    ?.copyWith(

        color:

Theme.of(context)

    .textTheme

    .headline1

    ?.color)

    :

Theme.of(context)

    .textTheme

    .headline6,

)),

const SizedBox(
    width: 20),

LayoutBuilder(builder:

    ((p0, p1) {
```

```

        if (customiseData[
    "customise"]
        [
            [
                [
                    [
                        index2][3] !=

                    "0.00") { //If
the price change is not nothing, display the price change in red

                    return Text(
                            "-
${customiseData["customise"][[index][7][index2][3]}",
                    style: Theme.of(
                            context)
                            .textTheme
                            .headline6
                            ?.copyWith(
                                color:
Theme.of(context)

                            .colorScheme
                            .tertiary),
                    );

```

```
        } else { //If the
price change is nothing, display dash in grey

            return Text(
                "-",
                style: Theme.of(
                    context)
                    .textTheme
                    .headline6
                    ?.copyWith(
                        color:
Theme.of(context)

.colorScheme

.onBackground

.withOpacity(0.5)),
            );
        }
    }))

]);
}

} else { //If the option is in
the list

return Row( //Display row
with the quantity, text and a delete button
```

```
mainAxisAlignment:  
    MainAxisAlignment  
    .spaceBetween,  
  children: [  
    CircleAvatar(  
      radius: 15,  
      backgroundColor:  
        Theme.of(  
          context)  
        .colorScheme  
        .error  
        .withOpacity(  
          0.5),  
      child: Text(  
        count  
        .toString(),  
        style: Theme.of(  
          context)  
        .textTheme  
        .headline6  
        ?.copyWith(  
          color:  
Theme.of(context)
```

```
.textTheme  
  
.headline1  
  
?.color)),  
),  
const SizedBox(  
width: 30,  
),  
Expanded(  
child: Text(  
customiseData[  
"customise"]  
[index][7]  
[index2][1],  
style:  
(customisedOptions[  
customiseData["customise"][index]  
[θ]]  
.contains(customiseData["customise"][index][7]  
[index2]
```

```
[0]))  
//If the option is in the customisedOptions list, display the text brighter  
?  
Theme.of(context)  
    .textTheme  
    .headline6  
    ?.copyWith(  
        color:  
Theme.of(context)  
  
.textTheme  
  
.headline1  
  
?.color)  
:  
Theme.of(context)  
    .textTheme  
    .headline6,  

```

```
        splashRadius: 15,  
  
        color: Theme.of(  
  
            context)  
  
.colorScheme  
  
.error,  
  
        onPressed: () { //On  
remove button pressed, remove from the list and add from changed price.  
  
        setState(() {  
  
customisedOptions[  
  
customiseData["customise"][index]  
  
[0]]  
  
.remove(customiseData["customise"][index][7]  
  
[  
  
index2][0]  
  
.toString());  
  
        changingPrice +=  
  
double.parse(customiseData["customise"]  
  
[  
  
index][7]
```

```
index2][3]);  
    );  
  },  
  ),  
]);  
}  
},  
});  
});  
});  
});  
const SizedBox(height: 50)  
],  
);  
} else { //If there is an unknown customise type  
(not SELECT, ADD or REMOVE) dispaly container with the text "Unknown Customise  
Option"  
return Container(  
width: double.infinity,  
padding: const EdgeInsets.symmetric(  
vertical: 30, horizontal: 20),  
decoration: BoxDecoration(  
borderRadius: const BorderRadius.all(  
Radius.circular(10)),
```

```
        color: Theme.of(context)

        .colorScheme

        .onSurface),

margin: const EdgeInsets.symmetric(
    vertical: 5, horizontal: 30),

child: Text(
    "Unknown Customise Option", //Return that
there was an error

    textAlign: TextAlign.center,

    style:

        Theme.of(context).textTheme.headline6,

    ),

);

}

}))

],

);

}));

}

}

} else {

//While loading, return progress indicator

return const LinearProgressIndicator()
```

```

        color: Color(0xff4100C4), backgroundColor: Color(0xffEBE0FF));

    }

}),

//Add to cart and quantity

LayoutBuilder(builder: ((context, constraints) {

    finalSinglePrice =

        ((double.parse(widget.price) * 100) + (changingPrice * 100)) / 100;
//In order to do correct calculations with a double type, multiply all by 100 to
get integer and divide back to get it in pence and pounds

    quantityPrice = finalSinglePrice * quantity; //To get the quantity price
multiply by quantity

return Column(crossAxisAlignment: CrossAxisAlignment.center, children: [
    Text("£${finalSinglePrice.toStringAsFixed(2)}", //Display individual
price by 2dp.

    style: Theme.of(context).textTheme.headline4),

    Padding(
        padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 20),
        child: Row( //Display row with the quantity and add and subtract
buttons

        mainAxisAlignment: MainAxisAlignment.center,
        children: [
            CircleAvatar( //Remove from quantity

```

```
radius: 30,  
  
backgroundColor: Theme.of(context).colorScheme.onSurface,  
  
child: IconButton(  
  
    icon: const Icon(Icons.remove),  
  
    iconSize: 25,  
  
    color: (quantity == 1) //if the quantity is 1 (unable to  
subtract from quantity), display at 0.1 opacity  
    ? Theme.of(context).primaryColor.withOpacity(0.1)  
    : Theme.of(context).primaryColor,  
  
    onPressed: () {  
  
        if (quantity > 1) { //If the quantity is greater than  
1, allow for removing 1 from quantity  
  
            setState(() {  
  
                quantity -= 1;  
  
            });  
  
        }  
  
    },  
  
)),  
  
const SizedBox(  
  
    width: 30,  
  
)  
  
SizedBox( //Quanity text with fixed width  
  
width: 30,
```

```

        child: Text(
            quantity.toString(),
            textAlign: TextAlign.center,
            style: Theme.of(context).textTheme.headline3,
        )),
        const SizedBox(
            width: 30,
        ),
        CircleAvatar( //Add from quantity
            radius: 30,
            backgroundColor: Theme.of(context).colorScheme.onSurface,
            child: IconButton(
                icon: const Icon(Icons.add),
                iconSize: 25,
                color: (quantity == 99)//if the quantity is 99 (unable to
add to quantity), display at 0.1 opacity
                    ? Theme.of(context).primaryColor.withOpacity(0.1)
                    : Theme.of(context).primaryColor,
                onPressed: () {
                    if (quantity < 99) {//If the quantity is less than 99,
allow for adding 1 to quantity. Max 99 to stop overloading restaurants.
                    setState(() {
                        quantity += 1;

```

```

        });

    }

    },
    )),
    ],
)),
Padding(
padding: const EdgeInsets.all(20),
child: Row(children: [
Expanded(child:
LayoutBuilder(builder: ((context, constraints) { //Cart
bool tempCheckRequiredFilled = true; //Check if required is
fullfilled

for (int i = 0; i < requiredFields.length; i++) { //For each
customise section that is marked as required, check if each customisedOption are
not empty. If it is empty, change tempCheckRequiredFilled to false. Disables add
to cart button

if (customisedOptions[requiredFields[i]].isEmpty) {

tempCheckRequiredFilled = false;

}

if (tempCheckRequiredFilled == true) { //if required fields are
filled

return ElevatedButton(

```

```

        onPressed: () async { //On press add to cart (add to
database)

            bool isAddedToCart = await addToCart();

            if (isAddedToCart == true) { //If is successfully adds
to cart, display success

                setState(() {

                    ScaffoldMessenger.of(context).showSnackBar(
                        const SnackBar(
                            content:
                                Text("Successfully added to Cart")));
                });
            } else { //If fails to add to cart, display failed

                setState(() {

                    ScaffoldMessenger.of(context).showSnackBar(
                        const SnackBar(
                            content: Text("Failed to add to Cart")));
                });
            }
        },
    ),
    child: Row( //Display row with text "add to cart" and the
price for the quantity

        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
            const Text("Add to Cart"),

```

```

        Text("£${quantityPrice.toStringAsFixed(2)}")

    ]));

} else {

    return Opacity( //Display add to cart button with low opacity
and disable onPressed

        opacity: 0.2,

        child: ElevatedButton(

            style: ButtonStyle(
                backgroundColor: MaterialStatePropertyAll(
                    Theme.of(context)

                        .colorScheme

                        .onBackground)),
                onPressed: () {},
                child: Row(
                    mainAxisAlignment:
                        MainAxisAlignment.spaceBetween,
                    children: [
                        const Text("Add to Cart"),
                        Text("£${quantityPrice.toStringAsFixed(2)}")
                    ]));
}

})));
]
),

```

```
        const SizedBox(height: 20),  
  
    ]);  
  
});  
  
]);  
  
}  
  
}
```

restaurant_main.dart

```
import 'package:alleat/screens/restaurant/restaurant_customise.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

import 'package:shared_preferences/shared_preferences.dart';

class RestaurantMain extends StatefulWidget {

    final String resid;

    final String resname;

    final String reslogo;

    final String resbanner;

    final String resdistance;

    final String resdelivery;
```

```
final String resordermin;

const RestaurantMain(
    //Get restaurant info from the restaurant list widget (passed from the
    container)

    {

        Key? key,
        required this.resid,
        required this.resname,
        required this.reslogo,
        required this.resbanner,
        required this.resdistance,
        required this.resdelivery,
        required this.resordermin,
    } : super(key: key);

    @override
    State<RestaurantMain> createState() => _RestaurantMainState();
}

class _RestaurantMainState extends State<RestaurantMain> {
    Future<List> getMenuCategories() async {
        String phpurl =
```

```
"https://alleat.cpur.net/query/restaurantmenucategories.php"; //Get the
list of menu categories associated with the restaurant id

try {

  var res = await http.post(Uri.parse(phiurl), body: {

    "restaurantid": widget.resid,
  });

  if (res.statusCode == 200) {

    //If successfully sent

    var data = converty.json.decode(res.body); //Decode to array

    if (data["error"]) {

      //If failed to query

      List error = [
        {
          "error": "true",
          "restaurantcategories": "[]"
        } //Return no menu categories
      ];
    }

    return error;
  } else {

    List listdata = [data];

    return listdata;
  }
} else {
```

```

List<error> = [
    {"error": "true", "restaurantcategories": "[]"}
];
return error;
}

} catch (e) {
    List<Map<String, String>> error = [
        {"error": "true", "restaurantcategories": "[]"}
    ];
    return error;
}
}

Future<String> favouriteRestaurant(restaurantID, action) async {
    String phpurl =
        "https://alleat.cpur.net/query/favouriterestaurant.php"; //Action of
    favourite/unfavourite restaurant by id (Unused in this version)

    final prefs = await SharedPreferences.getInstance();

    final String? email = prefs.getString('email');

    try {
        var res = await http.post(Uri.parse(phpurl), body: {
            "action": action.toString(),

```

```
"profileemail": email.toString(),

"restaurantid": restaurantID,

});

if (res.statusCode == 200) {

    var data = converty.json.decode(res.body); //Decode to array

    if (data["error"]) {

        return "failed";

    } else {

        getFavourites();

        return "success";

    }

} else {

    return "failed";

}

} catch (e) {

    return "failed";

}

}

Future<List> getFavourites() async {

String phpurl =
```

```
"https://alleat.cpur.net/query/favouriterestaurantlist.php"; //Get
favourite restaurant list associated with email of profile (unused in this
version)

final prefs = await SharedPreferences.getInstance();

final String? email = prefs.getString('email');

try {

var res = await http.post(Uri.parse(phiurl), body: {

"profileemail": email.toString(),

});

if (res.statusCode == 200) {

var data = convert.json.decode(res.body); //Decode to array

if (data["error"]) {

List error = [

{"error": "true", "favouriterestaurants": "[]"

];

return error;

} else {

List listdata = [data];

return listdata;

}

} else {

List error = [

{"error": "true", "favouriterestaurants": "[]"

}
```

```

    ];

    return error;
}

} catch (e) {

    List<Map<String, String>> error = [
        {"error": "true", "favouriterestaurants": "[]"}
    ];

    return error;
}

}

Future<List> getMenuItems(categoryid) async {

    String phpurl =

        "https://alleat.cpur.net/query/restaurantmenuitems.php"; //Get list of
    menu items, with the inforamtion associated with it. Grabs using the category id
    it falls under

    try {

        var res = await http.post(Uri.parse(phpurl), body: {
            "categoryid": categoryid,
        });

        if (res.statusCode == 200) {

            //If fails to send data

            var data = convert.json.decode(res.body); //Decode to array
        }
    }
}

```

```
if (data["error"]) {

    //If fails the query

    List error = [
        {"error": "true", "menuitems": "[]"} //Send nothing

    ];

    return error;

} else {

    List listdata = [
        data
    ]; //Send back the list of items associated with the category

    return listdata;

}

} else {

    List error = [
        {"error": "true", "menuitems": "[]"}
    ];

    return error;

}

} catch (e) {

    List error = [
        {"error": "true", "menuitems": "[]"}
    ];
}
```

```
        return error;
    }
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: SingleChildScrollView(
            //Make page scrollable
            child:
                Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
                    Stack(children: [
                        Container(
                            width: MediaQuery.of(context).size.width,
                            height: 240,
                            decoration: BoxDecoration(
                                image: DecorationImage(
                                    fit: BoxFit.fitWidth,
                                    image: NetworkImage(widget.resbanner),
                                ),
                            ),
                        ),
                    ],
                )));
}
```

```
Align(  
  alignment: Alignment.bottomCenter,  
  child: Container(  
    margin: const EdgeInsets.only(top: 215),  
    width: double.infinity,  
    height: 30,  
    decoration: BoxDecoration(  
      color: Theme.of(context).backgroundColor,  
      borderRadius:  
        const BorderRadius.vertical(top: Radius.circular(20)),  
    )),  
  Align(  
    alignment: Alignment.bottomCenter,  
    child: Container(  
      margin: const EdgeInsets.only(top: 180),  
      width: 80,  
      height: 80,  
      decoration: BoxDecoration(  
        shape: BoxShape.circle,  
        color: Theme.of(context).backgroundColor,  
      )),  
  Align(
```

```
        alignment: Alignment.bottomCenter,  
  
        child: Container(  
  
            margin: const EdgeInsets.only(top: 185),  
  
            width: 70,  
  
            height: 70,  
  
            decoration: BoxDecoration(  
  
                image: DecorationImage(  
  
                    fit: BoxFit.cover,  
  
                    image: NetworkImage(widget.reslogo.toString()),  
  
                ),  
  
                shape: BoxShape.circle,  
  
                color: Theme.of(context).colorScheme.onSurface,  
  
            )),  
  
        SafeArea(  
  
            child: InkWell(  
  
                onTap: () => Navigator.of(context).pop(),  
  
                child: Container(  
  
                    margin: const EdgeInsets.only(top: 20, left: 20),  
  
                    width: 50,  
  
                    height: 50,  
  
                    decoration: BoxDecoration(  
  
                        shape: BoxShape.circle,
```

```
        color: Theme.of(context).colorScheme.onSurface,  
    ),  
  
    child: Icon(  
  
        Icons.chevron_left,  
  
        color: Theme.of(context).colorScheme.onBackground,  
  
        size: 35,  
    ),  
  
)),  
  
Align(  
  
    alignment: Alignment.topRight,  
  
    child: SafeArea(  
  
        child: InkWell(  
  
            child: Container(  
  
                margin: const EdgeInsets.only(top: 20, right: 80),  
  
                width: 45,  
  
                height: 45,  
  
                decoration: BoxDecoration(  
  
                    shape: BoxShape.circle,  
  
                    color: Theme.of(context).colorScheme.onSurface.withOpacity(0.8),  
                ),  
  
                child: Icon(  
  
                    Icons.info_outline,  
    
```

```
        color: Theme.of(context).colorScheme.onBackground,  
  
        size: 30,  
  
    ),  
  
))),  
  
FutureBuilder<List>(  
  
    //Get favourites list from future  
  
    future: getFavourites(),  
  
    builder: ((context, snapshot) {  
  
        if (!snapshot.hasData) {  
  
            //While no data received, show loading bar  
  
            return SafeArea(  
  
                child: Align(  
  
                    alignment: Alignment.topRight,  
  
                    child: Container(  
  
                        margin: const EdgeInsets.only(top: 20, right: 20),  
  
                        width: 45,  
  
                        height: 45,  
  
                        decoration: BoxDecoration(  
  
                            shape: BoxShape.circle,  
  
                            color: Theme.of(context)  
  
                                .colorScheme  
  
                                .onSurface
```

```
        .withOpacity(0.8),  
    ),  
    child: Icon(  
        Icons.cached,  
        color: Theme.of(context).colorScheme.onBackground,  
    ))));  
  
} else if (snapshot.hasError) {  
  
    return SafeArea(  
        child: Align(  
            alignment: Alignment.topRight,  
            child: Container(  
                margin: const EdgeInsets.only(top: 20, right: 20),  
                width: 45,  
                height: 45,  
                decoration: BoxDecoration(  
                    shape: BoxShape.circle,  
                    color: Theme.of(context)  
                        .colorScheme  
                        .onSurface  
                        .withOpacity(0.8),  
                ),  
                child: Icon(  
                    Icons.error,
```

```
        Icons.error,  
  
        color: Theme.of(context).colorScheme.error,  
    ))));  
  
} else {  
  
    //If data received  
  
    List restaurantFavourites = snapshot.data ?? [];  
  
    if (restaurantFavourites[0]["error"] == true) {  
  
        return SafeArea(  
  
            child: Align(  
  
                alignment: Alignment.topRight,  
  
                child: Container(  
  
                    margin: const EdgeInsets.only(top: 20, right: 20),  
  
                    width: 45,  
  
                    height: 45,  
  
                    decoration: BoxDecoration(  
  
                        shape: BoxShape.circle,  
  
                        color: Theme.of(context)  
  
.colorScheme  
.onSurface  
.withOpacity(0.8),  
  
        ),  
  
        child: Icon(  

```

```
        Icons.error,  
  
        color: Theme.of(context).colorScheme.error,  
  
    ))));  
  
} else {  
  
    return SafeArea(  
  
        child: Align(  
  
            alignment: Alignment.topRight,  
  
            child: InkWell(  
  
                onTap: () {  
  
                    if (restaurantFavourites[0]["restaurantids"]  
  
                        .contains(widget.resid)) {  
  
                        favouriteRestaurant(  
  
                            widget.resid.toString(), "unfavourite");  
  
                        setState(() {  
  
                            getFavourites();  
  
                        });  
  
                    } else {  
  
                        favouriteRestaurant(  
  
                            widget.resid.toString(), "favourite");  
  
                        setState(() {  
  
                            getFavourites();  
  
                        });  
  
                    }  
                }  
            );  
        );  
    );  
}
```

```
        },  
  
        child: Container(  
  
          margin:  
            const EdgeInsets.only(top: 20, right: 20),  
  
          width: 45,  
  
          height: 45,  
  
          decoration: BoxDecoration(  
  
            shape: BoxShape.circle,  
  
            color: Theme.of(context)  
  
.colorScheme  
  
.onSurface  
  
.withOpacity(0.8),  
  
        ),  
  
        child: restaurantFavourites[0]  
  
          ["restaurantids"]  
  
.contains(widget.resid)  
  
? Icon(Icons.favorite,  
  
size: 30, // ? = favourited  
  
color: Theme.of(context)  
  
.colorScheme  
  
.secondary)
```

```
: Icon( // : = unfavourited Icons.favorite_border_outlined, size: 30, color: Theme.of(context).colorScheme.onBackground)))); } } }))) ], Padding( padding: const EdgeInsets.symmetric(vertical: 20, horizontal: 30), child: Column(children: [ Row( mainAxisAlignment: MainAxisAlignment.spaceBetween, children: [ Flexible( child: Container( padding: const EdgeInsets.only(right: 13.0), child: Text( widget.resname,
```

```
        overflow: TextOverflow.ellipsis,  
        style: Theme.of(context).textTheme.headline3,  
    )),  
  
Row(  
  
    children: [  
  
        Icon(  
  
            Icons.star,  
  
            size: 20,  
  
            color: Theme.of(context).primaryColor,  
        ),  
  
        const SizedBox(  
  
            width: 5,  
        ),  
  
        Text(  
  
            "N/A",  
  
            style: Theme.of(context).textTheme.bodyText1,  
        )  
    ],  
),  
],  
,  
const SizedBox(height: 10),
```

```
Align(
    alignment: Alignment.topLeft,
    child: Wrap(
        children: [
            Text("${widget.resdistance} km away",
                style: Theme.of(context).textTheme.bodyText1),
            const SizedBox(
                width: 5,
            ),
            Text(
                ".",
                style: Theme.of(context).textTheme.bodyText1,
            ),
            const SizedBox(
                width: 5,
            ),
            Text(
                (widget.resdelivery == "0.00")
                    ? "Free Delivery"
                    : "£${widget.resdelivery} Delivery",
                style: Theme.of(context).textTheme.bodyText1),
            Text(

```

```

        ".",
        style: Theme.of(context).textTheme.bodyText1,
    ),
    const SizedBox(
        width: 5,
    ),
    Text(
        (widget.resordermin == "0")
            ? "No Minimum Order"
            : "£${widget.resordermin} Order Minimum",
        style: Theme.of(context).textTheme.bodyText1,
    ],
)),
]),
FutureBuilder<List>(
    //Checks for updates in the restaurant menu category
    future: getMenuCategories(),
    builder: ((context, snapshot) {
        if (!snapshot.hasData) {
            //If no data has been received, show loading bar
            return LinearProgressIndicator(
                color: Theme.of(context).primaryColor,

```

```
    backgroundColor: const Color.fromARGB(0, 235, 224, 255));  
  
}  
  
if (snapshot.hasError) {  
  
    //If there is an error, grabbing data, show error  
  
    return const Text("An Error occured. Please try again");  
  
} else {  
  
    //If the data has been received  
  
    List restaurantcategories = snapshot.data ??  
[]; //Categories data associated with restaurantcategories  
  
    if (restaurantcategories[0]["error"] == true) {  
  
        return Text(  
  
            "Failed to get restaurant categories",  
  
            style: Theme.of(context).textTheme.headline6,  
  
        );  
  
    } else {  
  
        return ListView.builder(  
  
            //For each category  
  
            physics:  
  
                const NeverScrollableScrollPhysics(), //Disable  
scrolling. Scroll with whole page  
  
            scrollDirection: Axis.vertical,  
  
            shrinkWrap: true,  
  
            itemCount: restaurantcategories[0]["restaurantcategories"]
```

```
.length, //For each restaurant

itemBuilder: (context, index) {

  return LayoutBuilder(builder: (context, constraints) {

    if (restaurantcategories[0]["restaurantcategories"]

        .isEmpty) {

      //If there is no categories, display menu unavailable

      return const Text("Menu unavailable");

    } else {

      //If there are categories

      return Column(children: [

        //Add text of category name

        Padding(

          padding: const EdgeInsets.only(

            left: 30, top: 50, right: 10, bottom: 10),

          child: Text(

            restaurantcategories[0]

              ["restaurantcategories"][index][0],

            style: Theme.of(context).textTheme.headline3),

        ),

        FutureBuilder<List>(

          //For each category, use a future to get the list

          of items

            future: getMenuItems(restaurantcategories[0]
```

```
        ["restaurantcategories"][index][1]),

    builder: (context, snapshot) {

        if (!snapshot.hasData) {

            //While no data received, show loading bar

            return LinearProgressIndicator(
                color: Theme.of(context).primaryColor,
                backgroundColor: const Color.fromARGB(
                    0, 235, 224, 255));

        }

        if (snapshot.hasError) {

            //If there is an error, show failed to get
            items

            return const Text("Failed to get items");

        } else {

            //List of items for category

            List restaurantitems = snapshot.data ??
            [];

            //Get data from Future

            if (restaurantitems[0]["menuitems"]

                .isEmpty) {

                return Container(
                    width: double.infinity,
                    padding: const EdgeInsets.symmetric(
```

```
        vertical: 30, horizontal: 20),  
  
        decoration: BoxDecoration(  
  
          borderRadius:  
  
            const BorderRadius.all(  
  
              Radius.circular(10)),  
  
          color: Theme.of(context)  
  
            .colorScheme  
  
            .onSurface),  
  
        margin: const EdgeInsets.symmetric(  
  
          vertical: 5, horizontal: 30),  
  
        child: Text(  
  
          "Oh no! There are no items found under  
this category.",  
  
          textAlign: TextAlign.center,  
  
          style: Theme.of(context)  
  
            .textTheme  
  
            .headline6,  
  
        ),  
  
      );  
  
    } else {  
  
      return ListView.builder(  
  
        physics:
```

```
        const  
        NeverScrollableScrollPhysics(),  
  
        scrollDirection: Axis.vertical,  
  
        shrinkWrap: true,  
  
        itemCount: (restaurantitems[  
  
            0] //For each item, create  
a container  
  
            ["menuitems"]  
  
.length), //For each restaurant  
  
        itemBuilder: (context, index) {  
  
            return LayoutBuilder(builder:  
  
(context, constraints) {  
  
                // If there is items to display  
  
                return InkWell(  
  
                    //Clickable items  
  
                    onTap: () {  
  
                        Navigator.push(  
  
                            context,  
  
                            MaterialPageRoute(  
  
                                builder: (context) =>  
RestaurantItemCustomisePage(  
  
                                reslogo: widget  
  
.reslogo,
```

```
        itemid:  
restaurantitems[0]["menuitems"]  
  
        [index][0],  
  
        foodcategory:  
restaurantitems[0]  
  
        ["menuitems"]  
  
        [index][1],  
  
        subfoodcategory:  
restaurantitems[0]  
  
        ["menuitems"]  
  
        [index][2],  
  
        itemname:  
restaurantitems[0]  
  
        ["menuitems"]  
  
        [index][3],  
  
        description:  
restaurantitems[0]  
  
        ["menuitems"][index][4],  
  
        price:  
restaurantitems[0]["menuitems"][index][5],  
  
        itemimage:  
restaurantitems[0]["menuitems"][index][6]]));  
  
    },
```

```
        child: Container(  
  
            //Create clickable  
            container  
  
            width: double.infinity,  
  
            margin:  
  
            const EdgeInsets.only(  
  
                left: 10,  
  
                right: 10,  
  
                top: 10,  
  
                bottom: 10),  
  
            child: Row(  
  
                //Create a row containing  
                item image, name and price  
  
                children: [  
  
                    Container(  
  
                        width: 80,  
  
                        height: 80,  
  
                        decoration:  
                        BoxDecoration(  
  
                            borderRadius:  
  
                            const  
BorderRadius  
  
.all(
```

```
Radius.circular(  
    5)),  
  
        image:  
DecorationImage(  
    fit: BoxFit  
    .cover,  
  
        image:  
NetworkImage(restaurantitems[0]["menuitems"]  
[  
  
index][6]  
  
.toString()))),  
),  
  
const SizedBox(  
    width: 15),  
  
Expanded(  
    child: Column(  
        mainAxisAlignment:  
  
MainAxisAlignment  
        .start,  
  
        crossAxisAlignment:
```

```
CrossAxisAlignment  
    .start,  
  
    children: [  
        Text(  
  
restaurantitems[0]  
        [  
  
"menuitems"]  
        [  
  
index][3]  
  
.toString(),  
  
        textAlign:  
            TextAlign  
                .start,  
  
            style:  
Theme.of(  
  
context)  
  
.textTheme  
    .headline6!  
  
.copyWith(  
    .textTheme  
        .headline6!  
        .copyWith(  
            .textTheme  
                .headline6!
```

```
        color:  
Theme.of(context)  
  
.colorScheme  
  
.onBackground),  
),  
Text(  
  
restaurantitems[0]  
[  
  
"menuitems"]  
[  
  
index][4]  
  
.toString(),  
maxLines: 3,  
style:  
Theme.of(  
  
context)  
.textTheme  
.bodyText2!  
.copyWith(  
)
```

```
        color:  
Theme.of(context)  
  
.textTheme  
  
.headline5!  
  
.color,  
  
fontSize:  
  
14),  
  
        overflow:  
  
TextOverflow  
  
.clip,  
  
)  
  
],  
  
),  
  
),  

```

```
"£",  
    style: Theme.of(  
        context)  
            .textTheme  
            .headline6!  
            .copyWith(  
                color:  
Theme.of(context)  
  
.primaryColor),  
    ),  
    const SizedBox(  
        width: 2),  
    Text(  
  
restaurantitems[0]["menuitems"]  
[  
  
index]  
[5]  
  
.toString(),  
    style:  
Theme.of(  
    
```

```
context)

    .textTheme

    .headline6!

    .copyWith(
        color:

Theme.of(context)

.colorScheme

.onBackground))

    ],
),
const SizedBox(
    width: 10),
],
));
)));
});
}
}
})
]);
}
}
});
```

```
        });

    });

}

}());

])));

}

}
```

filtersort.dart

```
import 'package:alleat/screens/navigationscreens/browse.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:flutter/material.dart';

import 'dart:convert';

class FilterSort extends StatefulWidget {

const FilterSort({super.key});

}

@Override

State<FilterSort> createState() => _FilterSortState();

}
```

```
class _FilterSortState extends State<FilterSort> {

    List sortOptions = [
        //Sort options [Visual button text, icon, option saved in customiseSelected]
        ["Recommended", Icons.assistant_outlined, "default"],
        ["Top Rated", Icons.star_border_outlined, "top"],
        ["Popular", Icons.local_fire_department_outlined, "hot"],
        ["Distance", Icons.straighten_outlined, "distance"]
    ];

    List priceRangeOptions = [
        //Price range options [Visual Text, Option Saved in customiseSelected]
        ["£", 1],
        ["££", 2],
        ["£££", 3],
        ["££££", 4]
    ];

    bool isChecked = false; //Favourites selector

    double _currentMaxDeliveryFeeValue = 4;

    double _currentMinOrderPriceValue = 40;

    String? encodedCustomiseSelected;

    Map customiseSelected = {
        //initial filters and sort method
    };
}
```

```
"sort": "distance",
" favourite": false,
"price": [1, 2, 3, 4],
"maxDelivery": 4.0,
"minOrder": 40.0

};

@Override
void initState() {
    //On initiation, load filters from shared preferences
    super.initState();
    _loadFilter();
}

_loadFilter() async {
    SharedPreferences prefs =
        await SharedPreferences.getInstance(); //get from shared preferences
    setState(() {
        encodedCustomiseSelected = prefs.getString('filtersort');
    });
    if (encodedCustomiseSelected != null) {
        //If the filter and sort methods have been previously saved
    }
}
```

```

customiseSelected = json.decode(encodedCustomiseSelected

    .toString()); //Decode customise and split and replace default values

_currentMaxDeliveryFeeValue = customiseSelected["maxDelivery"];

_currentMinOrderPriceValue = customiseSelected["minOrder"];

isChecked = customiseSelected["favourite"];

} else {

//If the filter and sort methods have not been previously saved

customiseSelected = {

    //Set default values

    "sort": "default",

    "favourite": false,

    "price": [1, 2, 3, 4],

    "maxDelivery": 4.0,

    "minOrder": 40.0

};

_currentMaxDeliveryFeeValue = 4; //Change sliders to default values

_currentMinOrderPriceValue = 40;

}

}

@Override

Widget build(BuildContext context) {

```

```
return Scaffold(  
    body: SingleChildScrollView(  
        child:  
            Column(crossAxisAlignment: CrossAxisAlignment.start, children: [  
                Padding(  
                    padding: const EdgeInsets.only(left: 40, right: 20, top: 50),  
                    child: Row(  
                        //Row with the text sort and a button to clear filters and reset  
                        sort  
                        mainAxisAlignment: MainAxisAlignment.end,  
                        mainAxisSize: MainAxisSize.spaceBetween,  
                        children: [  
                            Text("Sort", style: Theme.of(context).textTheme.headline2),  
                            LayoutBuilder(builder: (context, constraints) {  
                                //If there have been any changes to the filters and sort  
                                methods, highlight clear all text  
                                if (customiseSelected["sort"] != "default" ||  
                                    customiseSelected["favourite"] != false ||  
                                    !customiseSelected["price"].contains(1) ||  
                                    !customiseSelected["price"].contains(2) ||  
                                    !customiseSelected["price"].contains(3) ||  
                                    !customiseSelected["price"].contains(4) ||  
                                    customiseSelected["maxDelivery"] != 4.0 ||
```

```
        customiseSelected["minOrder"] != 40.0) {

    return InkWell(
        onTap: () {
            //On tap, reset to defaults
            setState(() {
                customiseSelected = {
                    "sort": "default",
                    "favourite": false,
                    "price": [1, 2, 3, 4],
                    "maxDelivery": 4.0,
                    "minOrder": 40.0
                };
                _currentMaxDeliveryFeeValue = 4;
                _currentMinOrderPriceValue = 40;
                isChecked = false;
            });
        },
        child: Padding(
            padding: const EdgeInsets.symmetric(
                horizontal: 40, vertical: 20),
            child: Text(
                "Clear all",

```

```
        style: Theme.of(context)

            .textTheme

            .headline6!

            .copyWith(
                color:

                    Theme.of(context).colorScheme.error),

            )));

} else {

    return InkWell(
        onTap: () {
            setState(() {
                customiseSelected = {

                    "sort": "default",

                    "favourite": false,

                    "price": [1, 2, 3, 4],

                    "maxDelivery": 4.0,

                    "minOrder": 40.0

                };
                _currentMaxDeliveryFeeValue = 4;
                _currentMinOrderPriceValue = 40;
            });
        },
    },
}
```

```
        child: Padding(  
          padding: const EdgeInsets.symmetric(  
            horizontal: 40, vertical: 20),  
          child: Text(  
            "Clear all",  
            style: Theme.of(context)  
              .textTheme  
              .headline6!  
              .copyWith(  
                color: Theme.of(context)  
                  .colorScheme  
                  .error  
                  .withOpacity(0.1)),  
          )),  
        );  
      )  
    ]),  
  ),  
 Padding(  
  padding: const EdgeInsets.symmetric(horizontal: 30),  
  child:  
  Column(crossAxisAlignment: CrossAxisAlignment.start, children: [  
    const SizedBox(  

```

```
    height: 20,  
)  
  
ListView.builder(  
  
    //For each sort method  
  
    physics: const NeverScrollableScrollPhysics(),  
  
    scrollDirection: Axis.vertical,  
  
    shrinkWrap: true,  
  
    itemCount: sortOptions.length,  
  
    itemBuilder: (context, index) {  
  
        return Padding(  
  
            padding: const EdgeInsets.symmetric(vertical: 7),  
  
            child: ElevatedButton(  
  
                style: ElevatedButton.styleFrom(  
  
                    side: (customiseSelected["sort"] ==  
  
                        sortOptions[index][  
  
                            2]) //If the sort method is the same as  
the button, display with border  
  
                    ? BorderSide(  
  
                        color: Theme.of(context).primaryColor,  
  
                        width: 2)  
  
                    : BorderSide.none,  
  
                    backgroundColor:  
  
                        Theme.of(context).colorScheme.onSurface,
```

```
padding: const EdgeInsets.symmetric(  
    vertical: 15, horizontal: 25)),  
  
onPressed: () {  
  
    //On press, overwrite sort method with new sort  
method  
  
    setState(() {  
  
        customiseSelected["sort"] = sortOptions[index][2];  
  
    });  
  
},  
  
child: Row(  
  
    //Row with the icon and text for sort method grabbed  
from the list  
  
    children: [  
  
        Icon(  
  
            sortOptions[index][1],  
  
            color: customiseSelected["sort"] ==  
  
                sortOptions[index][2]  
  
            ? Theme.of(context)  
  
                .textTheme  
  
                .headline1  
  
                ?.color  
  
            : Theme.of(context)  
  
                .textTheme
```

```
        .headline5  
  
        ?.color,  
  
    ),  
  
    const SizedBox(width: 20),  
  
    Flexible(  
  
        child: Text(  
  
            sortOptions[index][0],  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline5!  
  
            .copyWith(  
  
                color: customiseSelected["sort"] ==  
  
                    sortOptions[index][2]  
  
                ? Theme.of(context)  
  
                    .textTheme  
  
                    .headline1  
  
                ?.color  
  
                : Theme.of(context)  
  
                    .textTheme  
  
                    .headline5  
  
                ?.color,  
  
        ),
```



```
contentPadding: const EdgeInsets.all(0),  
  
value: isChecked,  
  
controlAffinity: ListTileControlAffinity.leading,  
  
onChanged: (newValue) {  
  
  setState(() {  
  
    isChecked = !isChecked;  
  
});  
  
if (isChecked) {  
  
  customiseSelected["favourite"] = true;  
  
} else {  
  
  customiseSelected["favourite"] = false;  
  
}  
  
},  
  
)),  
  
],  
,  
  
const SizedBox(  
  
height: 20,  
,  
  
Text("Price Range", style: Theme.of(context).textTheme.headline3),  
  
const SizedBox(  
  
height: 20,
```

```
),

Container(
    //Container with a list of price brackets

    padding: const EdgeInsets.symmetric(horizontal: 10),

    color: Theme.of(context).colorScheme.onSurface,

    height: 70,

    child: ListView.builder(
        scrollDirection: Axis.horizontal,
        shrinkWrap: true,
        itemCount: priceRangeOptions.length, //For each restaurant
        itemBuilder: (context, index) {
            //For each price bracket
            return Padding(
                padding: const EdgeInsets.symmetric(
                    horizontal: 5, vertical: 10),
                child: SizedBox(
                    width: 70,
                    child: ElevatedButton(
                        style: ElevatedButton.styleFrom(
                            side: customiseSelected["price"].contains(
                                priceRangeOptions[index][
                                    1])) //If customiseSelected has
the price bracket, display with border
```

```

? BorderSide(
    color: Theme.of(context).primaryColor,
    width: 2)

: BorderSide.none,
backgroundColor: Theme.of(context).colorScheme.onSurface,
padding: const EdgeInsets.symmetric(
    vertical: 15, horizontal: 10)),

onPressed: () {
    // If customiseSelected has the price bracket,
remove otherwise add it.

    if (customiseSelected["price"]
        .contains(priceRangeOptions[index][1])) {
        setState(() {
            customiseSelected["price"]
                .remove(priceRangeOptions[index][1]);
        });
    } else {
        setState(() {
            customiseSelected["price"]
                .add(priceRangeOptions[index][1]);
        });
    }
}

```

```

        }

    },
    child: Text(
        //Display text with price bracket text (£, ££, £££, ££££)
        priceRangeOptions[index][0],
        style: Theme.of(context)

            .textTheme
            .headline6!
            .copyWith(
                color: customiseSelected["price"]
                .contains(
                    priceRangeOptions[index][1])
                ? Theme.of(context).primaryColor
                : Theme.of(context)

                    .textTheme
                    .headline6
                    ?.color,
            ),
    ),
));
}
),
const SizedBox(

```

```
        height: 40,  
    ),  
  
    Row(mainAxisAlignment: MainAxisAlignment.spaceBetween, children: [  
  
        //Row with title and the current soldier status. If price is 0,  
display as free  
  
        Text("Max Delivery Fee",  
            style: Theme.of(context).textTheme.headline3),  
  
        LayoutBuilder(builder: (context, constraints) {  
  
            if (_currentMaxDeliveryFeeValue == 0) {  
  
                return const Text("FREE");  
  
            } else {  
  
                return Text("£${_currentMaxDeliveryFeeValue}0");  
  
            }  
  
        })  
    ]),  
  
    const SizedBox(  
  
        height: 20,  
    ),  
  
    Slider(  
  
        value: _currentMaxDeliveryFeeValue,  
  
        max: 4,  
  
        min: 0,  
  
        thumbColor: Theme.of(context).primaryColor,  
    )
```

```
activeColor: Theme.of(context).primaryColor,  
  
inactiveColor:  
  
    Theme.of(context).colorScheme.onBackground.withOpacity(0.2),  
  
divisions: 4,  
  
onChanged: (double value) {  
  
    //On change, update filter and the slider with new value  
  
    setState(() {  
  
        _currentMaxDeliveryFeeValue = value;  
  
    });  
  
    customiseSelected["maxDelivery"] = _currentMaxDeliveryFeeValue;  
  
},  
  
(  
  
const SizedBox(  
  
height: 40,  
  
),  
  
Row(mainAxisAlignment: MainAxisAlignment.spaceBetween, children: [  
  
    //Row with title and the current soldier status. If price is 0,  
display as None  
  
    Text("Min Order Price",  
  
style: Theme.of(context).textTheme.headline3),  
  
LayoutBuilder(builder: (context, constraints) {  
  
if (_currentMinOrderPriceValue == 0) {  
  
return const Text("None");  
}
```

```

    } else {

        return Text("£${_currentMinOrderPriceValue}0");

    }

})

]),

const SizedBox(
    height: 20,
),

Slider(
    value: _currentMinOrderPriceValue,
    max: 40,
    min: 0,
    thumbColor: Theme.of(context).primaryColor,
    activeColor: Theme.of(context).primaryColor,
    inactiveColor:
        Theme.of(context).colorScheme.onBackground.withOpacity(0.2),
    divisions: 4,
    onChanged: (double value) {

        setState(() {

            _currentMinOrderPriceValue = value;
        });

        customiseSelected["minOrder"] = _currentMinOrderPriceValue;
    }
);

```

```
        },  
        ),  
        const SizedBox(  
          height: 40,  
        ),  
        ElevatedButton(  
          //Save button encodes the filter and sort dictionary and replaces  
          //the sharedpreference with the new value then closes the filter screen  
          style: ElevatedButton.styleFrom(  
            minimumSize: const Size.fromHeight(50)),  
          onPressed: () async {  
            final prefs = await SharedPreferences.getInstance();  
            String encodedCustomiseSelected =  
              json.encode(customiseSelected);  
            await prefs.setString('filtersort', encodedCustomiseSelected);  
            setState(() {  
              Navigator.of(context).push(  
                MaterialPageRoute(builder: (_) => const BrowsePage()),  
              );  
            });  
          },  
          child: const Text("Save"))  
        ]),  
      ),
```

```
        const SizedBox(  
            height: 40,  
        ),  
    ]));  
}  
}
```

locationselection.dart

```
import 'dart:async';

import 'package:alleat/widgets/genericlocading.dart';

import 'package:alleat/widgets/navigationbar.dart';

import 'package:flutter/services.dart';

import 'package:google_maps_flutter/google_maps_flutter.dart';

import 'package:flutter/material.dart';

import 'package:map_picker/map_picker.dart';

import 'package:geocoding/geocoding.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:geolocator/geolocator.dart';

class SelectLocation extends StatefulWidget {

  const SelectLocation({super.key});
```

```

@Override
State<SelectLocation> createState() => _SelectLocationState();

}

class _SelectLocationState extends State<SelectLocation> {

final _controller = Completer<GoogleMapController>();
static TextEditingController addresslineone = TextEditingController();
static TextEditingController addresslinetwo = TextEditingController();
static TextEditingController postcode = TextEditingController();
static TextEditingController city = TextEditingController();
MapPickerController mapPickerController = MapPickerController();

late var cameraPosition =
const CameraPosition(target: LatLng(0, 0), zoom: 20);

Future<List> getSavedPosition() async {
//Try to get saved location and current location

final prefs = await SharedPreferences
.getInstance(); // Get saved location from shared preferences

final double? savedLocationLat = prefs.getDouble('locationLatitude');

final double? savedLocationLng = prefs.getDouble('locationLongitude');

```

```

final List<String>? savedLocationText =
    prefs.getStringList('locationPlacemark');

}

if (savedLocationLat != null &&
    savedLocationLng != null &&
    savedLocationText != null) {

    //If not null returned, return the value

    addresslineone = TextEditingController(text: savedLocationText[0]);
    addresslinetwo = TextEditingController(text: savedLocationText[1]);
    city = TextEditingController(text: savedLocationText[2]);
    postcode = TextEditingController(text: savedLocationText[3]);

    return [savedLocationLat, savedLocationLng];
} else {
    //If null returned from saved location, get the approximate location
    return getCurrentLocation();
}
}

Future<bool> saveLocation() async {
    try {
        final prefs = await SharedPreferences.getInstance();
        await prefs.setDouble('locationLatitude', cameraPosition.target.latitude);
    }
}

```

```
    await prefs.setDouble(
        'locationLongitude', cameraPosition.target.longitude);

    await prefs.setStringList('locationPlacemark', <String>[
        addresslineone.text,
        addresslinetwo.text,
        city.text,
        postcode.text
    ]);

    return true;
} catch (e) {
    return false;
}
}

Future<List> getCurrentLocation() async {
    bool serviceEnabled;
    LocationPermission permission;

    // Test if location services are enabled.
    serviceEnabled = await Geolocator.isLocationServiceEnabled();

    if (!serviceEnabled) {
```

```

    await getPlacemark([false, 51.509865, -0.118092]);

    return [51.509865, -0.118092]; //London lat long

}

// Check if the location is denied

permission = await Geolocator.checkPermission();

if (permission == LocationPermission.denied) {

    permission = await Geolocator.requestPermission();

    if (permission == LocationPermission.denied) {

        await getPlacemark([false, 51.509865, -0.118092]);

        return [51.509865, -0.118092]; //London lat long

    }

}

if (permission == LocationPermission.deniedForever) {

    // Permissions are denied forever, handle appropriately.

    await getPlacemark([false, 51.509865, -0.118092]);

    return [51.509865, -0.118092]; //London lat long

}

// Access the current possition of the device

Position locationDevice = await Geolocator.getCurrentPosition();

```

```
await getPlacemark(  
    [false, locationDevice.latitude, locationDevice.longitude]);  
  
return [locationDevice.latitude, locationDevice.longitude];  
}  
  
  
Future<void> getPlacemark(placemarkLocation) async {  
  
try {  
  
switch (placemarkLocation[0]) {  
  
case true:  
  
List<Placemark> placemarks = await placemarkFromCoordinates(  
  
    cameraPosition.target.latitude,  
  
    cameraPosition.target.longitude,  
);  
  
addresslineone = TextEditingController(text: placemarks.first.name);  
  
addresslinetwo = TextEditingController(text: placemarks.first.street);  
  
postcode = TextEditingController(text: placemarks.first.postalCode);  
  
if ((placemarks.first.locality) == "") {  
  
city = TextEditingController(  
  
    text: placemarks.first.subAdministrativeArea);  
  
} else {  
  
city = TextEditingController(text: placemarks.first.locality);  
}  
}
```

```
        break;

    case false:

        List<Placemark> placemarks = await placemarkFromCoordinates(
            placemarkLocation[1], placemarkLocation[2]);

        addresslineone = TextEditingController(text: placemarks.first.name);

        addresslinetwo = TextEditingController(text: placemarks.first.street);

        postcode = TextEditingController(text: placemarks.first.postalCode);

        if ((placemarks.first.locality) == "") {

            city = TextEditingController(
                text: placemarks.first.subAdministrativeArea);

        } else {

            city = TextEditingController(text: placemarks.first.locality);

        }

        break;

    }

} catch (e) {

    textController.text = "";

}

}

var textController = TextEditingController();

@Override
```

```
Widget build(BuildContext context) {  
  
  return Scaffold(  
  
    body: FutureBuilder<List>(  
  
      future: getSavedPosition(),  
  
      builder: (context, snapshot) {  
  
        if (!snapshot.hasData) {  
  
          return const GenericLoading();  
  
        }  
  
        if (snapshot.hasData) {  
  
          var savedPosition = snapshot.data ?? [];  
  
  
  
          CameraPosition cameraPosition = CameraPosition(  
  
            target: LatLng(savedPosition[0], savedPosition[1]), zoom: 19);  
  
          return Stack(  
  
            alignment: Alignment.topCenter,  
  
            children: [  
  
              LayoutBuilder(  
  
                builder: (BuildContext context, BoxConstraints constraints) {  
  
                  return MapPicker(  
  
                    // pass icon widget  
  
                    iconWidget: const Icon(  
  
                      Icons.location_on,
```

```
        size: 65,  
  
    ),  
  
    //add map picker controller  
  
    mapPickerController: mapPickerController,  
  
    child: GoogleMap(  
  
        myLocationEnabled: false,  
  
        zoomControlsEnabled: false,  
  
        // hide location button  
  
        myLocationButtonEnabled: true,  
  
        mapType: MapType.normal,  
  
        rotateGesturesEnabled: false,  
  
        tiltGesturesEnabled: false,  
  
        // camera position  
  
        initialCameraPosition: cameraPosition,  
  
        onMapCreated: (GoogleMapController controller) {  
  
            _controller.complete(controller);  
  
        },  
  
        onCameraMoveStarted: () {  
  
            // notify map is moving  
  
            try {
```

```

        mapPickerController.mapMoving!();

    } catch (e) {

        Navigator.pop(context);

    }

    },
onCameraMove: (cameraPosition) {

    try {

        //If the pointer position is invalid, dont try to move
the map.

        this.cameraPosition = cameraPosition;

    } catch (e) {

        cameraPosition = cameraPosition;

    }

},
onCameraIdle: () async {

    // notify map stopped moving

    try {

        // if there is an error getting the placemark return
null

        mapPickerController.mapFinishedMoving!();

        //get address name from camera position

        getPlacemark([true]);

    } catch (e) {

```

```
        return;

    }

    },
));

}),
DraggableScrollableSheet(
    //Dragable bottom bar

    initialChildSize: 0.2,
    snap: true,
    minChildSize: 0.2,
    maxChildSize: 0.8,
    builder: ((context, scrollController) {

        return Container(
            //Setting bar attributes
            decoration: BoxDecoration(
                borderRadius: const BorderRadius.vertical(
                    top: Radius.circular(10)),
                color: Theme.of(context).backgroundColor,
                boxShadow: const [
                    BoxShadow(
                        spreadRadius: 2,
                        blurRadius: 10,
```

```
        color: Color.fromARGB(8, 0, 0, 0))

    ]),

    child: ListView.builder(
        controller: scrollController,
        itemCount: 1,
        itemBuilder: ((context, index) {
            return Column(
                children: [
                    Container(
                        width: 40,
                        height: 5,
                        decoration: BoxDecoration(
                            color: Colors.grey,
                            borderRadius:
                                BorderRadius.circular(5))),
                    Padding(
                        padding: const EdgeInsets.symmetric(
                            vertical: 30, horizontal: 20),
                        child: Row(
                            children: [
                                Expanded(
                                    child: InkWell(

```

```
//Save Location button

onTap: (() async {

    bool hasSavedLocation =
        await saveLocation();

//Try to save

    if (hasSavedLocation ==
        true) {

        setState(() {
            Navigator.of(context)
                .push(
                    MaterialPageRoute(
                        builder: (_) =>
                            const
Navigation(),
                    );
                );
            });
        } else {
            setState(() {
                //Display failed to
update
                ScaffoldMessenger.of(
                    context)
                        .showSnackBar(

```

```
const SnackBar(  
  content:  
  Text(  
    'Failed  
to update location.'));  
  
});  
}  
},  
child: Container(  
  decoration: BoxDecoration(  
    borderRadius:  
      BorderRadius  
        .circular(10),  
    color: Theme.of(context)  
      .primaryColor),  
  padding: const EdgeInsets  
    .symmetric(  
      vertical: 15,  
      horizontal: 30),  
  child: Text(  
    "Save Location",  
    textAlign:  
      TextAlign.center,
```

```
        style: Theme.of(context)

        .textTheme

        .headline6!

        .copyWith(
            color: Theme.of(
                context)
            .colorScheme
            .onSurface),
        ),
    ))),

const SizedBox(
    width: 20,
),
InkWell(
    //Current location button
    onTap: () async {
        // Get the current location of the
device
        List currentLocation =
        await getCurrentLocation();

        // Move the camera to the current
location
        final GoogleMapController
```

```
        controller =  
  
        await _controller.future;  
  
        controller.animateCamera(  
  
            CameraUpdate  
  
                .newCameraPosition(  
  
                    CameraPosition(  
  
                        target: LatLng(  
  
                            currentLocation[0],  
  
                            currentLocation[1]),  
  
                        zoom: 18,  
  
                    ),  
  
                )),  
  
        );  
  
    },  
  
    child: Container(  
  
        decoration: BoxDecoration(  
  
            borderRadius:  
  
                BorderRadius.circular(  
  
                    10),  
  
            color: Theme.of(context)  
  
                .backgroundColor),  
  
        padding:  
  
            const EdgeInsets.all(15),
```

```
        child: Icon(  
          Icons.my_location,  
          color: Theme.of(context)  
            .textTheme  
            .headline1!  
            .color,  
        )),  
      ),  
    ],  
  )),  
Container(  
  padding: const EdgeInsets.all(20),  
  alignment: Alignment.bottomLeft,  
  child: Column(  
    crossAxisAlignment:  
      CrossAxisAlignment.start,  
    children: [  
      Text("Edit your address",  
        style: Theme.of(context)  
          .textTheme  
          .headline3),  
      Padding(  
       
```

```
padding: const EdgeInsets.only(
    top: 5,
    right: 20,
    bottom: 30),
    child: Text(
        "Set your exact address on
the map and edit it below.",

        style: Theme.of(context)
            .textTheme
            .headline6),

    Padding(
        padding:
            const EdgeInsets.symmetric(
                vertical: 15),
        child: Column(
            crossAxisAlignment:
                CrossAxisAlignment.start,
            children: [
                Text(
                    "Address Line 1",
                    style: Theme.of(context)
                        .textTheme
```

```
        .headline6,  
    ),  
    TextFormField(  
        controller:  
            addresslineone,  
        style: Theme.of(context)  
            .textTheme  
            .bodyText2,  
        inputFormatters: [  
            //Only allows the input  
of letters a-z and A-Z and @,.-
```

FilteringTextInputFormatter

```
.allow(RegExp(  
    r'[a-zA-Z0-9,.-]  
    ]+|\s'))  
],  
)  
]),  
Padding(  
padding:  
const EdgeInsets.symmetric(  
    vertical: 15),  
child: Column(  

```

```
crossAxisAlignment:  
    CrossAxisAlignment.start  
    .start,  
  children: [  
    Text(  
      "Address Line 2",  
      style: Theme.of(context)  
        .textTheme  
        .headline6,  
    ),  
    TextFormField(  
      controller:  
        addresslinetwo,  
      style: Theme.of(context)  
        .textTheme  
        .bodyText2,  
      inputFormatters: [  
        //Only allows the input  
        of letters a-z and A-Z and ,.- and space  
        FilteringTextInputFormatter  
          .allow(RegExp(  
            r'[a-zA-Z0-9,.-\n]+|\s'))
```

```
        ],
      )
    ])),
Padding(
  padding:
    const EdgeInsets.symmetric(
      vertical: 15),
child: Column(
  mainAxisAlignment:
    MainAxisAlignment
      .start,
children: [
  Text(
    "City/County",
    style: Theme.of(context)
      .textTheme
      .headline6,
  ),
  TextFormField(
    controller: city,
    style: Theme.of(context)
      .textTheme
```

```
        .bodyText2,  
  
        inputFormatters: [  
  
            //Only allows the input  
            of letters a-z and A-Z and ,.- and space  
  
            FilteringTextInputFormatter  
  
                .allow(RegExp(  
  
                    r'[a-zA-Z0-9,.  
]|\s'))  
  
            ],  
  
        )  
  
    ])),  
  
    Padding(  
  
        padding:  
  
        const EdgeInsets.symmetric(  
  
            vertical: 15),  
  
        child: Column(  
  
            crossAxisAlignment:  
  
            CrossAxisAlignment  
  
            .start,  
  
            children: [  
  
                Text(  
  
                    "Postcode",  
  
                    style: Theme.of(context)
```

```
        .textTheme  
        .headline6,  
    ),  
    TextFormField(  
        controller: postcode,  
        style: Theme.of(context)  
            .textTheme  
            .bodyText2,  
        inputFormatters: [  
            //Only allows the input  
of letters a-z and A-Z and ,.- and space
```

FilteringTextInputFormatter

```
        .allow(RegExp(  
            r'[a-zA-Z0-9,.-]  
            ]+|\s'))  
    ],  
    )  
],),  
]))  
],  
);  
}));  
})),
```

```
        ],
    );
} else {
    return const GenericLoading();
}
},
));
}
}
```

setupverification.dart

```
import 'package:alleat/screens/profilesetup/welcomeScreen.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/widgets/genericlocading.dart';

import 'package:alleat/widgets/navbar.dart';

import 'package:flutter/material.dart';

class SetupWrapper extends StatefulWidget {

    const SetupWrapper({Key? key}) : super(key: key);
```

```
@override

State<SetupWrapper> createState() => _SetupWrapperState();

}

class _SetupWrapperState extends State<SetupWrapper> {

//Default setup not complete if something wrong happens

bool setup = false;

Future<bool> isSetupComplete() async {

List<Map> profileInfo = await SQLiteLocalProfiles

.getFirstProfile(); //Call Database for the first entry

if (profileInfo.isEmpty) {

//If the first entry is empty

//Then setup is not complete (pass to build)

return false;

} else {

//If the first entry exists

//Setup is complete (pass to build)

return true;

}

}

}
```

```
@override

Widget build(BuildContext context) => FutureBuilder<bool>(
    future: isSetupComplete(),
    builder: (context, snapshot) {
        if (!snapshot.hasData) {
            //While loading, go to loading page
            return const GenericLoading();
        }
        if (snapshot.hasError) {
            //If the app has an error, go to error page
            return const GenericLoading();
        } else {
            bool setup = snapshot.data ?? [] as bool; //Get data from Future
            if (setup == false) {
                //If setup is not complete
                return const ProfileSetupWelcome(); //Go to Setup page
            } else if (setup == true) {
                //If setup is complete
                return const Navigation(); //Go to main page
            } else {
                //If there is an error
                return const GenericLoading(); // Go to error page
            }
        }
    }
)
```

```
        }

    }

},
);

}
```

cartservice.dart

```
import 'package:alleat/services/localprofiles_service.dart';

import 'package:sqflite/sqflite.dart' as sql;

class SQLiteCartItems {

    // -----
    // Cart Items Table
    // -----



    static Future<void> createTableCartItems(sql.Database database) async {

        //Create cart table (Used to store items and their customised details)
        await database.execute("""
            CREATE TABLE cartItems(
                cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
                itemid INT,
                quantity INT,
                price REAL,
                total REAL
            )
        """);
    }
}
```

```
        customised TEXT,  
        quantity INT  
    )  
    """");  
}  
  
static Future<sql.Database> cartdb() async {  
  
    //If tables don't exist, create tables  
  
    return sql.openDatabase(  
        'alleatlocal.db',  
        version: 1,  
        onCreate: (sql.Database database, int version) async {  
  
            await SQLiteLocalProfiles.createTableProfile(database);  
  
            await createTableCartItems(database);  
        },  
    );  
}  
  
//-----  
  
// Edit Cart  
//-----
```

```
// Add to Cart

static Future<void> addToCart(
    int itemid, dynamic customised, int quantity) async {
    final db = await SQLiteCartItems.cartdb();
    db.insert("cartItems", {
        "itemid": itemid,
        "customised": customised,
        "quantity": quantity,
    });
}

}
```

dataencryption.dart

```
import 'package:encrypt/encrypt.dart' as encrypty;
import 'dart:convert' as converty;
import 'package:crypto/crypto.dart' as cryptoy;

class DataEncryption {
    static Future<String> encrpyt(plaintext) async {
```

```

dynamic encryptPassword;

String strkey =
    'ZC8cegCGG45d1IjIACEtrfypDXtkgJ1rA+4JABPncUE='; //Static key and iv

String striv = 'yvhstTh739b2yUw9NNlwrcKmHTtLTZNbjbiV3F/cSRzM=';

var iv = cryptojs.sha256 //Grab the substring of the key and iv

    .convert(converty.utf8.encode(striv))

    .toString()

    .substring(0, 16);

var key = cryptojs.sha256

    .convert(converty.utf8.encode(strkey))

    .toString()

    .substring(0, 16);

encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);

encrypty.Key keyObj = encrypty.Key.fromUtf8(key);

final encrypter =

    encrypty.Encrypter(encrypty.AES(keyObj, mode: encrypty.AESMode.cbc));

encryptPassword = encrypter.encrypt(plaintext,

    iv: ivObj); //Use the key and iv to encrypt the plaintext password

encryptPassword = encryptPassword.base64;

return encryptPassword;

}

}

```

localprofiles_service.dart

```
import 'dart:math';

import 'package:alleat/services/cart_service.dart';

import 'package:flutter/material.dart';

import 'package:sqflite/sqflite.dart' as sql;

class SQLiteLocalProfiles {

    // -----
    // Local Profiles Table
    // -----


    static Future<void> createTableProfile(sql.Database database) async {

        //Create localprofiles table (Used to store local logged in profiles)

        await database.execute("""CREATE TABLE localprofiles(
            id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
            profileid INT,
            firstname TEXT,
            lastname TEXT,
            email TEXT,
```

```
        password TEXT,  
        selected TEXT,  
        profilecolorred INT,  
        profilecolorgreen INT,  
        profilecolorblue INT,  
        createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP  
)  
""");  
}  
  
static Future<sql.Database> db() async {  
    //If tables don't exist, create tables  
    return sql.openDatabase(  
        'alleatlocal.db',  
        version: 1,  
        onCreate: (sql.Database database, int version) async {  
            await createTableProfile(database);  
            await SQLiteCartItems.createTableCartItems(database);  
        },  
    );  
}
```

```
//-----  
  
// Profile Creation  
  
//-----  
  
// Create new profile  
  
  
  
static Future<void> createProfile(dynamic profileid, String firstname,  
String lastname, String email, String password) async {  
  
    //Create profile using firstname, lastname, email and encrypted password  
  
    List colors = [  
  
        [46, 41, 78],  
  
        [239, 188, 213],  
  
        [190, 151, 198],  
  
        [134, 97, 193],  
  
        [77, 104, 184],  
  
        [112, 202, 209]  
  
    ];  
  
    Random random = Random();  
  
    List randomProfileColor = colors[random.nextInt(colors.length)];  
  
    final db = await SQLiteLocalProfiles.db();  
  
    db.insert("localprofiles", {  
  
        //Insert data into localprofiles table
```

```

    "profileid": profileid,
    "firstname": firstname,
    "lastname": lastname,
    "email": email,
    "password": password,
    "profilecolorred": randomProfileColor[0],
    "profilecolorgreen": randomProfileColor[1],
    "profilecolorblue": randomProfileColor[2],
  });
  selectProfile(email);
}

static Future<void> selectProfile(dynamic email) async {
  final db = await SQLiteLocalProfiles.db();
  db.rawUpdate(
    'UPDATE localprofiles SET selected = false'); //Set all profiles selected
  status to false
  db.rawUpdate(
    'UPDATE localprofiles SET selected = true WHERE email = "$email"');
}
//-----
// Profile Query

```

```
//-----  
  
// Get profiles  
  
static Future<List<Map<String, dynamic>>> getProfiles() async {  
  
    final db = await SQLiteLocalProfiles.db();  
  
    return db.query('localprofiles', orderBy: "id"); //get local profiles  
  
}  
  
// Get first profile in profiles table  
  
static Future<List<Map<String, dynamic>>> getFirstProfile() async {  
  
    final db = await SQLiteLocalProfiles.db();  
  
    return db.rawQuery('SELECT * FROM localprofiles ORDER BY id ASC LIMIT 1');  
  
}  
  
// Get currently selected profile color from table  
  
static Future<List<Map<String, Object?>>> getSelectedProfileColor() async {  
  
    final db = await SQLiteLocalProfiles.db();  
  
    return db.rawQuery(  
        'SELECT profilecolorred, profilecolorgreen, profilecolorblue FROM  
localprofiles ORDER BY id ASC LIMIT 1');
```

```
}

// Get all unselected profiles to be displayed

static Future<List<Map<String, Object?>>> getDisplayProfilesList() async {

    final db = await SQLiteLocalProfiles.db();

    return db.rawQuery(
        'SELECT id, profileid, firstname, lastname, profilecolorred,
profilecolorgreen, profilecolorblue FROM localprofiles ORDER BY selected DESC');

}

// Get profile info from ID

static Future<List<Map<String, Object?>>> getProfileFromID(id) async {

    final db = await SQLiteLocalProfiles.db();

    return db.rawQuery(
        'SELECT profileid, firstname, lastname, email FROM localprofiles WHERE id
= $id');

}

//-----
// Profile Modify
//-----
```

```
// Update an profile by id

static Future<void> updateProfile(int id, String firstname, String lastname,
    String email, String password) async {

    //Update profile using firstname, lastname, email and encrypted password

    final db = await SQLiteLocalProfiles.db();

    final data = {

        'firstname': firstname,
        'lastname': lastname,
        'email': email,
        'password': password,
    };

    await db.update('localprofiles', data, where: "id = ?", whereArgs: [id]);
}

// Delete profile

static Future<void> deleteProfile(int id) async {

    //Delete profile associated with the id inputted

    final db = await SQLiteLocalProfiles.db();

    try {

```

```

//Try to delete profile

await db.delete("localprofiles", where: "id = ?", whereArgs: [id]);

} catch (err) {

    // If it fails, prints the error to the console

    debugPrint("Something went wrong when deleting an item: $err");

}

}

}

```

queryserver.dart

```

import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

class QueryServer {

    static Future<Map> query(phurl, body) async { //Get url and data being sent

        try { //Try to send data to server

            var res = await http.post(Uri.parse(phurl), body: body);

            if (res.statusCode != 200) { //If the server rejects the data return the
error code

                return {"error": true, "message": "Error ${res.statusCode}"};

            } else { // If server successfully gets data

```

```
    var data = converty.json.decode(res.body); // Get the data sent back from  
the server and decode  
  
    if (data["error"]) { //If the data has an error  
  
        return {"error": true, "message": "Error 500"}; //Return error 500  
(server rejects data)  
  
    } else {  
  
        return {"error": false, "message": data};  
  
    }  
  
}  
  
}  
  
}  
  
}  
  
}
```

setselected.dart

```
import 'package:alleat/services/localprofiles_service.dart';  
  
import 'package:alleat/services/queryserver.dart';  
  
import 'package:shared_preferences/shared_preferences.dart';  
  
  
class SetSelected {
```

```

static Future<bool> selectProfile(
    serverid, firstname, lastname, email) async {

    dynamic favReataurantList = await QueryServer.query(
        //Get favourites from server
        'https://alleat.cpur.net/query/favouriterestaurantlist.php',
        {"profileemail": email});

    if (favReataurantList["Error"] == true) {
        //If getting favourites fails, return false
        return false;
    } else {
        try {
            //Try to change the shared preferences to the selected profile
            final prefs = await SharedPreferences.getInstance();
            await prefs.setString('serverprofileid', serverid.toString());
            await prefs.setString('firstname', firstname);
            await prefs.setString('lastname', lastname);
            await prefs.setString('email', email);
            await prefs.setString('favrestaurants',
                ((favReataurantList["message"])[ "restaurantids" ]).toString());
            SQLiteLocalProfiles.selectProfile(
                email); //Select profile in the database
        }
    }
}

```

```
        return true;

    } catch (e) {

        //If there is an error, return false

        return false;
    }
}
```

theme.dart

```
library globals;

import 'package:flutter/material.dart';

dynamic appthemepreference = 0;

class ThemeClass {

    static Color primaryLight = const Color(0xff5806FF);

    static Color secondaryLight = const Color(0xffAD84FF);

    static Color successLight = const Color(0xff07852A);
```

```
static Color errorLight = const Color(0xffAF0E17);

static Color textLight = const Color(0xff1A1A1A);

static Color bgLight = const Color(0xffFAFAFA);

static Color bottomAppBarLight = const Color(0xffffffff);

static Color primaryDark = const Color(0xffC1A3FF);

static Color secondaryDark = const Color(0xffAD84FF);

static Color successDark = const Color(0xff60F68A);

static Color errorDark = const Color(0xffF2555F);

static Color textDark = const Color(0xffEAEEAE);

static Color bgDark = const Color(0xff0C0C0C);

static Color bottomAppBarDark = const Color(0xff161616);

static Color mono900 = const Color(0xff0C0C0C);

static Color mono800 = const Color(0xff161616);

static Color mono700 = const Color(0xff292929);

static Color mono600 = const Color(0xff434343);

static Color mono500 = const Color(0xff676767);

static Color mono400 = const Color(0xff818181);

static Color mono300 = const Color(0xff979797);

static Color mono200 = const Color(0xffB6B6B6);

static Color mono100 = const Color(0xffD2D2D2);
```

```
static Color mono050 = const Color(0xffEAEAEA);

static Color mono000 = const Color(0xffffffff);

static Color primary900 = const Color(0xff060011);

static Color primary800 = const Color(0xff160041);

static Color primary700 = const Color(0xff2B0082);

static Color primary600 = const Color(0xff4100C4);

static Color primary500 = const Color(0xff5806FF);

static Color primary400 = const Color(0xff9866FF);

static Color primary300 = const Color(0xffAD84FF);

static Color primary200 = const Color(0ffc1A3FF);

static Color primary100 = const Color(0ffd6C2FF);

static Color primary050 = const Color(0xffEBE0FF);

static ThemeData lightTheme = ThemeData(

    //COLOUR

    primaryColor: primaryLight,

    colorScheme: ColorScheme.fromSwatch().copyWith(

        secondary: secondaryLight,

        error: errorLight,
```

```
tertiary: successLight,  
  
onBackground: textLight,  
  
onSurface: mono000,  
,  
  
// MAIN APP  
  
bottomNavigationBarTheme: BottomNavigationBarThemeData(  
  
    backgroundColor: mono000,  
  
    selectedItemColor: primaryLight,  
  
    unselectedItemColor: mono400),  
  
appBarTheme: AppBarTheme(backgroundColor: primaryLight),  
  
backgroundColor: bgLight,  
  
scaffoldBackgroundColor: bgLight,  
  
snackBarTheme: SnackBarThemeData(  
  
    contentTextStyle: TextStyle(  
  
        color: mono900,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w600,  
  
        fontSize: 14),  
  
    backgroundColor: mono100),
```

```
// TEXT

textTheme: TextTheme(
    headline1: TextStyle(
        color: textLight,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w800,
        fontSize: 32),
    headline2: TextStyle(
        color: mono900,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w700,
        fontSize: 28),
    headline3: TextStyle(
        color: mono800,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 21),
    headline4: TextStyle(
        color: mono800,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w500,
```

```
fontSize: 21),  
  
headline5: TextStyle(  
  
    color: mono500,  
  
    fontFamily: 'Satoshi',  
  
    fontWeight: FontWeight.w600,  
  
    fontSize: 20),  
  
headline6: TextStyle(  
  
    color: mono400,  
  
    fontFamily: 'Satoshi',  
  
    fontWeight: FontWeight.w600,  
  
    fontSize: 16),  
  
bodyText1: TextStyle(  
  
    color: textLight,  
  
    fontWeight: FontWeight.w600,  
  
    fontFamily: 'NotoSans',  
  
    fontSize: 14),  
  
bodyText2: TextStyle(  
  
    color: textLight,  
  
    fontWeight: FontWeight.w400,  
  
    fontFamily: 'NotoSans',  
  
    fontSize: 16)),
```

```
//BUTTONS

buttonTheme: ButtonThemeData(
    shape:
        RoundedRectangleBorder(borderRadius: BorderRadius.circular(10.0)),
    padding:
        const EdgeInsets.only(top: 18, bottom: 18, left: 30, right: 30)),
elevatedButtonTheme: ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
        foregroundColor: bgLight,
        backgroundColor: primaryLight,
        padding: const EdgeInsets.only(
            top: 15, bottom: 15, left: 30, right: 30),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(5), // <-- Radius
        ),
        textStyle: TextStyle(
            color: bgLight,
            fontFamily: 'Satoshi',
            fontWeight: FontWeight.w600,
            fontSize: 16))),
outlinedButtonTheme: OutlinedButtonThemeData(
```

```
style: OutlinedButton.styleFrom(  
    foregroundColor: primaryLight,  
    padding: const EdgeInsets.only(  
        top: 15, bottom: 15, left: 30, right: 30),  
    side: BorderSide(color: primaryLight, width: 3),  
    shape: RoundedRectangleBorder(  
        borderRadius: BorderRadius.circular(5), // <-- Radius  
    ),  
    textStyle: TextStyle(  
        color: primaryLight,  
        fontFamily: 'Satoshi',  
        fontWeight: FontWeight.w600,  
        fontSize: 16))),  
  
textButtonTheme: TextButtonThemeData(  
    style: TextButton.styleFrom(  
        foregroundColor: primaryLight,  
        padding: const EdgeInsets.only(  
            top: 18, bottom: 18, left: 30, right: 30),  
        textStyle: TextStyle(  
            color: primaryLight,  
            fontFamily: 'Satoshi',  
            fontWeight: FontWeight.w600,
```

```
        fontSize: 16))),

//FORM FIELD

inputDecorationTheme: InputDecorationTheme(
    filled: false,
    hintStyle: TextStyle(
        color: mono300,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16),
    contentPadding: const EdgeInsets.all(10),
    border: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono100)),
    focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: primaryLight)),
    disabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono050)),
    errorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorLight)),
    focusedErrorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorDark)),
```

```
enabledBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: mono100))));  
  
static ThemeData darkTheme = ThemeData(  
  
    //COLOUR  
  
    primaryColor: primaryDark,  
    colorScheme: ColorScheme.fromSwatch().copyWith(  
        secondary: secondaryDark,  
        error: errorDark,  
        tertiary: successDark,  
        onBackground: textDark,  
        onSurface: mono700,  
    ),  
  
    //MAIN APP  
  
    bottomNavigationBarTheme: BottomNavigationBarThemeData(  
        backgroundColor: mono800, unselectedItemColor: mono600),  
        appBarTheme: AppBarTheme(backgroundColor: mono800),  
        backgroundColor: bgDark,
```

```
scaffoldBackgroundColor: bgDark,  
  
// TEXT  
  
textTheme: TextTheme(  
  
    headline1: TextStyle(  
  
        color: textDark,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w800,  
  
        fontSize: 32),  
  
    headline2: TextStyle(  
  
        color: mono050,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w700,  
  
        fontSize: 28),  
  
    headline3: TextStyle(  
  
        color: mono100,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w600,  
  
        fontSize: 21),  
  
    headline4: TextStyle(  
  
        color: mono100,
```

```
fontFamily: 'Satoshi',
fontWeight: FontWeight.w500,
fontSize: 21),

headline5: TextStyle(
color: mono300,
fontFamily: 'Satoshi',
fontWeight: FontWeight.w500,
fontSize: 20),

headline6: TextStyle(
color: mono400,
fontFamily: 'Satoshi',
fontWeight: FontWeight.w600,
fontSize: 16),

bodyText1: TextStyle(
color: textDark,
fontWeight: FontWeight.w600,
fontFamily: 'NotoSans',
fontSize: 14),

bodyText2: TextStyle(
color: mono000,
fontWeight: FontWeight.w400,
fontFamily: 'NotoSans',
```

```
    fontSize: 16)),  
  
//BUTTONS  
  
buttonTheme: ButtonThemeData(  
  
    shape:  
  
        RoundedRectangleBorder(borderRadius: BorderRadius.circular(10.0)),  
  
    padding:  
  
        const EdgeInsets.only(top: 18, bottom: 18, left: 30, right: 30)),  
  
elevatedButtonTheme: ElevatedButtonThemeData(  
  
    style: ElevatedButton.styleFrom(  
  
        foregroundColor: bgDark,  
  
        backgroundColor: primaryDark,  
  
        padding: const EdgeInsets.only(  
  
            top: 15, bottom: 15, left: 30, right: 30),  
  
        shape: RoundedRectangleBorder(  
  
            borderRadius: BorderRadius.circular(5), // <-- Radius  
>),  
  
        textStyle: TextStyle(  
  
            color: mono900,  
  
            fontFamily: 'Satoshi',  
  
            fontWeight: FontWeight.w600,
```

```
        fontSize: 16))),

outlinedButtonTheme: OutlinedButtonThemeData(
    style: OutlinedButton.styleFrom(
        foregroundColor: primaryDark,
        padding: const EdgeInsets.only(
            top: 15, bottom: 15, left: 30, right: 30),
        side: BorderSide(color: primaryDark, width: 3),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(5), // <-- Radius
        ),
        // minimumSize: const Size.fromHeight(50),
    ),

    textStyle: TextStyle(
        color: primaryDark,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16)),
)

textButtonTheme: TextButtonThemeData(
    style: TextButton.styleFrom(
        foregroundColor: primaryDark,
        padding: const EdgeInsets.only(
            top: 18, bottom: 18, left: 30, right: 30),
    ),
)
```



```
errorBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: errorDark)),  
  
focusedErrorBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: errorLight)),  
  
enabledBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: mono600))));  
}  
}
```

browse_categories.dart

```
import 'package:alleat/widgets/elements/elements.dart';  
  
import 'package:flutter/material.dart';  
  
  
  
List categoriesList = [  
  
    //List of categories  
  
    [  
  
        "Burgers",  
  
        const Color(0xFFFFD8C2),  
  
        'lib/assets/images/categories/burger-category.png'  
  
    ],  
  
    [  
  
        "Drinks",  
  
        const Color(0xFFE0B080),  
  
        'lib/assets/images/categories/drinks-category.png'  
  
    ],  
  
    [  
  
        "Sides",  
  
        const Color(0xFFA0C0E0),  
  
        'lib/assets/images/categories/sides-category.png'  
  
    ],  
  
    [  
  
        "Desserts",  
  
        const Color(0xFFB0C0E0),  
  
        'lib/assets/images/categories/desserts-category.png'  
  
    ],  
  
    [  
  
        "Salads",  
  
        const Color(0xFFC0E0C0),  
  
        'lib/assets/images/categories/salads-category.png'  
  
    ],  
  
    [  
  
        "Breakfast",  
  
        const Color(0xFFD0C0E0),  
  
        'lib/assets/images/categories/breakfast-category.png'  
  
    ]
```

```
"American",

const Color(0xffeec4c4),
'lib/assets/images/categories/american-category.png'

],

[

"Breakfast",

const Color(0xffc6f0d9),
'lib/assets/images/categories/breakfast-category.png'

],

[

"Sushi",

const Color(0xffc2f7a9),
'lib/assets/images/categories/sushi-category.png'

],

[

"Chinese",

const Color(0xffffeec2),
'lib/assets/images/categories/chinese-category.png'

],

[

"Thai",

const Color(0xffff8c6aa),
```

```
'lib/assets/images/categories/thai-category.png'  
],  
[  
    "Coffee",  
    const Color(0xffff0c6df),  
    'lib/assets/images/categories/coffee-category.png'  
],  
[  
    "Greek",  
    const Color(0xfffc6e1f0),  
    'lib/assets/images/categories/greek-category.png'  
],  
];  
  
class Categories extends StatefulWidget {  
    const Categories({super.key});  
  
    @override  
    State<Categories> createState() => _CategoriesState();  
}  
  
class _CategoriesState extends State<Categories> {
```

```
@override

Widget build(BuildContext context) {

  return SizedBox(
    //Create profile selection widget with height 150px
    height: 170,
    child: ListView(scrollDirection: Axis.horizontal, children: <Widget>[
      Row(mainAxisAlignment: MainAxisAlignment.center, children: [
        Padding(
          padding: const EdgeInsets.only(left: 25),
          child: SizedBox(
            width: 825,
            child: ListView.builder(
              itemCount: 5,
              scrollDirection: Axis.horizontal, //Make scrollable list
              itemBuilder: (context, index) {
                List category = categoriesList[index];
                return Padding(
                  padding:
                    const EdgeInsets.symmetric(horizontal: 7),
                  child: InkWell(
                    onTap: (() {}),
                    child: Stack(

```

```
        alignment: AlignmentDirectional.topCenter,  
  
        children: [  
  
            Container(  
  
                decoration: BoxDecoration(  
  
                    borderRadius:  
  
                        const BorderRadius.all(  
  
                            Radius.circular(10)),  
  
                    color: category[1],  
  
                ),  
  
                width: 150,  
  
                height: 170,  
  
                padding: const EdgeInsets.symmetric(  
  
                    horizontal: 10),  
  
            ),  
  
            Padding(  
  
                padding: const EdgeInsets.symmetric(  
  
                    vertical: 15),  
  
                child: Align(  
  
                    alignment: Alignment.topCenter,  
  
                    child: Text(  
  
                        category[0],  
  
                        textAlign: TextAlign.center,  
                    ),  
                ),  
            ),  
        ),  
    ),  
);
```

```
        style: Theme.of(context)

            .textTheme

            .headline5!

            .copyWith(
                color: const Color(
                    0xff000000)),

        ))),
    Image.asset(
        //Fullscreen image of food
        category[2],
        alignment: Alignment.bottomRight,
    ),
])));
})))
])
]);
}
}

class CategoriesPage extends StatefulWidget {
const CategoriesPage({super.key});
```

```
@override

State<CategoriesPage> createState() => _CategoriesPageState();

}

class _CategoriesPageState extends State<CategoriesPage> {

@override

Widget build(BuildContext context) {

return Scaffold(

body: CustomScrollView(slivers: [

SliverFillRemaining(

hasScrollBody: false,

child:

Column(crossAxisAlignment: CrossAxisAlignment.start, children: [

const ScreenBackButton(), //Back button

const SizedBox(height: 20),

Padding(

padding: const EdgeInsets.symmetric(horizontal: 30),

child: Text(

"All Categories.",

textAlign: TextAlign.start,

style: Theme.of(context).textTheme.headline2,

)),
```

```
Padding(  
  padding: const EdgeInsets.only(  
    bottom: 40, top: 20, left: 20, right: 20),  
  child: SizedBox(  
    height: (categoriesList.length * 185) / 2.0.ceil(),  
    child: GridView.builder(  
      //Build grid of categories  
      itemCount: categoriesList.length,  
      physics: const NeverScrollableScrollPhysics(),  
      gridDelegate:  
        const SliverGridDelegateWithFixedCrossAxisCount(  
          crossAxisCount: 2,  
          crossAxisSpacing: 0.0,  
          mainAxisSpacing: 10.0),  
      itemBuilder: (context, index) {  
        List category = categoriesList[index];  
        return Padding(  
          padding: const EdgeInsets.symmetric(horizontal: 7),  
          child: InkWell(  
            onTap: (() {}),  
            child: Stack(  
              alignment: AlignmentDirectional.topCenter,
```

```
children: [  
  
    Container(  
  
        decoration: BoxDecoration(  
  
            borderRadius: const BorderRadius.all(  
  
                Radius.circular(10)),  
  
            color: category[1],  
  
>),  
  
        width: 150,  
  
        height: 170,  
  
        padding: const EdgeInsets.symmetric(  
  
            horizontal: 10),  
  
>),  
  
    Padding(  
  
        padding: const EdgeInsets.symmetric(  
  
            vertical: 15),  
  
        child: Align(  
  
            alignment: Alignment.topCenter,  
  
            child: Text(  
  
                category[0],  
  
                textAlign: TextAlign.center,  
  
                style: Theme.of(context)  
  
.textTheme
```

```
        .headline5!  
  
        .copyWith(  
            color: const Color(  
                0xFF000000)),  
  
        )),  
  
        Image.asset(  
            //Fullscreen image of food  
            category[2],  
            alignment: Alignment.bottomRight,  
  
        ),  
  
    ]));  
  
    })),  
  
)  
  
]),  
  
)  
  
]),  
);  
  
}  
}
```

elements.dart

```
import 'package:flutter/material.dart';

class ScreenBackButton extends StatelessWidget {
  final dynamic data;
  const ScreenBackButton({Key? key, this.data})
    : super(key: key); //Get any data that needs to be sent to previous screen

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(top: 50),
      child: TextButton.icon(
        onPressed: () =>
          Navigator.of(context).pop(data), //Remove current screen
        label: Text(
          "Back",
          style: Theme.of(context)
            .textTheme
            .headline6!
            .copyWith(color: Theme.of(context).primaryColor),
        ),
      ),
    );
  }
}
```

```
        icon: Icon(  
            Icons.chevron_left,  
            color: Theme.of(context).primaryColor,  
        ));  
    }  
}
```

profiles_buttons.dart

```
import 'package:flutter/material.dart';

class ProfileButton extends StatelessWidget {

    final dynamic icon;

    final dynamic name;

    final dynamic action;

    const ProfileButton({Key? key, this.icon, this.name, this.action})

        : super(key: key);

    @override

    Widget build(BuildContext context) {
```

```
return Container(  
  padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 15),  
  width: double.infinity,  
  height: 65,  
  margin: const EdgeInsets.symmetric(vertical: 7, horizontal: 25),  
  decoration: BoxDecoration(  
    borderRadius: const BorderRadius.all(Radius.circular(10)),  
    color: Theme.of(context).colorScheme.onSurface,  
    boxShadow: [  
      BoxShadow(  
        color: Theme.of(context).colorScheme.onBackground.withOpacity(0.05),  
        spreadRadius: 10,  
        blurRadius: 45,  
        offset: const Offset(0, 30), // changes position of shadow  
      ),  
    ],  
  ),  
  child: InkWell(  
    child: Row(  
      children: [  
        Icon(icon, color: Theme.of(context).colorScheme.onBackground),  
        const SizedBox(width: 20),  
      ],  
    ),  
  ),  
);
```

```
        Text(  
            name,  
            style: Theme.of(context).textTheme.headline5?.copyWith(  
                color: Theme.of(context).colorScheme.onBackground,  
                fontWeight: FontWeight.w700),  
            )  
        ],  
    ),  
    onTap: () {  
        action;  
    },  
),  
);  
}  
}  
}
```

profiles_selection.dart

```
import 'package:alleat/screens/profilesetup/profilesetup_existing.dart';  
  
import 'package:alleat/services/localprofiles_service.dart';  
  
import 'package:alleat/services/setselected.dart';  
  
import 'package:flutter/material.dart';
```

```
class ProfileList extends StatefulWidget {

  const ProfileList({super.key});

  @override

  State<ProfileList> createState() => _ProfileListState();
}

class _ProfileListState extends State<ProfileList> {

  bool edit = false;

  Future<List> getDisplayableProfiles() async {

    return await SQLiteLocalProfiles.getDisplayProfilesList();
  }

  Future<void> selectProfile(id) async {

    var getToSelect = (await SQLiteLocalProfiles.getProfileFromID(id))[0];

    bool trySelect = await SetSelected.selectProfile(
      getToSelect['profileid'],
      getToSelect['firstname'],
      getToSelect['lastname'],
    );
  }
}
```

```
getToSelect['email']);

if (trySelect == true) {

  setState(() {

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text(
          'Switched profiles successfully',
        ),
      ),
    );
  });
}

} else {
  setState(() {
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
      content: Text(
        'Failed to switch profiles.',
      ),
    ));
  });
}
}
```

```
@override

Widget build(BuildContext context) {

  return SizedBox(

    //Create profile selection widget with height 150px

    height: 150,

    child: ListView(scrollDirection: Axis.horizontal, children: <Widget>[

      Row(mainAxisAlignment: MainAxisAlignment.center, children: [

        FutureBuilder<List>(

          // Get selected profile

          future: getDisplayableProfiles(),

          builder: (context, snapshot) {

            if (!snapshot.hasData) {

              //While there is no information sent back

              return Column(

                //Display empty circle

                children: [
                  Container(
                    width: 120,
                    height: 120,
                    decoration: BoxDecoration(
                      shape: BoxShape.circle,
                      color: Theme.of(context)

```

```
        .navigationBarTheme  
        .backgroundColor))  
  
    ],  
  
);  
  
} else if (snapshot.hasData) {  
  
    // Once there is information  
  
    List? profileInfo = snapshot.data; // Store in profileInfo  
  
  
  
    if (profileInfo != null && profileInfo.isNotEmpty) {  
  
        //If it contains a profile  
  
        return Padding(  
  
            padding: const EdgeInsets.only(left: 30),  
  
            child: SizedBox(  
  
                //Set width of data to the number of profiles by  
120px  
  
                width: (profileInfo.length) * 120,  
  
                child: ListView.builder(  
  
                    //For each profile  
  
                    itemCount: profileInfo.length,  
  
                    scrollDirection:  
  
                        Axis.horizontal, //Make scrollable list  
  
                    itemBuilder: (context, index) {  
  
                        Map<String, dynamic> profile = profileInfo[
```

```
index]; //Store current profile being
created in profile

if (index == 0) {

    return InkWell(
        onTap: (() {}),
        child: Container(
            padding: const EdgeInsets.symmetric(
                horizontal: 10),
            child: Column(children: [
                Stack(
                    alignment: Alignment.center,
                    children: [
                        Container(
                            width: 93,
                            height: 93,
                            decoration:
BoxDecoration(
                            shape:
                                BoxShape.circle,
                            border: Border.all(
                                color: Theme.of(
                                    context)
```

```
.primaryColor,  
    width: 3,  
    style:  
        BorderStyle  
            .solid)),  
,  
Container(  
    // Create a circle with  
    a purple outline and and the first letter of first and last name  
    width: 80,  
    height: 80,  
    decoration:  
BoxDecoration(  
    shape: BoxShape  
        .circle,  
    color:  
Color.fromRGBO(  
    profile[  
  
'profilecolorred'],  
    profile[  
  
'profilecolorgreen'],  
    profile[
```

```
'profilecolorblue'],
    1)),
    child: Align(
        alignment:
            Alignment
                .center,
        child: Text(
            '${profile['firstname'][0]}${profile['lastname'][0]}',
            style:
Theme.of(
context)
            .textTheme
            .headline3
            ?.copyWith(
                color:
Theme.of(context).backgroundColor)))
    ],
    const SizedBox(height: 15),
    SizedBox(
        //Display profile name
```

```
        width: 100,  
  
        child: Text(  
  
          "${profile['firstname']}",  
  
          textAlign:  
            TextAlign.center,  
  
          style: Theme.of(context)  
  
            .textTheme  
  
            .headline5  
  
            ?.copyWith(  
  
              fontWeight:  
                FontWeight  
  
                .w600,  
  
              color: Theme.of(  
  
                context)  
  
                .textTheme  
  
                .headline1  
  
                ?.color),  
  
              overflow:  
                TextOverflow.ellipsis,  
  
            ))  
  
        ]));  
  
      } else if (index > 0) {
```

```
        return InkWell(  
          onTap: (() {  
            selectProfile(profile['id']);  
          }),  
          child: Container(  
            padding:  
              const EdgeInsets.symmetric(  
                horizontal: 10),  
            child: Column(children: [  
              Container(  
                // Create a circle with a  
                purple outline and the first letter of first and last name  
                width: 88,  
                height: 88,  
                decoration: BoxDecoration(  
                  shape: BoxShape.circle,  
                  color: Color.fromRGBO(  
                    profile[  
                      'profilecolorred'],  
                    profile[  
                      'profilecolorgreen'],  
                    profile[
```

```
'profilecolorblue'],
    1)),
    child: Align(
        alignment:
            Alignment.center,
        child: Text(
            '${profile['firstname'][0]}${profile['lastname'][0]}',
            style: Theme.of(
                context)
            .textTheme
            .headline3
            ?.copyWith(
                color:
                    Theme.of(
                context)
            .backgroundColor)))),
    const SizedBox(height: 15),
    SizedBox(
        //Display profile name
        width: 100,
```

```
        child: Text(  
            "${profile['firstname']}",  
            textAlign:  
                TextAlign.center,  
            style: Theme.of(context)  
                .textTheme  
                .headline5  
                ?.copyWith(  
                    fontWeight:  
                        FontWeight  
                            .w600,  
                    color: Theme.of(  
                        context)  
                            .textTheme  
                            .headline1  
                            ?.color),  
                    overflow:  
                        TextOverflow.ellipsis,  
                ))  
        ]));  
    } else {  
        return Container(  
    }
```

```
padding: const EdgeInsets.symmetric(
    horizontal: 10),
child: Column(children: [
    Container(
        // Create a circle with a purple
outline and and the first letter of first and last name
        width: 100,
        height: 100,
        decoration: BoxDecoration(
            shape: BoxShape.circle,
            color: Theme.of(context)
                .colorScheme
                .error),
    child: Align(
        alignment: Alignment.center,
        child: Text(
            "Error",
            style: Theme.of(context)
                .textTheme
                .headline5
                ?.copyWith(
                    color: const Color(
                        0xffffffff)),
    
```

```
        ),  
        ))  
    ]));  
}  
});  
}  
} else {  
    return Column(  
        children: [  
            Container(  
                width: 100,  
                height: 100,  
                decoration: BoxDecoration(  
                    shape: BoxShape.circle,  
                    color: Theme.of(context).errorColor))  
            ],  
        );  
}  
}  
} else {  
    return Column(  
        children: [  
            Container(  
                width: 100,  
                height: 100,  
                decoration: BoxDecoration(  
                    shape: BoxShape.circle,  
                    color: Theme.of(context).primaryColor))  
            ],  
        );  
}  
}
```

```
        height: 100,  
  
        decoration: BoxDecoration(  
  
            shape: BoxShape.circle,  
  
            color: Theme.of(context).errorColor)  
  
        ],  
  
    );  
  
}  
  
)),  
  
Column(  
  
    children: [  
  
    Padding(  
  
        padding: const EdgeInsets.symmetric(horizontal: 20),  
  
        child: Column(children: [  
  
        ElevatedButton(  
  
            onPressed: () {  
  
                Navigator.push(  
  
                    context,  
  
                    MaterialPageRoute(  
  
                        builder: (context) =>  
  
                            const ProfileSetupExisting()));  
  
            },  
  
            style: ElevatedButton.styleFrom(  

```

```
        shape: const CircleBorder(),
        padding: const EdgeInsets.all(22)),
        child: Icon(
            Icons.add,
            color: Theme.of(context).backgroundColor,
            size: 40,
        ),
    ),
    const SizedBox(height: 20),
])
],
)
])
]);
}
}
```

search.dart

```
import 'package:flutter/material.dart';

class SearchBar extends StatelessWidget {
```

```
const SearchBar({super.key});
```



```
@override
```

```
Widget build(BuildContext context) {
```

```
    return InkWell(
```

```
        child: Container(
```

```
            width: double.infinity,
```

```
            height: 60,
```

```
            margin: const EdgeInsets.symmetric(vertical: 10, horizontal: 30),
```

```
            padding: const EdgeInsets.symmetric(horizontal: 20),
```

```
            decoration: BoxDecoration(
```

```
                borderRadius: const BorderRadius.all(Radius.circular(10)),
```

```
                color: Theme.of(context).colorScheme.onSurface,
```

```
                boxShadow: [
```

```
                    BoxShadow(
```

```
                        color: Theme.of(context).colorScheme.onBackground.withOpacity(0.05),
```

```
                        spreadRadius: 10,
```

```
                        blurRadius: 45,
```

```
                        offset: const Offset(0, 30), // changes position of shadow
```

```
                    ),
```

```
                ],
```

```
            ),
```

```
        child: Row(children: [  
  
        Icon(  
  
            Icons.search,  
  
            color: Theme.of(context).textTheme.headline1?.color,  
        ),  
  
        const SizedBox(  
  
            width: 20,  
        ),  
  
        Text(  
  
            'Try searching "Pizza"',  
            style: Theme.of(context).textTheme.bodyText2,  
        )  
    )),  
));  
}  
}
```

genericloading.dart

```
import 'package:flutter/material.dart';  
  
class GenericLoading extends StatefulWidget {
```

```
const GenericLoading({super.key});
```

```
@override
```

```
State<GenericLoading> createState() => _GenericLoadingState();
```

```
}
```

```
class _GenericLoadingState extends State<GenericLoading> {
```

```
    @override
```

```
    Widget build(BuildContext context) {
```

```
        return MaterialApp(
```

```
            title: "Loading",
```

```
            home: Scaffold(
```

```
                resizeToAvoidBottomInset: false,
```

```
                body: Column(
```

```
                    mainAxisAlignment: MainAxisAlignment.center,
```

```
                    crossAxisAlignment: CrossAxisAlignment.center,
```

```
                    children: const [CircularProgressIndicator()])),
```

```
            theme:
```

```
                Theme.of(context)); //Show blank page with circular loading circle
```

```
    }
```

```
}
```

navigationbar.dart

```
import 'package:alleat/screens/navigationscreens/browse.dart';

import 'package:alleat/screens/navigationscreens/foryou.dart';

import 'package:alleat/screens/navigationscreens/homepage.dart';

import 'package:alleat/screens/navigationscreens/profiles.dart';

import 'package:flutter/material.dart';

class Navigation extends StatefulWidget {

  const Navigation({Key? key}) : super(key: key);

  @override

  State<Navigation> createState() => _NavigationState();

}

class _NavigationState extends State<Navigation> {

  int _selectedIndex = 0; //Start at Home page

  final List _screens = [
    {"screen": const HomePage()},
    {"screen": const ForYouPage()},
    {"screen": const BrowsePage()},
```

```
{"screen": const ProfilePage()}
```

```
];
```

```
void _onItemTapped(int index) {
```

```
    //When user click button on bottom navigation bar, change to that index
```

```
    setState(() {
```

```
        _selectedIndex = index;
```

```
    });
```

```
}
```

```
@override
```

```
Widget build(BuildContext context) {
```

```
    return WillPopScope(
```

```
        onWillPop: () async => false,
```

```
        child: MaterialApp(
```

```
            title: "All Eat.",
```

```
            theme: Theme.of(context),
```

```
            home: Scaffold(
```

```
                body: _screens[_selectedIndex]["screen"],
```

```
                bottomNavigationBar: Theme(
```

```
                    data: Theme.of(context).copyWith(
```

```
                        canvasColor: Theme.of(context)
```

```
.bottomNavigationBarTheme  
    .backgroundColor),  
  
    child: BottomNavigationBar(  
  
        // If button is selected, show purple. If button is not  
        selected show greyed out text and icon  
  
        selectedItemColor: Theme.of(context)  
  
        .bottomNavigationBarTheme  
  
        .selectedItemColor,  
  
        unselectedItemColor: Theme.of(context)  
  
        .bottomNavigationBarTheme  
  
        .unselectedItemColor,  
  
        backgroundColor: Theme.of(context)  
  
        .bottomNavigationBarTheme  
  
        .backgroundColor,  
  
        items: <BottomNavigationBarItem>[  
  
            //Bottom navigation bar items  
  
            BottomNavigationBarItem(  
  
                icon: Container(  
  
                    padding: const EdgeInsets.only(  
  
                        bottom:  
  
                        3), //Each have padding to separate from the  
text  
  
                child: const Icon(Icons
```

```
        .home_outlined)), //Unselected icon is outlined

        label: 'Home',

        activeIcon: Container(
            padding: const EdgeInsets.only(bottom: 3),
            child: const Icon(
                Icons.home)), // Selected icon is filled

        ),
    BottomNavigationBarItem(
        //Bottom navigation bar options

        icon: Container(
            padding: const EdgeInsets.only(bottom: 3),
            child: const Icon(Icons.assistant_outlined)),

        label: 'For You',

        activeIcon: Container(
            padding: const EdgeInsets.only(bottom: 3),
            child: const Icon(Icons.assistant)),

        ),
    BottomNavigationBarItem(
        icon: Container(
            padding: const EdgeInsets.only(bottom: 3),
            child: const Icon(Icons.manage_search_rounded)),

        label: 'Browse',
```

```
        activeIcon: Container(
          padding: const EdgeInsets.only(bottom: 3),
          child: const Icon(Icons.manage_search)),
        ),
      ),
    BottomNavigationBarItem(
      icon: Container(
        padding: const EdgeInsets.only(bottom: 3),
        child: const Icon(Icons.person_outlined)),
      label: 'Profile',
      activeIcon: Container(
        padding: const EdgeInsets.only(bottom: 3),
        child: const Icon(Icons.person)),
      ),
    ],
    onTap: _onItemTapped, //On tap, change selected option
    currentIndex:
      _selectedIndex, //Make current index the one last
      selected (defaults to home)
    ),
  ))));
}

}
```

restaurants_list.dart

```
import 'package:alleat/screens/restaurant/restaurant_main.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

import 'dart:math';

class RestaurantList extends StatefulWidget {

  const RestaurantList({Key? key}) : super(key: key);

  @override

  State<RestaurantList> createState() => _RestaurantListState();
}

class _RestaurantListState extends State<RestaurantList> {

  late bool error, sending, success, serverOffline;

  late String msg;

  Map metadataTemp = {

    "error": false,

    "sort": "default",
}
```

```
"favourite": false,  
"price": [1, 2, 3, 4],  
"maxDelivery": 4.0,  
"minOrder": 40.0,  
"latitude": 0.0,  
"longitude": 0.0  
};  
  
@override  
void initState() {  
    //Default values  
    error = false;  
    sending = false;  
    success = false;  
    msg = "";  
    super.initState();  
}  
  
Future<Map> getUserMetadata() async {  
    final prefs = await SharedPreferences.getInstance();  
    final String? filterSortEncoded = prefs.getString('filtersort');  
    final double? locationLatitude = prefs.getDouble('locationLatitude');
```

```
final double? locationLongitude = prefs.getDouble('locationLongitude');

final String? profileid = prefs.getString("serverprofileid");

metadataTemp["profileid"] = profileid;

if (filterSortEncoded != null) {

    Map filterSort = json.decode(filterSortEncoded);

    metadataTemp["sort"] = filterSort["sort"];

    metadataTemp["favourite"] = filterSort["favourite"];

    metadataTemp["price"] = filterSort["price"];

    metadataTemp["maxDelivery"] = filterSort["maxDelivery"];

    metadataTemp["minOrder"] = filterSort["minOrder"];

}

if (locationLatitude == null ||
    locationLatitude == 0 ||
    locationLongitude == null ||
    locationLongitude == 0) {

    metadataTemp["error"] = true;

    return metadataTemp;

} else {

    metadataTemp["latitude"] = locationLatitude;

    metadataTemp["longitude"] = locationLongitude;

    return metadataTemp;

}
```

```
}

Future<List> getRestaurants() async {

    Map metadata = await getUserMetadata();

    if (metadata["error"] != true) {

        metadata.remove("error");

        switch (metadata["favourite"]) {

            case (true):

                metadata.remove("favourite");

                metadata["favourite"] = "true";

                break;

            case (false):

                metadata.remove("favourite");

                metadata["favourite"] = "false";

                break;

        }

        metadata["price"] = metadata["price"].join(",");

        metadata["maxDelivery"] = metadata["maxDelivery"].toString();

        metadata["profileid"] = metadata["profileid"].toString();

        metadata["minOrder"] = metadata["minOrder"].toString();
    }
}
```

```

metadata["latitude"] = metadata["latitude"].toString();

metadata["longitude"] = metadata["longitude"].toString();

String phpurl =

"https://alleat.cpur.net/query/restaurantlist.php"; //Get restaurant
list

try {

  var res = await http.post(Uri.parse(phpurl), body: metadata);

  if (res.statusCode == 200) {

    //If sends successfully

    var data = json.decode(res.body); //Decode to array

    if (data["error"]) {

      //If fails to perform query

      List error = [

        {

          "error": true,

          "message": "Server Response: ${data["message"]}",

          "restaurants": "[]"

        } //Send blank list of restaurants

      ];

      return error;

    } else {

      List listdata = await getDistance(data);

      if (listdata[0] == false) {

```

```

List error = [
{
    "error": true,
    "message": "Failed to get Location. \nTry changing your destination address",
    "restaurants": "[]"
} //Send blank list of restaurants
];
return error;
} else {
    return [listdata[1]];
}
//If success, send the list of restaurants back
}
} else {
List error = [
{
    "error": true,
    "message": "Error ${res.statusCode}: Failed to connect to server.",
    "restaurants": "[]"
}
]

```

```
];

    return error;

}

} catch (e) {

    List error = [

    {

        "error": true,

        "message":


            "An unexpected error occurred.\n Try reopening the app. \n\n

ERROR: $e",

        "restaurants": "[]"

    }

];

    return error;

}

} else {

    List error = [

    {

        "error": true,

        "message":


            "Failed to get Location. \nTry changing your destination address",

        "restaurants": "[]"

    } //Send blank list of restaurants
```

```

];
return error;
}

}

Future<List> getDistance(restaurantdata) async {

var p = 0.017453292519943295; //Convert constant from degrees to radians

final prefs = await SharedPreferences.getInstance();

final double? savedLocationLat = prefs.getDouble('locationLatitude'); //lat2

final double? savedLocationLng =

prefs.getDouble('locationLongitude'); //lng2

if (savedLocationLat != null && savedLocationLng != null) {

for (var i = 0; i < restaurantdata["restaurants"].length; i++) {

double latRestaurant =

double.parse(restaurantdata["restaurants"][i][4]); //lat1

double lngRestaurant =

double.parse(restaurantdata["restaurants"][i][5]); //lng1

double distance = 12742 *

asin(sqrt(0.5 -

cos((savedLocationLat - latRestaurant) * p) / 2 +

cos(latRestaurant * p) *

cos(savedLocationLat * p) *

```

```

        (1 - cos((savedLocationLng - lngRestaurant) * p)) /
    2));

restaurantdata["restaurants"][i].add(distance);

}

return [true, restaurantdata];

} else {

return [false, restaurantdata];

}

}

Future<String> favouriteRestaurant(restaurantID, action) async {

String phpurl =

"https://alleat.cpur.net/query/favouriterestaurant.php"; //Favourite &
unfavourite restaurant for sepcific profile using their email

final prefs = await SharedPreferences.getInstance();

final String? email = prefs.getString('email');

try {

var res = await http.post(Uri.parse(phpurl), body: {

"action": action

.toString(), //Action determines if it will favourite or unfavourite
depending on input

"profileemail": email,

"restaurantid": restaurantID,

```

```
});

if (res.statusCode == 200) {

    //On successfull send

    var data = json.decode(res.body); //Decode to array

    if (data["error"]) {

        return "failed";

    } else {

        getFavourites(); //get favourite restaurant list

        return "success";

    }

} else {

    return "failed";

}

} catch (e) {

    return "failed";

}

}

Future<List> getFavourites() async {

    String phpurl =

        "https://alleat.cpur.net/query/favouriterestaurantlist.php"; //Get

favourite restaurant ids using profile email

    final prefs = await SharedPreferences.getInstance();

}
```

```
final String? email = prefs.getString('email');

try {

var res = await http.post(Uri.parse(phiurl), body: {

    "profileemail": email.toString(),

});

if (res.statusCode == 200) {

    //If successfull send data

    var data = json.decode(res.body); //Decode to array

    if (data["error"]) {

        //If error querying data

        List error = [

            {

                "error": true,

                "favouriterestaurants": "[]"

            } //Send empty favourite restaurant ids list

        ];

        return error;

    } else {

        //If successful query

        List listdata = [data]; //Send list of restaurant ids back

        return listdata;

    }

}
```

```
        } else {

            List error = [
                {"error": true, "favouriterestaurants": "[]"}
            ];

            return error;
        }

    } catch (e) {
        List error = [
            {"error": true, "favouriterestaurants": "[]"}
        ];

        return error;
    }

}

@Override

Widget build(BuildContext context) {

    return FutureBuilder<List>(
        //Get list of restaurants data from future (rebuild on change of data)
        future: getRestaurants(),
        builder: ((context, snapshot) {

            if (snapshot.hasData) {

                //If received data

```

```
List restaurantsdata =  
  
    snapshot.data ?? [];  
    //Data stored in restaurantsdata  
  
if (restaurantsdata[0]["error"] == true) {  
  
    return Padding(  
  
        padding: const EdgeInsets.only(  
  
            left: 20, right: 20, top: 10, bottom: 10),  
  
        child: Container(  
  
            decoration: BoxDecoration(  
  
                borderRadius:  
  
                    const BorderRadius.all(Radius.circular(20)),  
  
                color: Theme.of(context).colorScheme.onSurface,  
  
                boxShadow: [  
  
                    BoxShadow(  
  
                        color: Colors.black.withOpacity(0.1),  
  
                        spreadRadius: 2,  
  
                        blurRadius: 10,  
  
                        offset: const Offset(  
  
                            0, 10), // changes position of shadow  
  
                    ),  
  
                ]),  
  
            child: Column(children: [  
  
                Padding(  

```

```
padding: const EdgeInsets.all(30),  
  
child: SizedBox(  
  
width: double.infinity,  
  
child: Center(  
  
child: Column(children: [  
  
Text(  
  
"${restaurantsdata[0]["message"]}",  
  
textAlign: TextAlign.center,  
  
style: Theme.of(context).textTheme.headline6,  
,  
  
const SizedBox(  
  
height: 20,  
,  
  
ElevatedButton(  
  
onPressed: () {  
  
setState(() {  
  
getRestaurants();  
});  
  
},  
  
child: const Text("Retry"))  
))),  
)
```

```
        ]));  
  
    } else {  
  
        try {  
  
            List restaurants = restaurantsdata[0]["restaurants"];  
  
            return FutureBuilder<List>(  
  
                //Get favourites list from future  
  
                future: getFavourites(),  
  
                builder: ((context, snapshot) {  
  
                    if (!snapshot.hasData) {  
  
                        //While no data received, show loading bar  
  
                        return const LinearProgressIndicator(  
  
                            color: Color(0xff4100C4),  
  
                            backgroundColor: Color(0xffEBE0FF));  
  
                    } else {  
  
                        //If data received  
  
                        List restaurantFavourites = snapshot.data ?? [];  
  
                        return ListView.builder(  
  
                            //Show number of containers depending on number of  
                            restaurants  
  
                            physics:  
  
                                const NeverScrollableScrollPhysics(), //Dont allow  
                                scrolling (Done by main page)  
  
                                scrollDirection: Axis.vertical,  
                
```

```
        shrinkWrap: true,  
  
        itemCount: restaurantsdata[0]["restaurants"]  
  
        .length, //For each restaurant  
  
        itemBuilder: (context, index) {  
  
            return Center(child:  
  
                LayoutBuilder(builder: (context, constraints) {  
  
                    if (restaurants.isEmpty) {  
  
                        //If there are no restaurants show container saying  
no restaurants  
  
                        return Container(  
  
                            padding: const EdgeInsets.all(20),  
  
                            child: Center(  
  
                                child: Text("No Restaurants Found",  
                                    style: TextStyle(fontSize: 18, color: Colors.black)),  
  
                            ),  
  
                        );  
  
                    } else {  
  
                        return Container(  
  
                            padding: const EdgeInsets.only(left: 20, right: 20, top: 10, bottom: 10),  
  
                            child: Column(  
  
                                children: restaurantsdata[0]["restaurants"].map((rest){  
  
                                    return Container(  
  
                                        decoration: BoxDecoration(  
  
                                            borderRadius: const BorderRadius.all(  
  
                                                Radius.circular(20)),  
  
                                            color: const Color(0xffffffff),  
  
                                            boxShadow: [  
  
                                                BoxShadow(  
  
                                                    color:  
  
                                                    Colors.black.withOpacity(0.1),  
  
                                                    spreadRadius: 2,  
  
                                                    offset: Offset(0, 2),  
  
                                                ),  
  
                                            ],  
  
                                        child: Column(  
  
                                            children: [  
  
                                                Container(  
  
                                                    padding: const EdgeInsets.all(10),  
  
                                                    child: Row(  
  
                                                        mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  
                                                        children: [  
  
                                                            Container(  
  
                                                                width: 100,  
  
                                                                height: 100,  
  
                                                                child: Image.network(rest["image"])),  
  
                                                            Container(  
  
                                                                width: 100,  
  
                                                                height: 100,  
  
                                                                child: Image.network(rest["image"])),  
  
                                                        ],  
  
                                                    ),  
  
                                                Container(  
  
                                                    padding: const EdgeInsets.all(10),  
  
                                                    child: Row(  
  
                                                        mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  
                                                        children: [  
  
                                                            Container(  
  
                                                                width: 100,  
  
                                                                height: 100,  
  
                                                                child: Image.network(rest["image"])),  
  
                                                            Container(  
  
                                                                width: 100,  
  
                                                                height: 100,  
  
                                                                child: Image.network(rest["image"])),  
  
                                                        ],  
  
                                                    ),  
  
                                                ],  
  
                                            ],  
  
                                        ),  
  
                                    );  
  
                                }).  
  
                            ).  
  
                        );  
  
                    }  
  
                },  
  
            );  
  
        },  
  
    ),  
  
);  
  
return Scaffold(  
  
    appBar: AppBar(  
  
        title: Text("Restaurants"),  
  
        centerTitle: true,  
  
        backgroundColor: Colors.indigo[900],  
  
        foregroundColor: Colors.white,  
  
    ),  
  
    body:  
  
        Container(  
  
            padding: const EdgeInsets.all(10),  
  
            child:  
  
                ListView(  
  
                    scrollDirection: Axis.vertical,  
  
                    shrinkWrap: true,  
  
                    children: [  
  
                        Container(  
  
                            padding: const EdgeInsets.all(10),  
  
                            child:  
  
                                Column(  
  
                                    children: [  
  
                                        Container(  
  
                                            padding: const EdgeInsets.all(10),  
  
                                            child:  
  
                                                Row(  
  
                                                    mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  
                                                    children: [  
  
                                                        Container(  
  
                                                            width: 100,  
  
                                                            height: 100,  
  
                                                            child: Image.network(restaurantsdata[0]["restaurants"][0]["image"])),  
  
                                                        Container(  
  
                                                            width: 100,  
  
                                                            height: 100,  
  
                                                            child: Image.network(restaurantsdata[0]["restaurants"][0]["image"])),  
  
                                                    ],  
  
                                                ),  
  
                                                Container(  
  
                                                    padding: const EdgeInsets.all(10),  
  
                                                    child:  
  
                                                        Row(  
  
                                                            mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  
                                                            children: [  
  
                                                                Container(  
  
                                                                    width: 100,  
  
                                                                    height: 100,  
  
                                                                    child: Image.network(restaurantsdata[0]["restaurants"][1]["image"])),  
  
                                                                Container(  
  
                                                                    width: 100,  
  
                                                                    height: 100,  
  
                                                                    child: Image.network(restaurantsdata[0]["restaurants"][1]["image"])),  
  
                                                            ],  
  
                                                        ),  
  
                                                ),  
  
                                            ],  
  
                                        ),  
  
                                    ],  
  
                                ),  
  
                            ),  
  
                        ),  
  
                    ],  
  
                ),  
  
            ),  
  
        ),  
  
    ),  
  
);  
  
});
```

```

        blurRadius: 10,
        offset: const Offset(0,
10), // changes position of
shadow

),
[]),
child: Column(children: const [
Padding(
padding: EdgeInsets.all(30),
child: SizedBox(
width: double.infinity,
child: Center(
child: Text(
"No restaurants found"))),
//Display no restaurants text
)
])));
} else {
// If there is restaurant
return InkWell(
//Create clickable container
onTap: () {
List restaurantinfodata = restaurantsdata[0]

```

```
["restaurants"][index];

Navigator.push(
  context,
  MaterialPageRoute(
    builder:
      (context) => //On tap, send
restaurant data associated with container to main page and go to that new screen

    RestaurantMain(
      resid:
        restaurantinfodata[0],
      resname:
        restaurantinfodata[1],
      reslogo:
        restaurantinfodata[2],
      resbanner:
        restaurantinfodata[3],
      resdistance:
        (restaurantsdata[
          0][
            "restaurants"]
          [index][8])
        .toStringAsFixed(1),
      resordermin:
```

```
        restaurantinfodata[6],  
  
        resdelivery:  
  
        restaurantinfodata[  
  
        7]),  
  
    )),  
  
},  
  
child: Container(  
  
margin: const EdgeInsets.symmetric(  
  
horizontal: 30, vertical: 20),  
  
decoration: BoxDecoration(  
  
borderRadius: const BorderRadius.all(  
  
Radius.circular(10)),  
  
color: Theme.of(context)  
  
.colorScheme  
  
.onSurface,  
  
),  
  
child: Stack(  
  
children: [  
  
Container(  
  
//Container filled with restaurant  
banner  
  
width: MediaQuery.of(context)  
  
.size
```

```
        .width,  
  
        height: 150,  
  
        decoration: BoxDecoration(  
  
            borderRadius:  
  
                const BorderRadius.only(  
  
                    topLeft: Radius.circular(10),  
  
                    topRight: Radius.circular(10),  
  
                ),  
  
            image: DecorationImage(  
  
                fit: BoxFit.fitWidth,  
  
                image: NetworkImage(  
  
                    restaurants[index][3]  
  
                    .toString()),  
  
            ),  
  
        ),  
  
    ),  
  
    Align(  
  
        alignment: Alignment.topCenter,  
  
        child: Container(  
  
            margin: const EdgeInsets.only(  
  
                top: 110),  
  
            width: 70,
```

```
height: 70,  
  
decoration: BoxDecoration(  
  
    shape: BoxShape.circle,  
  
    color: Theme.of(context)  
  
        .colorScheme  
  
        .onSurface,  
  
)),  
  
Align(  
  
    alignment: Alignment.topCenter,  
  
    child: Container(  
  
        margin: const EdgeInsets.only(  
  
            top: 115),  
  
        width: 60,  
  
        height: 60,  
  
        decoration: BoxDecoration(  
  
            image: DecorationImage(  
  
                fit: BoxFit.cover,  
  
                image: NetworkImage(  
  
                    restaurants[index][2]  
  
                    .toString()),  
  
            ),  
  
            shape: BoxShape.circle,
```

```
        color: Theme.of(context)

        .colorScheme

        .onSurface,

    ))),

Align(
    alignment: Alignment.topRight,
    child: InkWell(
        onTap: () {
            if (restaurantFavourites[0]
                ["restaurantids"]
                .contains(
                    restaurants[index]
                [0]
                .toString())))
            {
                favouriteRestaurant(
                    restaurants[index][0]
                    .toString(),
                    "unfavourite");
                setState(() {
                    getFavourites();
                });
            } else {

```

```
favouriteRestaurant(  
    restaurants[index][0]  
        .toString(),  
    "favourite");  
  
    setState(() {  
        getFavourites();  
    });  
}  
  
,  
child: Container(  
    margin:  
        const EdgeInsets.only(  
            top: 10,  
            right: 10),  
    width: 45,  
    height: 45,  
    decoration: BoxDecoration(  
        shape: BoxShape.circle,  
        color: Theme.of(context)  
            .colorScheme  
            .onSurface  
            .withOpacity(0.8),
```

```
        ),  
  
        child:  
restaurantFavourites[0]  
  
        [  
  
"restaurantids"]  
  
.contains(  
  
restaurants[index]  
  
[0]  
  
.toString())  
  
? Icon(Icons.favorite,  
  
size:  
  
30, // ? =  
favourited  
  
color: Theme.of(  
  
context)  
  
.colorScheme  
  
.secondary)  
  
: Icon(  
  
// : = unfavourited  
Icons  
  
.favorite_border_outlined,
```

```
        size: 30,  
  
        color: Theme.of(  
  
            context)  
  
.colorScheme  
  
.onBackground()))),  
  
        Align(  
  
            alignment: Alignment.bottomLeft,  
  
            child: Padding(  
  
                padding:  
  
                    const EdgeInsets.only(  
  
                        top: 190,  
  
                        left: 20,  
  
                        right: 20),  
  
                child: Column(  
  
                    children: [  
  
                        Row(  
  
                            mainAxisAlignment:  
  
                                MainAxisAlignment  
  
.spaceBetween,  
  
                            children: [  
  
                                Flexible(  
  
                                    child: Container(  
))
```

```
padding: const  
EdgeInsets  
  
.only(  
  
right:  
  
13.0),  
  
child: Text(  
  
restaurants[  
  
index][1],  
  
overflow:  
  
TextOverflow  
  
.ellipsis,  
  
style:  
Theme.of(  
  
context)  
  
.textTheme  
  
.headline4,  
  
)),  
  
Row(  
  
children: [  
  
Icon(  
)
```

```
        Icons.star,  
        size: 20,  
        color: Theme.of(  
            context)  
  
.primaryColor,  
(),  
const SizedBox(  
    width: 5,  
(),  
Text(  
    "N/A",  
    style: Theme.of(  
        context)  
.textTheme  
.bodyText1,  
)  
],  
(),  
],  
(),  
const SizedBox(  
    height: 10),
```

```
Row(  
  children: [  
    Text(  
  
      "${(restaurantsdata[0]["restaurants"][index][8]).toStringAsFixed(1)} km away",  
      style: Theme.of(  
        context)  
          .textTheme  
          .bodyText1!  
          .copyWith(  
            fontSize:  
              12)),  
    const SizedBox(  
      width: 5,  
    ),  
    Text(  
      ". ",  
      style: Theme.of(  
        context)  
          .textTheme  
          .bodyText1!  
          .copyWith(  
            fontSize:
```

```
        12),  
        ),  
        const SizedBox(  
            width: 5,  
        ),  
        Text(  
  
(double.parse(restaurantsdata[0]["restaurants"]  
            [  
  
index]  
            [  
                7]) ==  
            0)  
            ? "Free  
Delivery"  
            :  
"£${(restaurantsdata[0]["restaurants"][index][7])} Delivery",  
            style: Theme.of(  
                context)  
                    .textTheme  
                    .bodyText1!  
                    .copyWith(  
                        fontSize:  
                            12)
```

```
        12)),  
        ],  
        ),  
        const SizedBox(  
          height: 20),  
        ],  
      )))  
      ],  
    )));  
  }  
});  
});  
}  
},  
);  
} catch (e) {  
  return Padding(  
    padding: const EdgeInsets.only(  
      left: 20, right: 20, top: 10, bottom: 10),  
    child: Container(  
      decoration: BoxDecoration(  
        borderRadius:
```

```
        const BorderRadius.all(Radius.circular(20)),  
  
        color: Theme.of(context).colorScheme.onSurface,  
  
        boxShadow: [  
  
            BoxShadow(  
  
                color: Colors.black.withOpacity(0.1),  
  
                spreadRadius: 2,  
  
                blurRadius: 10,  
  
                offset: const Offset(  
  
                    0, 10), // changes position of shadow  
  
            ),  
  
        ],  
  
        child: Column(children: [  
  
            Padding(  
  
                padding: const EdgeInsets.all(30),  
  
                child: SizedBox(  
  
                    width: double.infinity,  
  
                    child: Center(  
  
                        child: Column(children: [  
  
                            const Text(  
  
                                "An unexpected error occurred. \nPlease try  
again",  
  
                            textAlign: TextAlign.center,  
  
                        ),  
  
                    ),  
  
                ),  
  
            ),  
  
        ],  
  
    ),  
  
    body: Container(  
  
        decoration: BoxDecoration(  
  
            color: Colors.white,  
  
            border: Border.all(  
  
                color: Colors.grey[300],  
  
                width: 1,  
  
            ),  
  
            borderRadius: BorderRadius.all(  
  
                Radius.circular(10),  
  
            ),  
  
            boxShadow: [  
  
                BoxShadow(  
  
                    color: Colors.black.withOpacity(0.1),  
  
                    spreadRadius: 2,  
  
                    blurRadius: 10,  
  
                    offset: const Offset(  
  
                        0, 10), // changes position of shadow  
  
                ),  
  
            ],  
  
        ),  
  
        padding: const EdgeInsets.all(10),  
  
        child: Column(  
  
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
  
            children: [  
  
                Text("Welcome to the app!"),  
  
                Text("This is a sample text."),  
  
                Text("You can change the text and colors in the theme settings."),  
  
                Text("The background color is white with a light grey border and a soft shadow effect on the corners."),  
  
                Text("The text is centered and has a consistent font size and weight across all screens."),  
  
                Text("The overall design is clean and minimalist, focusing on readability and user experience."),  
  
                Text("Feel free to explore the code and make changes to see how it works."),  
  
                Text("Happy coding!"),  
  
            ],  
  
        ),  
  
    ),  
  
);  
  
void main() {  
  
    runApp(MyApp());  
  
}
```

```
        const SizedBox(  
            height: 20,  
        ),  
        ElevatedButton(  
            onPressed: () {  
                setState(() {  
                    getRestaurants();  
                });  
            },  
            child: const Text("Retry"))  
        ])),  
    )  
]);  
}  
}  
} else {  
    return const LinearProgressIndicator(  
        color: Color(0xff4100C4),  
        backgroundColor: Color.fromARGB(0, 235, 224, 255));  
}  
},  
);
```

}

topbar.dart

```
import 'package:alleat/screens/locationselection.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

class MainAppBar extends StatelessWidget implements PreferredSizeWidget {

    final double height;

    const MainAppBar({Key? key, required this.height}) : super(key: key);

    Future<String> getSavedLocation() async {

        final prefs = await SharedPreferences.getInstance();

        final List<String>? savedLocationText =
            prefs.getStringList('locationPlacemark');

        if (savedLocationText != null) {
            return savedLocationText[1];
        } else {
            return ("No Location Set");
        }
    }

}
```

```
@override

Widget build(BuildContext context) {

    return Container(
        //Container of height 120px

        height: 120,
        color: Theme.of(context).bottomNavigationBarTheme.backgroundColor,
        child: SafeArea(
            child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                    InkWell(
                        onTap: () {
                            Navigator.push(
                                //go to profile creation page
                                context,
                                MaterialPageRoute(
                                    builder: (context) => const SelectLocation()));
                        },
                        child: Padding(
                            padding: const EdgeInsets.all(10),

```

```
        child: Icon( //Location icon

            Icons.location_on_outlined,
            color: Theme.of(context).textTheme.headline1?.color,
        ))),

Column(mainAxisAlignment: MainAxisAlignment.center, children: [ //Column that will display the destination the food will be delivered to

Text(
    "Location",
    style: Theme.of(context)
        .textTheme
        .headline6
        ?.copyWith(fontWeight: FontWeight.w500),
),
const SizedBox(
    height: 2,
),
FutureBuilder<String>(
    future: getSavedLocation(),
    builder: (context, snapshot) {
        if (!snapshot.hasData) {

            return const Text("Location loading");
        }
    }
),

```


main.dart

```
import 'package:alleat/screens/setupverification.dart';

import 'package:flutter/services.dart';

import 'package:alleat/theme/theme.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:alleat/theme/theme.dart' as globals;

void main() {

    //Start App

    runApp(
        const MyApp(),
    );
}

class MyApp extends StatelessWidget {

    const MyApp({super.key});

    Future getTheme() async {

        final prefs = await SharedPreferences.getInstance();

        final int? appTheme = prefs.getInt('appTheme');

        globals.appthemepreference = appTheme;
```

```
}

@Override
Widget build(BuildContext context) {
    SystemChrome.setPreferredOrientations([
        //Force portrait mode
        DeviceOrientation.portraitUp,
        DeviceOrientation.portraitDown,
    ]);
    WidgetsFlutterBinding.ensureInitialized();
    getTheme();
    switch (globals.appthemepreference) {
        case 1:
            return MaterialApp(
                //Create main app
                title: 'AllEat.',
                themeMode: ThemeMode.light,
                theme: ThemeClass.lightTheme,
                home:
                    const SetupWrapper(), //SetupWrapper(), //Check if setup is
                    complete for app (reference)
            );
        case 2:
```

```
        return MaterialApp(  
  
            //Create main app  
  
            title: 'AllEat.',  
  
            themeMode: ThemeMode.dark,  
  
            theme: ThemeClass.darkTheme,  
  
            home:  
  
                const SetupWrapper(), //SetupWrapper(), //Check if setup is  
complete for app (reference)  
  
        );  
  
    }  
  
    return MaterialApp(  
  
        //Create main app  
  
        title: 'AllEat.',  
  
        themeMode: ThemeMode.system,  
  
        theme: ThemeClass.lightTheme,  
  
        darkTheme: ThemeClass.darkTheme,  
  
        home:  
  
            const SetupWrapper(), //SetupWrapper(), //Check if setup is complete  
for app (reference)  
  
        );  
  
    }  
}
```

Files - Server

favouriterestaurant.php

```
<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "success" ] = false;

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["action"]) && ($_POST["profileemail"]) &&
isset($_POST["restaurantid"]);

if($val){
    $email = $_POST[ "profileemail" ]; //grabing the data from headers
    $restaurantid = $_POST[ "restaurantid" ];
    $action = $_POST[ "action" ];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT
1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row[ 'ProfileID' ];
            }
            // Free result set
            mysqli_free_result($result);
            if ($action == "unfavourite"){
                $sqlunfavourite = "DELETE FROM favrestaurant WHERE profileid
= $ProfileID AND restaurantid = $restaurantid";
                if ($link->query($sqlunfavourite) === TRUE) {
                    $return[ "success" ] = true;
                } else{
                    $return[ "error" ] = true;
                }
            }
        }
    }
}
```

```

    }

}

else if ($action == "favourite"){
    $sqlfavourite = "INSERT INTO favrestaurant (profileid,
restaurantid) VALUES ($ProfileID, $restaurantid)";
    if ($link->query($sqlfavourite) === TRUE) {
        $return["success"] = true;
    } else{
        $return["error"] = true;
    }
}

else{
    $return["error"] = true;
    $return["message"] = "action unspecified";
}
}

else{
    $return["error"] = true;
    $return["message"] = "duplicate found";
}
}

else{
    $return["error"] = true;
    $return["message"] = "failed";
}
}

else{$return["error"] = true;
    $return["message"] = "data not specified";}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

favouriterestaurantdata.php

```
<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "FO7&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurants" ] = array();
$return[ "temp" ] = array();
$restaurantids = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileemail"]);

if($val){
    $email = $_POST["profileemail"];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row[ 'ProfileID' ];
            }
            // Free result set
            mysqli_free_result($result);
        }
    }
    else{
        $return[ "error" ] = true;
        $return[ "message" ] = "duplicate found";
    }
    $sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
profileid = $ProfileID";
    if($result2 = mysqli_query($link, $sqlfavrestaurant)){
        while($row2 = mysqli_fetch_assoc($result2)) {
            array_push($restaurantids, $row2[ "restaurantid" ]);
        }
    }
}
```

```

        }
        mysqli_free_result($result2);

    }

    foreach ($restaurantids as &$value) {
        $sqlrestaurantinfo = "SELECT res.restaurantid,
res.restaurantname, res.restaurantlogo, res.restaurantbanner FROM
restaurant res WHERE res.restaurantid = $value";
        $return["temp"] = $sqlrestaurantinfo;
        if($result3 = mysqli_query($link, $sqlrestaurantinfo)){
            while($row3 = mysqli_fetch_assoc($result3)) {
                array_push($return["restaurants"],
[$row3["restaurantid"], $row3["restaurantname"], $row3["restaurantlogo"],
$row3["restaurantbanner"]]);
            }
            mysqli_free_result($result3);
        }
    }

}

else{
    $return["error"] = true;
    $return["message"] = "failed";
}

}

else{
    $return["error"] = true;
    $return["message"] = "data not specified 2";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

favouriterestaurantlist.php

```
<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "FO7&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurantids" ] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileemail"]);

if($val){
    $email = $_POST["profileemail"];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";
    if($result = mysqli_query($link, $sql)){
        if(mysqli_num_rows($result) == 1){
            while($row = mysqli_fetch_array($result)){
                $ProfileID = $row[ 'ProfileID' ];
            }
            // Free result set
            mysqli_free_result($result);
        }
    } else{
        $return[ "error" ] = true;
        $return[ "message" ] = "duplicate found";
    }
} else{
    $return[ "error" ] = true;
    $return[ "message" ] = "failed";
```

```

        }
        $sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
profileid = $ProfileID";
        if($result2 = mysqli_query($link, $sqlfavrestaurant)){
            while($row2 = mysqli_fetch_assoc($result2)) {
                array_push($return["restaurantids"],
$row2["restaurantid"]);
            }
            mysqli_free_result($result2);
        }

    }
    else{
        $return["error"] = true;
        $return["message"] = "data not specified";
    }

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

itemcustomise.dart

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

```

```

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "customiseitem" ] = [];
$return[ "test" ] = [];

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["itemid"]);

if($val){
    $itemid = $_POST["itemid"];
    $sqlcustomisetitles = "SELECT * FROM customiseitem WHERE itemid =
$itemid ORDER BY viewingorder ASC";
    if($result = mysqli_query($link, $sqlcustomisetitles)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($return[ "customiseitem" ],
[$row[ "customiseid" ], $row[ "customisename" ], $row[ "description" ],
$row[ "optiontype" ], $row[ "required" ], $row[ "parentid" ],
$row[ "maxquantity" ], []]);
        }
        mysqli_free_result($result);
    }
    for ($i = 0; $i < count($return[ "customiseitem" ]); $i++){
        $sqlcustomiseoptions = "SELECT * FROM customiseitemselection WHERE
customiseid = {$return[ 'customiseitem' ][ $i ][ 0 ]}";
        if($result2 = mysqli_query($link, $sqlcustomiseoptions)){
            while($row2 = mysqli_fetch_assoc($result2)) {
                array_push($return[ "customiseitem" ][ $i ][ 7 ], [$row2[ "optionid" ],
$row2[ "name" ], $row2[ "image" ], $row2[ "pricechange" ]]);
            }
        }
    }
} else{
    $return[ "error" ] = true;
    $return[ "message" ] = "Unknown itemid";
}

```

```
mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>
```

login.php

```
<?php

$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";
$return["exists"] = false;
$return["profile"] = "";
$return["temp1"] = "";
$return["temp2"] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);
//connecting to database server

$val = isset($_POST["email"]) && isset($_POST["password"]);

if($val){
    //checking if there is POST data
    $email = $_POST["email"];
    $password = $_POST["password"];
    $return["temp1"] = $email;
    $return["temp2"] = $password;

    $secret_iv = '5fd0b31f582beb1f';
    $secret_key = 'e16a3f356b2366c1';
    $cipher = "AES-128-CBC";
```

```

$decryptedString = openssl_decrypt ($password, $cipher,
$secret_key, 0, $secret_iv);

$sql = "SELECT ProfileID, FirstName, LastName, Email, Password
FROM profile WHERE Email = '$email'";
if($result = mysqli_query($link, $sql)){
    if(mysqli_num_rows($result) == 1){
        while($row = mysqli_fetch_array($result)){
            $ProfileIDR = $row['ProfileID'];
            $FirstNameR = $row['FirstName'];
            $LastNameR = $row['LastName'];
            $EmailR = $row['Email'];
            $PasswordR = $row['Password'];
        }
        // Free result set
        mysqli_free_result($result);
        $ispasswordsame = password_verify($decryptedString,
$PasswordR);
        if ($ispasswordsame == true){
            $PasswordREncrypted = openssl_encrypt($PasswordR,
$cipher, $secret_key, 0, $secret_iv);
            $return["exists"] = true;
            $return["profile"] = array($ProfileIDR,
$FirstNameR, $LastNameR, $EmailR, $PasswordREncrypted);
        }
        else{
            $return["error"] = false;
            $return["message"] = "Email or password incorrect";
            $return["exists"] = false;
            $return["profile"] = null;
        }
    } else{
        $return["error"] = false;
        $return["message"] = "No matching profiles";
        $return["exists"] = false;
        $return["profile"] = null;
    }
} else{
    $return["error"] = true;
    $return["message"] = "Failed profile search";
}

```

```

}else{
    $return["error"] = true;
    $return["message"] = 'Send all parameters.';
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string

?>

```

register.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return["error"] = false;
$return["message"] = "";
$return["hash"] = "";
$return["profile"] = "";
$return["exist"] = false;
$return["temp"] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);
//connecting to database server

$val = isset($_POST["firstname"]) && isset($_POST["lastname"])
    && isset($_POST["email"]) && isset($_POST["password"]);

if($val){
    //checking if there is POST data
}

```

```

$firstname = $_POST["firstname"]; //grabbing the data from headers
$lastname = $_POST["lastname"];
$email = $_POST["email"];
$password = $_POST["password"];

$secret_iv = '5fd0b31f582beb1f';
$secret_key = 'e16a3f356b2366c1';
$cipher = "AES-128-CBC";
$decryptedString = openssl_decrypt ($password, $cipher,
$secret_key, 0, $secret_iv);
$hashpassword = password_hash($decryptedString, PASSWORD_DEFAULT);
$base64hash = base64_encode($hashpassword);
$encryptedhashedpassword = openssl_encrypt ($base64hash, $cipher,
$secret_key, 0, $secret_iv);
$return["hash"] = $encryptedhashedpassword; //Encrypt hashed
password to be sent back

if($return["error"] == false){
    $firstname = mysqli_real_escape_string($link, $firstname);
    $lastname = mysqli_real_escape_string($link, $lastname);
    $email = mysqli_real_escape_string($link, $email);
    $hashpassword = mysqli_real_escape_string($link,
$hashpassword);
    //escape inverted comma query conflict from string

    $sqlverify = "SELECT * FROM profile WHERE Email = '$email'";
    $res = mysqli_query($link, $sqlverify);
    if (mysqli_num_rows($result) > 0) {
        $return["exist"] = true;
    }
    else{
        $sql = "INSERT INTO profile SET
                FirstName = '$firstname',
                LastName = '$lastname',
                Email = '$email',
                Password = '$hashpassword'";
        $res = mysqli_query($link, $sql);

        $sqlget = "SELECT ProfileID, FirstName, LastName, Email,
Password FROM profile WHERE Email = '$email'";
        if($result = mysqli_query($link, $sqlget)){
            if(mysqli_num_rows($result) == 1){

```

```

        while($row = mysqli_fetch_array($result)){
            $ProfileIDR = $row[ 'ProfileID' ];
            $FirstNameR = $row[ 'FirstName' ];
            $LastNameR = $row[ 'LastName' ];
            $EmailR = $row[ 'Email' ];
            $PasswordR = $row[ 'Password' ];
        }
        // Free result set
        mysqli_free_result($result);
        $ispasswordsame = password_verify($decryptedString,
$PasswordR);
        if ($ispasswordsame == true){
            $PasswordREncrypted = openssl_encrypt($PasswordR,
$cipher, $secret_key, 0, $secret_iv);
            $return[ "exists" ] = true;
            $return[ "profile" ] = array($ProfileIDR,
$FirstNameR, $LastNameR, $EmailR, $PasswordREncrypted);
        }
        else{
            $return[ "error" ] = false;
            $return[ "message" ] = "Email or password incorrect";
            $return[ "exists" ] = false;
            $return[ "profile" ] = null;
        }
    } else{
        $return[ "error" ] = false;
        $return[ "message" ] = "No matching profiles";
        $return[ "exists" ] = false;
        $return[ "profile" ] = null;
    }
} else{
    $return[ "error" ] = true;
    $return[ "message" ] = "Failed profile search";
}

if($res){
    //write success
}else{
    $return[ "error" ] = true;
    $return[ "message" ] = "Database error";
}

```

```

        }
    }
}else{
    $return["error"] = true;
    $return["message"] = 'Send all parameters.';
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

restaurantlist.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "restaurants" ] = [];

$favrestaurantslist = [];
$filterquery = [];
$fullfilter;

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["sort"]) && isset($_POST["favourite"]) &&
isset($_POST["price"]) && isset($_POST["maxDelivery"]) &&
isset($_POST["minOrder"]) && isset($_POST["latitude"]) &&
isset($_POST["longitude"]) && isset($_POST["profileid"]);

$sort = $_POST[ "sort" ];

```

```

$profileid = $_POST["profileid"];
$favourite = $_POST["favourite"];
$maxDelivery = (double)$_POST["maxDelivery"];
$minOrder = (double)$_POST["minOrder"];
$destinationLatitude = (double)$_POST["latitude"];
$destinationLongitude = (double)$_POST["longitude"];
$priceTarget = explode(",", $_POST["price"]); //Convert string of
values to list

for($i = 0; $i < $priceTarget.length; $i++){
    $priceTarget[$i] = (double)$priceTarget[$i]; //Convert List of
strings to list of integers
}

if($val){

    if ($favourite == "true"){
        $sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
profileid = $profileid";
        if($result = mysqli_query($link, $sqlfavrestaurant)){
            while($row = mysqli_fetch_assoc($result)) {
                array_push($favrestaurantslist, $row["restaurantid"]);
            }
        }
        array_push($filterquery, "res.restaurantid IN (" . implode(',', $favrestaurantslist) . ")");
    }
    array_push($filterquery, "res.pricebracket IN (" . implode(',', $priceTarget) . ")",
    "res.delivery <= $maxDelivery", "res.ordermin <=
$minOrder");

    $fullfilter = "WHERE " . implode(" AND ", $filterquery); //Compile the
where statements into one
    if ($sort == "distance"){ //If the sort type is distance
        $sqldistance = "SELECT res.restaurantid, res.restaurantname,
res.restaurantlogo, res.restaurantbanner, res.latitude, res.longitude,
res.servicedistance, res.ordermin, res.delivery, ( 6371 * acos( cos(
radians($destinationLatitude) ) * cos( radians( latitude ) ) * cos(
radians( longitude ) - radians($destinationLongitude) ) + sin(
radians($destinationLatitude) ) * sin( radians( latitude ) ) ) ) AS
distance FROM restaurant res $fullfilter HAVING distance <
res.servicedistance ORDER BY distance ASC LIMIT 20";
    }
}

```

```

        if($result = mysqli_query($link, $sqldistance)){
            while($row = mysqli_fetch_assoc($result)) {
                array_push($return["restaurants"],
[$row["restaurantid"], $row["restaurantname"], $row["restaurantlogo"],
$row[ "restaurantbanner"], $row["latitude"], $row["longitude"],
$row[ "ordermin"], $row[ "delivery"]]);
            }
            mysqli_free_result($result);
        }
    }
    else{ //If the sort type is not distance, return error
        $return["error"] = true;
        $return["message"] = "Sort method unavailable";
    }
}
else{
    $return["error"] = true;
    $return["message"] = "Data not specified";
}
mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

restaurantmenucategories.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error"] = false;

```

```

$return[ "message" ] = "";
$return[ "restaurantcategories" ] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["restaurantid"]);

if($val){
    $resid = $_POST[ "restaurantid" ];

    $sql = "SELECT categoryname, categoryid FROM menucategories WHERE
restaurantid = '$resid' ORDER BY viewingorder ASC";
    if($result = mysqli_query($link, $sql)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($return[ "restaurantcategories" ],
[$row[ "categoryname" ], $row[ "categoryid" ]]);
        }
        mysqli_free_result($result);
    }
    else{
        $return[ "error" ] = true;
        $return[ "message" ] = "restaurant not sepcified";
    }
}
else{
    $return[ "error" ] = true;
    $return[ "message" ] = "data not specified";
}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

restaurantmenuitems.php

```
<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "FO7&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";
$return[ "menuitems" ] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["categoryid"]);

if($val){
    $rescategory = $_POST["categoryid"];

    $sql = "SELECT itemid, foodcategory, subfoodcategory, itemname,
description, price, itemimage FROM item WHERE categoryid = '$rescategory'
ORDER BY itemid ASC";
    if($result = mysqli_query($link, $sql)){
        while($row = mysqli_fetch_assoc($result)) {
            array_push($return[ "menuitems" ], [$row[ "itemid" ],
$row[ "foodcategory" ], $row[ "subfoodcategory" ], $row[ "itemname" ],
$row[ "description" ], $row[ "price" ], $row[ "itemimage" ]]);
        }
        mysqli_free_result($result);
    }
    else{
        $return[ "error" ] = true;
        $return[ "message" ] = "category not sepcified";
    }
}
else{
    $return[ "error" ] = true;
    $return[ "message" ] = "data not specified";
}
```

```
mysqli_close($link); //close mysqli

header('Content-Type: application/json');
    // tell browser that its a json data
echo json_encode($return);
    //converting array to JSON string
?>
```

Summary

Prototype 2 was an iteration on Prototype 1B, altering the location selection screen to use a map based system instead of a text based system. As well, filtering and sorting was added to the app. While only the distance sort method was added, the capability of doing other sorts was implemented. On top of filtering, the item customisation page was made which allows for multi-item selection, adding and removing from the item and saving the item to cart.

Evaluation

While developing Prototype 2, I had a few issues including dealing maps. Although they may seem basic, dealing with the many varieties of possibilities where people could live made it hard to design a fluid location system. In order to deal with this, I decided to make a basic map for people who live in single houses and if this fails due to map error, the user can drag up to edit the details manually. This ensures that it is accurate and easy for the delivery driver to know where to deliver. Another issue was dealing with the variety of different combinations of selections when customising an item. Although I could have made the algorithm more efficient by using separate files for each selection system, due to the lack of time, I was unfortunately unable to do that.

Prototype 3

Design

With prototype 3, I aim to complete the rest of the ordering process, from the cart all the way to when the user receives their food. Since a restaurant side app is not being made, the app will try to check the database for updates to the status and refresh the page adaptively.

Success Criteria

For this prototype, I aim to fulfil the following success criteria

- 1.3* - The database should be in third-normal form and normalised
- 5.1 - There should be a button to remove an item from the order before it has been checked out.
- 5.2 - For each item, it should show which profile the item is for
- 5.5 - When checking out, there should be a way to input the amount you would like to tip
- 5.10 - The status of the order should sync with all profiles on all devices they are signed in with

**Will be revisited in the future for when more features are added*

Concept Art

Cart

The cart updates depending on the items in the cart. If a different user selects their food, it will show as a separate section. When a user clicks on the edit button it will show a cross overlay on the image which, on clicking, removes the item from the cart and updates the food.

[Back](#)

Cart.

[!\[\]\(b7ff27ee77f030ef0e127e6554c79c2f_img.jpg\) Edit](#)



Peter Smith

2 items



1

Vegan Giardiniera

£ 13.45

Pizza Express

Which Base?

Romana

Would you like to add
any extra toppings

 Mushroom



1

Pollo Pesto

£ 13.75

Pizza Express

Subtotal

£49.52

Delivery

£4.98

Current Total

£56.72

[Continue to Checkout](#)

Checkout

On checkout, it shows the location of the restaurants and gives the ability to change the tip. During this time, the location is locked and cannot be changed.

[Back](#)

Checkout.



Location

11 Canal Reach, London



Subtotal £49.52

Delivery £4.98

Tip 5.00

Total £61.72

[Continue to Payment](#)

App Pseudocode

To start off with, since prototype 2 failed to save the profile associated with it, I would need to add that as a foreign key

The new database creation Future SQL would look like the following:

```
CREATE TABLE cartItems( cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, itemid  
INT, localprofileid int FOREIGN KEY REFERENCES localprofiles(id), customised TEXT, quantity  
INT )
```

Cart

The cart will do the following things:

- Display a list of items for each profile that is available.
- For each item, it will display the customised options.
- For each item, display the name, quantity and the new price including the price of any changed options.
- It should include the back button at the top which goes back to the previous screen
- The user should be able to swipe each item to remove it from the cart (instead of using the edit button)
- At the bottom of the screen, there should be a price for the items and a price for delivery and a current price including the food and the delivery fee
- There should be a button which brings you to the checkout page

By using Futurebuilders within the code, it can reduce the number of dimensions needed to temporarily store the data.

To do this, I will do the following:

availableProfiles: [[Profile ID, Profile Firstname, Avatar Color Red, Avatar Color Green, Avatar Color Blue]]

receivedItemInfo: [[Cart ID, Item ID, Customised List (json) - {CustomiseID: [OptionID]}, Quantity]]

itemInfo: {Cart ID: [[Item Name, Item Image, Original Price, Restaurant, Restaurant ID, delivery fee], {Customise ID: [[Customise title, type], [[Option Name, price, quantity]]] }] }

```
Future <List> getAvailableProfiles () async {  
  
    availableProfiles = [];  
  
    cartProfileIDs = await SQLiteCartItems.getProfilesInCart();  
  
    profileInfo = await SQLiteLocalProfiles.getProfiles();  
  
    for (int i = 0; i < cartProfileIDs.length; i++){  
  
        if (profileInfo[i][0] == cartProfileIDs[i]){  
  
            For each profile in the cart, return the profile info associated with it  
  
            availableProfiles.add([profileInfo[i][0], profileInfo[i][1], profileInfo[i][5],  
profileInfo[i][6], profileInfo[i][7]]);  
  
        }  
  
    }  
  
    return availableProfiles;  
  
}
```

```
Future <Map> getProfileCart(profileID) async {  
  
    itemInfo = {};  
  
    profileCart = await SQLiteCartItems.getProfileCart(profileID);  
  
    for (int i = 0; i < profileCart.length; i++){
```

```

basicItemInfo = await
QueryServer.query("https://alleat.cpur.net/query/cartiteminfo", {"type": "item", "term": profileCart[i][1]}); Returns [Item Name, Item Image, Original Price, Restaurant]

if (basicItemInfo["error"] == true){

    return {"Error": true};

}

itemInfo["${profileCart[i][1]}"] = [[basicItemInfo["message"][0],
basicItemInfo["message"][1], basicItemInfo["message"][2], basicItemInfo["message"][3]], {}]

customisedOptions = json.decode(profileCart[i][2])

for (int j = 0; j < customisedOptions.length; j++){ for each item title

    customisedOptionsList = customisedOptions.keys.toList()

    customiseTitleInfo = await
QueryServer.query("https://alleat.cpur.net/query/cartiteminfo", {"type": "title", "term": customisedOptionsList[j]});

    if (customiseTitleInfo["error"] == true){

        return {"Error": true};

    }

    itemInfo["${profileCart[i][1]}"]["${customisedOptionsList[j]}"] =
[[customiseTitleInfo["message"][0], customiseTitleInfo["message"][1]]]

    customisedTitleOptions =
customisedOptions["${customisedOptionsList[j]}"]

    customisedTitleOptionsQuantities = []

    for (int k = 0; k < customisedTitleOptions.length; k++){ For each
customise option

```

```

        bool containedOption = false;

        for (int m = 0; m < customisedTitleOptionsQuantities.length;
m++){ For each item that is already in the list of quantities

            if (customisedTitleOptionsQuantities[m][0] ==
customisedTitleOptions[k]){

                customisedTitleOptionsQuantities[m][1] += 1;

                containedOption = true;

            }

        }

        if (containedOption == true){

            customisedTitleOptionsQuantities.add([customisedTitleO
ptions[k], [1]);

        }

    }

    for (int n = 0; n < customisedTitleOptionsQuantities.length n++){ When
the final options are in the quantities list, for each option

        customiseOptionInfo = await
QueryServer.query("https://alleat.cpur.net/query/cartiteminfo",
{"type": "option", "term": customisedTitleOptionsQuantities[n][0]});

        if (customiseOptionInfo["error"] == true){

            return {"Error": true};

        }

        itemInfo["${profileCart[i][1]}"]["${customisedOptionsList[j]}"].add(
[customiseOptionInfo["message"]][0],

```

```
        customiseOptionInfo["message"][1],  
        customisedTitleOptionsQuantities[n][1]]);  
  
    }  
  
}  
  
return itemInfo;  
}
```

```
Future<bool> removeFromCart (cartID) async{  
  
try{  
  
bool isSuccess = await SQLiteCartItems.removeItem(cartID);  
  
if (isSuccess){  
  
    return true;  
  
} else{  
  
    return false;  
}  
  
} catch (e){  
  
    return false  
}  
  
}
```

```
Futurebuilder<List>{

    future: getAvailableProfiles,

    builder: ((context, snapshot) {

        if (snapshot.hasData) {

            List availableProfiles = snapshot.data

            for (int i = 0; i < availableProfiles.length; i++) { For each profile in
the cart

                Column( children: [

                    Row(children: [

                        CircleAvatar(


                            backgroundcolor: Color.fromRGBO(availableProfiles[i][2],
                            availableProfiles[i][3], availableProfiles[i][4], 1),

                            child: Text(availableProfiles[i][1].substring(0,1)

                        ),


                        Text(availableProfiles[i][1])

                    ]),
                ],
            FutureBuilder<List>{

                future: getProfileCart(availableProfiles[i][0]),

                builder: ((context, snapshot) {

                    if (snapshot.hasData) {

                        List profileCart = snapshot.data
```

```
        for (int j = 0; j < profileCart.length; j++)  
async { For each item in the cart for each profile  
        List itemInfo = await  
        getItemInfo(profileCart[j][0]);  
  
        Dismissible(  
            key: Key(profileCart[j][0]),  
            onDismissed(direction) async {  
                setState() {  
                    isSuccess = await  
                    removeFromCart(profileCar  
t[j][0]);  
  
                    if (isSuccess){  
                        ScaffoldMes  
senger.of(context).showSna  
ckBar(SnackBar(  
                            content:  
                            Text(  
                                "Successfully  
removed item"))  
  
                    else{  
                        ScaffoldMes  
senger.of(context).showSna  
ckBar(SnackBar(  
                            content:  
                            Text(  
                                "Error: Item  
not found or  
removed"))  
                }  
            }  
        }  
    }  
}
```

```
        "Failed to remove
        item"))
    }

}

}

background: Container(
    color: Theme.of
    (context).colorScheme.error)

    Child: Container(child: Column(children:
    [Row(children: [
        Container(
            width: 70,
            height: 70,
            decoration:
            BoxDecoration(
                image:
                DecorationImage(
                    fit: BoxFit.cover,
                    image:
                    NetworkImage(itemInfo["${profilec
                    art[i][0]}"])[0][1]),
            ),
        )),
        Text(profileCart[i][3], style:
        Theme.of(context).textTheme.head
```

```
line6?.copyWith(color:  
Theme.of(context).primaryColor))  
  
Column(children:[  
  
Text(itemInfo["${profilecart[i][0]}"][  
0][0], style:  
Theme.of(context).textTheme.head  
line5),  
  
Text(itemInfo["${profilecart[i][0]}"][  
0][3], style:  
Theme.of(context).textTheme.head  
line6.withOpacity(0.5)),  
  
]),]),  
  
LayoutBuilder(builder: (){
```

```
customisedOptionsKeys =  
itemInfo["${profilecart[i][0]}"][[1].ke  
ys.toList()  
  
if (customised[0][1] == "SELECT"){  
  
Listview.builder(  
  
count:  
customisedOptionsKeys.length  
  
Builder: (context, index){  
  
customised =  
customisedOptionsKeys["index"]  
  
Container(child: Row(children:[  
  
Text(customised[0][0]),
```

```

ListView.builder(
  count: customised[1].length
  builder: (context, index){
    Row(children:[
      Text(customised[1][index][0]),
      Text(" x "
        "${customised[1][index][2]}"))
    ],
  },
),

```

Variable Info

Variable Name	Explanation
availableProfiles	Available profiles variable is first set to be an empty list so that if there are no profiles in the cart, it will return an empty list
cartProfileIDs	This gets the ids of all the profiles in the cart. This is done in the cart service dart file which looks at the table of items and adds any unique profile ids to a list and returns them. For each item in this list, it is validated to check if it is the same as the any of the profiles in profile Info so that it can save the profile info associated with it.
profileInfo	This gets all the profiles currently logged in on the device and returns back the information about them. This uses the profile service dart file.
itemInfo	The item info variable is first set to be an empty dictionary/map. For each item id in the cart, it gets the item information including the customise titles and the options related to it compiled. Each item id is a key, having a unique id set by the primary key in the server's item database. This information is used to display the item information as a single widget.

profileCart	The profile cart variable is used to store a copy of the data stored in the local cart regarding the profile inputted into the Future function. This information includes the cartid, itemid, customised information and quantity.
basicItemInfo	The basic item info grabs the item info for "i" where "i" is the index for the item id. This is queried on the server and the relevant information is returned (Item Name, Item Image, Original Price, Restaurant). The basicItemInfo is validated to check if it contains an error and if it does, it sends the error information back and ends the search.
customisedOptions	The customisedOptions decodes the customised information for an item that was stored in the cart table stored on the client. The length is checked and for each title, it runs a loop to get information about the title id
customisedOptionsList	For each option that has been customised for the currently specified title, it is added to the list.
customiseTitleInfo	The server is queried for the info related to the customise title id. This is checked for errors and if there is an error, it returns the error information back.
customisedTitleOptions	This variable is used to store the option ids related to the customise title of the item.
customisedTitleOptionsQuantities	This stores the quantities for each option selected (option id, quantity). The program runs through the decoded customised options and finds the number of times mentioned and converts it into a quantity
containedOption	If the currently checked option id is already in the list, 1 is added to the quantity. By default it is set to false and if it is in it, it set to true.
customiseOptionInfo	The information about the currently selected option is grabbed from the server, getting the name and price change. It is checked to see if there is an error and if so, the error information is returned
isSuccess	Try to remove item from the cart table using the cart service dart file stored on the client. If it is removed successfully, set to true

Due to the limited time of the project, I was unfortunately unable to make it more optimised so at the current time it is a big O of n^4 which means with a large enough size, it could take a long time to load the page.

When the user clicks the save button, it will check to see if there is only one restaurant for all restaurant IDs and if there is, it will port over the price of the food and the delivery fee to the next screen. Since there are no cart summaries being shown, the data can be discarded from the variable.

Checkout

The checkout will use a static map showing both the restaurant and the delivery destination. This will be grabbed from the first item in the cart. Since the previous save button checked if they are all the same, there is no need to recheck. The restaurant info will then be gotten from the server and the latitude and longitude will be set as a marker as well as the destination stored in the shared preferences.

To do the tipping, I will use a TextField connected to the variable which will only take numbers and decimals.

There will be a button which leads to a different file which then takes the details and sends them to the server. Since I do not have enough time, I will not be allowing the user to pay with their card at the moment but if this were to be a real app, it is a feature I would implement.

Saving cart data

The following data will be saved in the server database

Order Information

OrderID	INC PRIMARY INT
RestaurantID	FOREIGN INT
Tip	DOUBLE (2,2)

Status	TEXT
DateOrdered	TIMESTAMP

Order Item Information

OrderItemID	INC PRIMARY INT
OrderID	FOREIGN INT
ProfileID	FOREIGN INT
ItemID	FOREIGN INT
Quantity	INT

Customised Item Information

CustomisedItemID	INC PRIMARY INT
OrderItemID	FOREIGN INT
CustomiseID	FOREIGN INT
Quantity	INT

orderslist.php

```
<?php  
  
$db = "dbu35...";
```

```

$dbuser = "username";

$dbpassword = "password123";

$dbhost = "localhost";

$return["error"] = false;

$return["message"] = "";

$itemlist = [];

$orderidlist = [];

$orderinfo = [];

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileid"]);

if($val){

    $profileid = $_POST["profileid"];

    $sql = "SELECT orderitem.orderid, item.itemname, item.itemimage FROM
orderitem INNER JOIN item ON orderitem.itemid=item.itemid WHERE orderitem.profileid =
$profileid";

    if($result = mysqli_query($link, $sql)) {

        while($row = mysqli_fetch_array($result)) {

            array_push($itemlist, [$row["orderitem.orderid"],
$row["item.itemname"], $row["item.itemimage"]]);
        }
    }
}

```

```

    }

    mysqli_free_result($result);

}

else {

    $return["error"] = true;

    $return ["message"] = "Failed to find order information";

}

for ($i = 0; $i < $itemlist.length(); $i++){

    if (!$orderidlist.contains($itemlist[$i][0])){

        $orderidlist.add($itemlist[$i][0]);

    }

}

$sql2 = "SELECT order.orderid, restaurant.deliveryprice, order.tip, order.status,
order.dateordered FROM orderinformation order INNER JOIN restaurant ON
order.restaurantid = restaurant.restaurantid WHERE order.orderid IN $orderidlist";

if($result2 = mysqli_query($link, $sql)) {

    while($row2 = mysqli_fetch_array($result)) {

        array_push($orderinfo, [$row2["order.orderid"],
$row2["restaurant.deliveryprice"], $row2["order.tip"], $row2["order.status"],
$row2["order.dateordered"], []]);

    }

}

else{

```

```

$return[“error”] = true;

$return [“message”] = “Failed to find order information”;

}

for ($i = 0; $i < $itemlist.length(); $i++){

    for ($j = 0; $j < $orderinfo.length(); $j++)

        if (itemlist[i][0] == orderinfo[j][0]){

            orderinfo[j][4].add(itemlist[i]);

        }

    }

}

$return[“message”] = $orderinfo

} else {

    $return[“error”] = true;

    $return [“message”] = “No profiles selected. Unable to preview order
information”;

}

mysqli_close($link);

header('Content-Type: application/json');

echo json_encode($return);

?>

```

Data Validation

\$val validates if the profile ID has been identified. This is important to ensure that the file is able to search the file for the profile and doesn't result in an error. If the client fails to send a profile id, it will return the error that not enough data has been sent to the server. This also prevents direct access to the server since even if a user knows the URL, they will need to know the correct profile id.

\$sql and \$sql2 has validation. The server will try to execute the SQL statement and save them in \$result and \$result2. If it is null, it returns to the client the error that it failed to query the server. This ensures that if the query fails, code related to it is not run causing the client to wait forever as the server tries to query the resulting information that doesn't exist.

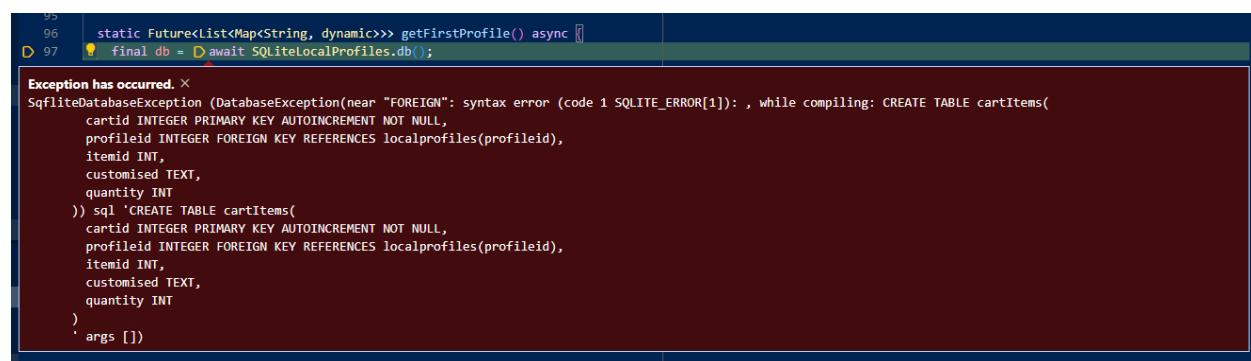
Development

Modifications to saving cart

In order to get the profile which it is saved under. To do this the first thing to do is to modify the cart table to include the profile id as a new field and make it a foreign key if ever there is to be references using the cart data or vice versa.

SQLite Modifications

While adding the SQL statement to the cart dart file, I encountered this error.



```
95
96     static Future<List<Map<String, dynamic>>> getFirstProfile() async {
97       final db = await SQLiteLocalProfiles.db();
Exception has occurred. ×
SqfliteDatabaseException (DatabaseException(near "FOREIGN": syntax error (code 1 SQLITE_ERROR[1]): , while compiling: CREATE TABLE cartItems(
    cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    profileid INTEGER FOREIGN KEY REFERENCES localprofiles(profileid),
    itemid INT,
    customised TEXT,
    quantity INT
)) sql 'CREATE TABLE cartItems(
    cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    profileid INTEGER FOREIGN KEY REFERENCES localprofiles(profileid),
    itemid INT,
    customised TEXT,
    quantity INT
)
args []
```

The code inputted was

```
await database.execute("""CREATE TABLE cartItems(
```

```
        cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
  
        profileid INTEGER FOREIGN KEY REFERENCES localprofiles(profileid),  
  
        itemid INT,  
  
        customised TEXT,  
  
        quantity INT  
  
)  
  
"""");
```

To fix this, I decided to do the foreign key references at the end of the statement.

```
await database.execute("""CREATE TABLE cartItems(  
  
        cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
  
        profileid INTEGER,  
  
        itemid INT,  
  
        customised TEXT,  
  
        quantity INT,  
  
        FOREIGN KEY (profileid) REFERENCES localprofiles(profileid)  
  
)  
  
""");
```

This seemed to fix the error so that the app would not crash when it was being created.

The new SQL Future is

```
static Future<bool> addToCart(  
  
        int itemid, dynamic customised, int quantity) async {
```

```

try {

    final db = await SQLiteCartItems.cartdb();

    final prefs = await SharedPreferences.getInstance();

    final String? profileid = prefs.getString('serverprofileid');

    if (profileid == null) {

        return false;

    }

    int profileidInt = int.parse(profileid);

    db.insert("cartItems", {

        "profileid": profileidInt,

        "itemid": itemid,

        "customised": customised,

        "quantity": quantity,

    });

    return true;

} catch (e) {

    return false;

}

}

```

On the customise page, the following was modified

```

Future<bool> addToCart() async { //Add item with cutomsie options to db

    try {

```

```

    String customisedOptionsEncoded = json.encode(customisedOptions);
    //Encode Dictionary into a string

    bool isAdded = await SQLiteCartItems.addToCart(
        int.parse(widget.itemid), customisedOptionsEncoded, quantity);

    if (isAdded) {

        return true;

    }

    else{

        return true; //If no error, return true

    }

} catch (e) {

    return false; //If error, return false

}

}

```

Creating the Cart

Using the design phase as a basis for the cart, I first made it so that the homepage cart button does the cart and displays a cart title.

```

import 'package:alleat/widgets/elements/elements.dart';

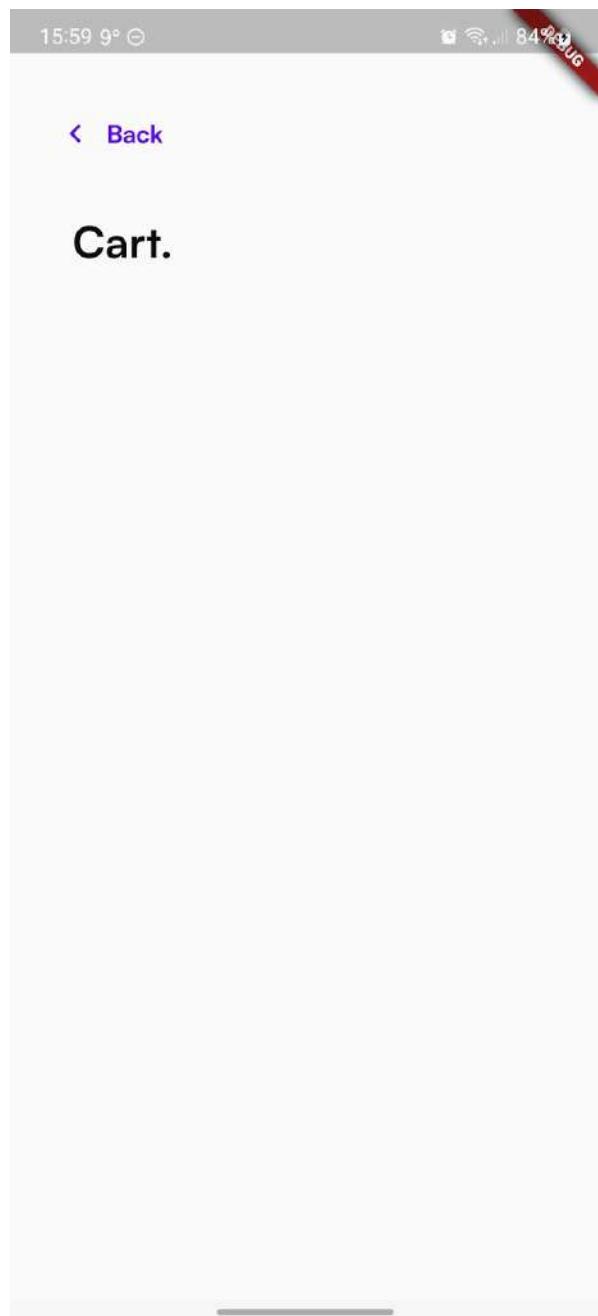
import 'package:flutter/material.dart';

class Cart extends StatefulWidget {

```



```
    }  
}
```



Getting Cart Profiles

The next thing to do is to display the profiles which are in the cart. To do this, I first need to get the items in the cart and create a list of profiles to get their details. A new query function is needed to do this which outputs a list containing the profiles in the cart

After adding the following to the app, what occurred was that from the cart, it was grabbing the profiles from the cart table but the final result would be null. Since there was no inherit error stopping the app from working, this meant that one of the checks was not working as intended.

```
static Future<List> getProfilesInCart() async {

    final db = await SQLiteCartItems.cartdb();

    List profilesInCart = await db.rawQuery("SELECT profileid FROM
cartItems");

    List singleProfilesInCart = [];

    print(profilesInCart);

    for (int i = 0; i < profilesInCart.length; i++) {

        if (singleProfilesInCart.contains(profilesInCart[i])) {

            singleProfilesInCart.add(profilesInCart[i]);

        }
    }

    return singleProfilesInCart;
}
```

```
class _CartState extends State<Cart> {

Future<List> getAvailableProfiles() async {

    List availableProfiles = [];

    List cartProfileIDs = await SQLiteCartItems.getProfilesInCart();
```

```

List profileInfo = await SQLiteLocalProfiles.getProfiles();

for (int i = 0; i < cartProfileIDs.length; i++) {

    if (profileInfo[i][0] == cartProfileIDs[i]) {

        availableProfiles.add([
            profileInfo[i][0],
            profileInfo[i][1],
            profileInfo[i][5],
            profileInfo[i][6],
            profileInfo[i][7]
        ]);
    }
}

return availableProfiles;
}

...

```

After some debugging, it seems that `cartProfileIDs` was receiving nothing. This is not correct so it therefore means that the cart service was not sending anything. As it turns out, it was checking for `{profileid: xx}` not `xx`. As well, other references was using an integer index instead of the key within the map.

```

static Future<List> getProfilesInCart() async {

    final db = await SQLiteCartItems.cartdb();

    List profilesInCart = await db.rawQuery("SELECT profileid FROM
    cartItems");

```

```

List singleProfilesInCart = [];

for (int i = 0; i < profilesInCart.length; i++) {

    if (!singleProfilesInCart.contains(profilesInCart[i]["profileid"]))
    {

        singleProfilesInCart.add(profilesInCart[i]["profileid"]);
    }
}

return singleProfilesInCart;
}

```

```

Future<List> getAvailableProfiles() async {

    List availableProfiles = [];

    List cartProfileIDs = await SQLiteCartItems.getProfilesInCart();

    List profileInfo = await SQLiteLocalProfiles.getProfiles();

    print(profileInfo);

    for (int i = 0; i < cartProfileIDs.length; i++) {

        if (profileInfo[i]["profileid"] == cartProfileIDs[i]) {

            availableProfiles.add([
                profileInfo[i]["profileid"],
                profileInfo[i]["firstname"],
                profileInfo[i]["lastname"],
                profileInfo[i]["profilecolorred"],
                profileInfo[i]["profilecolorgreen"],

```

```

        profileInfo[i] ["profilecolorblue"]

    ) ;

}

}

return availableProfiles;

}

```

Displaying Cart Profiles

To display the profiles, I will use a for a for statement to display each profile using the grabbed first name and lastname and display the profile icon with the generated profile colour associated on the device.

By using the code done for the profile selection file, I was able to easily create the profile icons simply by changing the heading size and circle radius.

```

ListView.builder(
    physics:
        const NeverScrollableScrollPhysics(), //Dont
allow scrolling (Done by main page)

    scrollDirection: Axis.vertical,
    shrinkWrap: true,
    itemCount:
        availableProfiles.length, //For each profile
    itemBuilder: (context, index) {
        return Container(
            margin: const EdgeInsets.symmetric(

```

```
        vertical: 10, horizontal: 20),  
  
        padding: const EdgeInsets.symmetric(  
  
            horizontal: 20, vertical: 20),  
  
        decoration: BoxDecoration(  
  
            color:  
  
Theme.of(context).colorScheme.onSurface,  
  
            borderRadius: const BorderRadius.all(  
  
                Radius.circular(20))),  
  
            child: Row(  
  
                children: [  
  
                    Container(  
  
                        // Create profile circle with  
                        first and last letter  
  
                        width: 50,  
  
                        height: 50,  
  
                        decoration: BoxDecoration(  
  
                            shape: BoxShape.circle,  
  
                            color: Color.fromRGBO(  
  
availableProfiles[index][3],  
  
availableProfiles[index][4],  
  
availableProfiles[index][5],
```

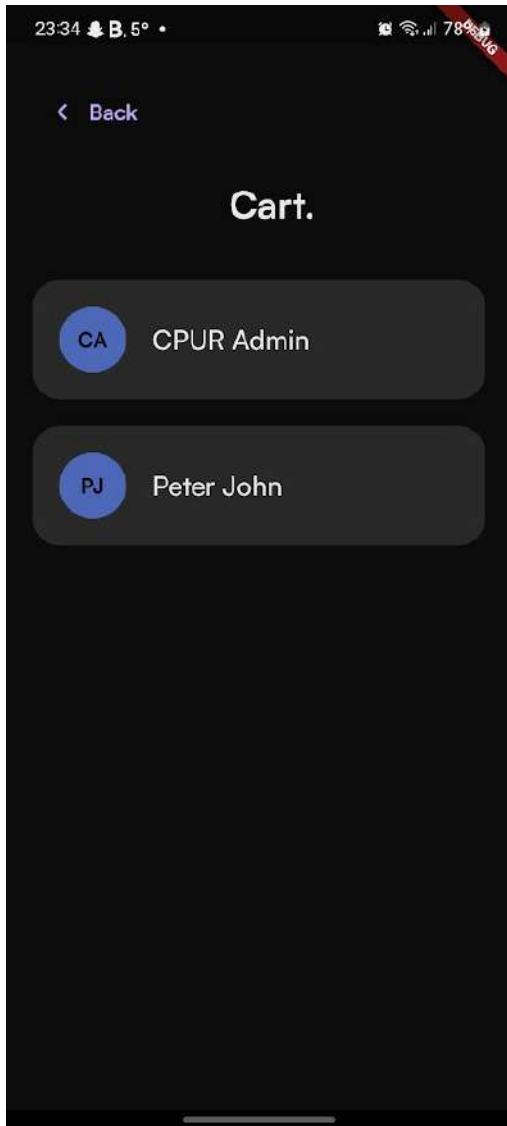
```
        1)),

        child: Align(
            alignment: Alignment.center,
            child: Text(
                '${availableProfiles[index][1][0]}${availableProfiles[index][2][0]}',
                style: Theme.of(context)
                    .textTheme
                    .headline6
                    ?.copyWith(
                        color:
                    Theme.of(context)

                .backgroundColor))),

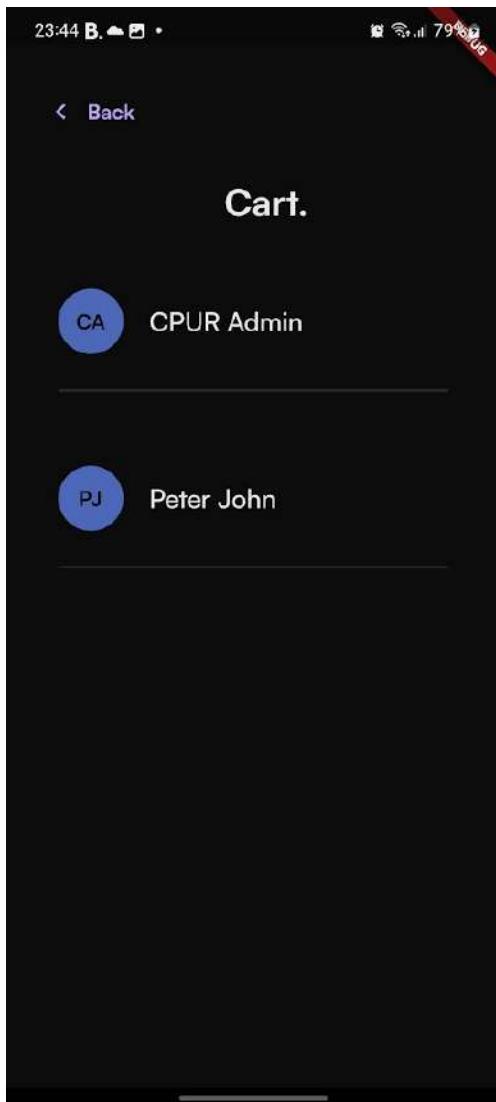
        const SizedBox(width: 20),
        Text(
            "${availableProfiles[index][1]}${availableProfiles[index][2]}",
            style: Theme.of(context)
                .textTheme
                .headline5
                ?.copyWith(
                    color: Theme.of(context)
                        .textTheme
```

```
.headline1  
    ?.color),  
    )  
],  
)) ;  
})
```



After looking at the concept art, I found that the container was not around the profile but instead around each item.

By making the container not have a colour, I was able to make it hold the same properties and keep it contained. I then added a Divider to separate it from each item.



After rebuilding the app and attempting to run the code, it resulted in no profiles appearing. After testing the following statement, it was found that because it was only checking the place of each with the other (eg. index 1 of list 1 with index 1 of list 2), if the second profile added an item and then the first, it would show nothing.

```
for (int i = 0; i < cartProfileIDs.length; i++) {  
    if (profileInfo[i]["profileid"] == cartProfileIDs[i]) {
```

To fix this, I just need to replace "equal to" to "contains".

```
for (int i = 0; i < cartProfileIDs.length; i++) {  
    if (cartProfileIDs.contains(profileInfo[i]["profileid"])) {
```

Getting Item Data

To get the item data, the program should first check the cart for each profile for items in the cart. It should then send the item id and the customised options to the server.

To do this, a function should be created in the cart service file, using the profile id as the input and output of a 3D array containing the item id, quantity and the customised options.

The first thing to do is to make the main structure that will call the item future function. This will ensure that during testing, the correct data is returned. The FutureBuilder is added after the divider.

```
FutureBuilder<List>(  
  
    future: getProfileCart(  
        availableProfiles[index][0]),  
  
    builder: (context, snapshot) {  
  
        if (snapshot.hasData) {  
  
            List profileCart = snapshot.data ??  
[];  
  
            return Text(profileCart.toString())  
);  
  
        }  
    } ),
```

The profileCart Future function first gets a list of items from the local database.

```
Future<void> getProfileCart(profileID) async {

    Map itemInfo = {};

    List profileCart = await SQLiteCartItems.getProfileCart(profileID);

    print(profileCart);

}
```

```
// Get a list of items that are under a profile ID

static Future<List> getProfileCart(profileID) async {

    final db = await SQLiteCartItems.cartdb();

    List itemsInCart = await db.rawQuery(
        "SELECT itemid, customised, quantity FROM cartItems WHERE
profileid = $profileID");

    return itemsInCart;

}
```

```
:[{"itemid": 22, "customised": {"20": ["229"], "21": ["231"], "22": ["236"]}, "quantity": 1}
, [{"itemid": 3, "customised": {"7": ["88"], "6": [], "9": ["101"]}, "quantity": 1}, {"itemid": 22, "customised": {"20": ["228"], "21": ["230"], "22": ["235"]}, "quantity": 1}]]
```

Doing this returned back all the information needed. To get the information from the ids, I will use a for loop to get the item information for each item

```
for (int i = 0; i < profileCart.length; i++) {

    Map basicItemInfo = await QueryServer.query(
```

```

        "https://alleat.cpur.net/query/cartiteminfo",
        {"type": "item", "term": profileCart[i][0]}));
    }
}

```

On the server, I will create the cart file to grab all the relevant information depending on what needs to grab. By using “type”, it reduces the amount of repeated code needed and compiles it all into one file.

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "FO7&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["type"]) && ($_POST["term"]);

if($val){
    $type = $_POST["type"];
    $term = $_POST["term"];
    if ($type == "item"){ //Get item information for the cart using item id
        $sqlitem = "SELECT item.itemid, item.itemname, item.price,
item.itemimage, restaruant.restaurantname FROM item JOIN menucategories ON
item.categoryid = menucategories.categoryid JOIN restaurant ON
menucategories.restaurantid = restaurant.restaurantid WHERE item.itemid =
$term";

        if($result = mysqli_query($link, $sqlitem)){
            $iteminfo = [];
            while($row = mysqli_fetch_assoc($result)) {
                array_push($iteminfo, $row["item.itemid"],
$row["item.itemname"], $row["item.price"], $row["item.itemimage"],
$row["restaruant.restaurantname"]);
            }
        }
    }
}

```

```

        $return["message"] = $iteminfo;

    } else {
        $return["error"] = true;
        $return["message"] = "Unknown item";
    }
} else {
    $return["error"] = true;
    $return["message"] = "Query type unknown";
}
}
else{$return["error"] = true;
      $return["message"] = "data not specified";}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

When testing the code, I got an error relating to how a variable was being called.

44 | "https://alleat.cpur.net/query/cartiteminfo",
D 45 | {"type": "item", "term": profileCart[i][0]});
Exception has occurred. ×
`_CastError (type 'int' is not a subtype of type 'String?' in type cast)`

While developing the pseudocode, I forgot that the data is declared in a dictionary form so to fix this, I replaced [0] with ["itemid"].

After fixing this, I got another error from the server

```
[{"itemid": 3, "customised": {"7": 88, "8": 0, "9": 101}, "quantity": 1}, {"itemid": 22, "customised": {"2": 0}}: {error: true, message: ERROR: type 'int' is not a subtype of type 'String' in type cast}
```

This issue was relating to how key "term" was sending the variable as an integer instead of a string.

```
: {error: true, message: Error 404}
```

I then got an error 40 since it was meant to go to the URL

<https://alleat.cpur.net/query/cartiteminfo.php> not <https://alleat.cpur.net/query/cartiteminfo>.

```
: {error: true, message: Error 500}
```

I then got an error 500. To debug, I decided that it might be that the SQL may have errors so I decided to just get the item information. This resulted in only null values returning meaning that no valid terms was returning

```
{error: false, message: [null, null, null, null]}}
```

To fix this, I thought it might be that it is not querying an integer using a string so the string was parsed to be an integer. This still resulted in null being returned, so I did some alteration to debug a variety of other things. As it turned out, the item part of each variable was breaking it.

After lots of debugging, I found that the problem was that you do not need to use the prefix for the name of the table when referencing it in the appending.

```
array_push($iteminfo, $row["itemid"], $row["itemname"], $row["price"],  
$row["itemimage"], $row["restaurantname"]);
```

```
{error: false, message: [3, BBQ Meat Feast, 17.99, https://alleat.cpur.net/assets/images/restaurant_item/bbqmeatfeast.webp, Papa John's Pizza], debug: 3}  
get {error: false, message: [22, Piccolo to Go, 7.25, https://alleat.cpur.net/assets/images/restaurant_item/piccolotogo.webp, PizzaExpress], debug: 22}}
```

In order to get a result if there was no items inputted, a default error value was set and returned when the for loop was not entered.

```
Future<Map> getProfileCart(profileID) async {  
  
    Map itemInfo = {};  
  
    List profileCart = await SQLiteCartItems.getProfileCart(profileID);  
  
    print(profileCart);  
  
    Map basicItemInfo = {"error": "true", "message": "No items found"}; //If  
there are no items, the default is an error
```

```

        for (int i = 0; i < profileCart.length; i++) { //For each item, return the
item info from server.

        basicItemInfo = await QueryServer.query(
            "https://alleat.cpur.net/query/cartiteminfo.php",
            {"type": "item", "term": profileCart[i]["itemid"].toString()});
    }

    return basicItemInfo;
}

```

While looking at the design stage, I found that the delivery fee and restaurant ID was something that was also returned back from the server so that was added to the SQL query.

An issue I faced which I did not consider in the design stage was error checking. The pseudocode written then would only work if there was no error from the server.

To fix this, what I did was that if there was an error during any of the cycles of the if loops, it would immediately return a map with error and the message. If there was no errors, after each cycle, it would dump the data for each item into itemInfo and after another cycle, it would move the data to the basicItemInfo message.

```

for (int i = 0; i <= profileCart.length; i++) {

    if (i == profileCart.length) {

        basicItemInfo["message"] = profileCart;

    }
}

```

To test my code, I used an if statement

```
FutureBuilder<Map>(
```

```
future: getProfileCart(  
    availableProfiles[index][0]),  
  
builder: (context, snapshot) {  
  
    if (snapshot.hasData) {  
  
        List profileCart = [snapshot.data ?? []];  
  
        if (profileCart[0]["error"] == "true") {  
  
            return Column(children: [  
  
                Text("ERROR"),  
  
                Text(profileCart[0]["message"]  
                    .toString())  
  
            ]);  
  
        } else {  
  
            return Column(children: [  
  
                Text("SUCCESS"),  
  
                Text(profileCart[0]["message"]  
                    .toString())  
  
            ]);  
  
        }  
  
    } else {  
  
        return LinearProgressIndicator(  
            color: Theme.of(context).primaryColor,  
            backgroundColor: const Color.fromARGB(  
                255, 255, 255, 255),  
            value: 0.5);  
  
    }  
}
```

```
    0, 235, 224, 255));  
}  
});
```

This resulted in nothing being returned.

As it turned out, the if statement was failing under the terms. To fix this, I ran a ".runtimeType" to get the type of each of the values and they were both booleans so I made the if statements use booleans instead of string values.

This made the if statement result in true but null was once again being returned for the values.

```
): {error: false, message: {null: [[null, null, null, null, null, null], {}]}}
```

As it turned out, it was getting only the last item in the cart and then returning null for the next

```
): success  
): {error: false, message: {error: false, message: [29, Calzone Nduja, 14.95, https://alleat.cpur.net/assets/images/restaurant_item/calzonem...  
): {error: false, message: {null: [[null, null, null, null, null, null], {}]}}
```

Due to the amount of errors, I decided that instead of trying to fix the previous code, I would remake the getProfileCart function. I first made it get the item ids and add them to a temporary item info map.

```
Future<Map> getProfileCart(profileID) async {  
  
    Map returnItemList = {"error": false, "message": "", "iteminfo": {}};  
  
    Map tempItemInfo = {};  
  
    List profileCart = await SQLiteCartItem.getProfileCart(  
        profileID); // {itemid, customised, quantity}  
  
    print(profileCart);  
  
    for (int i = 0; i < profileCart.length; i++) {
```

```

        tempItemInfo[profileCart[i]["itemid"]] = [[], {}];

    }

    returnItemList["iteminfo"] = tempItemInfo;

    return returnItemList;

}

```

The next thing I did was adding the item data grab from the server

```

for (int i = 0; i < profileCart.length; i++) {

    Map basicItemInfo = await QueryServer.query(
        "https://alleat.cpur.net/query/cartiteminfo.php",
        {"type": "item", "term": profileCart[i]["itemid"].toString()});

    if (basicItemInfo["error"] == true) {

        returnItemList["error"] = true;

        returnItemList["message"] = basicItemInfo["message"];

        return returnItemList;
    }
}

```

The next thing I did was adding it to a list

```

for (int i = 0; i < profileCart.length; i++) {

    Map basicItemInfo = await QueryServer.query(
        "https://alleat.cpur.net/query/cartiteminfo.php",
        {"type": "item", "term": profileCart[i]["itemid"].toString()});
}

```

```

        {"type": "item", "term": profileCart[i]["itemid"].toString()}));

    if (basicItemInfo["error"] == true) {

        returnItemList["error"] = true;

        returnItemList["message"] = basicItemInfo["message"];

        return returnItemList;

    } else {

        print(basicItemInfo["message"]["message"]);

        tempItemInfo[profileCart[i]["itemid"]][0].addAll([
            basicItemInfo["message"]["message"][1],
            basicItemInfo["message"]["message"][3],
            basicItemInfo["message"]["message"][2],
            basicItemInfo["message"]["message"][5],
            basicItemInfo["message"]["message"][4],
            basicItemInfo["message"]["message"][6]
        ]);

    }

}

```

Now that we were back to where we were before, the next thing to do was to decode the customised options and get the info about it. While before I was sending each customise option individually, I decided that in order to reduce network traffic, I would send the customise ids and the customise option ids in only 2 requests per item. To do this for each item it needs to create a copy of each of the customised lists and separate them out then send 2 queries.

To start with, we separate them out using for statements

```

Map customised = json.decode(profileCart[i]["customised"]);

List customisedtitleids = customised.keys.toList(); //Each customise
title is stored here

print(customisedtitleids);

List customisedOptions = []; //Each unique option stored here

for (int i = 0; i < customisedtitleids.length; i++) { //For each
customise title

    for (int j = 0; j < customised[customisedtitleids[i]].length; j++) { //For each item in list of customised options for customise title

        if (!customisedOptions

            .contains(customised[customisedtitleids[i]][j])) { //If the
option is not in the list of options, add it to the list

            customisedOptions.add(customised[customisedtitleids[i]][j]);

        }

    }

}

```

In order to help the stage after which gets the quantity as well, I decided to make an updated customised map which will be added to the final list when the data has been imported for the customised titles and options.

While trying to add one, I found that one was not being added if it was in the list. The issue was that it was checking the whole array instead of just the first entry for the id.

```

for (int k = 0;

    k < updatedCustomised[customisedtitleids[i]][1].length;

    k++) {

```

```

        if (updatedCustomised[customisedtitleids[i]][1][k][0] ==
            customised[customisedtitleids[i]][j]) {
            updatedCustomised[customisedtitleids[i]][1][k][1] += 1;
        }
    }
}

```

```

: {[26: [[], [[258, 1]]], 35: [[], [[297, 3]]]], 27: [[], [[265, 1]]]}

```

The next thing was to get the data from the server so a query was made using the QueryServer dart file.

```

Map CustomisedTitlesInfo = await QueryServer.query(
    "https://alleat.cpur.net/query/cartiteminfo.php",
    {"type": "title", "term": customisedtitleids.toString()});
if (CustomisedTitlesInfo["error"] == true) {
    returnItemList["error"] = true;
    returnItemList["message"] = CustomisedTitlesInfo["message"];
    return returnItemList;
} else {
    print(CustomisedTitlesInfo);
    Map CustomisedOptionInfo = await QueryServer.query(
        "https://alleat.cpur.net/query/cartiteminfo.php",
        {"type": "option", "term": customisedOptions.toString()});
    if (CustomisedOptionInfo["error"] == true) {

```

```

        returnItemList["error"] = true;

        returnItemList["message"] = CustomisedOptionInfo["message"];

        return returnItemList;

    } else {

        print(CustomisedOptionInfo);

        return returnItemList;

    }

}

```

As well, on the server, the title and option types needed to be made. These used the same stategory for sorting by favourites, converting the list to circle brackets from square brackets then running the titles and options through the filter.

After attempting to run the following, it resulted in nothing being returned but still having no errors.

```

} elseif ($type == "title"){
    $sqltitle = "SELECT customisename, optiontype FROM customiseitem
WHERE (" . implode(',', $term) . ")";

    if($result = mysqli_query($link, $sqltitle)){
        $titleinfo = [];
        while($row = mysqli_fetch_assoc($result)) {
            array_push($titleinfo, [$row["customisename"],
$row["optiontype"]]);
        }
        $return["message"] = $titleinfo;
    } else {
        $return["error"] = true;
        $return["message"] = "Failed to query customised item";
    }
}

```

To debug my code I removed the code `WHERE (" . implode(' , ', $term) . ")`; which after removing it, it returned all the correct values. This means the code inputted was incorrect.

The issue was later found to be that it was not converted from a string to an array. To fix this instead of sending the list as a string, it was encoded with json and decoded while on the server.

```
} elseif ($type == "title"){
    $term = json_decode($term);
    $return["debug"] = "(" . implode(' , ', $term) . ")";
    $sqltitle = "SELECT customisename, optiontype FROM customiseitem
WHERE customiseid IN (" . implode(' , ', $term) . ")";

    if($result = mysqli_query($link, $sqltitle)){
        $titleinfo = [];
        while($row = mysqli_fetch_assoc($result)) {
            array_push($titleinfo, [$row["customisename"],
$row["optiontype"]]);
        }
        $return["message"] = $titleinfo;
    } else {
        $return["error"] = true;
        $return["message"] = "Failed to query customised item";
    }
}
```

The next thing to do was to get the customise options data. For this, I copied the same code for the title but replaced it and used a slightly different SQL query in order to get the customised options.

```
} elseif ($type == "option"){
    $term = json_decode($term);
    $sqloption = "SELECT name, pricechange FROM customiseitemselection
WHERE optionid IN (" . implode(' , ', $term) . ")";

    if($result = mysqli_query($link, $sqloption)){
        $optioninfo = [];
        while($row = mysqli_fetch_assoc($result)) {
            array_push($optioninfo, [$row["name"],
$row["pricechange"]]);
        }
    }
}
```

```

        }
        $return["message"] = $optioninfo;
    } else {
        $return["error"] = true;
        $return["message"] = "Failed to query customised item";
    }
}

```

Since the data was still being copied with the error status as well, after verifying that there is no error, code was added to remove the extra metadata.

Before:

```

16: [], [[258, 1]], 35: [], [[298, 1], [303, 1], [314, 1]], 27: [], [[264, 1]]]
): {error: false, message: {error: false, debug: , message: [[Which Base?, SELECT], [Add a Side?,
): {error: false, message: {error: false, debug: , message: [[Romana Margherita, 1.95], [Classic Dough

```

After:

```

16: [], [[258, 1]], 35: [], [[298, 1], [303, 1], [314, 1]], 27: [], [[264, 1]]]
[[Which Base?, SELECT], [Add a Side?, ADD], [Would you like to add any extra toppings?, ADD]]
[[Romana Margherita, 1.95], [Classic Dough Balls, 4.50], [Mozzerella Cheese, 2.50], [Black Olives, 2.00], [Free-range Egg, 2.00]]

```

With this final piece of information grabbed from the server, the next thing to do is to compile the information into the returned variable.

While thinking about how to find the correct data grabbed, it was found that the best thing to do was to also get the server to return the id for each customised title and customised option.

To get the data to be formatted correctly, I had to use for statements to cycle through each array within the dictionary. I then had to cycle through each option for each title for each item in the dictionary in order to add the new information. If I were to remake this, I would have used separate Future functions and get each part separately instead of getting all the data and storing it in one array. The issue is that if there were enough profiles and enough unique customised items, an overflow can occur where it gets bigger than the assigned space for the variable.

```

for (int i = 0; i < customisedTitleIds.length; i++) {

    // For each customise title

    for (int j = 0; j < customisedTitlesInfoFormatted.length; j++) {

```

```

    //For each item in list of returned from server info about each
title

    if (customisedTitlesInfoFormatted[j][0] ==

        customisedtitleids[i]) {

            // If it has the id of the title in the first index, add the
info to the composite info

            updatedCustomised[customisedtitleids[i]][0].addAll([
                customisedTitlesInfoFormatted[j][1],
                customisedTitlesInfoFormatted[j][2]
            ]);

        }

    }

    for (int j = 0;

        j < updatedCustomised[customisedtitleids[i]][1].length;
        j++) {

        //For each option in the list of options for each title

        for (int k = 0; k < customisedOptionInfoFormatted.length; k++) {
//For each item in list of returned from server about option info

            if (customisedOptionInfoFormatted[k][0] ==

                updatedCustomised[customisedtitleids[i]][1][j][0]) { // If
it has the id of the option in the first index, add the info to the list

                updatedCustomised[customisedtitleids[i]][1][j].addAll([
                    customisedOptionInfoFormatted[k][1],
                    customisedOptionInfoFormatted[k][2]
                ]

```

```
]);
}

}

}

}
```

Since there can be issues when trying to sort through this data, I surrounded the statements with a try and catch which will return an error if anything occurs.

The final thing to do is move the Map to the final variable to be returned. To do this I added a single line of code:

```
returnItemList["iteminfo"] = tempItemInfo;
```

After testing this, I found that it overwrites the value instead of adding it to the item info.

The fixed code is as follows

```
tempItemInfo[profileCart[i]["itemid"]][1] = updatedCustomised;

returnItemList["iteminfo"] = tempItemInfo;

return returnItemList;
```

Displaying Item Data

To process this item data, I will use the pseudocode as a basis for how the data is represented since the returned data has a very similar format.

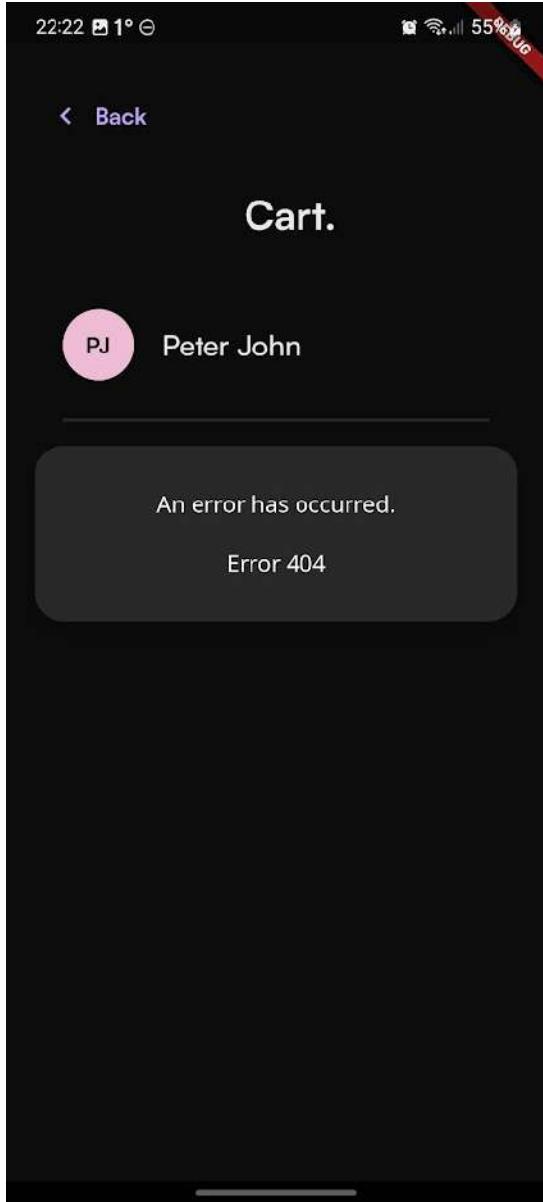
When there is an error returned, the following is returned

```
if (profileCart[0]["error"] == true) {  
  
    return Padding(  
  
        padding: const EdgeInsets.only(  
  
            left: 20,  
  
            right: 20,  
  
            top: 10,  
  
            bottom: 10),  
  
        child: Container(  
  
            decoration: BoxDecoration(  
  
                borderRadius:  
  
                const BorderRadius.all(  
  
                    Radius.circular(  
  
                        20)),  
  
            color: Theme.of(context)  
  
                .colorScheme  
  
                .onSurface,  
  
            boxShadow: [  
  
                BoxShadow(  
  
                    color: Colors.black  
  
                    .withOpacity(0.1),  
  
                    spreadRadius: 2,  
  
                    blurRadius: 10,  
            ]  
    );  
}  
});
```

```
        offset: const Offset(0,  
                          10), // changes  
position of shadow  
  
        ),  
  
    ],  
  
    child: Column(children: [  
  
        Padding(  
  
            padding:  
  
            const EdgeInsets.all(  
                30),  
  
            child: SizedBox(  
  
                width: double.infinity,  
  
                child: Center(  
  
                    child: Column(  
  
                        children: [  
  
                            const Text(  
  
                                "An error has  
occurred.",  
  
                                textAlign:  
  
                                TextAlign  
  
.center,  
  
                    ),  
  
                    const SizedBox(  
                ),  
            ),  
        ),  
    ],  
);
```

```
height: 20,  
),  
Text(  
profileCart[0]  
[ "message"],  
textAlign:  
TextAlign  
.center,  
(  
),  
]),  
)  
]);
```

To test this, I broke the URL to show what it would look like.



To display the success data, I will get the list of the keys and iterate through them using a listview builder and making each a slidable element.

To start with, I made it first just display the title in a dismissable container

```
List itemKeyValues = profileCart[0]  
        ["iteminfo"]  
        .keys  
        .toList();
```

```

        return ListView.builder(
            physics:
                const
            NeverScrollableScrollPhysics(), //Dont allow scrolling (Done by main page)
            scrollDirection: Axis.vertical,
            shrinkWrap: true,
            itemCount: itemKeyValues
                .length, //For each item
            itemBuilder: (context, index) {
                List currentItem = profileCart[0]
                    ["iteminfo"]
                    [itemKeyValues[index]];
                return Dismissible(
                    key: Key(itemKeyValues[index])
                        .toString(),
                    onDismissed: (direction) {
                        print("Dismissed");
                    },
                    child: Container(
                        child:
                            Text(currentItem[0][0]),
                    )));
    });

```

After testing this, it worked so I continued by adding formatting and the rest of the entries.

```
return ListView.builder(  
    physics:  
        const  
        NeverScrollableScrollPhysics(), //Dont allow scrolling (Done by main page)  
    scrollDirection: Axis.vertical,  
    shrinkWrap: true,  
    itemCount: itemKeyValues  
        .length, //For each item  
    itemBuilder: (context, index) {  
        List currentItem = profileCart[0]  
            ["iteminfo"]  
            [itemKeyValues[index]];  
        print(currentItem);  
        return Padding(  
            padding: const EdgeInsets  
                .symmetric(  
                    horizontal: 20,  
                    vertical: 20),  
            child: Dismissible(  
                key: Key(  
                    itemKeyValues[index]
```

```
        .toString()),

    onDismissed: (direction) {
        print("Dismissed");
    },
    background: Container(
        color:
            Theme.of(context)
                .colorScheme
                .error,
        child: Row(
            mainAxisAlignment:
                MainAxisAlignment
                    .spaceBetween,
            children: [
                Padding(
                    padding: const
                        EdgeInsets
                            .symmetric(
                                horizontal:
                                    30),
                child: Icon(
                    Icons
```

```
        .delete,  
  
        color:  
Theme.of(  
  
context)  
  
.colorScheme  
  
.onSurface,  
  
)),  
  
Padding(  
  
padding: const  
EdgeInsets  
  
.symmetric(  
  
horizontal:  
  
30),  
  
child: Icon(  
  
Icons  
  
.delete,  
  
color:  
Theme.of(  
  
context)  
  
.colorScheme
```

```
.onSurface,  
        ))  
    ],  
    ),  
    child: Container(  
        width: double.infinity,  
        decoration:  
BoxDecoration(  
        borderRadius:  
        const  
BorderRadius(  
        .all(  
        Radius  
  
.circular(  
        10)),  
        color: Theme.of(  
        context)  
        .colorScheme  
        .onSurface),  
        padding:  
        const EdgeInsets  
        .symmetric(  
        .
```

```
        vertical: 10,  
        horizontal: 10),  
  
        child: Row(  
  
            children: [  
  
                Container(  
  
                    width: 80,  
                    height: 80,  
                    decoration:  
BoxDecoration(  
  
        borderRadius:  
const BorderRadius  
  
.all(  
  
Radius.circular(  
  
        5)),  
  
        image:  
DecorationImage(  
  
        fit: BoxFit  
  
.cover,  
  
        image:  
NetworkImage(currentItem[0]  
  
        [  
  
        1]  
  
.toString()))),
```

```
        ),  
  
        const SizedBox(  
            width: 20,  
        ),  
  
        Container(  
            alignment:  
                Alignment  
                    .center,  
  
            height: 30,  
            width: 30,  
            decoration:  
BoxDecoration(  
                color:  
Theme.of(  
  
context)  
  
.primaryColor,  
                borderRadius:  
BorderRadius.circular(  
                    30)),  
  
            child: Text(  
                currentItem[0]
```

```
[4],  
    style:  
Theme.of(  
  
context)  
    .textTheme  
    .headline6  
    ?.copyWith(  
        color:  
Theme.of(context)  
  
.colorScheme  
  
.onSurface),  
    )),  
    const SizedBox(  
        width: 20,  
    ),  
    Column(  
        crossAxisAlignment:  
CrossAxisAlignment  
        .start,  
        children: [  
            Text(  

```

```
        currentItem[0]

        [0],

        style:

Theme.of(

context)

        .textTheme

        .headline6

        ?.copyWith(

            color:

Theme.of(context)

        .textTheme

        .headline1

        ?.color),

        ),

        Text(

        currentItem[0]

        [3],

        style:

Theme.of(

context)

        .textTheme
```

```
.bodyText1

    ?.copyWith(
        color:
Theme.of(context)

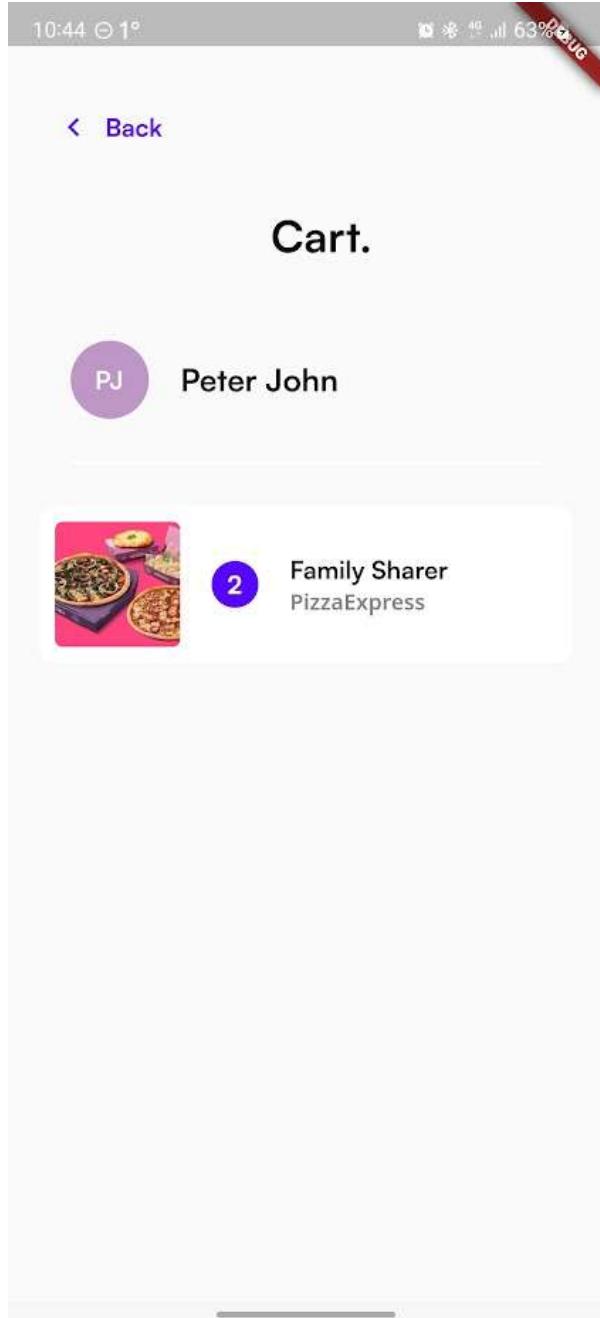
.textTheme

.headline6

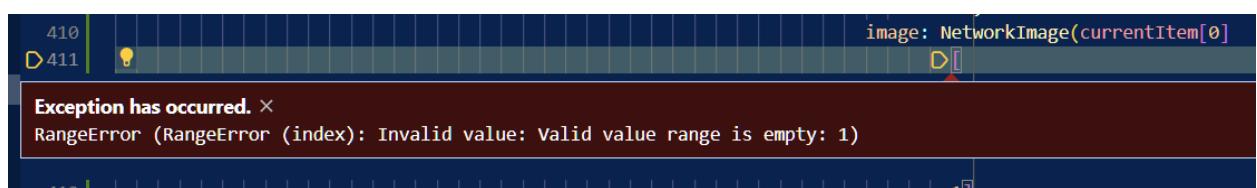
?.color),
    )
],
)
],
),
),
)));
});

}
} else {
    return LinearProgressIndicator(
        color: Theme.of(context).primaryColor,
        backgroundColor: const Color.fromARGB(
            0, 235, 224, 255));
}
}
```

```
        },  
        const SizedBox(  
          height: 30,  
        )  
      );  
    })
```



While testing it with more than one item in the cart for a profile, it resulted in an error occurring.



```
trace[MainActivity]( 505). viewPostime pointer 1  
: [[Family Sharer, https://alleat.cpur.net/assets/images/restaurant_item/familysharer.webp, 28.95, PizzaExpress],  
: [[American Hot, https://alleat.cpur.net/assets/images/restaurant_item/americanhot.webp, 15.99, Papa John's Pizza],  
: [[[], {}]]
```

it seems that it is not able to increment or get any information about the item.

After doing some debugging and formatting the returned variable, I found that the items were getting the keys but not the information about it

```
{2:  
    [  
        [American Hot, https://alleat.cpur.net/assets/images/restaurant_item/americanhot.webp, 15.99, Papa John's Pizza, 1, 2.50],  
    {4: [  
            [Please Choose a Pizza Type, SELECT],  
            [[45, 1, Small Original, 0.00]]  
        ],  
    5: [  
            [Defaults, REMOVE], []],  
    6: [[Optional, ADD], []]  
    }  
],  
31: [[], {}]}
```

This could be a result of a variety of things:

1. The data may not be added back to the variable after it is searched
2. There is not enough memory to add it to the variable
3. The items are not searched for each item in the cart
4. Other

To check for which one it is, I looked through the code for if it cycles through each item in the list. The result, it is not searched for each item in the cart.

To fix this, I went through the code to find the error. It seems that the code to return the data was being done after the first iteration, being inside the for loop.

While fixing this, I also found another error; if there was no customise options, it would crash because it would send nothing to the server. To fix this, I surrounded the customised section with an if statement which only uses the server if it is not null.

```
try {  
  
    if (customisedTitleIds.isNotEmpty) {  
  
        Map customisedTitlesInfo = await QueryServer.query(  
            "https://alleat.cpur.net/query/cartiteminfo.php",
```

```
{"type": "title", "term": json.encode(customisedtitleids)});  
  
if (customisedTitlesInfo["error"] == true) {  
  
    returnItemList["error"] = true;  
  
    returnItemList["message"] = customisedTitlesInfo["message"];  
  
    print("ERROR");  
  
    print(customisedtitleids);  
  
    return returnItemList;
```

After fixing this, I was able to display both multiple items and ones without any customise options. To do the customised options, I have to use a layout builder which will determine the customise type.

While trying to get the list of keys, it would tell me it was the wrong data type and after getting the data type, it gave me this:

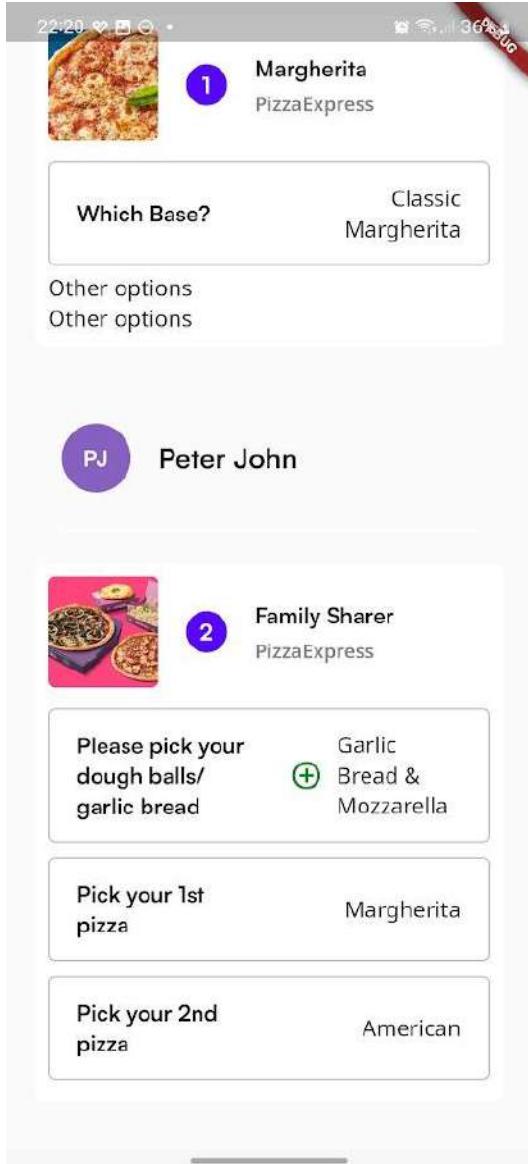
```
: _InternalLinkedHashMap<dynamic, dynamic>  
ibraries in 685ms (compile: 136 ms, reload: 194
```

To fix this, I did the following.

```
List CustomiseIDs = Map.from(currentItem[1]).keys.toList();
```

I started by working on the selection section. I used multiple listview blocks for first displaying for each customise option and then to get the list of options for the options that had any selections.

While making the views, I noticed that there was too much information to fit on a vertical small display meaning everything would get pushed to the sides of the screen.



To fix this, I made each option take their own line and fixed the formatting to also show the quantity

```
return Container(  
  padding:  
  const EdgeInsets.symmetric(vertical: 20, horizontal: 20),  
  margin:  
  const EdgeInsets.symmetric(horizontal: 0, vertical: 10),
```

```
decoration: BoxDecoration(  
  
    color: Theme.of(context).backgroundColor,  
  
    border: Border.all(  
  
        color: Theme.of(context).colorScheme.onBackground.withOpacity(0.3),  
  
        width: 1),  
  
    borderRadius: BorderRadius.circular(10)),  
  
    child:  
Column(  
  
    mainAxisAlignment: MainAxisAlignment.start,  
  
    crossAxisAlignment: CrossAxisAlignment.start,  
  
    children: [  
  
    Text(currentItem[1][customiseIDs[i]][0][0],  
  
        textAlign: TextAlign.start,  
  
        style: Theme.of(context).textTheme.headline6?.copyWith(  
  
            color: Theme.of(context).textTheme.headline1?.color)),
```

```
const SizedBox(height: 15),  
  
ListView.builder(  
  
physics: const NeverScrollableScrollPhysics(),  
  
shrinkWrap: true,  
  
itemCount: currentItem[1][customiseIDs[i]][1].length,  
  
itemBuilder: (context, index) {  
  
return Padding(  
  
padding: const EdgeInsets.only(left: 10),  
  
child: Row(children: [  
  
CircleAvatar(  
  
backgroundColor:  
  
Theme.of(context).primaryColor.withOpacity(0.5),  
  
radius: 10,  
  
child: Text(  
)
```

```
currentItem[1][customiseIDs[i]][1][index][1].toString(),

style: Theme.of(context).textTheme.bodyText1?.copyWith(

color: Theme.of(context).backgroundColor)),


),

const SizedBox(


width: 20,


),


Text(currentItem[1][customiseIDs[i]][1][index][2].toString(),


style: Theme.of(context).textTheme.bodyText1


]),


);


}),


]));
```

For the other options, they were displayed the same but with different text colours.

Removing Items

By using a Dismissible widget, I am able to use a future command to remove the item from the cart using the cart id stored in each dismissible widget key.

To start with I made the function in the cart_service dart file

```
//Remove item from cart

static Future<void> removeItem(cartID) async {
    final db = await SQLiteCartItems.cartdb();
    await db.rawDelete("DELETE FROM cartItems WHERE cartid = $cartID");
}
```

I then made the future on the cart

```
//Remove Item

Future<bool> removeItem(cartID) async {
    try {
        await SQLiteCartItems.removeItem(cartID);
        return true;
    } catch (e) {
        return false;
    }
}
```

And within the main program

```
onDismissed: (direction) async {  
    print(itemKeyValues[index]);  
  
    bool hasDeleted = await  
removeItem(itemKeyValues[index]);  
  
    if (hasDeleted == false) {  
  
        print("deleted");  
  
        setState() {  
  
ScaffoldMessenger.of(context)  
  
.showSnackBar(const  
SnackBar(content: Text("Successfully deleted item.")));  
  
    }  
  
} else {  
  
    print("Not deleted");  
  
    setState(() {  
  
ScaffoldMessenger.of(context).showSnackBar(  
  
        const  
SnackBar(content: Text("Failed to remove. Reopen cart and try again.")));  
  
    });  
  
}  
  
},
```

When running this, no snackbar is shown but no error is shown.

The reason it doesn't return false when it fails is because it is working but the wrong id is being sent to the remove function so when it filters, it removes nothing so there is then no error since everything works as intended. As it turns out, the key was not the cart id but instead the item id.

To fix this, I have to change the getProfileCart from the cart service to send the cart id

```
List profileCart = await SQLiteCartItems.getProfileCart(profileID); // {cartid,
itemid, customised, quantity}

for (int i = 0; i < profileCart.length; i++) {

    // For each item, add it as an index i

    tempItemInfo[profileCart[i]["itemid"]] = [[], {}];

    Map basicItemInfo = await
QueryServer.query("https://alleat.cpur.net/query/cartiteminfo.php", {

        "type": "item",

        "term": profileCart[i]["itemid"].toString()

}); //returns item id, item name, price, item image, restaurant id,
restaurant name, delivery price

if (basicItemInfo["error"] == true) {

    returnItemList["error"] = true;

    returnItemList["message"] = basicItemInfo["message"];

    return returnItemList;

} else {
```

```
tempItemInfo[profileCart[i]["itemid"]][0].addAll([
    basicItemInfo["message"]["message"][1],
    basicItemInfo["message"]["message"][3],
    basicItemInfo["message"]["message"][2],
    basicItemInfo["message"]["message"][5],
    basicItemInfo["message"]["message"][4],
    basicItemInfo["message"]["message"][6],
    profileCart[i]["quantity"],
    profileCart[i]["cartid"],
]);
```

In order to reduce the amount of changes, I put the cart id at the end so that no code would break as a result of the addition. The next thing to do is to change the keys for the dismissible widgets. To do this, I changed the key to currentItem[0][7]

After removing the item, I got an error that it was still part of the tree. To fix this, I added the line

```
profileCart[0]["iteminfo"].remove(itemKeyValues[index]);
```

Displaying price

To display the item price, I will create a variable for the total price then for each available profile, it will create a temporary list with the following details:

- Profile ID
- First name
- Last name
- Profile icon red

- Profile icon green
- Profile icon blue
- Total price of items
- Map of item details

For each item it will display the price of the item with the changes applied and then add the price to the total price for each profile

At the bottom of the page it will display a list of total prices for each profile and then display the subtotal price excl delivery and tips.

Displaying item price

To display the item price, I need to create a temporary price for each item and then alter it for each price.

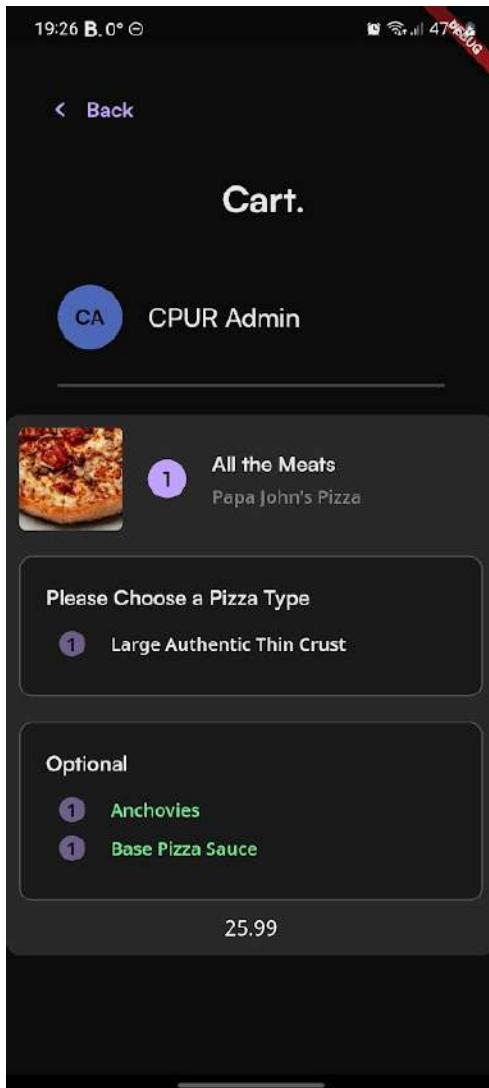
I started by getting the initial price of the item

```
double itemPrice = double.parseDouble(currentItem[0][2]);
```

For each change that is selected, the price is added to.

```
if (currentItem[1][customiseIDs[i]][0][1] == "SELECT" &&
    currentItem[1][customiseIDs[i]][1].isNotEmpty) {
    for (int s = 0; s < currentItem[1][customiseIDs[i]][1].length; s++) {
        itemPrice =
            ((itemPrice * 100) +
                double.parseDouble(currentItem[1][customiseIDs[i]][1][s][3]) * 100) /
                100;
    }
}
```

After learning my mistake when making the customise page, I remembered to multiply all values by 100 before dividing the total by 100. This prevented the numbers from breaking when calculating the price. I added a temporary number displayer to debug the code to see if it updates and it does.



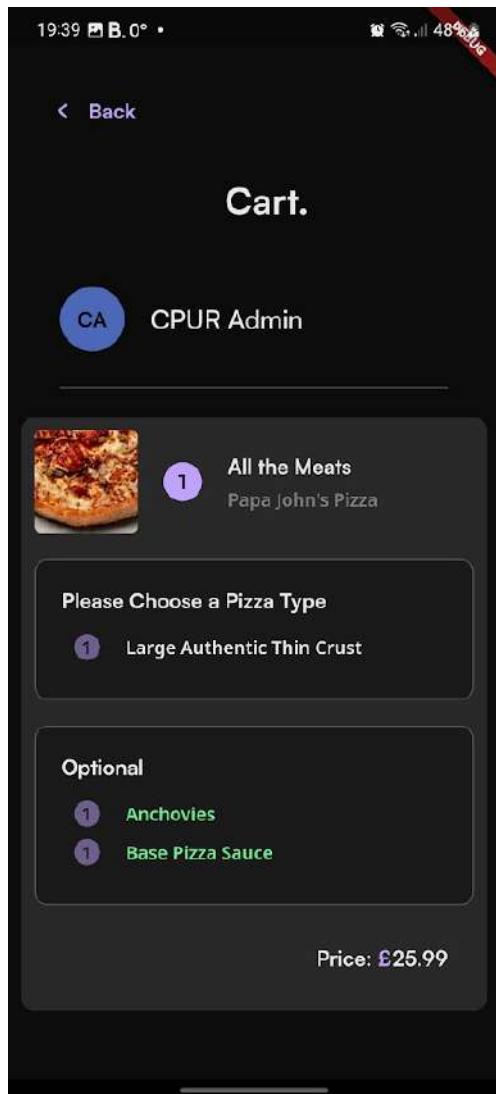
While looking at the screenshot, I noticed that it would look better if there was a slight margin in order for it to look better so a 10px gap was added on each side.

After getting the code working, I improved on the visuals

```
return Padding(
```

```
padding: const  
EdgeInsets.symmetric(horizontal: 20, vertical: 20),  
  
child:  
Row(mainAxisAlignment: MainAxisAlignment.end, children: [  
  
Text(  
  
"Price: ",  
  
style:  
Theme.of(context)  
  
.textTheme  
  
.headline6  
  
?.copyWith(color: Theme.of(context).textTheme.headline1?.color),  
( ),  
  
Text("£",  
  
style:  
Theme.of(context)  
  
.textTheme  
  
.headline6  
  
?.copyWith(color: Theme.of(context).primaryColor)),  
Text(  
  
itemPrice.toString(),  
  
style:  
Theme.of(context)
```

```
? .copyWith(color: Theme.of(context).textTheme.headline1?.color),  
)  
]);
```

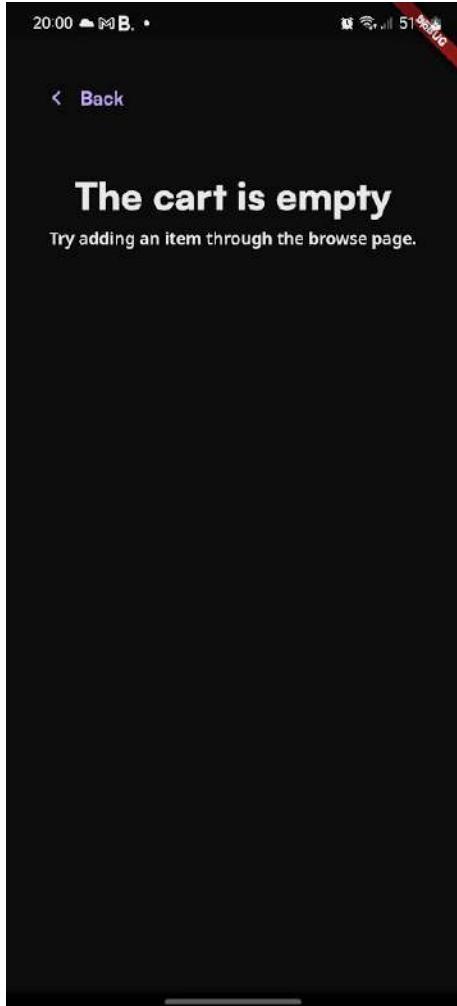


Displaying total price

In order to prevent errors, I added an if statement which displays that the cart is empty if there are no available profiles that are in the cart.

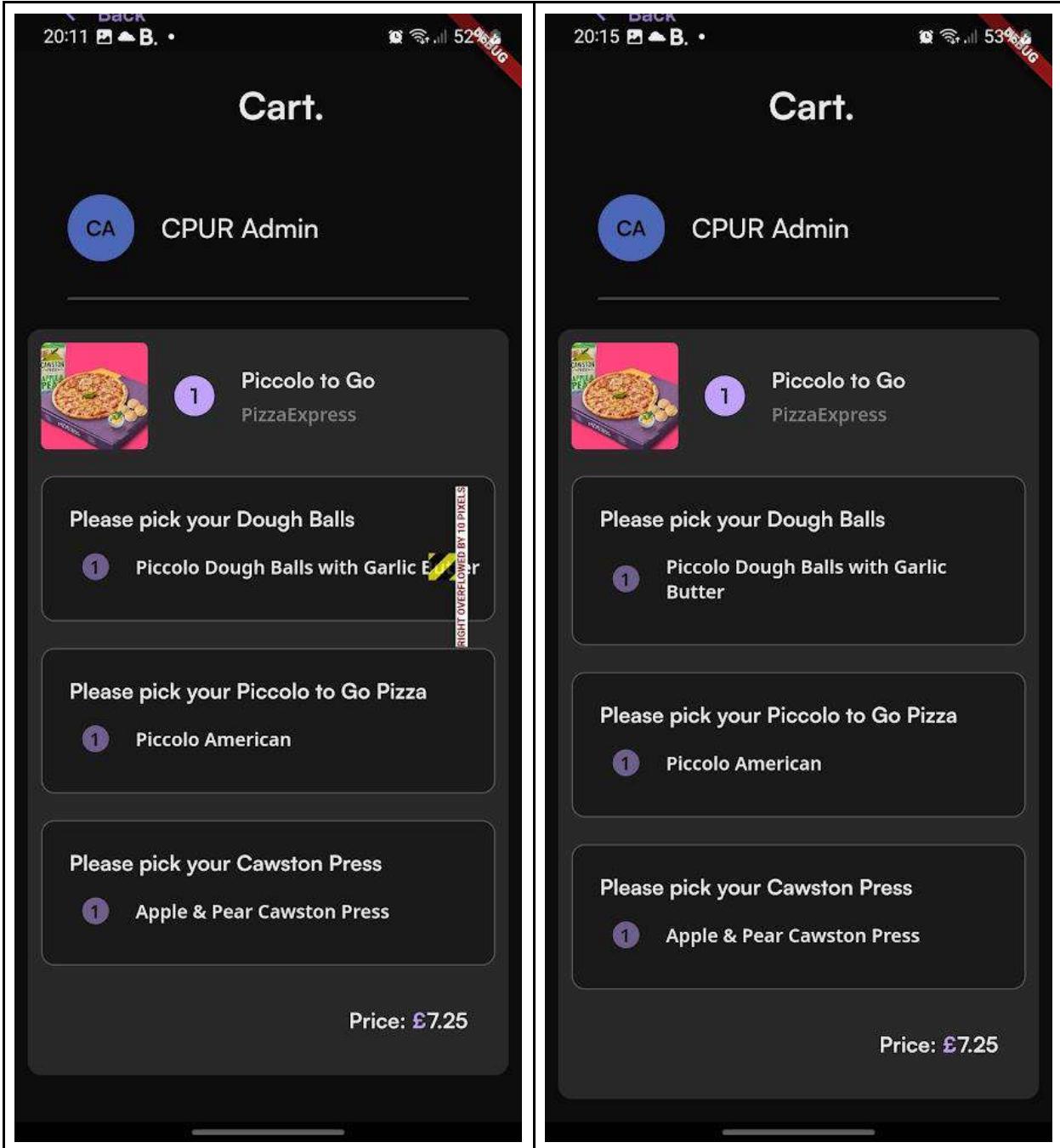
```
if (availableProfiles.isNotEmpty) {  
    ...  
} else {  
    return Scaffold(  
        body: SingleChildScrollView(  
            child: Column(crossAxisAlignment: CrossAxisAlignment.start,  
            children: [  
                const ScreenBackButton(),  
                Padding(  
                    padding: const EdgeInsets.all(30),  
                    child: SizedBox(  
                        width: double.infinity,  
                        child: Center(  
                            child: Column(children: [  
                                Text(  
                                    "The cart is empty",  
                                    textAlign: TextAlign.center,  
                                    style: Theme.of(context).textTheme.headline1,  
                                ),  
                            ],  
                        ),  
                    ),  
                ),  
            ],  
        ),  
    );  
}
```

```
        const SizedBox(  
          height: 5,  
        ),  
  
        Text(  
          "Try adding an item through the browse page.",  
          textAlign: TextAlign.center,  
          style: Theme.of(context).textTheme.bodyText1,  
        ),  
  
        const SizedBox(  
          height: 20,  
        ),  
      ])),  
    )  
  ]));  
}
```



While testing with some more items, I found that text could overflow for either the title of the customise type and the options. An expanded was added to these

Before	After
--------	-------



After adding this, I got an error

```
===== Exception caught by widgets library =====  
Incorrect use of ParentDataWidget.
```

As, it turns out, I had placed extra Expanded widgets in the wrong location so these were removed.

Another error I found was that if the first profile removes an item, it says the cart is empty. Looking at the code, it is apparent of what the issue is

```
for (int i = 0; i < cartProfileIDs.length; i++) {  
  
    if (cartProfileIDs.contains(profileInfo[i]["profileid"])) {  
  
        availableProfiles.add([  
  
            profileInfo[i]["profileid"],  
  
            profileInfo[i]["firstname"],  
  
            profileInfo[i]["lastname"],  
  
            profileInfo[i]["profilecolorred"],  
  
            profileInfo[i]["profilecolorgreen"],  
  
            profileInfo[i]["profilecolorblue"]  
  
        ]);  
  
    }  
  
}
```

Since both are lists, you would need to iterate through both lists meaning you need 2 for statements.

```
for (int i = 0; i < cartProfileIDs.length; i++) {  
  
    for (int j = 0; j < profileInfo.length; j++) {  
  
        if (cartProfileIDs[i] == profileInfo[j]["profileid"]) {  
  
            availableProfiles.add([  
  
                profileInfo[j]["profileid"],  
  
                profileInfo[j]["firstname"],  
  
                profileInfo[j]["lastname"],  
  
                profileInfo[j]["profilecolorred"],  
  
                profileInfo[j]["profilecolorgreen"],  
  
                profileInfo[j]["profilecolorblue"]  
  
            ]);  
  
        }  
  
    }  
  
}
```

```

        profileInfo[j]["firstname"],
        profileInfo[j]["lastname"],
        profileInfo[j]["profilecolorred"],
        profileInfo[j]["profilecolorgreen"],
        profileInfo[j]["profilecolorblue"]
    ]);
}

}

```

In order to get the total price for all the food, I started by getting the details of each profile and placing it into a 2D array, setting the price for the profile at 0 and the customise options as a blank map.

```
List finalCartData = [ ];
```

```

for (int p = 0; p < availableProfiles.length; p++) {
    finalCartData.addAll([
        [
            availableProfiles[p][0],
            availableProfiles[p][1],
            availableProfiles[p][2],
            availableProfiles[p][3],
            availableProfiles[p][4],

```

```
availableProfiles[p][5],  
0.00,  
{}  
]  
);  
}
```

For each item, it checks for which profile the program is at and adds it to that index

```
for (int l = 0; l < finalCartData.length; l++) {  
  
    if (finalCartData[l][0].toString()  
== availableProfiles[avPrIndex][0].toString()) {  
  
        finalCartData[l][6] =  
(finalCartData[l][6] * 100 + itemPrice * 100) / 100;  
  
    }  
}
```

In order for it to find the profile index, I had to change the index to avPrIndex so that it would not be overwritten.

One thing that I struggled to fix was getting the updated price. Since the whole cart was being run in a future, it meant that I could not update the page using a setState. To fix this, I tried a variety of things including making it run as another profile at the end, using a boolean to change the status of if it is complete but none of them worked.

None of my solutions worked so I decided the best course of action was to try and remake the file to have a more optimised design. Due to the amount of changes made, the file resulted in having too many issues which were roughly patched.

Remaking the Cart

I started with a basic design

```
class _CartState extends State<Cart> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(
      body: SingleChildScrollView(
        child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
          const ScreenBackButton(),
        ]));
  }
}
```

I then added the previous Futures with the basic shape of the previous Future Builder

```
FutureBuilder<List>(
  future: getAvailableProfiles(),
  builder: ((context, snapshot) {
    if (!snapshot.hasData) {
      List availableProfiles = snapshot.data ?? [];
      if (availableProfiles.isNotEmpty) {
        return Row(children: [Text("Test")]);
      }
    }
  }));
}
```

```
    } else {

        return Padding(
            padding: const EdgeInsets.all(30),
            child: SizedBox(
                width: double.infinity,
                child: Center(
                    child: Column(children: [
                        Text(
                            "The cart is empty",
                            textAlign: TextAlign.center,
                            style: Theme.of(context).textTheme.headline1,
                        ),
                        const SizedBox(
                            height: 5,
                        ),
                        Text(
                            "Try adding an item through the browse page.",
                            textAlign: TextAlign.center,
                            style: Theme.of(context).textTheme.bodyText1,
                        ),
                        const SizedBox(
                            height: 20,
```

```

        ),
    ])),
);

}

} else {

    return Row(mainAxisAlignment: MainAxisAlignment.center, children: [
        Padding(
            padding: const EdgeInsets.all(50),
            child: CircularProgressIndicator(
                color: Theme.of(context).primaryColor,
            )),
    ]);
}

})

```

I decided that the best thing to do is to integrate both future functions into one so that the result will be a single one.

```

Future<Map> getCart() async {
    Map returnCartInfo = {"error": false, "message": "", "cartinfo": []};

    List availableProfiles = [];

    List cartProfileIDs = await SQLiteCartItems.getProfilesInCart(); //Get
profile ids that are in the cart

    List profileInfo = await SQLiteLocalProfiles.getProfiles();
}

```

```

//Get profiles

for (int i = 0; i < cartProfileIDs.length; i++) {

    for (int j = 0; j < profileInfo.length; j++) {

        if (cartProfileIDs[i] == profileInfo[j]["profileid"]) {

            availableProfiles.add([
                profileInfo[j]["profileid"],
                profileInfo[j]["firstname"],
                profileInfo[j]["lastname"],
                profileInfo[j]["profilecolorred"],
                profileInfo[j]["profilecolorgreen"],
                profileInfo[j]["profilecolorblue"]
            ]);

        }
    }
}

for (int iProfile = 0; iProfile < availableProfiles.length; iProfile++) {

    Map tempItemInfo = {};

    List profileCart = await
SQLiteCartItems.getProfileCart(availableProfiles[iProfile][0]); //{cartid,
itemid, customised, quantity}

```

```

for (int i = 0; i < profileCart.length; i++) {

    // For each item, add it as an index i

    tempItemInfo[profileCart[i]["itemid"]] = [[], {}];

    Map basicItemInfo = await
QueryServer.query("https://alleat.cpur.net/query/cartiteminfo.php", {

    "type": "item",

    "term": profileCart[i]["itemid"].toString()

}); //returns item id, item name, price, item image, restaurant id,
restaurant name, delivery price

if (basicItemInfo["error"] == true) {

    returnCartInfo["error"] = true;

    returnCartInfo["message"] = basicItemInfo["message"];

    return returnCartInfo;

} else {

    tempItemInfo[profileCart[i]["itemid"]][0].addAll([
        basicItemInfo["message"]["message"][1],
        basicItemInfo["message"]["message"][3],
        basicItemInfo["message"]["message"][2],
        basicItemInfo["message"]["message"][5],
        basicItemInfo["message"]["message"][4],
        basicItemInfo["message"]["message"][6],
    ];
}
}

```

```
profileCart[i]["quantity"],

profileCart[i]["cartid"],

]);

Map customised = json.decode(profileCart[i]["customised"]);

List customisedtitleids = customised.keys.toList(); //Each customise
title is stored here

List customisedOptions = []; //Each unique option stored here

Map updatedCustomised = {};//Data that will be sent back in the
iteminfo key, correctly formatted

for (int i = 0; i < customisedtitleids.length; i++) {

    //For each customise title

    updatedCustomised[customisedtitleids[i]] = [[], []]; //Add customised
key to new Map

    for (int j = 0; j < customised[customisedtitleids[i]].length; j++) {

        //For each item in list of customised options for customise title

        if
(!customisedOptions.contains(customised[customisedtitleids[i]][j])) {

            //If the option is not in the list of options, add it to the list

updatedCustomised[customisedtitleids[i]][1].add([customised[customisedtitleids[i]]
][j], 1);

            customisedOptions.add(customised[customisedtitleids[i]][j]);
        }
    }
}
```

```

        for (int k = 0; k <
updatedCustomised[customisedtitleids[i]][1].length; k++) {

            if (updatedCustomised[customisedtitleids[i]][1][k][0] ==
customised[customisedtitleids[i]][j]) {

                updatedCustomised[customisedtitleids[i]][1][k][1] += 1;

            }

        }

    }

}

try {

    if (customisedtitleids.isNotEmpty) {

        Map customisedTitlesInfo = await QueryServer.query(

            "https://alleat.cpur.net/query/cartiteminfo.php", {"type": "title", "term": json.encode(customisedtitleids)});



        if (customisedTitlesInfo["error"] == true) {

            returnCartInfo["error"] = true;

            returnCartInfo["message"] = customisedTitlesInfo["message"];

            return returnCartInfo;

        } else {

            List customisedTitlesInfoFormatted =
customisedTitlesInfo["message"]["message"];

            if (customisedOptions.isNotEmpty) {

```

```

Map customisedOptionInfo = await QueryServer.query(
    "https://alleat.cpur.net/query/cartiteminfo.php", {"type": "option", "term": customisedOptions.toString()});

if (customisedOptionInfo["error"] == true) {
    returnCartInfo["error"] = true;
    returnCartInfo["message"] = customisedOptionInfo["message"];
    return returnCartInfo;
} else {
    List customisedOptionInfoFormatted =
    customisedOptionInfo["message"]["message"];

    for (int i = 0; i < customisedTitleIds.length; i++) {
        // For each customise title
        for (int j = 0; j < customisedTitlesInfoFormatted.length;
j++) {
            //For each item in list of returned from server info
            //about each title
            if (customisedTitlesInfoFormatted[j][0] ==
customisedTitleIds[i]) {
                // If it has the id of the title in the first index,
                add the info to the composite info
                updatedCustomised[customisedTitleIds[i]][0]
                    .addAll([customisedTitlesInfoFormatted[j][1],
customisedTitlesInfoFormatted[j][2]]);


```

```

        }

    }

    for (int j = 0; j <
updatedCustomised[customisedtitleids[i]][1].length; j++) {

        //For each option in the list of options for each title

        for (int k = 0; k < customisedOptionInfoFormatted.length;
k++) {

            //For each item in list of returned from server about
option info

            if (customisedOptionInfoFormatted[k][0] ==
updatedCustomised[customisedtitleids[i]][1][j][0]) {

                // If it has the id of the option in the first index,
add the info to the list

                updatedCustomised[customisedtitleids[i]][1][j]

                    .addAll([customisedOptionInfoFormatted[k][1],
customisedOptionInfoFormatted[k][2]]);

            }

        }

    }

    tempItemInfo[profileCart[i]["itemid"]][1] =
updatedCustomised;

}

}

}

```

```
        }

    } catch (e) {

        returnCartInfo["error"] = true;

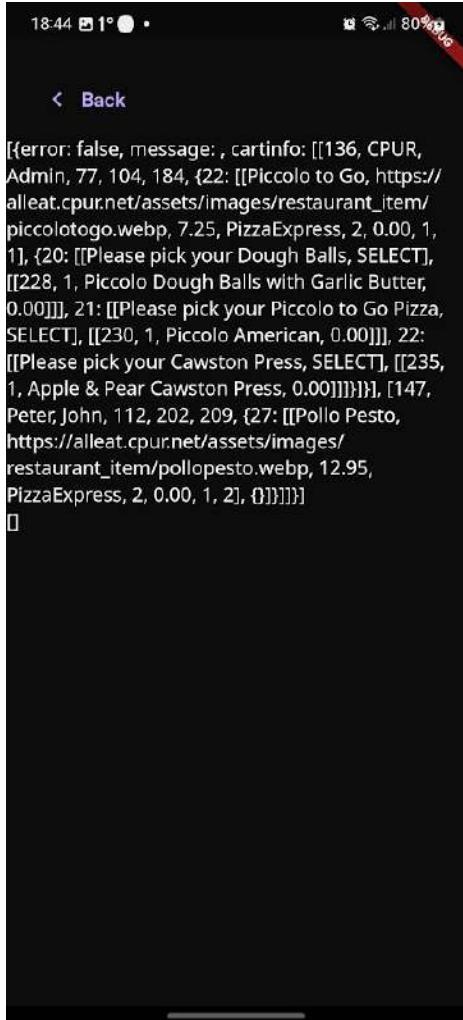
        returnCartInfo["message"] = "An error occurred while attempting to
process the item information.';

    }

}

returnCartInfo["cartinfo"].add([
    availableProfiles[iProfile][0],
    availableProfiles[iProfile][1],
    availableProfiles[iProfile][2],
    availableProfiles[iProfile][3],
    availableProfiles[iProfile][4],
    availableProfiles[iProfile][5],
    tempItemInfo
]);
}

return returnCartInfo;
}
```



To get the item prices, I will process the original price and return back the item price of the customised price. Before, it was returning the normal price back before there were any prices.

```
List customiseOptionsKeys =  
tempItemInfo[profileCart[i]["itemid"]][1].keys.toList();  
  
for (int iCust = 0; iCust < customiseOptionsKeys.length; iCust++) { //For  
each customise title  
  
    if  
(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==  
"SELECT" ||  
  
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
```

```

"ADD") { //If it is either a selection or an add function add it to the item
price

    for (int iCustOptions = 0;

        iCustOptions <
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1].length;

        iCustOptions++) {

            tempItemInfo[profileCart[i]["itemid"]][0][2] =
((double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2]) * 100 +

double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust
]][1][iCustOptions][3]) * 100) /

100)

            .toString();

        }

    } else if
(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
"REMOVE") { //If it is a remove function, remove from the item price

    for (int iCustOptions = 0;

        iCustOptions <
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1].length;

        iCustOptions++) {

            tempItemInfo[profileCart[i]["itemid"]][0][2] =
((double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2]) * 100 -

double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust
]][1][iCustOptions][3]) * 100) /

100)

```

```
        .toString();

    }

}

}
```

To format the data, I used a similar system to the previous cart but instead of using futurebuilders I used for statements for each profile and for each item

The basic format is as follows

```
if (snapshot.hasData) {

    List cartInfo = [snapshot.data ?? []];

    if (cartInfo[0]["error"] == true) {

        //If there is an error getting the cart info display error box

        return Padding(
            padding: const EdgeInsets.only(left: 20, right: 20, top: 10,
bottom: 10),
            child: Container(
                decoration: BoxDecoration(
                    borderRadius: const
BorderRadius.all(Radius.circular(20)),
                    color: Theme.of(context).colorScheme.onSurface,
                    boxShadow: [
                        BoxShadow(
                            color: Colors.black.withOpacity(0.1),
                            spreadRadius: 2,
```

```
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes position  
of shadow  
  
    ),  
  
    ]),  
  
    child: Column(children: [  
  
        Padding(  
  
            padding: const EdgeInsets.all(30),  
  
            child: SizedBox(  
  
                width: double.infinity,  
  
                child: Center(  
  
                    child: Column(children: [  
  
                        const Text(  
  
                            "An error has occurred.",  
  
                            textAlign: TextAlign.center,  
                        ),  
  
                        const SizedBox(  
  
                            height: 20,  
                        ),  
  
                        Text(  
  
                            cartInfo[0]["message"],  
  
                            textAlign: TextAlign.center,  
                        ),  
                    ],  
                ),  
            ),  
        ),  
    ],  
);
```

```
        ])),  
    )  
]);  
} else {}  
  
if (cartInfo[0]["cartinfo"].isEmpty) {  
  
    //If there are profiles in the cart  
  
    return Text(cartInfo[0]["cartinfo"].toString());  
} else {  
  
    return Padding(  
  
        padding: const EdgeInsets.all(30),  
  
        child: SizedBox(  
  
            width: double.infinity,  
  
            child: Center(  
  
                child: Column(children: [  
  
                    Text(  
  
                        "The cart is empty",  
  
                        textAlign: TextAlign.center,  
  
                        style: Theme.of(context).textTheme.headline1,  
                    ),  
  
                    const SizedBox(  
  
                        height: 5,  
                    ),  
                ],  
            ),  
        ),  
    );  
}
```

```

        Text(
            "Try adding an item through the browse page.",
            textAlign: TextAlign.center,
            style: Theme.of(context).textTheme.bodyText1,
        ),
        const SizedBox(
            height: 20,
        ),
    ])),),
);
}
}

```

I then added the “for each profile” section. From learning my mistake using just letters for the for statement, I replaced them with relevant variable names so that they can be referenced.

```

return ListView.builder(
    physics: const NeverScrollableScrollPhysics(), //Dont allow
scrolling (Done by main page)

    scrollDirection: Axis.vertical,
    shrinkWrap: true,
    itemCount: cartInfo[0]["cartinfo"].length + 1, //For each
profile (add one for the total price)
    itemBuilder: (context, indexProfile) {
        if (indexProfile != cartInfo[0]["cartinfo"].length) {

```

```
    List currentProfile =
cartInfo[0]["cartinfo"][indexProfile];

    return Column(children: [
        Container(
            //Contain the profile within a container
            margin: const EdgeInsets.symmetric(vertical: 10,
horizontal: 20),
            padding: const EdgeInsets.symmetric(horizontal: 20,
vertical: 10),
            child: Row(
                children: [
                    Container(
                        // Create profile circle with first and
last letter
                        width: 50,
                        height: 50,
                        decoration: BoxDecoration(
                            shape: BoxShape.circle, color:
Color.fromRGBO(currentProfile[3], currentProfile[4], currentProfile[5], 1)),
                        child: Align(
                            alignment: Alignment.center,
                            child:
Text('${currentProfile[1][0]}${currentProfile[2][0]}',
style:
Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).backgroundColor)))),
                ],
            ),
        ],
    );
}
```

```
        const SizedBox(width: 20), //Profile firstname  
and lastname  
  
        Text(  
          "${currentProfile[1]} ${currentProfile[2]}",  
          style:  
Theme.of(context).textTheme.headline5?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
          overflow: TextOverflow.fade,  
        )  
      ],  
    )),  
  
    Divider(  
      thickness: 2,  
      color:  
Theme.of(context).textTheme.headline6?.color?.withOpacity(0.5),  
      indent: 40,  
      endIndent: 40,  
    ),  
    const SizedBox(  
      height: 10,  
    ),  
  ]);  
} else {
```

```
        return const Text("Total");

    }

});
```

For each item, it runs another ListView for each item

```
ListView.builder(
    //Create a container for each profile containing
    the customised options and a summary of the item
    physics: const NeverScrollableScrollPhysics(),
    //Dont allow scrolling (Done by main page)
    scrollDirection: Axis.vertical,
    shrinkWrap: true,
    itemCount: itemIDs.length, //For each item
    itemBuilder: (context, indexItem) {
        List currentItem =
        currentProfile[6][itemIDs[indexItem]];
        return Padding(
            padding: const EdgeInsets.symmetric(vertical:
            5, horizontal: 10),
            child: Dismissible(
                //Create a slide to delete container
                (dismissible)
                key: Key(currentItem[0][7].toString()),
                //Key is the cart id
                onDismissed: (direction) async {
```

```
        bool hasDeleted = await
removeItem(currentItem[0][7]); //Remove item from local cart database

        if (hasDeleted) {

            //If it deleted

cartInfo[0]["cartinfo"][indexProfile][6].remove(itemIDs[indexItem]); //removing
the item from the list

        setState(() {

ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content:
Text("Successfully deleted item.")));

    });

}

else{

    setState(() {

ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content: Text("Failed
to deleted item. Please reopen the cart")));

    });

}

},
background: Container(
color:
Theme.of(context).colorScheme.error,
child: Row(
```

```
        mainAxisAlignment:  
MainAxisAlignment.spaceBetween,  
  
        children: [  
  
            Padding(  
  
                padding: const  
EdgeInsets.symmetric(horizontal: 30),  
  
                child: Icon(  
  
                    Icons.delete,  
  
                    color:  
Theme.of(context).colorScheme.onSurface,  
  
                )),  
  
            Padding(  
  
                padding: const  
EdgeInsets.symmetric(horizontal: 30),  
  
                child: Icon(  
  
                    Icons.delete,  
  
                    color:  
Theme.of(context).colorScheme.onSurface,  
  
                )),  
  
        ],  
  
    )),  
  
    child: Container(  
  
        width: double.infinity,  
  
        decoration: BoxDecoration(  

```

```
        borderRadius: const  
BorderRadius.all(Radius.circular(10)),  
  
        color:  
Theme.of(context).colorScheme.onSurface),  
  
        padding: const  
EdgeInsets.symmetric(vertical: 10, horizontal: 10),  
  
        child: Column(children: [  
  
        Row(  
  
        children: [  
  
        Container(  
  
        width: 80,  
  
        height: 80,  
  
        decoration: BoxDecoration(  
  
        borderRadius: const  
BorderRadius.all(Radius.circular(5)),  
  
        image:  
DecorationImage(fit: BoxFit.cover, image:  
NetworkImage(currentItem[0][1].toString()))),  
  
(  
),  
  
const SizedBox(  
  
width: 20,  
  
(  
),  
  
Container(  
  
alignment:  
Alignment.center,
```

```
        height: 30,  
  
        width: 30,  
  
        decoration: BoxDecoration(  
  
            color:  
Theme.of(context).primaryColor, borderRadius: BorderRadius.circular(30)),  
  
            child: Text(  
  
currentItem[0][6].toString(),  
  
style: Theme.of(context)  
  
.textTheme  
  
.headline6  
  
?.copyWith(color:  
Theme.of(context).colorScheme.onSurface),  
  
)),  
  

```

```
        style: Theme.of(context)

            .textTheme

            .headline6

            ?.copyWith(color:

Theme.of(context).textTheme.headline1?.color),

        ),

        const SizedBox(

            height: 5,

        ),

        Text(

            currentItem[0][3],

            style: Theme.of(context)

                .textTheme

                .bodyText1

                ?.copyWith(color:

Theme.of(context).textTheme.headline6?.color),

        )

    ],
))

],
)

]))));

}
}
```

The next thing to do is to add the customise options to the listview

```
ListView.builder(  
      
        //Create a container with the  
        title of the option that has been customised with a list of options changed  
          
        physics: const  
        NeverScrollableScrollPhysics(), //Dont allow scrolling (Done by main page)  
          
        scrollDirection: Axis.vertical,  
          
        shrinkWrap: true,  
          
        itemCount:  
        itemCustomiseIDs.length, //For each customise title  
          
        itemBuilder: (context,  
        indexCustomise) {  
              
            List currentCustomiseTitle =  
            currentItem[1][itemCustomiseIDs[indexCustomise]];  
              
            if  
            (currentCustomiseTitle[0][1] == "SELECT" &&  
            currentCustomiseTitle[1].toList().length != 0) {  
                  
                return Container(  
                      
                    padding: const  
                    EdgeInsets.symmetric(vertical: 20, horizontal: 20),  
                      
                    margin: const  
                    EdgeInsets.symmetric(horizontal: 0, vertical: 10),  
                      
                    decoration:  
                    BoxDecoration(  
                          
                        color:  
                        Theme.of(context).backgroundColor.withOpacity(0.5),  
                    ),  
                );  
            }  
        },  
    ),  
);
```

```
border: Border.all(  
    color:  
Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),  
  
borderRadius:  
BorderRadius.circular(10)),  
  
child: Column(  
  
    mainAxisAlignment:  
MainAxisAlignment.start,  
  
    crossAxisAlignment:  
CrossAxisAlignment.start,  
  
    children: [  
  
Text(currentCustomiseTitle[0][0],  
  
        textAlign:  
 TextAlign.start,  
  
        style:  
Theme.of(context)  
  
.textTheme  
  
.headline6  
  
?.copyWith(color: Theme.of(context).textTheme.headline1?.color)),  
  
const  
SizedBox(height: 15),  
  
ListView.builder(  

```

```
physics:  
const NeverScrollableScrollPhysics(),  
  
shrinkWrap:  
true,  
  
itemCount:  
currentCustomiseTitle[1].length, //For each customise option  
  
itemBuilder:  
(context, indexOption) {  
  
List  
currentCustomiseOption = currentCustomiseTitle[1][indexOption];  
  
return  
Padding(  
  
padding:  
const EdgeInsets.only(left: 10, bottom: 10),  
  
child:  
Row(children: [  
  
CircleAvatar(  
  
backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
  
radius: 10,  
  
child: Text(currentCustomiseOption[1].toString(),  
  
style: Theme.of(context)  
  
.textTheme
```

```
.bodyText1

?.copyWith(color: Theme.of(context).backgroundColor)),
),
const
SizedBox(
width: 20,
),

Expanded(
child: Text(currentCustomiseOption[2].toString(),
style: Theme.of(context).textTheme.bodyText1))
],
);

},
],
}),

]);
}

} else if
(currentCustomiseTitle[0][1] == "ADD" &&
currentCustomiseTitle[1].toList().length != 0) {
return Container(
padding: const
EdgeInsets.symmetric(vertical: 20, horizontal: 20),
```

```
margin: const  
EdgeInsets.symmetric(horizontal: 0, vertical: 10),  
  
decoration:  
BoxDecoration(  
  
color:  
Theme.of(context).backgroundColor.withOpacity(0.5),  
  
border: Border.all(  
  
color:  
Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),  
  
borderRadius:  
BorderRadius.circular(10)),  
  
child: Column(  
  
mainAxisAlignment:  
MainAxisAlignment.start,  
  
crossAxisAlignment:  
CrossAxisAlignment.start,  
  
children: [  
  
Text(currentCustomiseTitle[0][0],  
  
textAlign:  
 TextAlign.start,  
  
style:  
Theme.of(context)  
  
.textTheme  
  
.headline6
```

```
? .copyWith(color: Theme.of(context).textTheme.headline1?.color)),  
    const  
SizedBox(height: 15),  
  
        ListView.builder(  
  
            physics:  
const NeverScrollableScrollPhysics(),  
  
            shrinkWrap:  
true,  
  
            itemCount:  
currentCustomiseTitle[1].length, //For each customise option  
  
            itemBuilder:  
(context, indexOption) {  
  
                List  
currentCustomiseOption = currentCustomiseTitle[1][indexOption];  
  
                return  
Padding(  
  
                    padding:  
const EdgeInsets.only(left: 10, bottom: 10),  
  
                    child:  
Row(children: [  
  
CircleAvatar(  
  
backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
  
radius: 10,
```

```
child: Text(currentCustomiseOption[1].toString(),  
  
style: Theme.of(context)  
  
.textTheme  
  
.bodyText1  
  
?.copyWith(color: Theme.of(context).backgroundColor)),  
  
),  
  
const  
SizedBox(  
  
width: 20,  
  
),  
  
Expanded(  
  
child: Text(currentCustomiseOption[2].toString(),  
  
style: Theme.of(context)  
  
.textTheme  
  
.bodyText1  
  
?.copyWith(color: Theme.of(context).colorScheme.tertiary)))
```

```

        ],
      );
    },
  ],
));
} else if
(currentCustomiseTitle[0][1] == "REMOVE" &&
currentCustomiseTitle[1].toList().length != 0) {
  return Container(
    padding: const
    EdgeInsets.symmetric(vertical: 20, horizontal: 20),
    margin: const
    EdgeInsets.symmetric(horizontal: 0, vertical: 10),
    decoration:
    BoxDecoration(
      color:
      Theme.of(context).backgroundColor.withOpacity(0.5),
      border: Border.all(
        color:
        Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),
      borderRadius:
      BorderRadius.circular(10)),
      child: Column(
        mainAxisSize:
        MainAxisSize.start,
        crossAxisAlignment:
        CrossAxisAlignment.start,

```

```
children: [  
    Text(currentCustomiseTitle[0][0],  
        textAlign:  
        TextAlign.start,  
        style:  
        Theme.of(context)  
  
.textTheme  
  
.headline6  
  
?.copyWith(color: Theme.of(context).textTheme.headline1?.color)),  
    const  
SizedBox(height: 15),  
    ListView.builder(  
        physics:  
        const NeverScrollableScrollPhysics(),  
        shrinkWrap:  
        true,  
        itemCount:  
        currentCustomiseTitle[1].length, //For each customise option  
        itemBuilder:  
        (context, indexOption) {  
            List  
            currentCustomiseOption = currentCustomiseTitle[1][indexOption];  
            return  
            Padding(  
                child: Text(currentCustomiseOption),  
                padding: EdgeInsets.all(10),  
            ),  
        },  
    ),  
],  
);
```

```
padding:  
const EdgeInsets.only(left: 10, bottom: 10),  
  
child:  
Row(children: [  
  
CircleAvatar(  
  
backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
  
radius: 10,  
  
child: Text(currentCustomiseOption[1].toString(),  
  
style: Theme.of(context)  
  
.textTheme  
  
.bodyText1  
  
? .copyWith(color: Theme.of(context).backgroundColor)),  
(  
),  
const  
SizedBox(  

```

```

child: Text(currentCustomiseOption[2].toString(),

style: Theme.of(context)

.textTheme

.bodyText1

?.copyWith(color: Theme.of(context).colorScheme.error)))

]),

);

}),

],));

} else {

return const SizedBox(
height: 0,
);

}

})

```

The final thing to do is to add the total price to the end. Unlike in the previous cart, since all the calculations are done in the Future, I just need to return the value of the price stored in the returned Future data.

```
Padding(
```

```
padding: const  
EdgeInsets.symmetric(vertical: 10, horizontal: 20),  
  
child: Row(mainAxisAlignment:  
MainAxisAlignment.end, children: [  
  
    Text(  
  
        "Price: ",  
  
        style: Theme.of(context)  
            .textTheme  
            .headline6  
  
            ?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
    ),  
  
    Text(  
  
        "\u00a3",  
  
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).primaryColor),  
  
    ),  
  
    Text(  
  
        currentItem[0][2].toString(),  
  
        style: Theme.of(context)  
            .textTheme  
            .headline6
```

```
? .copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
)  
]))
```

Displaying the profile price

To get the profile price for the sum of items, I will use a LayoutBuilder in order to add logic and calculate the price. Since there is no need to get any information, once the data is received, the price will be constant.

While I had a few issues getting the right variable out of the returned data, overall it was painless and was a repeat of previous parts I had done before.

```
return LayoutBuilder(  
builder: (p0, p1) {  
  
    for (int iProfilePrice = 0; iProfilePrice <  
cartInfo[0]["cartinfo"].length; iProfilePrice++) {  
  
        //For each profile in the cart  
  
        cartInfo[0]["cartinfo"][iProfilePrice].add(0.00);  
//Add total price to the end of the profile index in the cartInfo  
  
        List itemPriceKeys =  
cartInfo[0]["cartinfo"][iProfilePrice][6].keys.toList(); //Create a list of keys  
for each item  
  
        for (int iItemPrice = 0; iItemPrice <  
itemPriceKeys.length; iItemPrice++) {  
  
            //For each item  
  
            cartInfo[0]["cartinfo"][iProfilePrice][7] =  
(cartInfo[0]["cartinfo"][iProfilePrice][7] * 100 +
```

```

double.parse(cartInfo[0]["cartinfo"][iProfilePrice][6][itemPriceKeys[iItemPrice]]
[0][2]) * 100) /
100; //Add price to the total price for the
profile

}

}

return Column(children: [
const SizedBox(height: 50),
LayoutBuilder(builder: ((p0, p1) {
for (int iProfilePrice = 0; iProfilePrice <
cartInfo[0]["cartinfo"].length; iProfilePrice++) {
//For each profile, display their total price
return Padding(
padding: const
EdgeInsets.symmetric(horizontal: 40, vertical: 10),
child: Row(
mainAxisAlignment:
MainAxisAlignment.end,
children: [
Text(
"${cartInfo[0]["cartinfo"][iProfilePrice][1]}
${cartInfo[0]["cartinfo"][iProfilePrice][2]}: ",
style: Theme.of(context)

```

```

        .textTheme
        .headline6
        ?.copyWith(color:
Theme.of(context).textTheme.headline1?.color),
),
const SizedBox(
width: 50,
),
),

Text("£${cartInfo[0]["cartinfo"][iProfilePrice][7].toStringAsFixed(2)}",
style:
Theme.of(context).textTheme.headline6?.copyWith(fontWeight: FontWeight.w500))
],
));
}

return const Text("");
}),

```

After testing on multiple profiles, it only displayed the first profile. To fix this, I replaced the for statement with a listview in order to get back multiple widgets without replacing the current one.

```
ListView.builder(
```

```

physics: const
NeverScrollableScrollPhysics(),
shrinkWrap: true,
itemCount: cartInfo[0]["cartinfo"].length,
//For each profile

itemBuilder: (context, iProfilePrice) {

    //For each profile, display the profile
total price

    return Padding(
padding: const
EdgeInsets.symmetric(horizontal: 40, vertical: 10),

child: Row(
mainAxisAlignment:
MainAxisAlignment.end,
children: [
Text(
"${cartInfo[0]["cartinfo"][[iProfilePrice][1]}${cartInfo[0]["cartinfo"][[iProfilePrice][2]]}: ",
style: Theme.of(context)

.textTheme
.headline6
?.copyWith(color:
Theme.of(context).textTheme.headline1?.color),
),
const SizedBox(

```

```
        width: 50,  
    ),  
  
Text("£${cartInfo[0]["cartinfo"][[iProfilePrice][7].toStringAsFixed(2)}",  
    style:  
Theme.of(context).textTheme.headline6?.copyWith(fontWeight: FontWeight.w500))  
    ],  
));  
},
```

Displaying subtotal

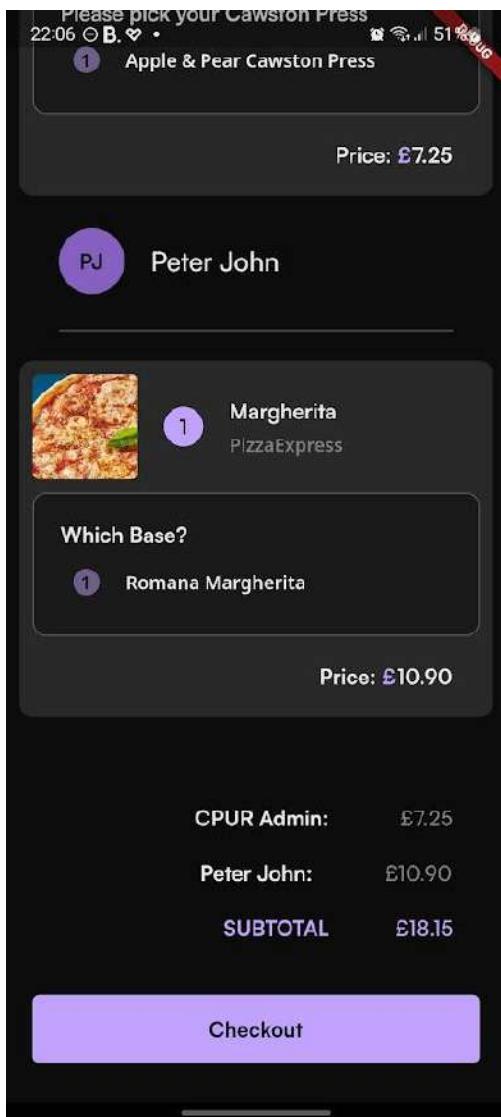
To display the subtotal, I will create a variable for the total price and add each profile's total price to it.

```
for (int iSubtotalProfile = 0; iSubtotalProfile < cartInfo[0]["cartinfo"].length;  
iSubtotalProfile++){  
  
    subtotal = (subtotal * 100 +  
cartInfo[0]["cartinfo"][[iSubtotalProfile][7] * 100) /  
  
    100; //Add the total price for profile  
to subtotal  
  
}
```

I then output the subtotal

```
Padding(  
    padding: EdgeInsets.all(10),  
    child:
```

```
        padding: const  
EdgeInsets.symmetric(horizontal: 40, vertical: 10),  
  
        child: Row(mainAxisAlignment:  
MainAxisAlignment.end, children: [  
  
            Text("SUBTOTAL", style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).primaryColor)),  
  
            const SizedBox(  
  
                width: 50,  
  
(),  
  
Text("£${subtotal.toStringAsFixed(2)}",  
style: Theme.of(context)  
    .textTheme  
    .headline6  
  
    ?.copyWith(fontWeight:  
FontWeight.w500, color: Theme.of(context).primaryColor))  
        ])),
```



Checkout verification

To determine if the user should be able to checkout, the following must be true

- Must have at least one item in the cart
- The items should be more than the minimum order price specified by the restaurant
- The distance from the restaurant must be less than the maximum distance the restaurant serves
- There must be no errors in the cart
- The items must only come from one restaurant

Since there is already an if statement for if there is an item in the cart, that can be skipped. As well, the error checking is already done in the FutureBuilder. By looking at the rest, I noticed that

in order to get the verifications in the correct order, the first thing to check is if there is only one restaurant. The next thing to check can either be distance or items.

Check for One Restaurant

To do the checking for if there is one restaurant, I will make use of the previous section where each profile added each item to get the total price for the profile. By adding some extra code, I am able to check for if it has the same restaurant ID for each

One issue I faced was that when returning the value of which restaurants display as not being the same, it said the same restaurant id was not the same

```
if (cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][4] !=  
tempRestaurantID) {  
  
    print(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][4]);  
  
    print(tempRestaurantID);  
  
    isOneRestaurant == false;  
  
    print("Not one restaurant");  
  
}
```

```
I/flutter (17969): [I/marqherita, https:  
② I/flutter (17969): 2  
I/flutter (17969): Not one restaurant
```

As it turns out one was being passed as an integer while the other was not

```
I/flutter (17969): String  
I/flutter (17969): int  
I/flutter (17969): Not one restaurant
```

To fix this I did an int.parse to convert the string to an integer. The code is as follows to the iterated code:

```
bool isOneRestaurant = true;  
  
int tempRestaurantID = -1;
```

```

        for (int iProfile = 0; iProfile <
cartInfo[0]["cartinfo"].length; iProfile++) {

            //For each profile in the cart

            cartInfo[0]["cartinfo"][iProfile].add(0.00);
//Add total price to the end of the profile index in the cartInfo

            List itemPriceKeys =
cartInfo[0]["cartinfo"][iProfile][6].keys.toList(); //Create a list of keys for
each item

            for (int iItem = 0; iItem < itemPriceKeys.length;
iItem++) {

                if (tempRestaurantID == -1) {

                    tempRestaurantID =
int.parse(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][4]);

                } else {

                    if
(int.parse(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][4]) !=
tempRestaurantID) {

                        isOneRestaurant == false;

                    }

                }

            }

            //For each item

            cartInfo[0]["cartinfo"][iProfile][7] =
(cartInfo[0]["cartinfo"][iProfile][7] * 100 +

double.parseDouble(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][2]) *
100) /

```

```
    100; //Add price to the total price for the
profile

}

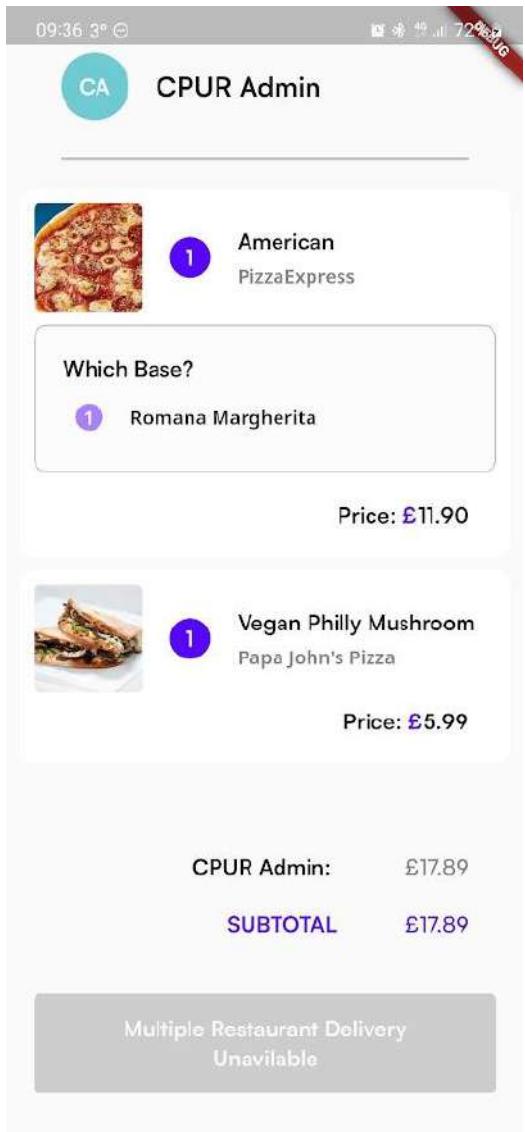
}
```

In order for the user to know if they can checkout and the reason, the text for the button is replaced with the issue

```
LayoutBuilder(
  builder: (p0, p1) {
    if (isOneRestaurant == true) {
      return Padding(
        padding: const
        EdgeInsets.symmetric(horizontal: 20, vertical: 30),
        child: Row(children: [Expanded(child:
ElevatedButton(onPressed: (() {}), child: const Text("Checkout")))]);
    } else {
      return Padding(
        padding: const
        EdgeInsets.symmetric(horizontal: 20, vertical: 30),
        child: Row(children: [
          Expanded(
            child: Opacity(
              opacity: 0.2,
```

```
        child: ElevatedButton(  
  
            style: ButtonStyle(  
  
                backgroundColor:  
MaterialStatePropertyAll(Theme.of(context).colorScheme.onBackground)),  
  
                onPressed: null,  
  
                child: Text(  
  
                    "Multiple Restaurant  
Delivery Unavailable",  
  
                    textAlign:  
TextAlign.center,  
  
                    style:  
Theme.of(context)  
  
.textTheme  
.headline6  
  
? .copyWith(color:  
Theme.of(context).backgroundColor),  
  
        ))))  
  

```



Minimum Order Price

The next thing to check is if the cart meets the minimum price for delivery requirement. To do this, I can use the if statement for checking if there is only one restaurant and then get the first restaurant from the cart and find the minimum order price stored within it.

After checking the restaurant information imported I found that the minimum order price was never imported. To fix this, I went into the cart php file stored on the server and added it to the SQL statement and export lines.

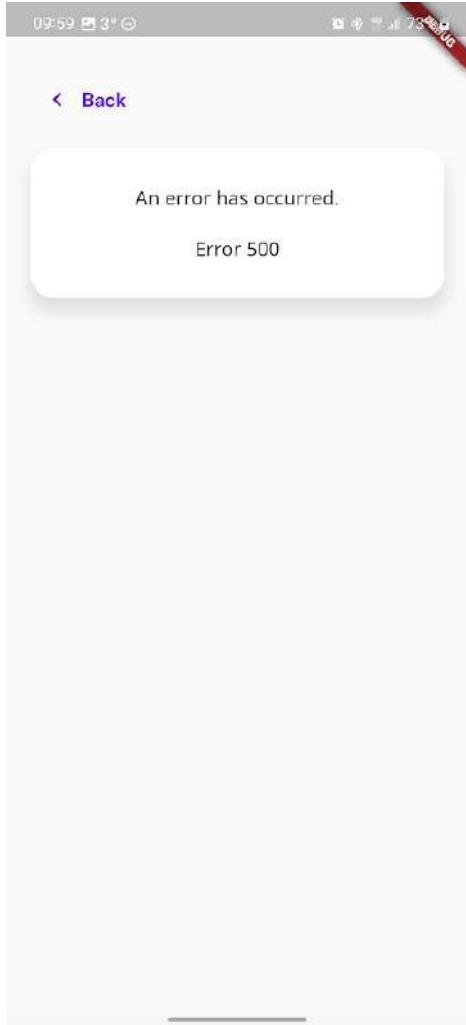
```
if ($type == "item") {
```

```
$term = intval($term);
$sqlitem = "SELECT item.itemid, item.itemname, item.price,
item.itemimage, restaurant.restaurantid, restaurant.restaurantname,
restaurant.delivery restaurant.ordermin FROM item AS item INNER JOIN
menucategories AS menucategories ON item.categoryid =
menucategories.categoryid INNER JOIN restaurant AS restaurant ON
menucategories.restaurantid = restaurant.restaurantid WHERE item.itemid =
$term";

if($result = mysqli_query($link, $sqlitem)){
    $iteminfo = [];
    while($row = mysqli_fetch_assoc($result)) {
        array_push($iteminfo, $row["itemid"], $row["itemname"],
$row["price"], $row["itemimage"], $row["restaurantid"],
$row["restaurantname"], $row["delivery"], $row["ordermin"]);
    }
    $return["message"] = $iteminfo;

} else{
    $return["error"] = true;
    $return["message"] = "Failed item query";
}
```

After adding the following, I got an error 500 from the server.



The problem was that there was no comma with the SQL statement.

Even after this fix, there was still an issue with the error 500 still being shown.

The issue was that the bracket was also not closed and after this, it worked.

To then add it to the cart, I had to add index 7 to the list

```
tempItemInfo[profileCart[1]["itemid"]][0].addAll([  
    basicItemInfo["message"]["message"][1],  
    basicItemInfo["message"]["message"][3],  
    basicItemInfo["message"]["message"][2],
```

```
basicItemInfo["message"]["message"][5],  
basicItemInfo["message"]["message"][4],  
basicItemInfo["message"]["message"][6],  
basicItemInfo["message"]["message"][7],  
profileCart[i]["quantity"],  
profileCart[i]["cartid"],
```

I then added the code to fix the first index and check if it was less than

```
List itemKeys = cartInfo[0]["cartinfo"][0][6].keys.toList();  
  
if  
(double.parseDouble(cartInfo[0]["cartinfo"][0][6][itemKeys[0]][0][6]) <= subtotal) {  
  
    return Padding(  
  
        padding: const  
EdgeInsets.symmetric(horizontal: 20, vertical: 30),  
  
        child:  
  
            Row(children: [Expanded(child:  
ElevatedButton(onPressed: (() {}), child: const Text("Checkout")))]));  
  
} else {  
  
    return Padding(  
  
        padding: const  
EdgeInsets.symmetric(horizontal: 20, vertical: 30),  
  
        child: Row(children: [  
  
            Expanded(  
  
                child: Opacity(
```

```

        opacity: 0.2,

        child: ElevatedButton(
            style: ButtonStyle(
                backgroundColor:
MaterialStatePropertyAll(Theme.of(context).colorScheme.onBackground)),
                onPressed: null,
                child: Text(
                    "Minimum order
£${cartInfo[0]["cartinfo"][0][6][itemKeys[0]][0][6]}",
                    textAlign:
 TextAlign.center,
                    style:
Theme.of(context)
                    .textTheme
                    .headline6
                    ?.copyWith(color:
Theme.of(context).backgroundColor),
                    ))),
                ]));
}

```

One thing that messed up a lot was that because I was appending to an array, the previous changes made which added the value to the list, this new change broke the quantity and cart ID.

After looking over the code, luckily, this appending is done at the beginning and can be fixed by changing when it is appended. Although this fixes it, it is not a good fix and if I had more time, I

would change it to use a Map instead of an array since they are referenced by key and because they have a set format, they can be assigned.

```
tempItemInfo[profileCart[i]["itemid"]][0].addAll([
    basicItemInfo["message"]["message"][1],
    basicItemInfo["message"]["message"][3],
    basicItemInfo["message"]["message"][2],
    basicItemInfo["message"]["message"][5],
    basicItemInfo["message"]["message"][4],
    basicItemInfo["message"]["message"][6],
    profileCart[i]["quantity"],
    profileCart[i]["cartid"],
    basicItemInfo["message"]["message"][7],
]);
```

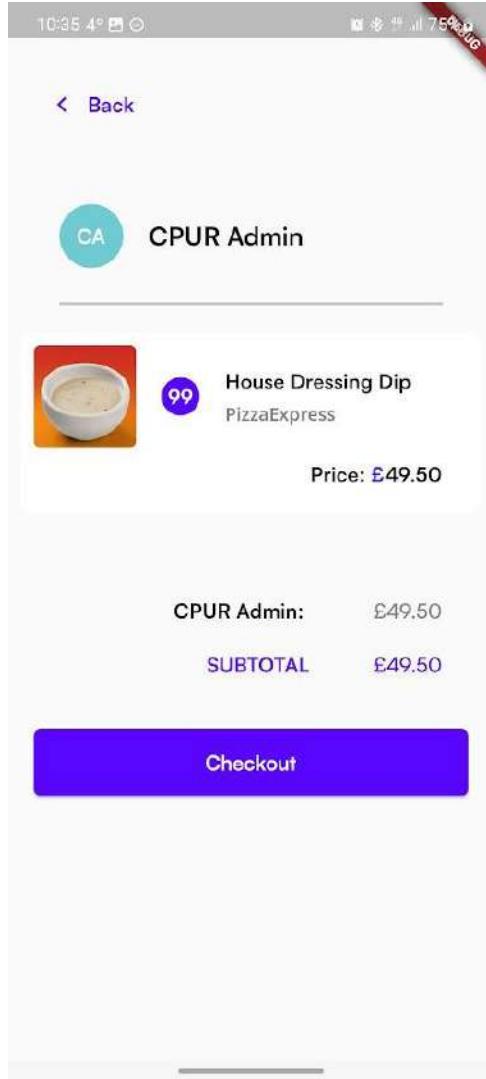
Fixing Price Quantity

During this time, I found the quantity is never put into account when determining the price for each item.

To fix this, I will multiply the price by the quantity.

In the Future after determining the price with the customised option modifications I multiplied the item price by the quantity

```
tempItemInfo[profileCart[i]["itemid"]][0][2] =
(double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2]) *
tempItemInfo[profileCart[i]["itemid"]][0][6]).toString();
```



After fixing this, I tried to remove the item from the cart but this caused an error.

```
669     builder: (p0, p1) {
670         if (isOneRestaurant == true) {
671             List itemKeys = cartInfo[0][\"cartinfo\"][0][6].keys.toList();
672             if (double.parseDouble(cartInfo[0][\"cartinfo\"][0][6][itemKeys[0]][0][8]) <= subtotal) {
```

Exception has occurred. ×
RangeError (RangeError (index): Invalid value: Valid value range is empty: 0)

The issue is caused by the program looking for the first entry in the list but if it has been removed, it breaks so the way I fixed this was using a catch statement to show that it failed to get the cart minimum. Since it catches the error, the build can be updated and the temporary button can be replaced with the correct value.

try

```

        if
(double.parseDouble(cartInfo[0]["cartinfo"][0][6][itemKeys[0]][0][8]) <= subtotal) {

    return Padding(
        padding: const
EdgeInsets.symmetric(horizontal: 20, vertical: 30),

        child: Row(
            children: [Expanded(child:
ElevatedButton(onPressed: (() {}), child: const Text("Checkout"))))];

    } else {

        return Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 20, vertical: 30),

            child: Row(children: [
                Expanded(
                    child: Opacity(
                        opacity: 0.2,
                    child: ElevatedButton(
                        style: ButtonStyle(
                            backgroundColor:
MaterialStatePropertyAll(Theme.of(context).colorScheme.onBackground)),
                            onPressed: null,
                            child: Text(
                                "Minimum order
£${cartInfo[0]["cartinfo"][0][6][itemKeys[0]][0][8]}",

```

```
        textAlign:  
        TextAlign.center,  
  
        style:  
        Theme.of(context)  
  
            .textTheme  
  
            .headline6  
  
    ?.copyWith(color: Theme.of(context).backgroundColor),  
  
        ))))  
  
    ]));  
  
}  
  
} catch (e) {  
  
    return Padding(  
  
        padding: const  
EdgeInsets.symmetric(horizontal: 20, vertical: 30),  
  
        child: Row(children: [  
  
        Expanded(  
  
            child: ElevatedButton(  
  
                style:  
  
                ButtonStyle(backgroundColor:  
MaterialStatePropertyAll(Theme.of(context).colorScheme.error)),  
  
                onPressed: null,  
  
                child: Text(  
        
```

```
        "Unknown minimum cart
price",
        textAlign:
        TextAlign.center,
        style:
        Theme.of(context).textTheme.headline6?.copyWith(color:
        Theme.of(context).backgroundColor),
        )));
])
);
}
```

Delivery Distance

Since the restaurants shown are only the ones available, in order to check if it is in the range, I will reset the cart when the user decides to change their location. In this case, I need to add a cancel button to the location setting as well as informing the user that their cart will be cleared when they change their location.

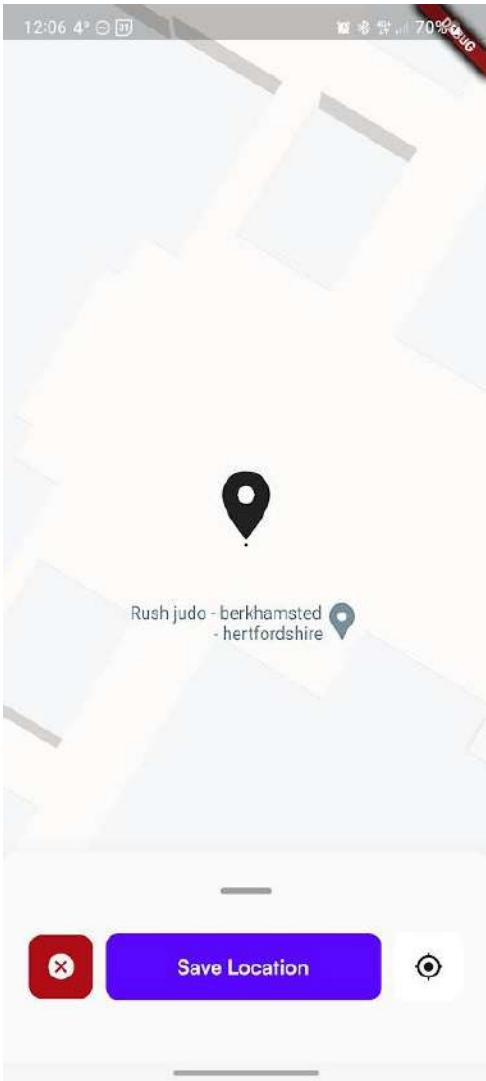
I will start by creating a function in `cart_service` to remove all items from the cart.

```
//Remove all items from cart
static Future<void> clearCart() async {
    final db = await SQLiteCartItems.cartdb();
    await db.rawDelete("DELETE FROM cartItems");
}
```

Cancel Button

Adding the cancel button was easy, by adding an InkWell to the start of the row where the save button and popping the location screen, it undos any changes since the changes are only saved at the save button tap.

```
InkWell(  
    child: Container(  
        width: 50,  
        height: 50,  
        decoration: BoxDecoration(  
            borderRadius:  
            BorderRadius.circular(10), color: Theme.of(context).colorScheme.error),  
        child: Icon(  
            Icons.cancel,  
            color:  
            Theme.of(context).backgroundColor,  
        )),  
    onTap: () {  
        setState(() {  
            Navigator.pop(context);  
        });  
    },  
,
```



Save Warning

In order to warn the user that the cart will clear, instead of immediately saving, it will open a popup and if the user clicks yes it will then save but if they click cancel, it will go back to the location screen.

Since I did not know how to create a popup, I used the following documentation to get an example of how to do it.

<https://api.flutter.dev/flutter/material/AlertDialog-class.html>

I modified the ok button to do the original function and left the cancel button to just close the popup.

```
Expanded(  
    child: InkWell(  
        //Save Location button  
        onTap: (() {  
            showDialog<String>(  
                //Display popup to confirm  
                context: context,  
                builder: (BuildContext  
context) => AlertDialog(  
                    title: Text(  
                        'Change Saved  
Destination',  
                        style:  
                        Theme.of(context)  
                            .textTheme  
                            .headline5  
                            ?.copyWith(color:  
                                Theme.of(context).textTheme.headline1?.color),  
                    ),  
                    content: Text(  
                        'Changing your delivery  
destination will clear any items currently saved in the cart.',  
                        style:  
                        Theme.of(context).textTheme.bodyText2,  
                    ),  
                ),  
            );  
        }),  
    ),  
);
```

```
actions: <Widget>[  
    TextButton(  
        //Cancel button to  
        close the popup bring user back to the location page  
        onPressed: () =>  
        Navigator.pop(context, 'Cancel'),  
        child: const  
        Text('Cancel'),  
    ),  
    TextButton(  
        //Ok button to save  
        the location  
        onPressed: (() async  
{  
        bool  
        hasSavedLocation = await saveLocation(); //Try to save  
  
        if  
        (hasSavedLocation == true) {  
            //If the location  
            has saved to shared preferences reopen the navigation page  
            setState(() {  
  
            Navigator.pop(context);  
  
            Navigator.pop(context);  
        }  
    })  
}
```

```
Navigator.of(context).push(  
  
MaterialPageRoute(builder: (_) => const Navigation()),  
  
);  
  
});  
  
} else {  
  
//If it fails to  
save  
  
setState(() {  
  
//Display  
failed to update  
  
ScaffoldMessenger.of(context)  
  
.showSnackBar(const SnackBar(content: Text('Failed to update location.')));  
  
});  
  
}  
  
},  
  
child: const  
Text('OK'),  
  
,  
  
],  
  
,  
  
);
```

```
        }),

        child: Container(
            decoration:
                BoxDecoration(borderRadius: BorderRadius.circular(10), color:
                    Theme.of(context).primaryColor),
            padding: const
                EdgeInsets.symmetric(vertical: 15, horizontal: 30),
            child: Text(
                "Save Location",
                textAlign:
                    TextAlign.center,
                style: Theme.of(context)
                    .textTheme
                    .headline6!
                    .copyWith(color:
                        Theme.of(context).colorScheme.onSurface),
            ),
        )),
    ),
)
```

Clearing the Cart

To clear the cart, at the beginning of the save function it will reset the cart in order to prevent incorrect ordering if something happens to the save process.

After implementing the following code, it ran without any issues, resetting the cart.

```
Future<bool> saveLocation() async {
```

```
try {

    await SQLiteCartItems.clearCart();

    final prefs = await SharedPreferences.getInstance();

    await prefs.setDouble('locationLatitude', cameraPosition.target.latitude);

    await prefs.setDouble('locationLongitude',
cameraPosition.target.longitude);

    await prefs.setStringList('locationPlacemark',
<String>[addresslineone.text, addresslinetwo.text, city.text, postcode.text]);




    return true;
}

} catch (e) {

    return false;
}

}
```

Sending Cart Data to Checkout

To send the cart data to the checkout, I will start by making the checkout file

```
import 'package:alleat/widgets/elements/elements.dart';

import 'package:flutter/material.dart';




class Checkout extends StatefulWidget {

    const Checkout({super.key});

}
```

```

@Override

State<Checkout> createState() => _CheckoutState();

}

class _CheckoutState extends State<Checkout> {

@Override

Widget build(BuildContext context) {

return Scaffold(

body: SingleChildScrollView(


child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [


const ScreenBackButton(),

Text(


"Checkout",


style: Theme.of(context).textTheme.headline1,


)

])));

}

}

```

The main piece which needs to be moved over is the cartInfo as it contains all the information about the cart. By not having it rerun in the checkout, the details can be saved and not modified.

```
class Checkout extends StatefulWidget {
```

```

final List cartInfo;

const Checkout({Key? key, required this.cartInfo}) : super(key: key);

@override

State<Checkout> createState() => _CheckoutState();
}

```

In the cart, since the checks are done previously, there doesn't need to be any if statements within the button

```

Expanded(
    child: ElevatedButton(
        onPressed: () {
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder:
                        (context) => Checkout(
                            cartInfo: cartInfo[0]["cartinfo"],
                            )));
        },
        child: const
Text("Checkout")))

```

Checkout

The checkout section is built from a few different modules.

- Destination
- Subtotal
- Delivery price
- Changeable tip value
- Total
- Button to payment provider (will not be implemented for security)

Delivery Destination

The destination can be grabbed using a FutureBuilder with the location being stored in the shared preferences under the key "locationPlacemark" and the lat and long of "locationLatitude" & "locationLongitude". I will use a FutureBuilder since it can run asynchronous code and wait for the shared preferences to return the value stored.

```
Future<List> getDeliveryDestination() async {

    final prefs = await SharedPreferences.getInstance(); // Get saved location
from shared preferences

    final double? savedLocationLat = prefs.getDouble('locationLatitude');

    final double? savedLocationLng = prefs.getDouble('locationLongitude');

    final List<String>? savedLocationText =
prefs.getStringList('locationPlacemark');

    if (savedLocationLat == null || savedLocationLng == null || savedLocationText
== null) {

        return [];
    } else {

        savedLocationText.addAll([savedLocationLat.toString(),
        savedLocationLng.toString()]);
    }

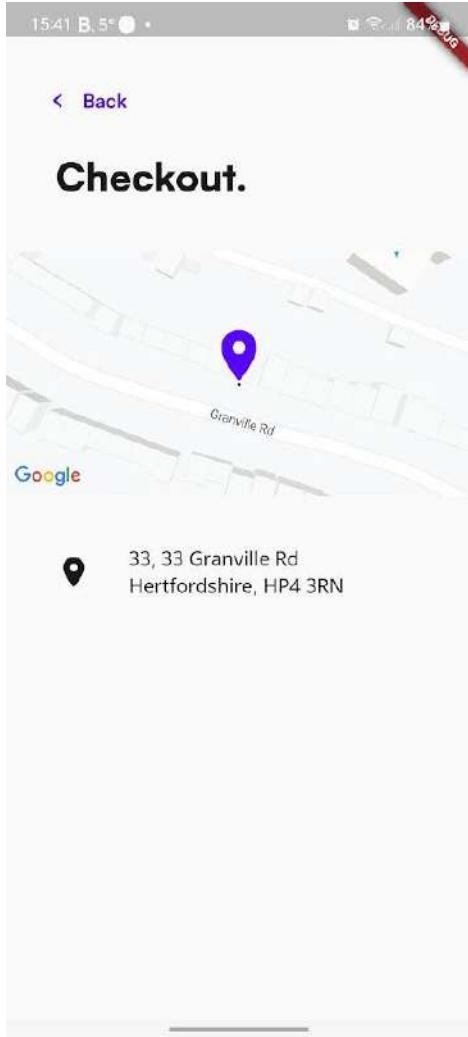
    return savedLocationText.toList();
}
```

```
    }  
  
}
```

```
FutureBuilder<List>(  
  
    future: getDeliveryDestination(),  
  
    builder: (context, snapshot) {  
  
        if (snapshot.hasData) {  
  
            List Destination = snapshot.data ?? [];  
  
            return Text(Destination.toString());  
  
        } else {  
  
            return Text("Getting Destination");  
  
        }  
  
    },  
  
) ,
```

In order for the user to know if they have got the right location, it will display a small image of the destination with the destination displayed under.

To do this, I used the same code used with the map for location setting but removed the movement code so that the map cannot be moved



```
if (snapshot.hasData) {  
  
    List destination = snapshot.data ?? [];  
  
    if (destination != []) {  
  
        final _controller = Completer<GoogleMapController>();  
  
        MapPickerController mapPickerController = MapPickerController();  
  
        final double destinationLat = double.parse(destination[4]);  
  
        final double destinationLng = double.parse(destination[5]);
```

```
    CameraPosition cameraPosition = CameraPosition(target:  
LatLng(destinationLat, destinationLng), zoom: 18);  
  
    return Column(children: [  
  
      SizedBox(  
  
        width: double.infinity,  
  
        height: 200,  
  
        child: LayoutBuilder(builder: (BuildContext context,  
BoxConstraints constraints) {  
  
          return MapPicker(  
  
            // pass icon widget  
  
            iconWidget: Icon(  
  
              Icons.location_on,  
  
              size: 50,  
  
              color: Theme.of(context).primaryColor,  
  
,  
  
            //add map picker controller  
  
            mapPickerController: mapPickerController,  
  
            child: GoogleMap(  
  
              myLocationEnabled: false,  
  
              scrollGesturesEnabled: false,  
  
              onCameraMove: (position) {  
  
                null;  
  
              },  
            ),  
          );  
        },  
      ),  
    ],  
  );  
}
```

```
        compassEnabled: false,  
  
        mapToolbarEnabled: false,  
  
        zoomControlsEnabled: false,  
  
        // hide location button  
  
        myLocationButtonEnabled: false,  
  
        mapType: MapType.normal,  
  
  
  
        rotateGesturesEnabled: false,  
  
        tiltGesturesEnabled: false,  
  
        // camera position  
  
        initialCameraPosition: cameraPosition,  
  
        onMapCreated: (GoogleMapController controller) {  
  
            _controller.complete(controller);  
  
        },  
  
    )),  
  
,  
  
const SizedBox(  
  
    height: 20,  
  
,  
  
Padding(  

```

```
padding: const EdgeInsets.symmetric(horizontal: 40, vertical:  
20),  
  
        child: Row(  
  
            children: [  
  
                Icon(  
  
                    Icons.location_on,  
  
                    size: 30,  
  
                    color: Theme.of(context).colorScheme.onBackground,  
                ),  
  
                const SizedBox(  
  
                    width: 30,  
                ),  
  
                Expanded(  
  
                    child: Column(  
  
                        crossAxisAlignment: CrossAxisAlignment.start,  
  
                        children: [  
  
                            Text(  
  
                                "${destination[0]}, ${destination[1]}",  
  
                                style: Theme.of(context).textTheme.bodyText2,  
                            ),  
  
                            Text(  
  
                                "${destination[2]}, ${destination[3]}",  
  
                                style: Theme.of(context).textTheme.bodyText2,  
                            ),  
                        ],  
                    ),  
                ),  
            ],  
        ),  
    ),  
);
```

```

        ),
        ],
        ))
    ],
    ))
]);
} else {
    return Text("Failed to get location");
}

```

Restaurant Location

I noticed that it is missing the restaurant location so in order for the user to know how far the restaurant is away, I changed the cart to request the location, latitude and longitude.

I started by changing the server cart file

```

if ($type == "item"){
    $term = intval($term);
    $sqlitem = "SELECT item.itemid, item.itemname, item.price,
item.itemimage, restaurant.restaurantid, restaurant.restaurantname,
restaurant.delivery, restaurant.ordermin, restaurant.address,
restaurant.latitude, restaurant.longitude FROM item AS item INNER JOIN
menucategories AS menucategories ON item.categoryid =
menucategories.categoryid INNER JOIN restaurant AS restaurant ON
menucategories.restaurantid = restaurant.restaurantid WHERE item.itemid =
$term";

    if($result = mysqli_query($link, $sqlitem)){
        $iteminfo = [];
        while($row = mysqli_fetch_assoc($result)) {
            array_push($iteminfo, $row["itemid"], $row["itemname"],
$row["price"], $row["itemimage"], $row["restaurantid"],

```

```

$row["restaurantname"], $row["delivery"], $row["ordermin"],
$row["address"], $row["latitude"], $row["longitude"]);
}
$return["message"] = $iteminfo;

} else{
$return["error"] = true;
$return["message"] = "Failed item query";
}

```

I then added the address and latitude and longitude to the list in the cart file

```

tempItemInfo[profileCart[i]["itemid"]][0].addAll([
    basicItemInfo["message"]["message"][1], //Item name
    basicItemInfo["message"]["message"][3], //Item image link
    basicItemInfo["message"]["message"][2], //Item price
    basicItemInfo["message"]["message"][5], //Restaurant name
    basicItemInfo["message"]["message"][4], //Restaurant id
    basicItemInfo["message"]["message"][6], //Delivery price
    profileCart[i]["quantity"], //Item quantity
    profileCart[i]["cartid"], //Cart ID
    basicItemInfo["message"]["message"][7], //Minimum order price for
    restaurant
    basicItemInfo["message"]["message"][8], //Restaurant address
    basicItemInfo["message"]["message"][9], //Restaurant location
    latitude
    basicItemInfo["message"]["message"][10] //Restaurant location
    longitude
]);

```

Within the map section of the checkout, I added a function to get the latitude and longitude of the restaurant

```
List locationItemKeys = widget.cartInfo[0][6].keys.toList();

    final String restaurantAddress =
widget.cartInfo[0][6][locationItemKeys[0]][0][9];

    final double restaurantLat =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][10]);

    final double restaurantLng =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][11]);
```

I then adjusted my map code to include the destination of both the restaurant and the delivery destination. Since it was just using Google Maps, the selector package was removed from the code.

In order to add multiple markers, I used the following resource to help build the structure of the Map

<https://www.fluttercampus.com/guide/73/how-to-add-multiple-markers-on-google-map-flutter/>

I was required to change the single marker made in the location selector to now have a separate function to assign the markers to a list then assign them to the map

```
Set<Marker> getmarkers(restaurantAddress, restaurantLat, restaurantLng,
destination, destinationLat, destinationLng) {

    //markers to place on map

    markers.add(Marker(
        //add first marker
    ));
```

```
markerId: const MarkerId("Restaurant"),  
  
position: LatLng(restaurantLat, restaurantLng), //position of restaurant  
  
infoWindow: const InfoWindow(  
  
    //popup info  
  
    title: 'Restaurant',  
  
,  
  
icon: BitmapDescriptor.defaultMarker, //Icon for restaurant  
));  
  
  
  
markers.add(Marker(  
  
    markerId: const MarkerId("Delivery Address"),  
  
    position: LatLng(destinationLat, destinationLng), //position of delivery  
address  
  
    infoWindow: const InfoWindow(  
  
        //popup info  
  
        title: 'Delivery Address',  
  
,  
  
icon: BitmapDescriptor.defaultMarker, //Icon for delivery address  
));  
  
  
  
return markers;  
}
```

```
List locationItemKeys = widget.cartInfo[0][6].keys.toList();

    final List restaurantAddress =
widget.cartInfo[0][6][locationItemKeys[0]][0][9].split(',');

    final double restaurantLat =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][10]);

    final double restaurantLng =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][11]);

    final controllerMap = Completer<GoogleMapController>(); //Google
Maps Controller

    final double destinationLat = double.parse(destination[4]);

    final double destinationLng = double.parse(destination[5]);

    CameraPosition cameraPosition = CameraPosition(target:
LatLng(destinationLat, destinationLng), zoom: 18);

return Column(children: [
    SizedBox(
        width: double.infinity,
        height: 200,
        child: LayoutBuilder(builder: (BuildContext context,
BoxConstraints constraints) {
            return GoogleMap(
                myLocationEnabled: false,
                scrollGesturesEnabled: false,
                compassEnabled: false,
                mapToolbarEnabled: false,
                zoomControlsEnabled: false,

```

```
// hide location button

myLocationButtonEnabled: false,

mapType: MapType.normal,

rotateGesturesEnabled: false,

tiltGesturesEnabled: false,

// camera position

initialCameraPosition: cameraPosition,

onMapCreated: (GoogleMapController controller) {

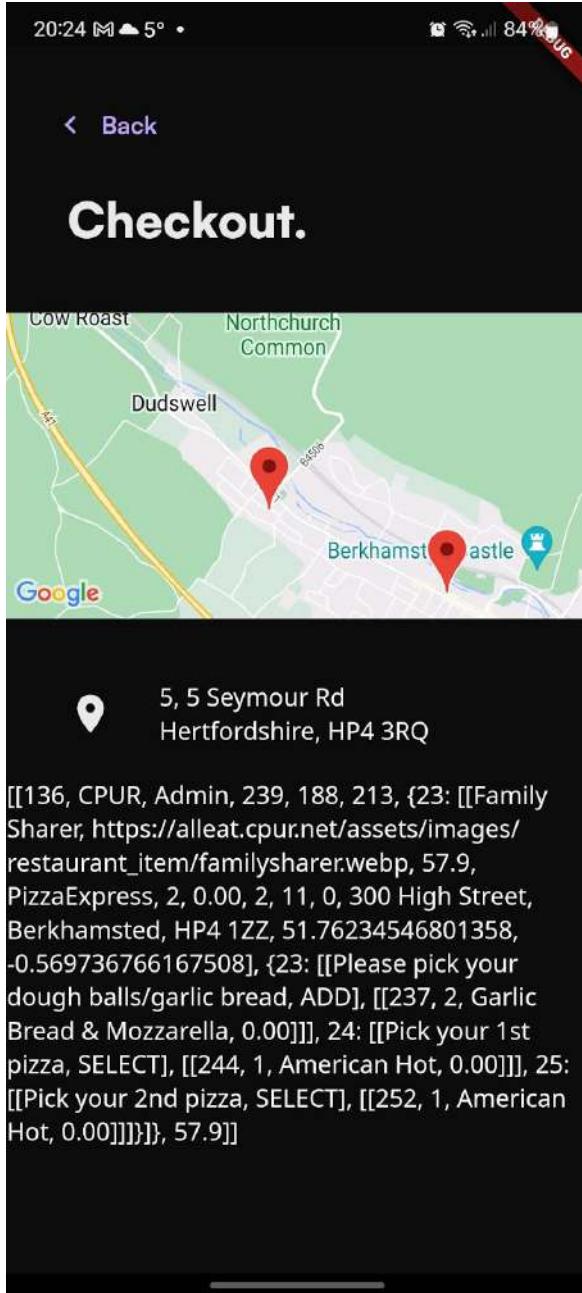
    controllerMap.complete(controller);

},

markers: getmarkers(restaurantAddress, restaurantLat,
restaurantLng, destination, destinationLat, destinationLng));

}),

),
```



An issue I faced was getting the destination and the restaurant to show on the same map. To fix this, I found the midpoint between these two and then make the map zoom out until both were visible.

```
CameraPosition(target: LatLng((destinationLat + restaurantLat) / 2,  
(destinationLng + restaurantLng) / 2), zoom: 12);
```

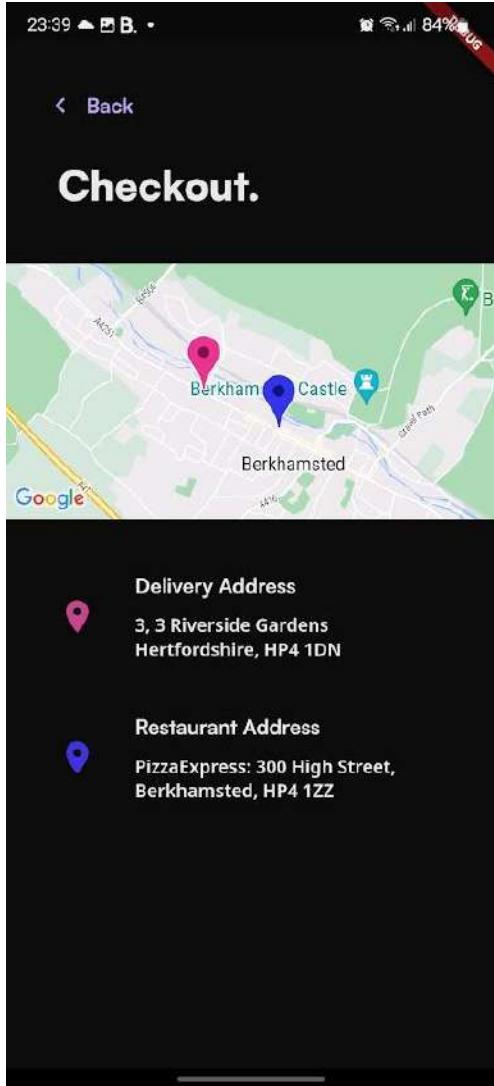
After trying to figure out how to get the map to bound multiple markers to the screen, I found that it would be very difficult and so to fix this, I will set it so that by default it shows the destination but it can be moved to show the restaurant.

The Google Map also uses a different icon format to the map selector, an asset icon has to be imported as an image not an icon. I used the following to assist me in this.

<https://medium.com/flutter-community/ad-custom-marker-images-for-your-google-maps-in-flutter-68ce627107fc>

Using Adobe Illustrator, I made an icon for the restaurant and one for the delivery destination

After much trial and error, I could not figure out a solution where it would not require a FutureBuilder since the result was asynchronous. In the end, I decided that the best thing to do was to colour code the default icons with the location



```
Set<Marker> getmarkers(restaurantAddress, restaurantLat, restaurantLng,
destination, destinationLat, destinationLng) {

    //markers to place on map

    BitmapDescriptor restaurantMarkerIcon =
BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueBlue);

    BitmapDescriptor destinationMarkerIcon =
BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueRose);

    markers.add(Marker(
        //add first marker
```

```
markerId: const MarkerId("Restaurant"),  
  
position: LatLng(restaurantLat, restaurantLng), //position of restaurant  
  
infoWindow: const InfoWindow(  
  
//popup info  
  
title: 'Restaurant',  
  
,  
  
icon: restaurantMarkerIcon //Icon for restaurant  
  
));  
  
  
markers.add(Marker(  
  
markerId: const MarkerId("Delivery Address"),  
  
position: LatLng(destinationLat, destinationLng), //position of delivery  
address  
  
infoWindow: const InfoWindow(  
  
//popup info  
  
title: 'Delivery Address',  
  
,  
  
icon: destinationMarkerIcon //Icon for delivery address  
  
));  
  
  
return markers;  
}
```

```
Widget build(BuildContext context) {  
  
    return Scaffold(  
  
        body: SingleChildScrollView(  
  
            child: Column(crossAxisAlignment: CrossAxisAlignment.start, children:  
[  
  
    const ScreenBackButton(),  
  
    Padding(  
  
        padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 10),  
  
        child: Text(  
  
            "Checkout.",  
  
            style: Theme.of(context).textTheme.headline1,  
  
        )),  
  
    const SizedBox(  
  
        height: 30,  
  
    ),  
  
    FutureBuilder<List>(  
  
        future: getDeliveryDestination(),  
  
        builder: (context, snapshot) {  
  
            if (snapshot.hasData) {  
  
                List destination = snapshot.data ?? [];  
  
                if (destination != []) {  
  
                    List locationItemKeys = widget.cartInfo[0][6].keys.toList();  
                }  
            }  
        },  
    ),  
],  
);  
}  
  
void getDeliveryDestination() {  
  
    FirebaseFirestore.instance.collection("Delivery").get().then((value) {  
  
        List destination = value.docs.map((e) {  
            return e.data();  
        }).toList();  
  
        setState(() {  
            deliveryDestination = destination;  
        });  
    }).catchError((error) {  
        print("Error getting documents: $error");  
    });  
}
```

```
        final List restaurantAddress =
widget.cartInfo[0][6][locationItemKeys[0]][0][9].split(',');

        final double restaurantLat =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][10]);

        final double restaurantLng =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][11]);

        final controllerMap = Completer<GoogleMapController>(); //Google
Maps Controller

        final double destinationLat = double.parse(destination[4]);
        final double destinationLng = double.parse(destination[5]);



return Column(children: [
    SizedBox(
        width: double.infinity,
        height: 200,
        child: LayoutBuilder(builder: (BuildContext context,
BoxConstraints constraints) {
            return GoogleMap(
                myLocationEnabled: false,
                compassEnabled: false,
                mapToolbarEnabled: false,
                zoomGesturesEnabled: true,
                // hide location button
                myLocationButtonEnabled: false,

```

```
        mapType: MapType.normal,  
  
        zoomControlsEnabled: false,  
  
        rotateGesturesEnabled: false,  
  
        tiltGesturesEnabled: false,  
  
        // camera position  
  
        initialCameraPosition: CameraPosition(target:  
LatLng(destinationLat, destinationLng), zoom: 12),  
  
        onMapCreated: (GoogleMapController controller) {  
  
            controllerMap.complete(controller);  
  
        },  
  
        markers: getmarkers(restaurantAddress, restaurantLat,  
restaurantLng, destination, destinationLat, destinationLng));  
  
    }),  
  
(,),  
  
const SizedBox(  
  
height: 20,  
  
(,),  
  
Padding(  
  
padding: const EdgeInsets.symmetric(horizontal: 40, vertical:  
20),  
  
child: Row(  
  
children: [  
  
const Icon(  
  
Icons.location_on,
```

```
        size: 30,  
  
        color: Color(0xffca458f),  
  
,  
  
const SizedBox(  
  
        width: 30,  
  
,  
  
Expanded(  
  
        child: Column(  
  
        crossAxisAlignment: CrossAxisAlignment.start,  
  
        children: [  
  
        Text(  
  
        "Delivery Address",  
  
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
,  
  
        const SizedBox(  
  
        height: 10,  
  
,  
  
        Text(  
  
        "${destination[0]}, ${destination[1]}",  
  
        style: Theme.of(context).textTheme.bodyText1,  
  
,
```

```
        Text(
            "${destination[2]}, ${destination[3]}",
            style: Theme.of(context).textTheme.bodyText1,
        ),
    ],
))
],
),
Padding(
    padding: const EdgeInsets.symmetric(horizontal: 40, vertical:
20),
    child: Row(
        children: [
            const Icon(
                Icons.location_on,
                size: 30,
                color: Color(0xff4133e3),
            ),
            const SizedBox(
                width: 30,
            ),
            Expanded(
                child: Column(

```

```
crossAxisAlignment: CrossAxisAlignment.start,  
children: [  
    Text(  
        "Restaurant Address",  
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
,  
    const SizedBox(  
        height: 10,  
,  
    ),  
    Text(  
        "${widget.cartInfo[0][6][locationItemKeys[0]][0][3]}:  
${widget.cartInfo[0][6][locationItemKeys[0]][0][9]}",  
        style: Theme.of(context).textTheme.bodyText1,  
,  
    ],  
    ))  
,  
)),  
]);  
}  
else {  
    return const Text("Failed to get location");  
}
```

```

        }

    } else {

        return const Text("Getting Destination");

    }

},
),
])));

}

```

Tipping

After some analysis of POS systems, I found that the way it usually works is that there are buttons with set amounts, 15%, 20%, 25%, custom tip and no tip. This means that the user is mutually obliged to tip 20%. To make this work, I will create Layoutbuilder and calculate the total price then multiply it by 0.15, 0.2 & 0.25.

```

LayoutBuilder(builder: (p0, p1) {

    double subtotal = 0;

    for (int i = 0; i < widget.cartInfo.length; i++) {

        subtotal += widget.cartInfo[0][7];

    }

    return Padding(
        padding: const EdgeInsets.symmetric(horizontal: 10),

        child: Column(children: [
            Row(

```

```
children: [  
    Expanded(  
        child: Padding(  
            padding: const EdgeInsets.all(10),  
            child: Container(  
                alignment: Alignment.center,  
                decoration:  
                    BoxDecoration(color:  
Theme.of(context).colorScheme.onSurface, borderRadius: BorderRadius.circular(10),  
boxShadow: [  
    BoxShadow(  
        color: Colors.black.withOpacity(0.01),  
        spreadRadius: 2,  
        blurRadius: 10,  
        offset: const Offset(0, 10), // changes  
position of shadow  
    ),  
],  
        child: AspectRatio(  
            aspectRatio: 1 / 1,  
            child: Column(  
                mainAxisAlignment: MainAxisAlignment.center,  
                children: [  
                    Text(  

```

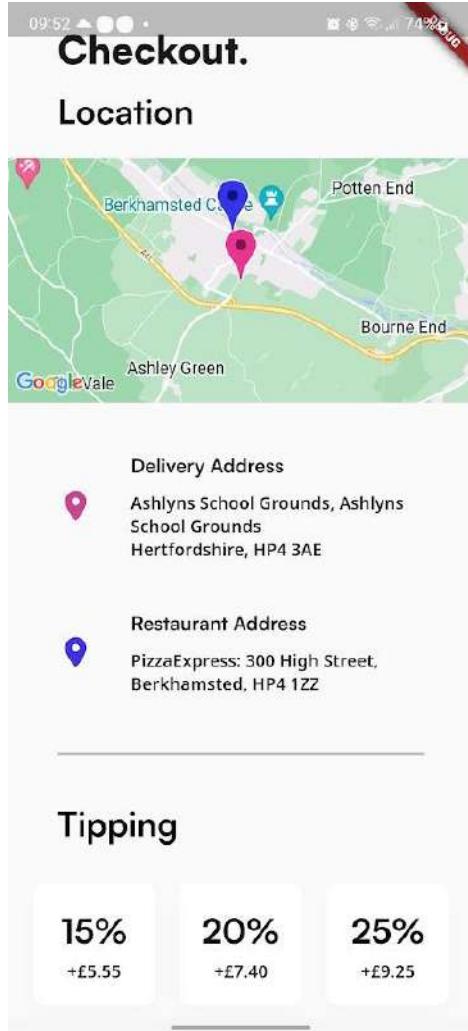
```
        "15%",

        style:
Theme.of(context).textTheme.headline2,
        ),
        const SizedBox(
            height: 5,
        ),
        Text(
            "+\${(((subtotal * 100) * 0.15) /
100).toStringAsFixed(2)}",
            style:
Theme.of(context).textTheme.bodyText1,
        )
    ],
)))),
Expanded(
    child: Padding(
        padding: const EdgeInsets.all(10),
        child: Container(
            alignment: Alignment.center,
            decoration:
BoxDecoration(color:
Theme.of(context).colorScheme.onSurface, borderRadius: BorderRadius.circular(10),
boxShadow: [
```

```
BoxShadow(  
    color: Colors.black.withOpacity(0.01),  
    spreadRadius: 2,  
    blurRadius: 10,  
    offset: const Offset(0, 10), // changes  
position of shadow  
,  
],  
child: AspectRatio(  
    aspectRatio: 1 / 1,  
    child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
            Text(  
                "20%",  
                style:  
Theme.of(context).textTheme.headline2,  
,  
const SizedBox(  
    height: 5,  
,  
Text(  
    "+${((subtotal * 100) * 0.20) /  
100).toStringAsFixed(2)}",
```

```
        style:  
Theme.of(context).textTheme.bodyText1,  
        )  
    ],  
))));  
Expanded(  
    child: Padding(  
        padding: const EdgeInsets.all(10),  
        child: Container(  
            alignment: Alignment.center,  
            decoration:  
                BoxDecoration(color:  
Theme.of(context).colorScheme.onSurface, borderRadius: BorderRadius.circular(10),  
boxShadow: [  
            BoxShadow(  
                color: Colors.black.withOpacity(0.01),  
                spreadRadius: 2,  
                blurRadius: 10,  
                offset: const Offset(0, 10), // changes  
position of shadow  
            ),  
        ]),  
        child: AspectRatio(  
            aspectRatio: 1 / 1,  
        )  
    )  
);
```

```
        child: Column(  
  
            mainAxisAlignment: MainAxisAlignment.center,  
  
            children: [  
  
                Text(  
  
                    "25%",  
  
                    style:  
Theme.of(context).textTheme.headline2,  
  
                ),  
  
                const SizedBox(  
  
                    height: 5,  
  
                ),  
  
                Text(  
  
                    "+£${(((subtotal * 100) * 0.25) /  
100).toStringAsFixed(2)}",  
  
                    style:  
Theme.of(context).textTheme.bodyText1,  
  
                )  
  
            ],  
  
        )))),  
  
    ],  
  
)  
  
]);  
})
```



I also added a no tip and custom tip amount buttons.

```
Padding(  
    padding: EdgeInsets.all(10),  
  
    child: Container(  
  
        alignment: Alignment.center,  
  
        height: 50,  
  
        decoration: BoxDecoration(color:  
Theme.of(context).colorScheme.onSurface, borderRadius: BorderRadius.circular(10),  
boxShadow: [
```

```
        BoxShadow(  
  
            color: Colors.black.withOpacity(0.01),  
  
            spreadRadius: 2,  
  
            blurRadius: 10,  
  
            offset: const Offset(0, 10), // changes position of  
shadow  
  
  
    )),  
  
    child: Text(  
  
        "Custom Tip Amount",  
  
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline3?.color),  
  
    )),  
  
Padding(  
  
    padding: EdgeInsets.all(10),  
  
    child: Container(  
  
        alignment: Alignment.center,  
  
        height: 50,  
  
        decoration: BoxDecoration(color:  
Theme.of(context).colorScheme.onSurface, borderRadius: BorderRadius.circular(10),  
boxShadow: [  
  
        BoxShadow(  
  
            color: Colors.black.withOpacity(0.01),
```

```
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes position of  
shadow  
  
    ),  
  
    ]),  
  
    child: Text(  
  
        "No Tip",  
  
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline3?.color),  
  
    )))  
  
]);
```



Delivery Address

 Ashlyns School Grounds, Ashlyns
School Grounds
Hertfordshire, HP4 3AE

Restaurant Address

 PizzaExpress: 300 High Street,
Berkhamsted, HP4 1ZZ

Tipping

15% 20% 25%

+£2.77

+£3.70

+£4.63

Custom Amount

No Tip

To then add functionality, I first made all containers be InkWells, I then made a variable to store the currently selected option. Depending on the option, each button will check if it has been selected and display it with a different outline.

Expanded(

```
        child: Padding(  
  
          padding: const EdgeInsets.all(10),  
  
          child: InkWell(  
  
            onTap: () {
```

```
        setState(() {
            selected = "25%";
        });
    },
    child: Container(
        alignment: Alignment.center,
        decoration: BoxDecoration(
            color:
Theme.of(context).colorScheme.onSurface,
            borderRadius: BorderRadius.circular(10),
            border: Border.all(
                color: (selected == "25%") ?
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,
                width: 2),
            boxShadow: [
                BoxShadow(
                    color: Colors.black.withOpacity(0.01),
                    spreadRadius: 2,
                    blurRadius: 10,
                    offset: const Offset(0, 10), // changes
position of shadow
                ),
            ],
            child: AspectRatio(
```

```

        aspectRatio: 1 / 1,

        child: Column(
          mainAxisAlignment:
MainAxisAlignment.center,
          children: [
            Text(
              "25%",
              style:
Theme.of(context).textTheme.headline3,
            ),
            const SizedBox(
              height: 5,
            ),
            Text(
              "+\${(((subtotal * 100) * 0.25) /
100).toStringAsFixed(2)}",
              style:
Theme.of(context).textTheme.bodyText1,
            )
          ],
        )))))));

```

For setting a custom amount, I used the same method used for setting the border but for the child widget of the container. In this way, it removes the need for a LayoutBuilder since the logic is within the container.

Using the following package allowed me to only allow money formatting

https://pub.dev/packages/currency_text_input_formatter

By using this package, it only allowed numbers as well as formatting the text for the user. They just enter the numbers and it enters from the right.

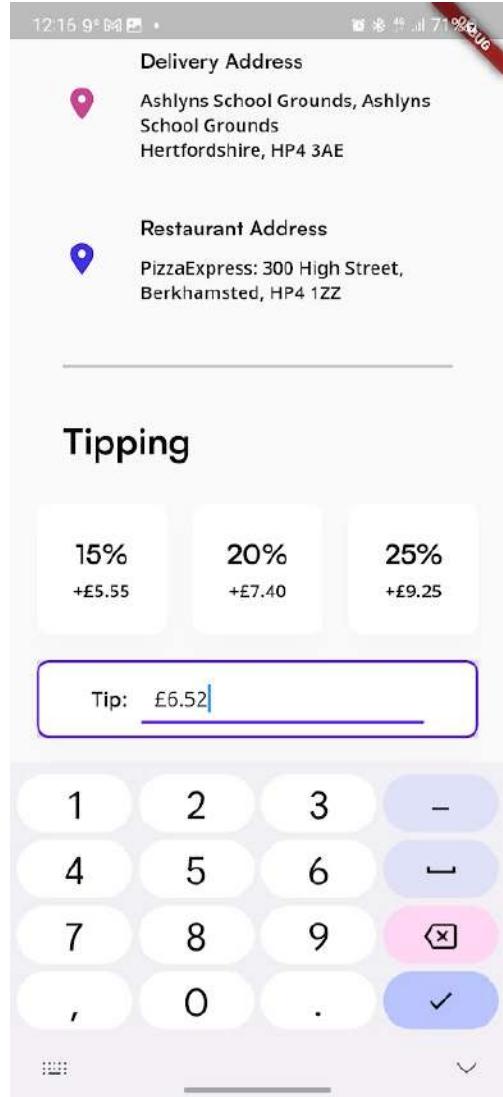
```
Padding(  
  padding: const EdgeInsets.all(10),  
  child: InkWell(  
    onTap: () {  
      setState(() {  
        selected = "custom";  
      });  
    },  
    child: Container(  
      alignment: Alignment.center,  
      height: 60,  
      decoration: BoxDecoration(  
        color: Theme.of(context).colorScheme.onSurface,  
        border: Border.all(  
          color: (selected == "custom") ?  
            Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface, width:  
            2),  
        borderRadius: BorderRadius.circular(10),  
        boxShadow: [  
          BoxShadow(  
        ]  
      )  
    )  
  )  
)
```

```
        color: Colors.black.withOpacity(0.01),  
  
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes  
position of shadow  
  
  
    ],  
  
    child: (selected == "custom")  
    ? Padding(  
  
        padding: const EdgeInsets.symmetric(horizontal:  
40, vertical: 10),  
  
        child: Row(  
  
            children: [  
  
                Text(  
  
                    "Tip:",  
  
                    style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
                ),  
  
                const SizedBox(  
  
                    width: 10,  
  
                ),  
  
                Expanded(  
  
                    child: TextFormField(  

```

```
        controller: customAmount,  
  
        keyboardType: TextInputType.number,  
  
        style:  
Theme.of(context).textTheme.bodyText2,  
  
        inputFormatters:  
[CurrencyTextInputFormatter(symbol: '£')],  
  
        decoration: (InputDecoration(  
  
            hintText: "£3.21",  
  
            contentPadding:  
Theme.of(context).inputDecorationTheme.contentPadding,  
  
            border:  
Theme.of(context).inputDecorationTheme.border,  
  
            focusedBorder:  
Theme.of(context).inputDecorationTheme.focusedBorder,  
  
            enabledBorder:  
Theme.of(context).inputDecorationTheme.enabledBorder,  
  
            floatingLabelBehavior:  
FloatingLabelBehavior.never))),  
  
    )  
  
],  
  
))  
  
: Text(  
  
    "Custom Tip Amount",  
  
    style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline3?.color),
```

```
    )))),
```



To then calculate the tip price, at the end of the layout builder, I added a calculation which sets the tip price to the new value.

While trying to calculate the custom price, although the formatting worked visually, when trying to get the value it took a lot of manipulation.

```
Padding(
```

```
    padding: const EdgeInsets.all(10),
```

```
        child: InkWell(
```

```
            onTap: () {
```

```
                setState(() {
```

```
                    selectedTip = "custom";
```

```
                    print(customAmount.text.split("£"));
```

```
                    try {
```

```
                        tipPrice =
```

```
                            double.parse((double.parse((customAmount.text.toString().split("£"))[1])).toStringAsFixed(2));
```

```
                    } catch (e) {
```

```
                        tipPrice = 0;
```

```
                    }
```

```
                };
```

```
            },
```

```
            child: Container(
```

```
                alignment: Alignment.center,
```

```
                height: 60,
```

```
                decoration: BoxDecoration(
```

```
                    color: Theme.of(context).colorScheme.onSurface,
```

```
                    border: Border.all(
```

```
                        color: (selectedTip == "custom") ?
```

```
                            Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,
```

```
                        width: 2),
```

```
                    borderRadius: BorderRadius.circular(10),
```

```
        boxShadow: [  
  
            BoxShadow(  
  
                color: Colors.black.withOpacity(0.01),  
  
                spreadRadius: 2,  
  
                blurRadius: 10,  
  
                offset: const Offset(0, 10), // changes  
position of shadow  
  
            ),  
  
        ],  
  
        child: (selectedTip == "custom")  
        ? Padding(  
  
            padding: const EdgeInsets.symmetric(horizontal:  
40, vertical: 10),  
  
            child: Row(  
  
                children: [  
  
                    Text(  
  
                        "Tip:",  
  
                        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
                ),  
  
                const SizedBox(  
  
                    width: 10,  
  
                ),  
  
            ),  
  
        ),  
  
    ),  
  
);
```

```
        Expanded(  
          child: TextFormField(  
            onChanged: (value) {  
              try {  
                tipPrice =  
  
double.parseDouble(double.parseDouble(customAmount.text.toString().split("£"))[1])).toStringAsFixed(2));  
  
            } catch (e) {  
              tipPrice = 0;  
  
            }  
  
,  
            controller: customAmount,  
            keyboardType: TextInputType.number,  
            style:  
Theme.of(context).textTheme.bodyText2,  
            inputFormatters:  
[CurrencyTextInputFormatter(symbol: '£')],  
            decoration: (InputDecoration(  
              hintText: "£3.21",  
              contentPadding:  
Theme.of(context).inputDecorationTheme.contentPadding,  
              border:  
Theme.of(context).inputDecorationTheme.border,
```

```

        focusedBorder:
Theme.of(context).inputDecorationTheme.focusedBorder,
                enabledBorder:
Theme.of(context).inputDecorationTheme.enabledBorder,
                floatingLabelBehavior:
FloatingLabelBehavior.never))),
)
],
))
: Text(
    "Custom Tip Amount",
    style:
Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).textTheme.headline3?.color),
    ))),

```

As well, when displaying the price for the percentages, it would sometimes give a price to 3dp so to fix this I first get the value as a double then change it to a fixed string of 2dp and then parsed back to a double.

```
tipPrice = double.parse(((subtotal * 100) * 0.25) / 100).toStringAsFixed(2));
```

Displaying the price

To the price is split up into 4 pieces of information:

- Subtotal
- Delivery price
- Tip

- Final price

While it would have been great to include the price splitter, it would require payment options to be inputted which is not part of this prototype and due to the limited time remaining, it would require too much time to calculate the prices and make 2 new screens.

While trying to display the output, it looked like it was being modified by the tip since when a tip option was pressed, it was being added by almost £50. After some debugging of what changed the subtotal price, it was found that the subtotal is never reset when it is recalculating so when a tip button is pressed, it rebuilds the screen and as a result, the price was reset. To fix this, before the for loop is initiated for each profile, the subtotal is set to 0;

```
subtotal = 0;

for (int i = 0; i < widget.cartInfo.length; i++) {

    subtotal = (subtotal * 100 + widget.cartInfo[0][7] * 100) / 100;

}
```

To then display the price, I compiled all the previous data together

```
LayoutBuilder(builder: ((p0, p1) {

    List locationItemKeys = widget.cartInfo[0][6].keys.toList();

    double total = (subtotal * 100 +
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][5]) * 100 + tipPrice *
100) / 100;

    return Padding(
        padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 20),
        child: Row(mainAxisAlignment: MainAxisAlignment.end, children: [
            Row(
                children: [
                    Column(
```

```
crossAxisAlignment: CrossAxisAlignment.start,  
children: [  
    Text(  
        "Subtotal",  
        style: Theme.of(context).textTheme.headline6,  
    ),  
    Text(  
        "Delivery Fee",  
        style: Theme.of(context).textTheme.headline6,  
    ),  
    Text(  
        "Tip",  
        style: Theme.of(context).textTheme.headline6,  
    ),  
    const SizedBox(height: 20),  
    Text(  
        "Total",  
        style:  
            Theme.of(context).textTheme.headline5?.copyWith(color:  
                Theme.of(context).primaryColor),  
    )  
,  
],  
,
```

```

const SizedBox(
  width: 50,
),
Column(
  mainAxisAlignment: MainAxisAlignment.end,
  children: [
    Text(
      "\$${subtotal.toStringAsFixed(2)}",
      style: Theme.of(context).textTheme.bodyText2,
    ),
    Text(
      "\$${widget.cartInfo[0][6][locationItemKeys[0]][0][5].toString()}",
      style: Theme.of(context).textTheme.bodyText2,
    ),
    Text(
      "\$${tipPrice.toStringAsFixed(2)}",
      style: Theme.of(context).textTheme.bodyText2,
    ),
  ],
  const SizedBox(height: 20),
  Text("\$${total.toStringAsFixed(2)}",
    style:
    Theme.of(context).textTheme.headline5?.copyWith(color:
    Theme.of(context).textTheme.headline1?.color)),

```

```
],  
)  
],  
,  
]);  
}))
```

The screenshot shows a mobile application interface for a delivery order. At the top, it displays the delivery address: "63, 63 Granville Rd, Hertfordshire, HP4 3RN". Below this, there's a "Restaurant Address" section showing "PizzaExpress: 300 High Street, Berkhamsted, HP4 1ZZ" with a location pin icon. The main part of the screen is titled "Tipping" and features three buttons for different tip percentages: "15% +£2.70", "20% +£3.60", and "25% +£4.50". The "25%" button is highlighted with a blue outline. Below these buttons is a "Custom Tip Amount" input field. Underneath the input field is a "No Tip" button. At the bottom of the screen, a summary of the total bill is shown:

Subtotal	£18.00
Delivery Fee	£0.00
Tip	£4.50
Total	£22.50

Sending Cart Data to Server

The final step to complete the transaction is to send the cart data to the server. To get this done, I first need to create the button. Since there is no chance for the user to need to make changes that impact the result, there are no checks needed.

```
Row(mainAxisAlignment: MainAxisAlignment.center, children: [  
    Expanded(  
        child: Padding(  
            padding: const EdgeInsets.symmetric(vertical: 30, horizontal:  
20),  
            child: ElevatedButton(  
                onPressed: () {  
                    print("ok");  
                },  
                child: Text("Complete Order")))  
    )  
)
```

A Future was made to send the cart to the server

```
Future<Map> sendCart(cart, tip, address, latitude, longitude) async {  
  
    var res = await QueryServer.query("https://alleat.cpur.net/query/orders.php",  
{  
  
        //Send data to orders.php . If there is an error, it returns back the error  
        code  
  
        "type": "add",  
  
        "cart": cart,  
    })  
    return res;
```

```
        "tip": tip,  
        "address": address,  
        "latitude": latitude,  
        "longitude": longitude,  
    });  
  
    return res;  
  
}
```

With the button pressed, it should encode the cartInfo and then return the address as a single string separated by commas.

```
 onPressed: () async {  
  
    var encodedCart = json.encode(widget.cartInfo);  
  
    List destination = await getDeliveryDestination();  
  
    String destinationAddress = [destination[0],  
    destination[1], destination[2], destination[3]].join(" ,");  
  
    Map orderCreated = await sendCart(encodedCart, tipPrice,  
    destinationAddress, destination[4], destination[5]);  
  
},
```

After trying to run the code, I received an error that it had an error regarding json encoding where it was not able to encode multiple data types. To fix this I converted the variable to a string then encoded it. After trying to run it, I received another error where it would not allow doubles to be send. To fix this, I converted the tip price, longitude and latitude to a string.

```
[{"id": 1, "name": "John Doe", "age": 30, "city": "New York"}, {"id": 2, "name": "Jane Smith", "age": 25, "city": "Los Angeles"}, {"id": 3, "name": "Mike Johnson", "age": 35, "city": "Chicago"}, {"id": 4, "name": "Sarah Davis", "age": 28, "city": "Houston"}, {"id": 5, "name": "David Wilson", "age": 32, "city": "Phoenix"}]
```

Processing Cart Data

To start with, I made the orders.php file, validating if the data is there to start with.

```
<?php

$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDfIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["type"]);

if($val){
    if ($_POST["type"] == "add"){
        $addval = isset($_POST["cart"]) && isset($_POST["tip"]) &&
        isset($_POST["address"]) && isset($_POST["latitude"]) &&
        isset($_POST["longitude"]);
        if($addval){
            $type = $_POST["type"];
            $cart = $_POST["cart"];
            $tip = $_POST["tip"];
            $address = $_POST["address"];
            $latitude = $_POST["latitude"];
            $longitude = $_POST["longitude"];
        } else{$return[ "error" ] = true;
            $return[ "message" ] = "data not specified";}
    } else{
        $return[ "error" ] = true;
        $return[ "message" ] = "order query type unknown";
    }
}

}
```

```

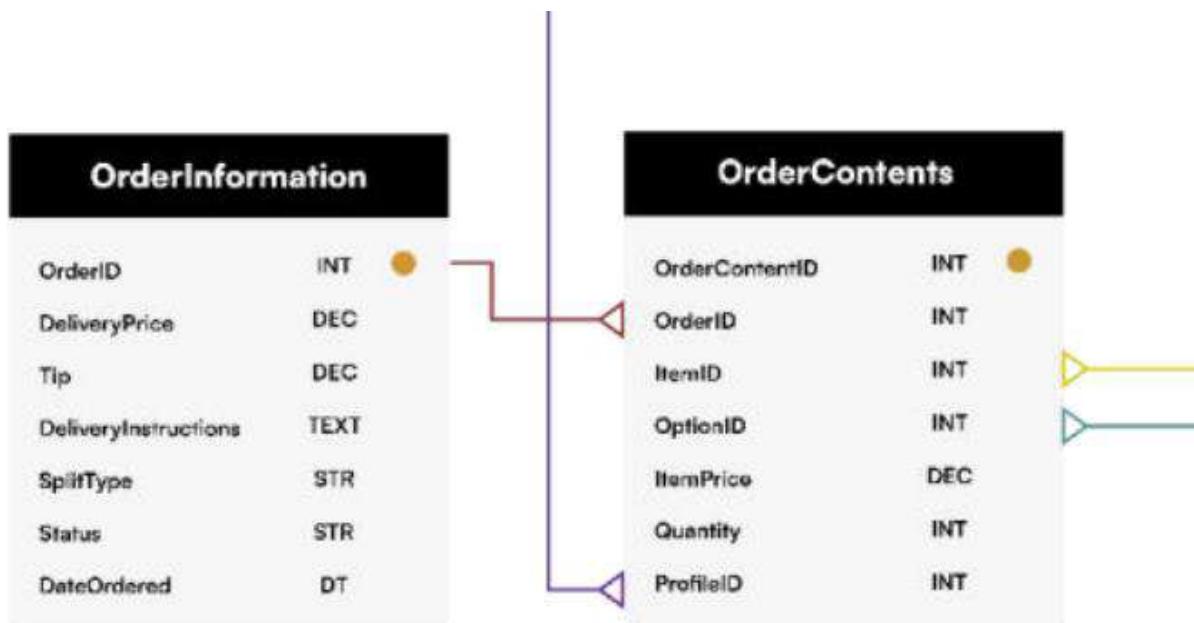
else{$return["error"] = true;
      $return[ "message" ] = "data not specified";}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>

```

To figure out how the data is formatted, I used the diagram made in prototype 1 showing how the tables are formatted.



For order information, delivery instructions have not been added to the app so it is not included. In order for the database to be 3rd normal form, it was found that the delivery price should be restaurant id instead since it is already included in the restaurant table. Split type is also not included as payments have not been implemented. By default status is set to “confirming”. The date and the order ID is also automatically filled when an order is added.

With the order contents table, it in fact needs to be 2 tables not one since each item has multiple customised options. To fix the table, I used the design made in the design stage. I modified the order information to include the delivery destination as well.

Order Information

OrderID	INC PRIMARY INT
RestaurantID	FOREIGN INT
Tip	DOUBLE (2,2)
Delivery Address	TEXT
Delivery Latitude	DOUBLE
Delivery Longitude	DOUBLE
Status	TEXT
DateOrdered	TIMESTAMP

Order Item Information

OrderItemID	INC PRIMARY INT
OrderID	FOREIGN INT
ProfileID	FOREIGN INT
ItemID	FOREIGN INT
Quantity	INT

Order Customised Item Information

CustomisedItemID	INC PRIMARY INT
OrderItemID	FOREIGN INT

CustomiseID	FOREIGN INT
Quantity	INT

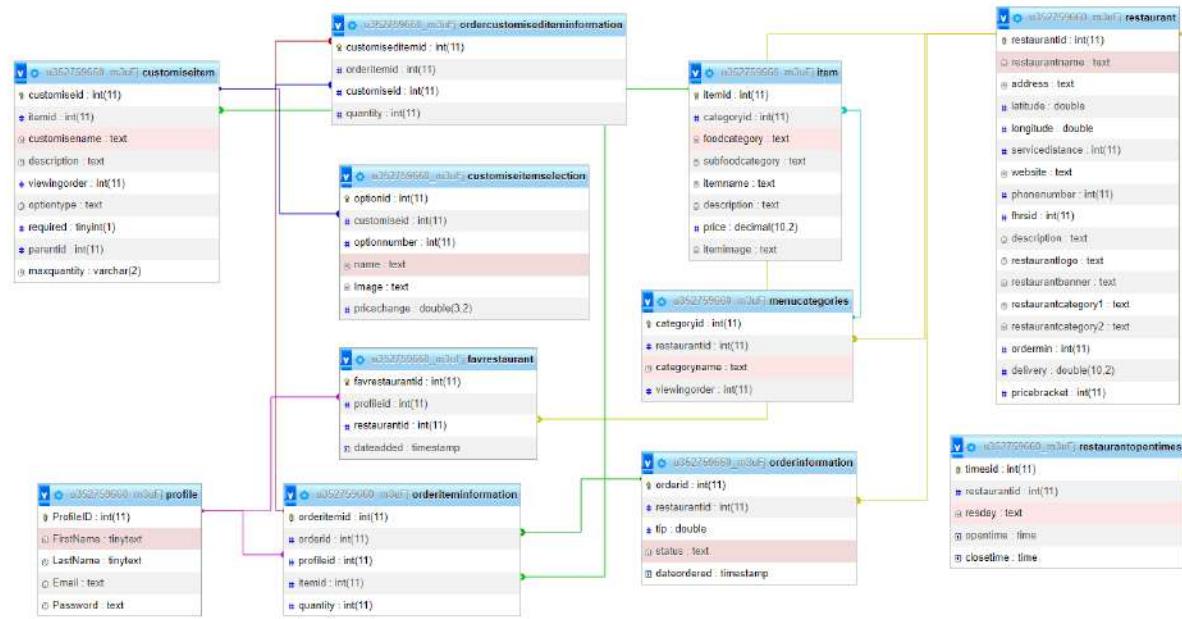
These tables were added to the database

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	orderid 🛡	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	restaurantid 🛡	int(11)			No	None			Change Drop More
3	tip	double			No	None			Change Drop More
4	deliveryaddress	text	utf8mb4_unicode_ci		No	None			Change Drop More
5	deliverylatitude	double			No	None			Change Drop More
6	deliverylongitude	double			No	None			Change Drop More
7	status	text	utf8mb4_unicode_ci		No	None			Change Drop More
8	dateordered	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Change Drop More

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	orderitemid 🛡	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	orderid 🛡	int(11)			No	None			Change Drop More
3	profileid 🛡	int(11)			No	None			Change Drop More
4	itemid 🛡	int(11)			No	None			Change Drop More
5	quantity	int(11)			No	None			Change Drop More

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	customiseditemid 🛡	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	orderitemid 🛡	int(11)			No	None			Change Drop More
3	customiseid 🛡	int(11)			No	None			Change Drop More
4	quantity	int(11)			No	None			Change Drop More

The relationship view is as follows:



With the tables being made, the next thing is to separate the cart info into separate variables and export them to the tables.

By using if statements and splitters, I can break down the cart into meaningful variables which can be used

In order to debug the code, I made use of a popup which will show after sending data to the server. This contains the response from the server in text form. I attempted to use a Snackbar but the information was removed quicker than required. As well, I added a debug variable to the server return which I can send data to view what is within it.



Fixing cartinfo Exports

While trying to work out how to process the data, I found that it could possibly be easier to split the data within the app.

The map exported should be the following

{

 "**type**": **add**

 "**restaurantid**": **restaurantid**,

 "**tip**": **tip**,

```

"address": address,
"latitude": latitude,
"longitude": longitude,
"iteminfo": [[indexid (for i incremented), profileid, itemid, quantity]],
"customiseinfo": [[indexid, customiseid, quantity]]
}

```

Since only lists can be encoded for php, it needs to be exported in that format otherwise it will cause other issues.

From within the future, I started by creating the following

```

int indexid = 0;

List iteminfo = [];

List customiseid = [];

for (int i = 0; i < cart.length; i++) {
    //For each profile
    int profileid = cart[i][0]; //Save the profile id as profileid
    List itemkeys = cart[i][6].keys.toList();
    for (int j = 0; j < itemkeys.length; j++) {
        List item = cart[i][6][itemkeys[j]];
        int itemid = cart[i][6][itemkeys[j]];
    }
}

```

This code gets the profileid and item from the cart. One problem I found was that the item id is never put into the cart info. As it turns out, it is exported from the server but it was never added to the cartinfo. The fixed version is as follows

```
tempItemInfo[profileCart[i]["itemid"]][0].addAll([
    basicItemInfo["message"]["message"][1], //Item name
    basicItemInfo["message"]["message"][3], //Item image link
    basicItemInfo["message"]["message"][2], //Item price
    basicItemInfo["message"]["message"][5], //Restaurant name
    basicItemInfo["message"]["message"][4], //Restaurant id
    basicItemInfo["message"]["message"][6], //Delivery price
    profileCart[i]["quantity"], //Item quantity
    profileCart[i]["cartid"], //Cart ID
    basicItemInfo["message"]["message"][7], //Minimum order price for
    restaurant
    basicItemInfo["message"]["message"][8], //Restaurant address
    basicItemInfo["message"]["message"][9], //Restaurant location
    latitude
    basicItemInfo["message"]["message"][10], //Restaurant location
    longitude
    basicItemInfo["message"]["message"][0] //Item ID
]);
```

When adding it as itemid, I had an error

Exception has occurred. ×

`_TypeError (type 'String' is not a subtype of type 'int')`

To fix this assigned I did an int parse to convert it to an integer.

I then built the general format of the formatter,

```
int indexid = 0;

List iteminfo = [];

List customiseinfo = [];

for (int i = 0; i < cart.length; i++) {
    //For each profile

    int profileid = cart[i][0]; //Save the profile id as profileid

    List itemkeys = cart[i][6].keys.toList();

    for (int j = 0; j < itemkeys.length; j++) { //For each item

        List item = cart[i][6][itemkeys[j]];

        int itemid = int.parse(item[0][12]); //Save item id as itemid

        List customisekeys = item[1].keys.toList();

        for (int k = 0; k < customisekeys.length; k++) { //For each customise
option

            List customiseOption = item[1][customisekeys[k]];

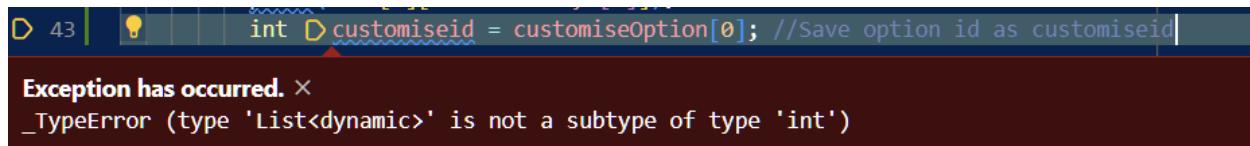
            int customiseid = customiseOption[0]; //Save option id as customiseid

        }

    }

}
```

When I tried to run it, it gave me an error.



A screenshot of a code editor showing a type error. The code is as follows:

```
int customiseid = customiseOption[0]; //Save option id as customiseid
```

The line `int customiseid = customiseOption[0];` is highlighted with a red underline, indicating a type mismatch. A tooltip or error message box is displayed below the line, showing the error message: `Exception has occurred. × _TypeError (type 'List<dynamic>' is not a subtype of type 'int')`.

What I had forgotten was that `customised` was built up of the title and the option. The server is able to handle the title due to each option referencing the id of the title using relationships.

The fixed code is as follows

```
for (int k = 0; k < customisekeys.length; k++) {  
    //For each customise title  
  
    List customiseTitle = item[1][customisekeys[k]];  
  
    for (int l = 0; l < customiseTitle[1].length; l++) { // For each option  
  
        List customiseOption = customiseTitle[1][l];  
  
        int customiseOptionid = customiseOption[0]; //Save option id as  
        customiseoptionid  
    }  
}
```

To then get the lists for each order, I compiled the information gathered and used the incrementing index as the main index that will be used to find which customise option is related to the item.

For item info, the following is the code:

```
for (int j = 0; j < itemkeys.length; j++) {  
    //For each item  
  
    List item = cart[i][6][itemkeys[j]];  
  
    int itemid = int.parse(item[0][12]); //Save item id as itemid
```

```

int itemQuantity = item[0][6];

iteminfo.add([
    indexid,
    profileid,
    itemid,
    itemQuantity,
]);

```

For the customise options, the following is the code added:

```

int optionQuantity = customiseOption[1];

customiseinfo.add([indexid, customiseOptionid, optionQuantity]);

```

After testing, it worked successfully

```

[[0, 136, 3, 99], [1, 136, 10, 2]]
[[0, 89, 1], [0, 95, 1], [0, 103, 1], [0, 104, 1], [1, 220, 1]]

```

Processing New Cart Data in Server

To build the SQL, I first convert all the strings back to doubles and integers. I also round the price values to 2 just in case there is an issue with the app where it fails to round. I then create the SQL sequence for the order information page

```

if($addval){
    $tip = round((floatval($_POST["tip"])), 2);
    $restaurantid = intval($_POST["restaurantid"]);
    $address = $_POST["address"];
    $latitude = floatval($_POST["latitude"]);
    $longitude = floatval($_POST["longitude"]);
    $sqlorderinfo = "INSERT INTO orderinformation SET

```

```

restaurantid = '$restaurantid', tip = '$tip', deliveryaddress = '$address',
deliverylatitude = '$latitude', deliverylongitude = '$longitude', status =
'confirming'";
    $resorderinfo = mysqli_query($link, $sqlorderinfo);
}

```

To confirm that the sql has worked successfully, it is verified

by putting resorderinfo into an if statement since if it successfully inserts the data it returns data.

In order to place the order id as a foreign key in the order item info, the primary key needs to be grabbed but due to the order id being automatic incrementing primary key, the only way to grab the primary key is by checking the most recent record put in the table and taking the key from it. Although it would be great to perform a table lock to prevent alterations and additions during this process, with the limited time, I was not able to implement this.

In order to limit the chance of getting the incorrect id, I filtered by the latitude and longitude. Since each makes use of a double (to 12 decimal places) this means that unless the app glitches and tries to make two orders at the same time, there is only a 1×10^{30} chance of it getting the same destination.

```

$orderid = "SELECT orderid WHERE deliverylatitude = '$latitude' AND
deliverylongitude = '$longitude', ORDER BY orderid DESC LIMIT 1";

```

To then add the item info I will create a for loop for the length of the list of the exported iteminfo since each has a unique id then create a record and get the id using another sql statement.

(While taking a look at the iteminfo, the item index can be removed since each relates to its item index)

For each item, it iterates through the customiseinfo for if the first index is equal to "i" (the item index) then runs an sql statement to add it to the table.

I tried the following:

```

if ($_POST["type"] == "add"){
    $addval = isset($_POST["tip"]) && isset($_POST["restaurantid"])
&& isset($_POST["address"]) && isset($_POST["latitude"]) &&

```

```

isset($_POST["longitude"]) && isset($_POST["iteminfo"]) &&
isset($_POST["customiseinfo"]);
    if($addval){
        $tip = round((floatval($_POST["tip"])), 2);
        $restaurantid = intval($_POST["restaurantid"]);
        $address = $_POST[ "address" ];
        $latitude = floatval($_POST[ "latitude" ]);
        $longitude = floatval($_POST[ "longitude" ]);
        $sqlorderinfo = "INSERT INTO orderinformation SET
restaurantid = '$restaurantid', tip = '$tip', deliveryaddress = '$address',
deliverylatitude = '$latitude', deliverylongitude = '$longitude', status =
'confirming'";
        if($resorderinfo = mysqli_query($link, $sqlorderinfo)){
            $sqlorderid = "SELECT orderid FROM orderinformation WHERE
deliverylatitude = '$latitude' AND deliverylongitude = '$longitude' ORDER
BY orderid DESC LIMIT 1";
            if($orderidinfo = mysqli_query($link, $sqlorderid)){
                $orderidrecord = mysqli_fetch_assoc($orderidinfo);
                $orderid = $orderidrecord[ "orderid" ];
                for ($i = 0; $i < count($_POST[ "iteminfo" ]); $i++){
                    $iteminfo = $_POST[ "iteminfo" ][ $i ];
                    $sqlorderiteminfo = "INSERT INTO
orderiteminformation SET orderid = '$orderid', profileid = '$iteminfo[0]',
itemid = '$iteminfo[1]', quantity = '$iteminfo[2]'";
                    if($orderiteminfo = mysqli_query($link,
$sqlorderiteminfo)){
                        $sqlorderitemid = "SELECT orderitemid FROM
orderinformation WHERE profile = '$iteminfo[0]' AND itemid = '$iteminfo[1]'
ORDER BY orderitemid DESC LIMIT 1";
                        if($orderitemidinfo = mysqli_query($link,
$sqlorderitemid)){
                            $orderitemidrecord =
mysqli_fetch_assoc($orderitemidinfo);
                            $orderitemid =
$orderitemidrecord[ "orderitemid" ];
                            for ($j = 0; $j <
count($_POST[ "customiseinfo" ]); $j++){
                                if ($_POST[ "customiseinfo" ][ $j ][ 0 ] ==
$i){
                                    $sqlorderitemoptioninfo = "INSERT
INTO ordercustomisediteminformation SET orderitemid = '$orderitemid',
customiseid = '$customiseid[1]', quantity = '$customiseid[2]'";

```

```

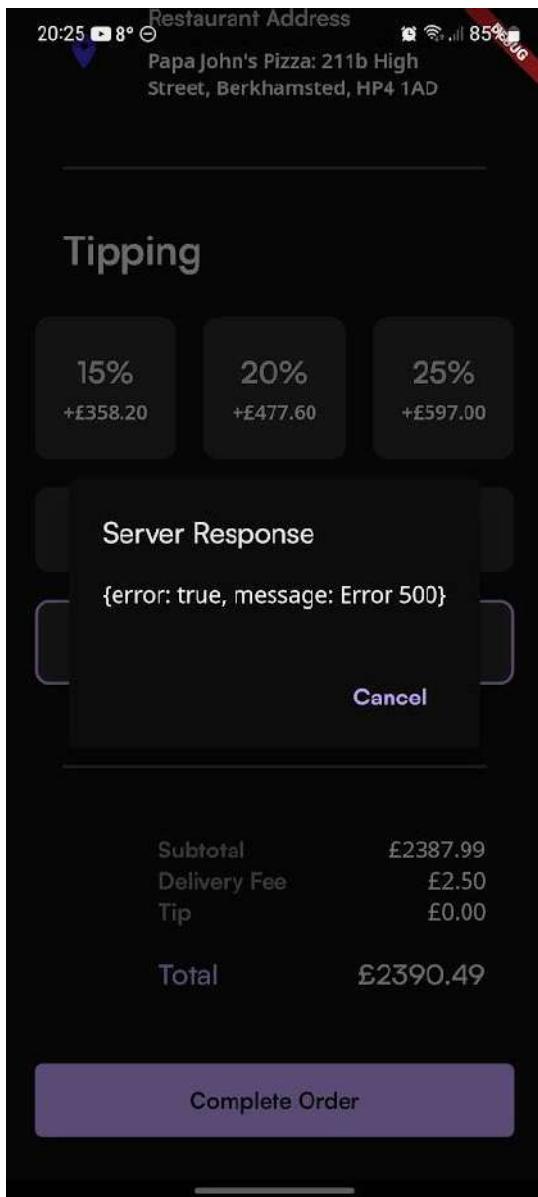
        if($orderitemoptioninfo =
mysqli_query($link, $sqlorderitemoptioninfo)){
            $return[ "message" ] = "success";
        } else{
            $return[ "error" ] = true;
            $return[ "message" ] = "failed to
save customised item";
        }
    }
}
} else {
    $return[ "error" ] = true;
    $return[ "message" ] = "failed to verify item
info";
}
} else{
    $return[ "error" ] = true;
    $return[ "message" ] = "failed to save item info";
}
}
}
}else{
    $return[ "error" ] = true;
    $return[ "message" ] = "failed to verify order info";
}
}else{
    $return[ "error" ] = true;
    $return[ "message" ] = "failed to save order";
}

} else{$return[ "error" ] = true;
$return[ "message" ] = "data not specified";}

} else{
    $return[ "error" ] = true;
    $return[ "message" ] = "order query type unknown";
}

```

I got an error 500 from this meaning there is an issue with the code:



Checking the database, I found that only the order information table is added to.

Since I get an error 500 this means that an error catch doesn't return so that reduces the number of lines that it could be.

Using nested if statements and for statements means allows me to remove a further nested code to find when the error 500 appears.

The error occurs around here:

```
$sqlorderiteminfo = "INSERT INTO
```

```
orderiteminformation SET orderid = '$orderid', profileid = '$iteminfo[0]',  
itemid = '$iteminfo[1]', quantity = '$iteminfo[2]'" ;
```

While looking at the error showing, I noticed that it is a set error code which I had forgotten about. Using a set error code instead of the returned value.

After changing this to be the returned message, it was found that the message returned was "failed to save item info".

Although the sql is valid, the data imported must be invalid. By removing the catch statements and returning the sql variable as text through the debug key, I was able to determine that the profile id, item id and quantity were all 0.

The reason was because it was a string not an array. By doing a json decode, I was able to get it into a php array.

I continued debugging. As it turned out another issue was that the foreign keys were incorrectly referencing the wrong table.

The fixed code is as follows:

```
if ($_POST["type"] == "add"){
    $addval = isset($_POST["tip"]) && isset($_POST["restaurantid"])
&& isset($_POST["address"]) && isset($_POST["latitude"]) &&
isset($_POST["longitude"]) && isset($_POST["iteminfo"]) &&
isset($_POST["customiseinfo"]);
    if($addval){
        $tip = round((floatval($_POST["tip"])), 2);
        $restaurantid = intval($_POST["restaurantid"]);
        $address = $_POST["address"];
        $latitude = floatval($_POST["latitude"]);
        $longitude = floatval($_POST["longitude"]);
        $iteminfo = json_decode($_POST["iteminfo"], true);
        $customiseinfo = json_decode($_POST["customiseinfo"], true);
        $sqlorderinfo = "INSERT INTO orderinformation SET
restaurantid = '$restaurantid', tip = '$tip', deliveryaddress = '$address',
deliverylatitude = '$latitude', deliverylongitude = '$longitude', status =
'confirming'";
        if($resorderinfo = mysqli_query($link, $sqlorderinfo)){
            $sqlorderid = "SELECT orderid FROM orderinformation WHERE
deliverylatitude = '$latitude' AND deliverylongitude = '$longitude' ORDER
```

```

BY orderid DESC LIMIT 1";
    if($orderidinfo = mysqli_query($link, $sqlorderid)){
        $orderidrecord = mysqli_fetch_assoc($orderidinfo);
        $orderid = intval($orderidrecord["orderid"]);
        for ($i = 0; $i < count($_POST["iteminfo"]); $i++){
            $profileid = intval($iteminfo[$i][0]);
            $itemid = intval($iteminfo[$i][1]);
            $itemquantity = intval($iteminfo[$i][2]);

            $sqlorderiteminfo = "INSERT INTO
orderiteminformation SET orderid = '$orderid', profileid = '$profileid',
itemid = '$itemid', quantity = '$itemquantity'";

            if($orderiteminfo = mysqli_query($link,
$sqlorderiteminfo)){
                $sqlorderitemid = "SELECT orderitemid FROM
orderiteminformation WHERE profileid = '$profileid' AND itemid = '$itemid'
ORDER BY orderitemid DESC LIMIT 1";
                if($orderitemidinfo = mysqli_query($link,
$sqlorderitemid)){
                    $orderitemidrecord =
mysqli_fetch_assoc($orderitemidinfo);
                    $orderitemid =
intval($orderitemidrecord["orderitemid"]);
                    for ($j = 0; $j < count($customiseinfo);
$j++){
                        if ($customiseinfo[$j][0] == $i){
                            $customiseid =
intval($customiseinfo[$j][1]);
                            $quantity =
intval($customiseinfo[$j][2]);
                            $sqlorderitemoptioninfo = "INSERT
INTO ordercustomisediteminformation SET orderitemid = '$orderitemid',
customiseid = '$customiseid', quantity = '$quantity'";
                            if($orderitemoptioninfo =
mysqli_query($link, $sqlorderitemoptioninfo)){
                                $return[ "message" ] = "success";
                            } else{
                                $return[ "error" ] = true;
                                $return[ "message" ] = "failed to
save customised item";
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }
}
} else {
    $return["error"] = true;
    $return["message"] = "failed to verify item
info";

}

} else{
    $return["error"] = true;
    $return["message"] = "failed to save item info";
}
}
}
}else{
    $return["error"] = true;
    $return["message"] = "failed to verify order info";
}
}else{
    $return["error"] = true;
    $return["message"] = "failed to save order";
}

} else{$return["error"] = true;
    $return["message"] = "data not specified";}

} else{
    $return["error"] = true;
    $return["message"] = "order query type unknown";
}

```

Responding to order creation

When the cart is sent to the server, it either returns that there is an error or not. If there is an error. The app should return the user back to the main page and not clear the cart. This is because when there is an issue with the order creation page, it is simply just processing the data sent from the cart. By kicking the user, it allows for the app to re-attempt getting the correct

information. when the user tries to checkout again. If there is no error when the server completes the order it should remove the items from the cart to prevent double ordering.

```
Map orderCreated = await sendCart(  
    widget.cartInfo, tipPrice.toString(),  
destinationAddress, destination[4].toString(), destination[5].toString());  
  
    if (orderCreated["error"] == true) {  
  
        setState(() {  
  
            Navigator.of(context).pop();  
  
            Navigator.of(context).pop();  
  
ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("ERROR:  
${orderCreated["message"]}")));  
  
    });  
  
} else {  
  
    try {  
  
        setState(() async {  
  
            await SQLiteCartItems.clearCart();  
  
        });  
  
    } catch (e) {  
  
        setState(() {  
  
            Navigator.of(context).pop();  
  
            Navigator.of(context).pop();  
  
ScaffoldMessenger.of(context)  
    }));  
    }  
}
```

```

        .showSnackBar(const SnackBar(content:
Text("ERROR: Successfully created order but failed to clear cart.")));

    });

}

setState(() {

    Navigator.of(context).pop();

    Navigator.of(context).pop();

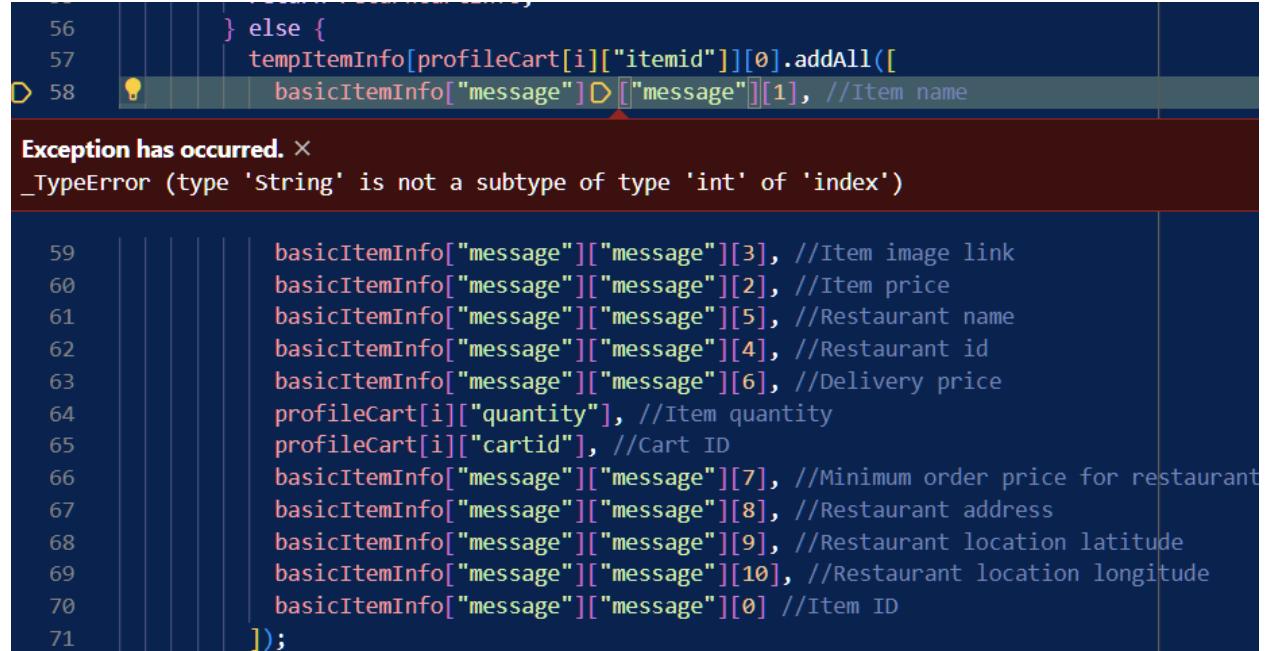
    ScaffoldMessenger.of(context).showSnackBar(const
SnackBar(content: Text("Successfully ordered.")));

});

}

```

After running through the app, I found that many server queries were broken due to the fact that I had fixed the code to not return a double error map for each query. One of these was the cart:



```

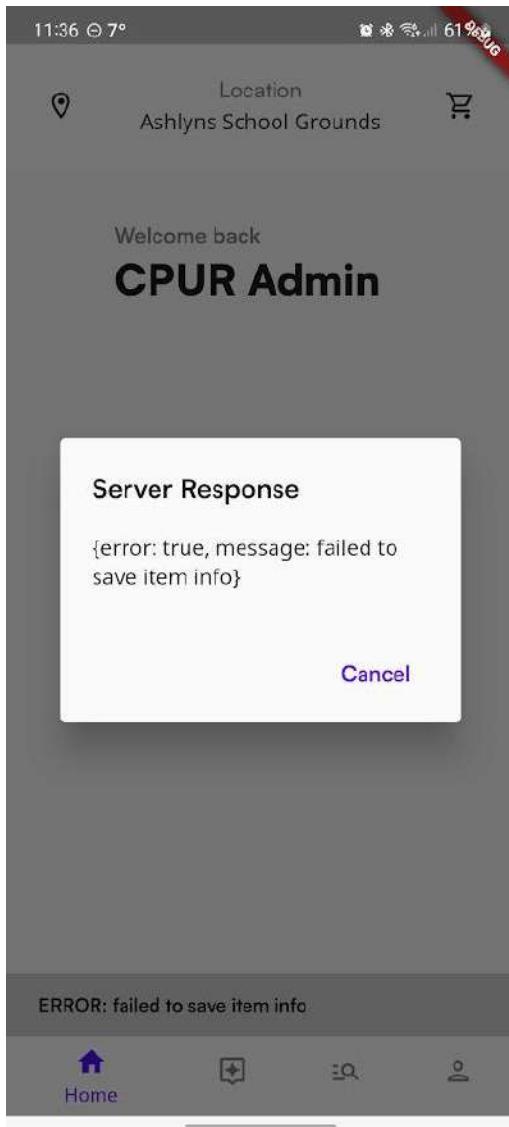
56
57
58 } else {
    tempItemInfo[profileCart[i]["itemid"]][0].addAll([
        basicItemInfo["message"][1], //Item name
        basicItemInfo["message"][3], //Item image link
        basicItemInfo["message"]["message"][2], //Item price
        basicItemInfo["message"]["message"][5], //Restaurant name
        basicItemInfo["message"]["message"][4], //Restaurant id
        basicItemInfo["message"]["message"][6], //Delivery price
        profileCart[i]["quantity"], //Item quantity
        profileCart[i]["cartid"], //Cart ID
        basicItemInfo["message"]["message"][7], //Minimum order price for restaurant
        basicItemInfo["message"]["message"][8], //Restaurant address
        basicItemInfo["message"]["message"][9], //Restaurant location latitude
        basicItemInfo["message"]["message"][10], //Restaurant location longitude
        basicItemInfo["message"]["message"][0] //Item ID
    ]);
}

```

Exception has occurred. ×
`_TypeError (type 'String' is not a subtype of type 'int' of 'index')`

To fix this, I removed one of the ["message"].

Even after this fix though, I got an error returned from the server informing me that it "failed to save item info".



The issue was caused by an incorrectly synced codebase. In order to edit my code everywhere, I use Github to sync my changes over both my laptop and my desktop and had forgotten to sync the most recent change causing this error.

Another error I faced was that the cart was clearing but it was returning an error. This was because I ran the future through a setState sequence which is incorrect. I removed this and it was complete

Post-Development

Testing

Testing Summary

Test Number	Success Criteria	What it is	Expected	Actual	Pass/Fail
1	1.3	The database should be in third-normal form and normalised	The database should be in third-normal form, having no duplicate data	The program makes use of third-normal form on both the server and the client app	PASS
2	5.1	There should be a button to remove an item from the order before it has been checked out	There should be a way to remove an item if a user makes a mistake or wants to order something else	The app makes use of dismissible elements, using a slide to delete instead of a button to edit the cart	PASS
3	5.2	For each item, it should show which profile the item is for	For each item it should indicate which profile is associated with it	Each profile has their own section of the cart containing the items associated with the profile. At the bottom of the cart there is a final price for	PASS

				each profile	
4	5.5	When checking out, there should be a way to input the amount you would like to tip	There should be an input field to select how much you want to tip	The checkout page contains a tip section which allows for both percentage tip and manual tip amounts.	PASS
5	5.10	The status of the order should sync with all profiles on all devices they are signed in with	The orders should be synced with the cloud and allow for any device signed in to see order updates	Due to the limited time, I was unfortunately not able to display the order status for each order created on the app but each order is synced with the cloud so there is potential to query the server for the status of each order	FAIL*
6	5.12	The checkout and cart should display the prices of all the items as a total	Each item should have a price of the customised item and at the bottom of the screen there should be the total price, adding	Although the cart contains the correct information, the total price is incorrect, having the completely incorrect price	FAIL

			to the total		
7	5.12	The checkout and cart should display the prices of all the items as a total	Each item should have a price of the customised item and at the bottom of the screen there should be the total price, adding to the total	The checkout is successfully displaying the correct price after delivery and tips, with the total price of the items also adding to the subtotal	PASS

Testing Criteria (1.3)

The database should be in third-normal form and normalised

Justification

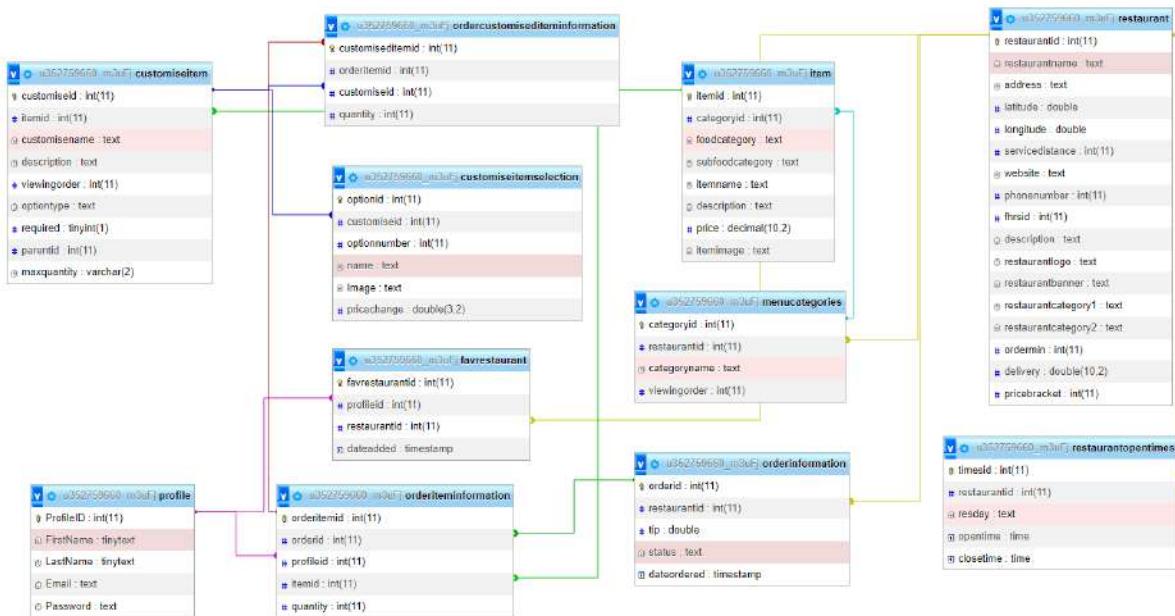
It reduces the amount of duplicate data, avoids data anomalies, ensures referential integrity, and simplifies data management.

Expected result

The database should be in third-normal form, having no duplicate data

Actual result

The program makes use of third-normal form on both the server and the client app



Pass or Fail?

— PASS —

Testing Criteria (5.1)

There should be a button to remove an item from the order before it has been checked out.

Justification

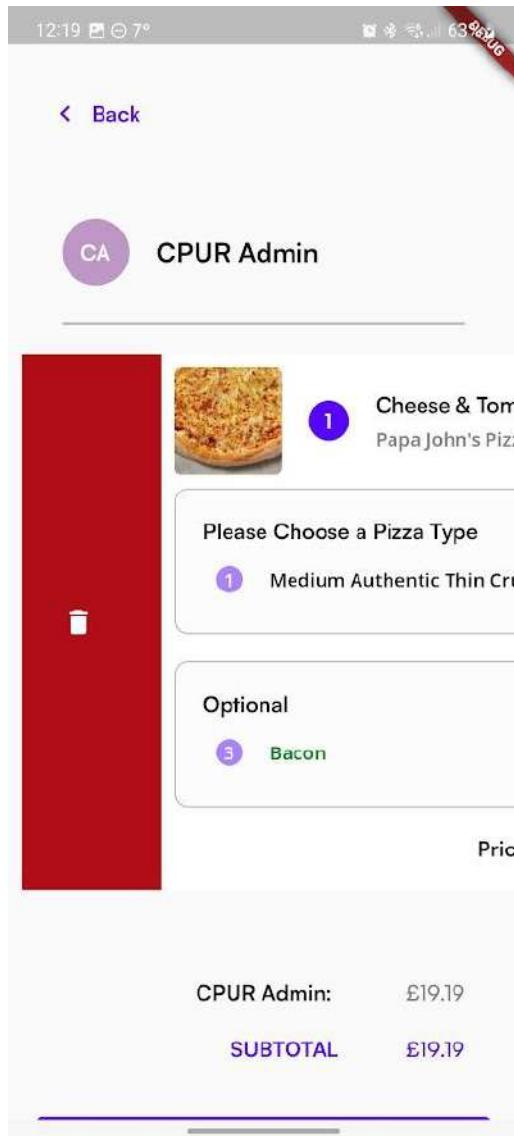
This means that if you make a mistake or want to change it, you can

Expected Result

There should be a way to remove an item if a user makes a mistake or wants to order something else

Actual Result

The app makes use of dismissible elements, using a slide to delete instead of a button to edit the cart



Pass or Fail?

— PASS —

Testing Criteria (5.2)

For each item, it should show which profile the item is for

Justification

This ensures that you know which item is for which person and removes the confusion

Expected result

For each item it should indicate which profile is associated with it

Actual result

Each profile has their own section of the cart containing the items associated with the profile. At the bottom of the cart there is a final price for each profile

The screenshot shows a mobile application interface with two distinct sections for different users:

- Top Section (CPUR Admin):** Shows a pizza item with a count of 1, labeled "Cheese & Tomato" from "Papa John's Pizza". Below it is a selection box for "Please Choose a Pizza Type" containing "Medium Authentic Thin Crust".
 - Optional Additions:** Shows an item with a count of 3, labeled "Bacon".
- Bottom Section (Peter John):** Shows a dish of chicken poppers with a count of 1, labeled "10 Chicken Poppers" from "Papa John's Pizza". Below it is a selection box for "Additions".

At the bottom of the screen, the total price is displayed as "Price: £19.19".

Pass or Fail?

— PASS —

Testing Criteria (5.5)

When checking out, there should be a way to input the amount you would like to tip

Justification

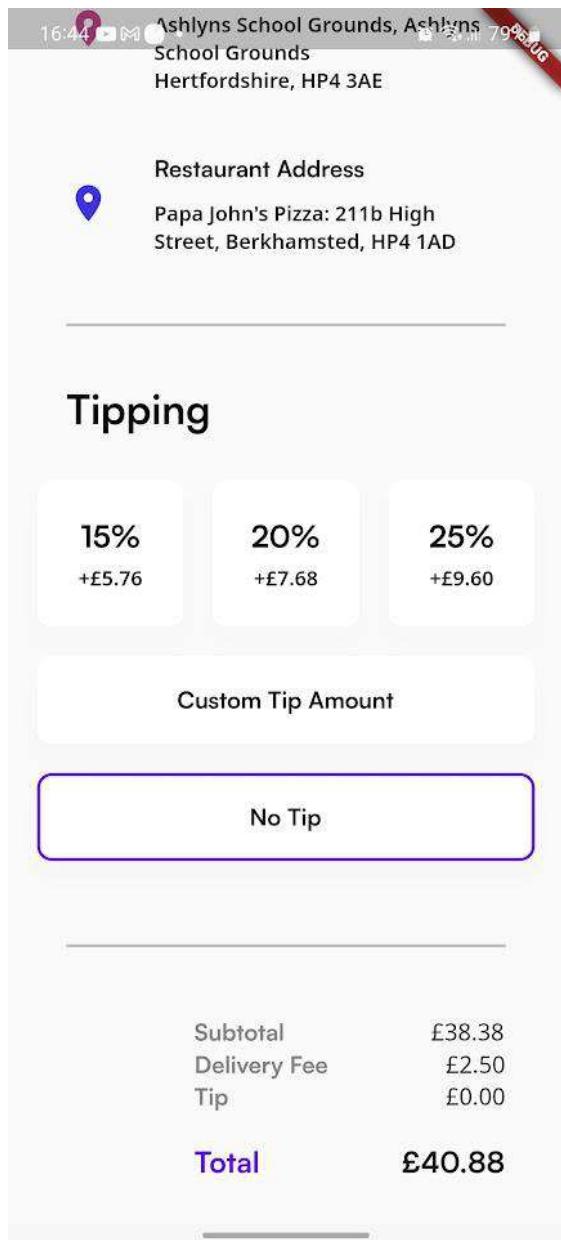
This helps out delivery drivers and encourages them to continue

Expected result

There should be an input field to select how much you want to tip

Actual result

The checkout page contains a tip section which allows for both percentage tip and manual tip amounts.



Pass or Fail?

— PASS —

Testing Criteria (5.10)

The status of the order should sync with all profiles on all devices they are signed in with

Justification

This will mean if they order on one phone, they can see the status on another phone

Expected Result

The orders should be synced with the cloud and allow for any device signed in to see order updates

Actual Result

Due to the limited time, I was unfortunately not able to display the order status for each order created on the app but each order is synced with the cloud so there is potential to query the server for the status of each order

	orderid	restaurantid	tip	deliveryaddress	deliverylatitude	deliverylongitude	status	dateordered
<input type="checkbox"/>   	4	1	955.2	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 12:17:41
<input type="checkbox"/>   	5	1	0	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 14:58:30
<input type="checkbox"/>   	6	1	0	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 19:13:26
<input type="checkbox"/>   	7	1	0	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 19:14:04
<input type="checkbox"/>   	8	1	0	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 19:15:19
<input type="checkbox"/>   	9	1	0	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 19:16:00
<input type="checkbox"/>   	10	1	0	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 19:16:48
<input type="checkbox"/>   	11	1	0	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 19:16:50
<input type="checkbox"/>   	12	1	0	Ashlyns School Grounds ,Ashlyns School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 20:25:07
<input type="checkbox"/>   	13	1	477.8	Ashlyn School Grounds ,Ashlyn School Grounds ,He...	51.753849674679	-0.56749507784843	confirming	2023-02-01 20:31:05

	orderitemid	orderid	profileid	itemid	quantity
<input type="checkbox"/>   	41	54	136	24	1
<input type="checkbox"/>   	42	55	136	24	1
<input type="checkbox"/>   	43	56	136	24	1
<input type="checkbox"/>   	44	57	136	24	1
<input type="checkbox"/>   	45	58	136	24	1
<input type="checkbox"/>   	46	59	136	24	1

		customiseditemid	orderitemid	customiseid	quantity
<input type="checkbox"/>	Edit Copy Delete	21	64	260	1
<input type="checkbox"/>	Edit Copy Delete	22	71	129	1
<input type="checkbox"/>	Edit Copy Delete	23	72	51	1
<input type="checkbox"/>	Edit Copy Delete	24	73	178	1

Pass or Fail?

— FAIL* —

* The order is synced with the cloud but it is not displayed. This will not be fixed.

Testing Criteria (5.12)

The checkout and cart should display the prices of all the items as a total

Justification

This means that the user knows how much they will be paying

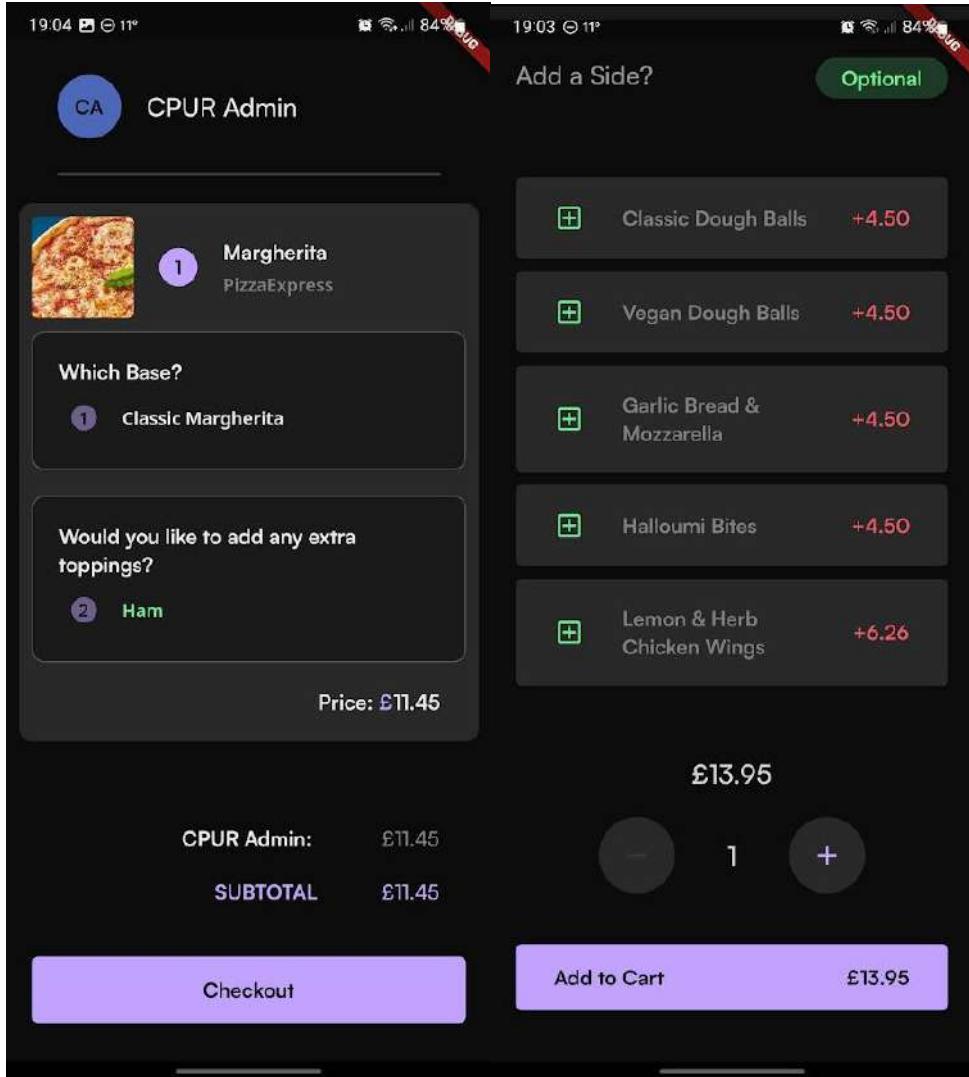
Expected Result

Each item should have a price of the customised item and at the bottom of the screen there should be the total price, adding to the total

Actual Result

Although the cart contains the items with the added customised price, what it does not do is take into account quantities of the customised options.

Left image shows the cart. Right image shows the price of the customised item.



Pass or Fail?

— FAIL —

Fixing Issues

In order to fix this, I remembered that the item price is changed in part of the cart code, taking a look at it, I found that it only got the price of the customised option and added it to the item price without multiplying it by the quantity.

I used a print statement to find the place where the option is stored to find the quantity's location

```
print(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1][iCustOptions]);
```

After finding that it was index 1 that the customise option is multiplied to, I added the code to modify the price added.

```
for (int iCust = 0; iCust < customiseOptionsKeys.length; iCust++) {  
    //For each customise title  
    if  
(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==  
"SELECT" ||  
  
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==  
"ADD") {  
    //If it is either a selection or an add function add it to the item  
    price (multiplied by the quantity)  
    for (int iCustOptions = 0;  
        iCustOptions <  
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1].length;  
        iCustOptions++) {  
  
    tempItemInfo[profileCart[i]["itemid"]][0][2] =  
((double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2]) * 100 +  
  
double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]]  
][1][iCustOptions][3]) * 100 *  
int.parseInt(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]]  
[1][iCustOptions][1])) /  
100)
```

```

        .toString();

    }

} else if
(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
"REMOVE") {

    //If it is a remove function, remove from the item price (multiplied
by the quantity)

    for (int iCustOptions = 0;

        iCustOptions <
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1].length;

        iCustOptions++) {

        tempItemInfo[profileCart[i]["itemid"]][0][2] =
((double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2]) * 100 -

double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]]
][1][iCustOptions][3]) * 100 *
int.parseInt(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]]
[1][iCustOptions][1])) /

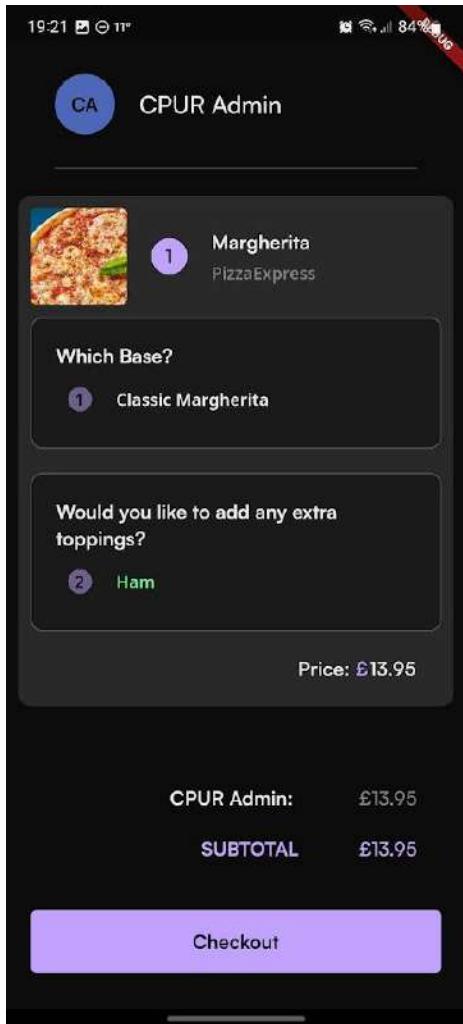
100)

        .toString();
    }
}
}

```

After adding it and running, I got an error indicating that it is not a string but a quantity so the *int.parse()* was removed

The final result was correct and accurate to the customised price



Pass or Fail?

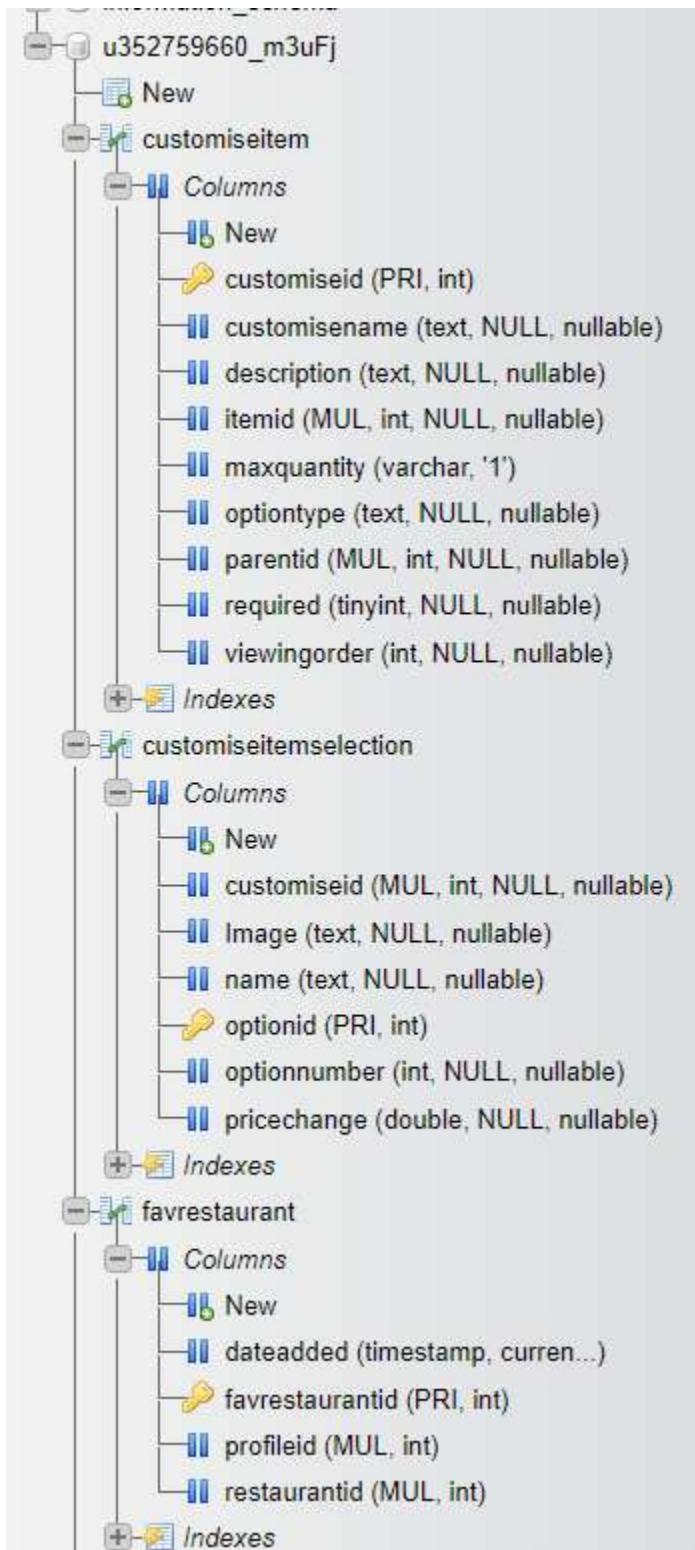
— PASS —

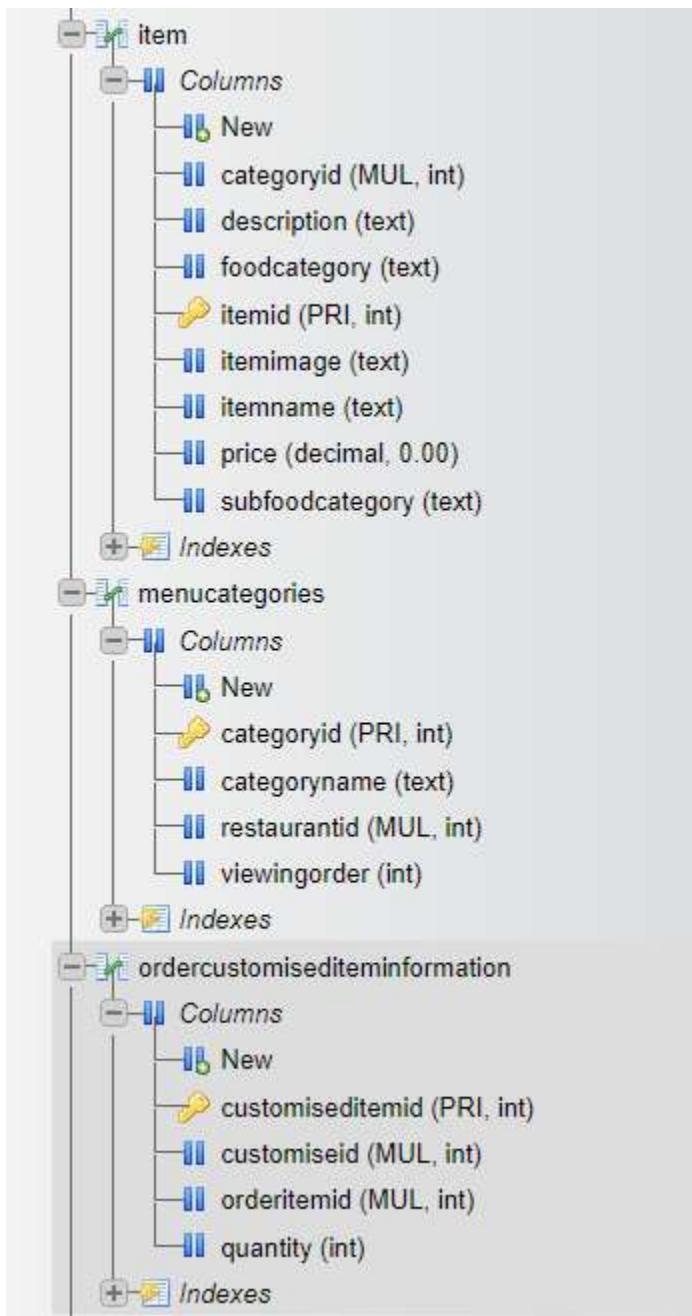
Prototype 3 Final Code

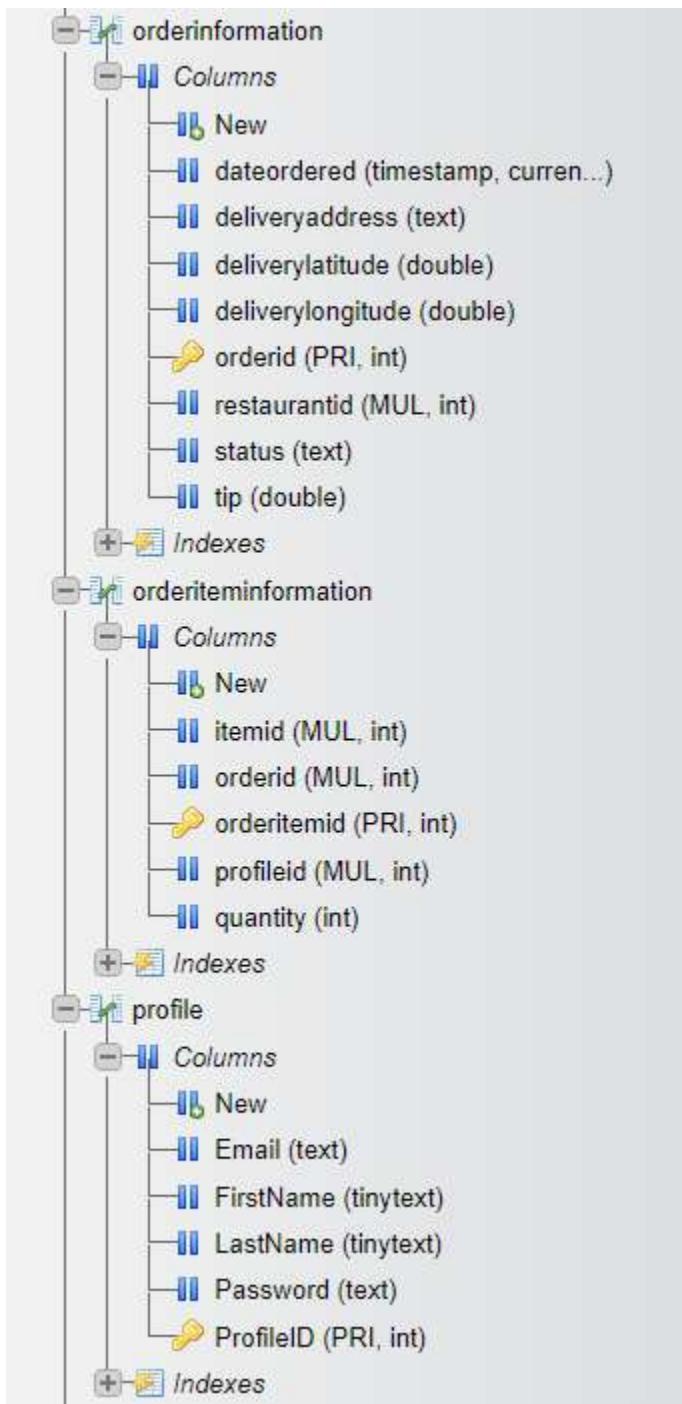
File Structure

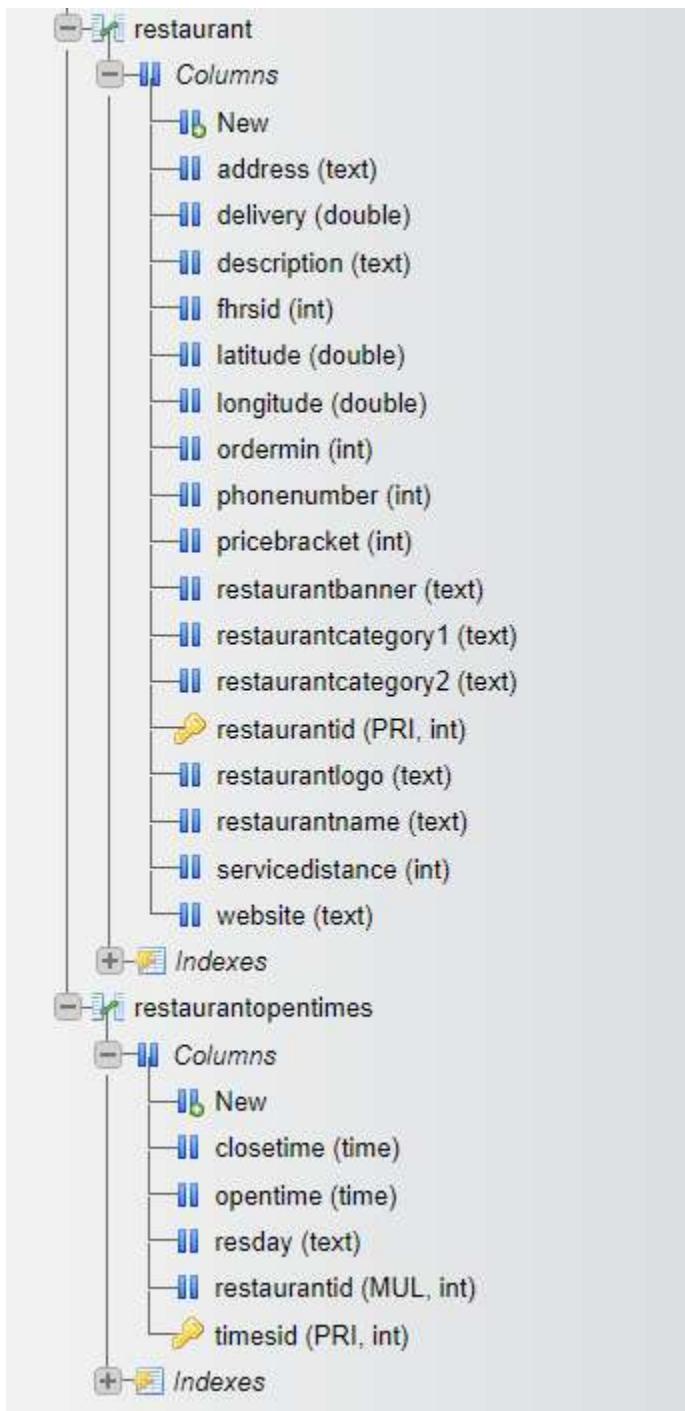
```
└─ lib
    └─ assets
        ├─ fonts
        └─ images
    └─ screens
        └─ checkout
            └─ cart.dart
            └─ checkout.dart
        └─ navigationscreens
            └─ browse.dart
            └─ foryou.dart
            └─ homepage.dart
            └─ profiles.dart
        └─ profilesetup
            └─ profilesetup_create.dart
            └─ profilesetup_existing.dart
            └─ profilesetup_login.dart
            └─ welcomescreen.dart
        └─ restaurant
            └─ restaurant_customise.dart
            └─ restaurant_main.dart
            └─ filtersort.dart
```

```
locationselection.dart
setupverification.dart
services
  cart_service.dart
  dataencryption.dart
  localprofiles_service.dart
  queryserver.dart
  setselected.dart
theme
  theme.dart
widgets
  elements
    browse_categories.dart
    elements.dart
    profiles_buttons.dart
    profiles_selection.dart
    search.dart
    genericlocading.dart
    navigationbar.dart
    restaurants_list.dart
    topbar.dart
main.dart
```









Name ↑	Size	Last modified	
<> cartiteminfo.php	3.24 KB	5 days ago	-rw-r--x--x
<> favouriterestaurant.php	2.38 KB	5 months ago	-rw-r--x--x
<> favouriterestaurantdata.php	2.49 KB	5 months ago	-rw-r--x--x
<> favouriterestaurantlist.php	1.81 KB	4 months ago	-rw-r--x--x
<> itemcustomise.php	1.7 KB	a month ago	-rw-r--x--x
<> login.php	2.9 KB	21 days ago	-rw-r--x--x
<> orders.php	5.15 KB	9 hours ago	-rw-r--x--x
<> register.php	4.61 KB	a day ago	-rw-r--x--x
<> restaurantlist.php	3.38 KB	a month ago	-rw-r--x--x
<> restaurantmenucategories.php	1.25 KB	5 months ago	-rw-r--x--x
<> restaurantmenuitems.php	1.30 KB	5 months ago	-rw-r--x--x

Files - Application

cart.dart

```
import 'package:alleat/screens/checkout/checkout.dart';

import 'package:alleat/services/cart_service.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/queryserver.dart';

import 'package:alleat/widgets/elements/elements.dart';

import 'package:flutter/material.dart';

import 'dart:convert';

class Cart extends StatefulWidget {

  const Cart({super.key});

  @override

  State<Cart> createState() => _CartState();
}
```

```

class _CartState extends State<Cart> {

    Future<Map> getCart() async {

        //Get cart data including restaurant info, profile info, item data, customise
titles for each item and the customised options performed on the item

        Map returnCartInfo = {"error": false, "message": "", "cartinfo": []}; //Set
returned cart info to be empty and there to be no errors

        List availableProfiles =
            [] //Each profile that is in the cart is stored in available profiles,
storing their basic information including profileid, firstname, lastname, profile
icon colour

        List cartProfileIDs = await SQLiteCartItems.getProfilesInCart(); //Get
profile ids that are in the cart

        List profileInfo = await SQLiteLocalProfiles.getProfiles(); //Get all info
stored for each profile within the local profiles database

        //Get profiles

        for (int i = 0; i < cartProfileIDs.length; i++) {
            //For each profile id that is in the available profiles

            for (int j = 0; j < profileInfo.length; j++) {
                //For each profile in the list of total profiles

                if (cartProfileIDs[i] == profileInfo[j]["profileid"]) {

                    //If the id of the current available profile in the cart is equal to
the currently searched profile, add its details to availableprofiles

                    availableProfiles.add([

```

```

        profileInfo[j]["profileid"],

        profileInfo[j]["firstname"],

        profileInfo[j]["lastname"],

        profileInfo[j]["profilecolorred"],

        profileInfo[j]["profilecolorgreen"],

        profileInfo[j]["profilecolorblue"]

    ));

}

}

}

for (int iProfile = 0; iProfile < availableProfiles.length; iProfile++) {

    //For each available profile

    Map tempItemInfo = {};
```

List profileCart = await SQLiteCartItems.getProfileCart(

availableProfiles[iProfile][0]); //Get list of items for profile.

Returns in format {cartid, itemid, customised, quantity}

for (int i = 0; i < profileCart.length; i++) {

// For each item, add it as an index i

tempItemInfo[profileCart[i]["itemid"]] = [[], {}];

Map basicItemInfo = await

QueryServer.query("https://alleat.cpur.net/query/cartiteminfo.php", {

```

    "type": "item",

    "term": profileCart[i]["itemid"].toString()

}); //returns item id, item name, price, item image, restaurant id,
restaurant name, delivery price

if (basicItemInfo["error"] == true) {

    returnCartInfo["error"] = true;

    returnCartInfo["message"] = basicItemInfo["message"];

    return returnCartInfo;

} else {

    tempItemInfo[profileCart[i]["itemid"]][0].addAll([
        basicItemInfo["message"]["message"][1], //Item name
        basicItemInfo["message"]["message"][3], //Item image link
        basicItemInfo["message"]["message"][2], //Item price
        basicItemInfo["message"]["message"][5], //Restaurant name
        basicItemInfo["message"]["message"][4], //Restaurant id
        basicItemInfo["message"]["message"][6], //Delivery price
        profileCart[i]["quantity"], //Item quantity
        profileCart[i]["cartid"], //Cart ID
        basicItemInfo["message"]["message"][7], //Minimum order price for
        restaurant
        basicItemInfo["message"]["message"][8], //Restaurant address
        basicItemInfo["message"]["message"][9], //Restaurant location
        latitude
    ])
}

```

```

    basicItemInfo["message"]["message"][10], //Restaurant location
longitude

    basicItemInfo["message"]["message"][0] //Item ID
]);

Map customised = json.decode(profileCart[i]["customised"]);

List customisedtitleids = customised.keys.toList(); //Each customise
title is stored here

List customisedOptions = []; //Each unique option stored here

Map updatedCustomised = {};//Data that will be sent back in the
iteminfo key, correctly formatted

for (int i = 0; i < customisedtitleids.length; i++) {

    //For each customise title

    updatedCustomised[customisedtitleids[i]] = [[], []]; //Add customised
key to new Map

    for (int j = 0; j < customised[customisedtitleids[i]].length; j++) {

        //For each item in list of customised options for customise title

        if
(!customisedOptions.contains(customised[customisedtitleids[i]][j])) {

            //If the option is not in the list of options, add it to the list

updatedCustomised[customisedtitleids[i]][1].add([customised[customisedtitleids[i]]
][j], 1));

            customisedOptions.add(customised[customisedtitleids[i]][j]);
        } else {
    }
}

```

```

        for (int k = 0; k <
updatedCustomised[customisedtitleids[i]][1].length; k++) {

            //If there is more than one of an option add it to the quantity
eg. (192, 162, 192) = [192, 2]

            if (updatedCustomised[customisedtitleids[i]][1][k][0] ==
customised[customisedtitleids[i]][j]) {

                updatedCustomised[customisedtitleids[i]][1][k][1] += 1;

            }

        }

    }

}

try {

    if (customisedtitleids.isNotEmpty) {

        //If there is at least one customisable title get the details of
all the customise titles using the list of titles that have been gotten from the
keys of the map (each key is the title id)

        Map customisedTitlesInfo = await QueryServer.query(
            "https://alleat.cpur.net/query/cartiteminfo.php", {"type": "title", "term": json.encode(customisedtitleids)});

    }

    if (customisedTitlesInfo["error"] == true) {

        //If there is an error, return error=true with the message
returned from the server and end future

        returnCartInfo["error"] = true;
    }
}

```

```

        returnCartInfo["message"] = customisedTitlesInfo["message"];

        return returnCartInfo;

    } else {

        List customisedTitlesInfoFormatted =
customisedTitlesInfo["message"]["message"];

        if (customisedOptions.isNotEmpty) {

            Map customisedOptionInfo = await QueryServer.query(
                "https://alleat.cpur.net/query/cartiteminfo.php", {"type": "option", "term": customisedOptions.toString()});
        }

        if (customisedOptionInfo["error"] == true) {

            //If there is an error, return error=true with the message
            returned from the server and end future

            returnCartInfo["error"] = true;

            returnCartInfo["message"] = customisedOptionInfo["message"];

            return returnCartInfo;
        } else {

            List customisedOptionInfoFormatted =
customisedOptionInfo["message"]["message"];

            for (int i = 0; i < customisedTitleIds.length; i++) {

                // For each customise title

                for (int j = 0; j < customisedTitlesInfoFormatted.length;
j++) {

```

```

        //For each item in list of returned from server info
about each title

        if (customisedTitlesInfoFormatted[j][0] ==
customisedtitleids[i]) {

            // If it has the id of the title in the first index,
add the info to the composite info

            updatedCustomised[customisedtitleids[i]][0]

                .addAll([customisedTitlesInfoFormatted[j][1],
customisedTitlesInfoFormatted[j][2]]);

        }

    }

    for (int j = 0; j <
updatedCustomised[customisedtitleids[i]][1].length; j++) {

        //For each option in the list of options for each title

        for (int k = 0; k < customisedOptionInfoFormatted.length;
k++) {

            //For each item in list of returned from server about
option info

            if (customisedOptionInfoFormatted[k][0] ==
updatedCustomised[customisedtitleids[i]][1][j][0]) {

                // If it has the id of the option in the first index,
add the info to the list

                updatedCustomised[customisedtitleids[i]][1][j]

                    .addAll([customisedOptionInfoFormatted[k][1],
customisedOptionInfoFormatted[k][2]]);

            }

        }

    }

```

```

        }

    }

    tempItemInfo[profileCart[i]["itemid"]][1] =
updatedCustomised;

}

}

}

}

}

} catch (e) {

returnCartInfo["error"] = true;

returnCartInfo["message"] = "An error occurred while attempting to
process the item information./";

}

}

// Update item price to be the new customised price

List customiseOptionsKeys =
tempItemInfo[profileCart[i]["itemid"]][1].keys.toList();

for (int iCust = 0; iCust < customiseOptionsKeys.length; iCust++) {

//For each customise title

if
(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
"SELECT" ||

```

```

tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
"ADD") {

    //If it is either a selection or an add function add it to the item
    price (multiplied by the quantity)

    for (int iCustOptions = 0;

        iCustOptions <
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1].length;

        iCustOptions++) {

        tempItemInfo[profileCart[i]["itemid"]][0][2] =
((double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2]) * 100 +

double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]]
][1][iCustOptions][3]) *

100 *

tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1][iCustOptions][1]) /
100)

        .toString();

    }

} else if
(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
"REMOVE") {

    //If it is a remove function, remove from the item price (multiplied
    by the quantity)

    for (int iCustOptions = 0;

```

```

        iCustOptions <
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1].length;

        iCustOptions++) {

            tempItemInfo[profileCart[i]["itemid"]][0][2] =
((double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2])) * 100 -

double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1][iCustOptions][3]) *

100 *

tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1][iCustOptions][1]) /

100)

.toString();

}

}

//Multiply price of each item by the quantity of the item

tempItemInfo[profileCart[i]["itemid"]][0][2] =

(double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2])) *

tempItemInfo[profileCart[i]["itemid"]][0][6]).toString();

}

returnCartInfo["cartinfo"].add([
availableProfiles[iProfile][0],

```

```
availableProfiles[iProfile][1],  
availableProfiles[iProfile][2],  
availableProfiles[iProfile][3],  
availableProfiles[iProfile][4],  
availableProfiles[iProfile][5],  
tempItemInfo  
]); //Return the item information of profile with the profile information  
}  
  
return returnCartInfo;  
}  
  
//Remove Item from cart in local profiles table  
  
Future<bool> removeItem(cartID) async {  
  
try {  
  
await SQLiteCartItems.removeItem(cartID);  
  
return true;  
} catch (e) {  
  
return false;  
}  
}  
  
@override
```

```
Widget build(BuildContext context) {  
  
  return Scaffold(  
  
    body: SingleChildScrollView(  
  
      child: Column(crossAxisAlignment: CrossAxisAlignment.start, children:  
[  
  
  const ScreenBackButton(),  
  
  FutureBuilder<Map>(  
  
    future: getCart(),  
  
    builder: ((context, snapshot) {  
  
      if (snapshot.hasData) {  
  
        List cartInfo = [snapshot.data ?? []];  
  
        if (cartInfo[0]["error"] == true) {  
  
          //If there is an error getting the cart info display error box  
  
          return Padding(  
  
            padding: const EdgeInsets.only(left: 20, right: 20, top: 10,  
bottom: 10),  
  
            child: Container(  
  
              decoration: BoxDecoration(  
  
                borderRadius: const  
BorderRadius.all(Radius.circular(20)),  
  
                color: Theme.of(context).colorScheme.onSurface,  
  
                boxShadow: [  
  
                  BoxShadow(  
  
                    color: Colors.black.withOpacity(0.1),
```

```
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes position  
of shadow  
  
    ),  
  
]),  
  
child: Column(children: [  
  
    Padding(  
  
        padding: const EdgeInsets.all(30),  
  
        child: SizedBox(  
  
            width: double.infinity,  
  
            child: Center(  
  
                child: Column(children: [  
  
                    const Text(  
  
                        "An error has occurred.",  
  
                        textAlign: TextAlign.center,  
  
                    ),  
  
                    const SizedBox(  
  
                        height: 20,  
  
                    ),  
  
                    Text(  
  
                        cartInfo[0]["message"],  
  
                        textAlign: TextAlign.center,  
                ]),  
            ),  
        ),  
    ),  
],  
),
```

```

        ),
        ]))),
    )
])));

} else {

    if (cartInfo[0]["cartinfo"].isNotEmpty) {

        //If there are profiles in the cart

        return ListView.builder(
            physics: const NeverScrollableScrollPhysics(), //Dont allow
scrolling (Done by main page)

            scrollDirection: Axis.vertical,
            shrinkWrap: true,
            itemCount: cartInfo[0]["cartinfo"].length + 1, //For each
profile (add one for the total price)

            itemBuilder: (context, indexProfile) {

                if (indexProfile != cartInfo[0]["cartinfo"].length) {

                    List currentProfile =
cartInfo[0]["cartinfo"][indexProfile];

                    List itemIDs = currentProfile[6].keys.toList();

                    return Column(children: [
                        Container(
                            //Contain the profile within a container
                            margin: const EdgeInsets.symmetric(vertical: 10,
horizontal: 20),

```

```

padding: const EdgeInsets.symmetric(horizontal:
20, vertical: 10),

        child: Row(
            children: [
                Container(
                    // Create profile circle with first and
last letter

                    width: 50,
                    height: 50,
                    decoration: BoxDecoration(
                        shape: BoxShape.circle,
                        color:
Color.fromRGBO(currentProfile[3], currentProfile[4], currentProfile[5], 1)),
                    child: Align(
                        alignment: Alignment.center,
                        child:
Text('${currentProfile[1][0]}${currentProfile[2][0]}',
style:
Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).backgroundColor))),

                    const SizedBox(width: 20), //Profile
firstname and lastname
                ),
                Text(
                    "${currentProfile[1]}"
                    "${currentProfile[2]}",

```

```
        style:  
Theme.of(context).textTheme.headline5?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
        overflow: TextOverflow.fade,  
    )  
],  
)),  
  
Divider(  
  
    thickness: 2,  
  
    color:  
Theme.of(context).textTheme.headline6?.color?.withOpacity(0.5),  
  
    indent: 40,  
  
    endIndent: 40,  
,  
  
const SizedBox(  
  
    height: 10,  
,  
  
ListView.builder(  
  
    //Create a container for each profile containing  
the customised options and a summary of the item  
  
    physics: const NeverScrollableScrollPhysics(),  
//Dont allow scrolling (Done by main page)  
  
    scrollDirection: Axis.vertical,  
  
    shrinkWrap: true,  
)
```

```

        itemCount: itemIDs.length, //For each item

        itemBuilder: (context, indexItem) {

            List currentItem =
currentProfile[6][itemIDs[indexItem]];

            List itemCustomiseIDs =
currentItem[1].keys.toList();

            return Padding(
                padding: const
EdgeInsets.symmetric(vertical: 5, horizontal: 10),

                child: Dismissible(
                    //Create a slide to delete container
(dismissible)

                    key: Key(currentItem[0][7].toString()),

//Key is the cart id

                    onDismissed: (direction) async {

                        bool hasDeleted = await
removeItem(currentItem[0][7]); //Remove item from local cart database

                        if (hasDeleted) {

                            //If it deleted

cartInfo[0]["cartinfo"][indexProfile][6].remove(itemIDs[indexItem]); //removing
the item from the list

                            setState(() {

                                ScaffoldMessenger.of(context)
                                    .showSnackBar(const
SnackBar(content: Text("Successfully deleted item.")));

```

```
    });

} else {
    setState(() {
        ScaffoldMessenger.of(context)
            .showSnackBar(const
SnackBar(content: Text("Failed to deleted item. Please reopen the cart")));
    });
}

},
background: Container(
color:
Theme.of(context).colorScheme.error,
child: Row(
mainAxisAlignment:
MainAxisAlignment.spaceBetween,
children: [
Padding(
padding: const
EdgeInsets.symmetric(horizontal: 30),
child: Icon(
Icons.delete,
color:
Theme.of(context).colorScheme.onSurface,
)),
]),
```

```
Padding(  
  padding: const  
EdgeInsets.symmetric(horizontal: 30),  
  child: Icon(  
    Icons.delete,  
    color:  
Theme.of(context).colorScheme.onSurface,  
  ))  
,  
(),  
child: Container(  
  width: double.infinity,  
  decoration: BoxDecoration(  
    borderRadius: const  
BorderRadius.all(Radius.circular(10)),  
    color:  
Theme.of(context).colorScheme.onSurface),  
  padding: const  
EdgeInsets.symmetric(vertical: 10, horizontal: 10),  
  child: Column(children: [  
    Row(  
      children: [  
        Container(  
          width: 80,
```

```
        height: 80,  
  
        decoration: BoxDecoration(  
  
          borderRadius: const  
BorderRadius.all(Radius.circular(5)),  
  
          image:  
  
DecorationImage(fit: BoxFit.cover, image:  
NetworkImage(currentItem[0][1].toString()))),  
  
        ),  
  
        const SizedBox(  
  
          width: 20,  
  
        ),  
  
        Container(  
  
          alignment:  
Alignment.center,  
  
          height: 30,  
  
          width: 30,  
  
          decoration:  
BoxDecoration(  
  
            color:  
Theme.of(context).primaryColor, borderRadius: BorderRadius.circular(30)),  
  
            child: Text(  
  
currentItem[0][6].toString(),  
  
style:  
Theme.of(context)
```

```
.textTheme  
.headline6  
?.copyWith(color:  
Theme.of(context).colorScheme.onSurface),  
)),  
const SizedBox(  
width: 20,  
),  
Expanded(  
child: Column(  
crossAxisAlignment:  
CrossAxisAlignment.start,  
children: [  
Text(  
currentItem[0][0],  
style:  
Theme.of(context)  
.textTheme  
.headline6  
?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
),  
const SizedBox(  
height: 5,
```

```

        ),
        Text(
            currentItem[0][3],
            style:
                Theme.of(context)
                    .textTheme
                    .bodyText1
                    ?.copyWith(color:
Theme.of(context).textTheme.headline6?.color),
            )
        ],
        )),
    ],
),
ListView.builder(
    //Create a container with the
title of the option that has been customised with a list of options changed
    physics: const
NeverScrollableScrollPhysics(), //Dont allow scrolling (Done by main page)
    scrollDirection:
Axis.vertical,
    shrinkWrap: true,
    itemCount:
itemCustomiseIDs.length, //For each customise title

```

```

        itemBuilder: (context,
indexCustomise) {

    List currentCustomiseTitle
= currentItem[1][itemCustomiseIDs[indexCustomise]];

    if
(currentCustomiseTitle[0][1] == "SELECT" &&
currentCustomiseTitle[1].toList().length != 0) {

        //If the customise title
type is select and there is at least one customised option for the title return
the customised options in black text

        return Container(
            padding: const
EdgeInsets.symmetric(vertical: 20, horizontal: 20),
            margin: const
EdgeInsets.symmetric(horizontal: 0, vertical: 10),
            decoration:
BoxDecoration(
            color:
Theme.of(context).backgroundColor.withOpacity(0.5),
            border:
Border.all(
            color:
Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),
            borderRadius:
BorderRadius.circular(10)),
            child: Column(
mainAxisAlignment: MainAxisAlignment.start,

```

```
crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
  
    Text(currentCustomiseTitle[0][0],  
        textAlign:  
        TextAlign.start,  
        style:  
        Theme.of(context)  
  
.textTheme  
  
.headline6  
  
?.copyWith(color: Theme.of(context).textTheme.headline1?.color)),  
        const  
SizedBox(height: 15),  
  
ListView.builder(  
        physics:  
        const NeverScrollableScrollPhysics(),  
        shrinkWrap:  
        true,  
        itemCount:  
        currentCustomiseTitle[1].length, //For each customise option  
  
itemBuilder: (context, indexOption) {
```

```
List  
currentCustomiseOption = currentCustomiseTitle[1][indexOption];  
  
return  
Padding(  
  
padding: const EdgeInsets.only(left: 10, bottom: 10),  
  
child:  
Row(children: [  
  
CircleAvatar(  
  
backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
  
radius: 10,  
  
child: Text(currentCustomiseOption[1].toString(),  
  
style: Theme.of(context)  
  
.textTheme  
  
.bodyText1  
  
? .copyWith(color: Theme.of(context).backgroundColor)),  
  
),  
  
const  

```

```

width: 20,
),
),

Expanded(
child: Text(currentCustomiseOption[2].toString(),
style: Theme.of(context).textTheme.bodyText1))
]),

);
}),

]),
]));

} else if
(currentCustomiseTitle[0][1] == "ADD" &&

currentCustomiseTitle[1].toList().length != 0) {
//If the customise title
type is add and there is at least one customised option for the title return the
customised options in green text

return Container(
padding: const
EdgeInsets.symmetric(vertical: 20, horizontal: 20),
margin: const
EdgeInsets.symmetric(horizontal: 0, vertical: 10),
decoration:
BoxDecoration(

```

```
        color:  
Theme.of(context).backgroundColor.withOpacity(0.5),  
  
        border:  
Border.all(  
  
        color:  
Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),  
  
        borderRadius:  
BorderRadius.circular(10)),  
  
        child: Column(  
  
mainAxisAlignment: MainAxisAlignment.start,  
  
crossAxisAlignment: CrossAxisAlignment.start,  
  
        children: [  
  
Text(currentCustomiseTitle[0][0],  
  
        textAlign:  
 TextAlign.start,  
  
        style:  
Theme.of(context)  
  
.textTheme  
  
.headline6  
  
?.copyWith(color: Theme.of(context).textTheme.headline1?.color)),  
  
        const  
SizedBox(height: 15),
```

```
ListView.builder(  
    physics:  
    const NeverScrollableScrollPhysics(),  
    shrinkWrap:  
    true,  
    itemCount:  
    currentCustomiseTitle[1].length, //For each customise option  
  
    itemBuilder: (context, indexOption) {  
        List  
        currentCustomiseOption = currentCustomiseTitle[1][indexOption];  
  
        return  
        Padding(  
  
            padding: const EdgeInsets.only(left: 10, bottom: 10),  
  
            child:  
            Row(children: [  
  
                CircleAvatar(  
  
                    backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
  
                    radius: 10,  
  
                    child: Text(currentCustomiseOption[1].toString(),  
  
                    style: Theme.of(context)
```

```
.textTheme

.bodyText1

?.copyWith(color: Theme.of(context).backgroundColor)),  
),  
const  
SizedBox(  
  
width: 20,  
),  
  
Expanded(  
  
child: Text(currentCustomiseOption[2].toString(),  
  
style: Theme.of(context)  
  
.textTheme  
  
.bodyText1

?.copyWith(color: Theme.of(context).colorScheme.tertiary)))  
]),  
);  
},  
},
```

```

        ]));

    } else if
(currentCustomiseTitle[0][1] == "REMOVE" &&

currentCustomiseTitle[1].toList().length != 0) {

    //If the customise title
type is remove and there is at least one customised option for the title return
the customised options in red text

    return Container(
        padding: const
EdgeInsets.symmetric(vertical: 20, horizontal: 20),
        margin: const
EdgeInsets.symmetric(horizontal: 0, vertical: 10),
        decoration:
BoxDecoration(
            color:
Theme.of(context).backgroundColor.withOpacity(0.5),
            border:
Border.all(
                color:
Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),
            borderRadius:
BorderRadius.circular(10)),
            child: Column(
mainAxisAlignment: MainAxisAlignment.start,
crossAxisAlignment: CrossAxisAlignment.start,

```

```
children: [  
  
    Text(currentCustomiseTitle[0][0],  
  
          textAlign:  
          TextAlign.start,  
  
          style:  
          Theme.of(context)  
  
          .textTheme  
  
          .headline6  
  
          ?.copyWith(color: Theme.of(context).textTheme.headline1?.color)),  
  
          const  
          SizedBox(height: 15),  
  
          ListView.builder(  
  
              physics:  
              const NeverScrollableScrollPhysics(),  
  
              shrinkWrap:  
              true,  
  
              itemCount:  
              currentCustomiseTitle[1].length, //For each customise option  
  
              itemBuilder: (context, indexOption) {  
  
                  List  
                  currentCustomiseOption = currentCustomiseTitle[1][indexOption];
```

```
        return  
Padding(  
  
padding: const EdgeInsets.only(left: 10, bottom: 10),  
  
        child:  
Row(children: [  
  
CircleAvatar(  
  
backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
  
radius: 10,  
  
child: Text(currentCustomiseOption[1].toString(),  
  
style: Theme.of(context)  
  
.textTheme  
  
.bodyText1  
  
? .copyWith(color: Theme.of(context).backgroundColor)),  
  
,  
  
const  

```

```
Expanded(  
  
    child: Text(currentCustomiseOption[2].toString(),  
  
    style: Theme.of(context)  
  
.textTheme  
  
.bodyText1  
  
?.copyWith(color: Theme.of(context).colorScheme.error)))  
  
        ],  
  
    );  
  
    },  
  
    ]));  
  
} else {  
  
    return const SizedBox(  
  
        height: 0,  
  
    );  
  
}  
  
},  
  
Padding(  
  
    padding: const  
EdgeInsets.symmetric(vertical: 10, horizontal: 20),
```

```
        child: Row(mainAxisAlignment:  
MainAxisAlignment.end, children: [  
  
        Text(  
  
            "Price: ",  
  
            style: Theme.of(context)  
                .textTheme  
                .headline6  
  
            ?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
        ),  
  
        Text(  
  
            "£",  
  
            style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).primaryColor),  
  
        ),  
  
        Text(  
  
            double.parseDouble(currentItem[0][2]).toStringAsFixed(2),  
  
            style: Theme.of(context)  
                .textTheme  
                .headline6  
  
            ?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
        )  
    )
```

```

        ]))

    ))));

}

]);

} else {

    return LayoutBuilder(
        //Calculate the price for each profile in the botttom
section of the cart

        builder: (p0, p1) {

            double subtotal = 0;

            bool isOneRestaurant = true;

            int tempRestaurantID = -1;

            for (int iProfile = 0; iProfile <
cartInfo[0]["cartinfo"].length; iProfile++) {

                //For each profile in the cart

                cartInfo[0]["cartinfo"][iProfile].add(0.00);

//Add total price to the end of the profile index in the cartInfo

                List itemPriceKeys =
cartInfo[0]["cartinfo"][iProfile][6].keys.toList(); //Create a list of keys for
each item

                for (int iItem = 0; iItem < itemPriceKeys.length;
iItem++) {

                    //For the first check, the default value is -1
so replace with the first restaurant id. For the remaining items check if it is
equal to the current restaurant id and if it isnt set OneRestaurant to false

                    if (tempRestaurantID == -1) {

```

```

tempRestaurantID =
int.parse(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][4]);

} else {

    if
(int.parse(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][4]) != tempRestaurantID) {

        isOneRestaurant = false;

    }

}

//For each item

cartInfo[0]["cartinfo"][iProfile][7] =
(cartInfo[0]["cartinfo"][iProfile][7] * 100 +


double.parseDouble(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][2]) * 100) /

100; //Add price to the total price for the profile

}

}

for (int iSubtotalProfile = 0; iSubtotalProfile <
cartInfo[0]["cartinfo"].length; iSubtotalProfile++) {

    subtotal = (subtotal * 100 +
cartInfo[0]["cartinfo"][iSubtotalProfile][7] * 100) /

```

```

        100; //Add the total price for profile to
subtotal

    }

    return Column(children: [
        const SizedBox(height: 50),
        ListView.builder(
            physics: const
NeverScrollableScrollPhysics(),
            shrinkWrap: true,
            itemCount: cartInfo[0]["cartinfo"].length,
//For each profile

            itemBuilder: (context, iProfilePrice) {
                //For each profile, display the profile
total price

                return Padding(
                    padding: const
EdgeInsets.symmetric(horizontal: 40, vertical: 10),
                    child: Row(
                        mainAxisAlignment:
MainAxisAlignment.end,
                        children: [
                            Text(
                                "${cartInfo[0]["cartinfo"][iProfilePrice][1]}"
                                "${cartInfo[0]["cartinfo"][iProfilePrice][2]}: ",
                                style: Theme.of(context)

```

```
        .textTheme  
        .headline6  
        ?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
        ),  
        const SizedBox(  
        width: 50,  
        ),  
  
Text("£${cartInfo[0]["cartinfo"][iProfilePrice][7].toStringAsFixed(2)}",  
        style:  
Theme.of(context).textTheme.headline6?.copyWith(fontWeight: FontWeight.w500))  
        ],  
        ));  
    },  
    Padding(  
        //Display subtotal  
        padding: const  
EdgeInsets.symmetric(horizontal: 40, vertical: 10),  
        child: Row(mainAxisAlignment:  
MainAxisAlignment.end, children: [  
            Text("SUBTOTAL", style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).primaryColor)),  
            const SizedBox(  
        
```

```

        width: 50,
    ),
    Text("£${subtotal.toStringAsFixed(2)}",
        style: Theme.of(context)
            .textTheme
            .headline6
            ?.copyWith(fontWeight:
FontWeight.w500, color: Theme.of(context).primaryColor))
    ])),
LayoutBuilder(
    //Checkout button with checks for multiple
conditions
    builder: (p0, p1) {
        if (isOneRestaurant == true) {
            //If there is only one restaurant being
ordered from
            List itemKeys =
cartInfo[0]["cartinfo"][0][6].keys.toList(); // Get a list of item ids
            try {
                if
(double.parseDouble(cartInfo[0]["cartinfo"][0][6][itemKeys[0]][0][8]) <= subtotal) {
                    //If the subtotal is more than the
minimum order price for the restaurant return the
                    return Padding(

```

```

padding: const
EdgeInsets.symmetric(horizontal: 20, vertical: 30),

child: Row(children: [
    Expanded(
        child: ElevatedButton(
            onPressed: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder:
                    (context) => Checkout(
                        cartInfo: cartInfo[0]["cartinfo"],
                    )));
            },
        ),
        child: const
    Text("Checkout")))
    ]));

} else {
    // If it isn't more than the minimum
    // order price, display the minimum order price in greyed out button
    return Padding(
        padding: const
    EdgeInsets.symmetric(horizontal: 20, vertical: 30),
        child: Row(children: [

```

```
        Expanded(  
          child: Opacity(  
            opacity: 0.2,  
            child: ElevatedButton(  
              style: ButtonStyle(  
                backgroundColor:  
  
MaterialStatePropertyAll(Theme.of(context).colorScheme.onBackground)),  
                onPressed: null,  
                child: Text(  
                  "Minimum order  
£${cartInfo[0]["cartinfo"][0][6][itemKeys[0]][0][8]}",  
                  textAlign:  
                  TextAlign.center,  
                  style:  
                  Theme.of(context)  
                    .textTheme  
                    .headline6  
  
?.copyWith(color: Theme.of(context).backgroundColor),  
          ))))  
        ]));  
      }  
    } catch (e) {
```

```
// If it fails to check the price, return  
that there is an error. This fixes when the first item is removed and app tries  
to check if it is equal during the rebuild  
  
        return Padding(  
            padding: const  
EdgeInsets.symmetric(horizontal: 20, vertical: 30),  
  
            child: Row(children: [  
  
                Expanded(  
                    child: ElevatedButton(  
                        style:  
  
ButtonStyle(backgroundColor:  
MaterialStatePropertyAll(Theme.of(context).colorScheme.error)),  
  
                        onPressed: null,  
  
                        child: Text(  
  
                            "Unknown minimum cart  
price",  
  
                            textAlign:  
TextAlign.center,  
  
                        style:  
  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).backgroundColor),  
  
                    ))  
  
                ]));  
        }  
    }  
}
```

```
    } else {

        //Return that there is more than one
        restaurant in the cart

        return Padding(
            padding: const
            EdgeInsets.symmetric(horizontal: 20, vertical: 30),
            child: Row(children: [
                Expanded(
                    child: Opacity(
                        opacity: 0.2,
                        child: ElevatedButton(
                            style: ButtonStyle(
                                backgroundColor:
                                MaterialStatePropertyAll(Theme.of(context).colorScheme.onBackground)),
                            onPressed: null,
                            child: Text(
                                "Multiple Restaurant
Delivery Unavailable",
                                textAlign:
                                TextAlign.center,
                                style:
                                Theme.of(context)
                                    .textTheme
                                    .headline6
                            )
                        )
                    )
                )
            ],
        );
    }
}
```

```
? .copyWith(color:  
Theme.of(context).backgroundColor),  
))));  
}  
},  
)  
]);  
},  
);  
}  
});  
}  
} else {  
return Padding(  
padding: const EdgeInsets.all(30),  
child: SizedBox(  
width: double.infinity,  
child: Center(  
child: Column(children: [  
Text(  
"The cart is empty",  
textAlign: TextAlign.center,  
style: Theme.of(context).textTheme.headline1,
```

```
        ),  
  
        const SizedBox(  
  
            height: 5,  
  
  
        Text(  
  
            "Try adding an item through the browse page.",  
  
            textAlign: TextAlign.center,  
  
            style: Theme.of(context).textTheme.bodyText1,  
  
        ),  
  
        const SizedBox(  
  
            height: 20,  
  
        ),  
  
    ]))),  
  
);  
  
}  
  
}  
  
} else {  
  
    return Row(mainAxisAlignment: MainAxisAlignment.center, children: [  
  
        Padding(  
  
            padding: const EdgeInsets.all(50),  
  
            child: CircularProgressIndicator(  
  
                color: Theme.of(context).primaryColor,  

```

cart_service.dart

```
import 'package:alleat/services/localprofiles_service.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:sqflite/sqflite.dart' as sql;

class SQLiteCartItems {

    // -----
    // Cart Items Table
    // -----


static Future<void> createTableCartItems(sql.Database database) async {
    //Create cart table (Used to store items and their customised details)
    await database.execute("""
        CREATE TABLE cartItems(
            cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
            itemid INTEGER NOT NULL,
            quantity INTEGER NOT NULL,
            price REAL NOT NULL,
            total REAL NOT NULL
        )
    """);
}
```

```
profileid INTEGER,  
itemid INT,  
customised TEXT,  
quantity INT,  
FOREIGN KEY (profileid) REFERENCES localprofiles(profileid)  
)  
""");  
}  
  
static Future<sql.Database> cartdb() async {  
    //If tables don't exist, create tables  
    return sql.openDatabase(  
        'alleatlocal.db',  
        version: 1,  
        onCreate: (sql.Database database, int version) async {  
            await SQLiteLocalProfiles.createTableProfile(database);  
            await createTableCartItems(database);  
        },  
    );  
}  
//-----
```

```
// Edit Cart

//-----

// Add to Cart

static Future<bool> addToCart(int itemid, dynamic customised, int quantity)
async {

    try {

        final db = await SQLiteCartItems.cartdb();

        final prefs = await SharedPreferences.getInstance();

        final String? profileid = prefs.getString('serverprofileid'); //Try to get
profileid of current user and convert

        if (profileid == null) {

            //If there is no current user selected

            return false;
        }

        int profileidInt = int.parse(profileid);

        await db.insert("cartItems", {

            "profileid": profileidInt,

            "itemid": itemid,

            "customised": customised,

            "quantity": quantity,
        });
    }
}
```

```
        return true;

    } catch (e) {
        //If fails to add to cart
        return false;
    }
}

// Get a list of profiles which are in the cart

static Future<List> getProfilesInCart() async {

    final db = await SQLiteCartItems.cartdb();

    List profilesInCart = await db.rawQuery("SELECT profileid FROM cartItems");

    List singleProfilesInCart = [];

    for (int i = 0; i < profilesInCart.length; i++) {

        if (!singleProfilesInCart.contains(profilesInCart[i]["profileid"])) {

            singleProfilesInCart.add(profilesInCart[i]["profileid"]);
        }
    }

    return singleProfilesInCart;
}

// Get a list of items that are under a profile ID
```

```
static Future<List> getProfileCart(profileID) async {

    final db = await SQLiteCartItems.cartdb();

    List itemsInCart = await db.rawQuery("SELECT cartid, itemid, customised,
quantity FROM cartItems WHERE profileid = $profileID");

    return itemsInCart;

}

//Remove item from cart

static Future<void> removeItem(cartID) async {

    final db = await SQLiteCartItems.cartdb();

    await db.rawDelete("DELETE FROM cartItems WHERE cartid = $cartID");

}

//Remove all items from cart

static Future<void> clearCart() async {

    final db = await SQLiteCartItems.cartdb();

    await db.rawDelete("DELETE FROM cartItems");

}

}
```

checkout.dart

```
import 'dart:async';

import 'package:alleat/services/cart_service.dart';

import 'package:alleat/services/queryserver.dart';

import 'package:alleat/widgets/elements/elements.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:google_maps_flutter/google_maps_flutter.dart';

import 'package:currency_text_input_formatter/currency_text_input_formatter.dart';

import 'dart:convert';

class Checkout extends StatefulWidget {

    final List cartInfo;

    const Checkout({Key? key, required this.cartInfo}) : super(key: key);

    @override

    State<Checkout> createState() => _CheckoutState();

}

class _CheckoutState extends State<Checkout> {

    final Set<Marker> markers = {};//markers for google map

    String selectedTip = "none";//Percentage tip (15%, 20%, 25%, custom, none)
```

```

double tipPrice = 0; //Tip price

double subtotal = 0; //Subtotal for items

static TextEditingController customAmount = TextEditingController(text:
"£0.00"); //Custom price for tip (text field)

Future<Map> sendCart(cart, tip, address, latitude, longitude) async {

    //Send cart to the server - formatted before sending

    int indexid = 0; //Index of the current item so that the customise 2D array
can link to the item 2D array

    List iteminfo = []; //(profile id, item id, item quantity)

    List customiseinfo = []; //(index id, customise option id, option quantity)

    int restaurantid = 0; //restaurant id

    for (int i = 0; i < cart.length; i++) {

        //For each profile

        int profileid = cart[i][0]; //Save the profile id as profileid

        List itemkeys = cart[i][6].keys.toList();

        for (int j = 0; j < itemkeys.length; j++) {

            //For each item

            List item = cart[i][6][itemkeys[j]];

            restaurantid = int.parse(item[0][4]);

            int itemid = int.parse(item[0][12]); //Save item id as itemid

            int itemQuantity = item[0][6];

            iteminfo.add([

```

```

profileid,
itemid,
itemQuantity,
]);

List customisekeys = item[1].keys.toList();

for (int k = 0; k < customisekeys.length; k++) {

    //For each customise title

    List customiseTitle = item[1][customisekeys[k]];

    for (int l = 0; l < customiseTitle[1].length; l++) {

        // For each option

        List customiseOption = customiseTitle[1][l];

        int customiseOptionid = int.parse(customiseOption[0]); //Save option
id as customiseoptionid

        int optionQuantity = customiseOption[1];

        customiseinfo.add([indexid, customiseOptionid, optionQuantity]);

    }

}

indexid++; //Add one to index (incremental for each item)

}

var res = await QueryServer.query("https://alleat.cpur.net/query/orders.php",
{

```

```

//Send data to orders.php . If there is an error, it returns back the error
code

    "type": "add",

    "tip": tip,

    "address": address,

    "latitude": latitude,

    "longitude": longitude,

    "restaurantid": restaurantid.toString(),

    "iteminfo": json.encode(iteminfo),

    "customiseinfo": json.encode(customiseinfo),

});

return res; //return result (contains basic message either with error=true
and the error or error=false and success)

}

```

```

Future<List> getDeliveryDestination() async {

    final prefs = await SharedPreferences.getInstance(); // Get saved location
from shared preferences

    final double? savedLocationLat = prefs.getDouble('locationLatitude');

    final double? savedLocationLng = prefs.getDouble('locationLongitude');

    final List<String>? savedLocationText =
prefs.getStringList('locationPlacemark');

    if (savedLocationLat == null || savedLocationLng == null || savedLocationText
== null) {

```

```

    //If either the latitude, longitude or the text is null return empty array
    to ensure that it cannot return a partial result ACID

    return [];
}

} else {

    savedLocationText.addAll([savedLocationLat.toString(),
    savedLocationLng.toString()]);

    return savedLocationText.toList();
}

}

}

Set<Marker> getmarkers(restaurantAddress, restaurantLat, restaurantLng,
destination, destinationLat, destinationLng) {

    //Create the two markers on the map with their positions

    //markers to place on map

    BitmapDescriptor restaurantMarkerIcon =
BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueBlue);

    BitmapDescriptor destinationMarkerIcon =
BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueRose);

    markers.add(Marker(
        //add first marker

        markerId: const MarkerId("Restaurant"),

        position: LatLng(restaurantLat, restaurantLng), //position of restaurant

        infoWindow: const InfoWindow(
            //popup info

```



```
        child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
          const ScreenBackButton(),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 10),
            child: Text(
              "Checkout.",
              style: Theme.of(context).textTheme.headline1,
            )),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 40),
            child: Text(
              "Location",
              style: Theme.of(context).textTheme.headline2,
            )),
        ],
        const SizedBox(
          height: 20,
        ),
        FutureBuilder<List>(
          future: getDeliveryDestination(), //Get the delivery location from shared preferences
          builder: (context, snapshot) {
            if (snapshot.hasData) {
```

```

List destination = snapshot.data ?? [];

if (destination != []) {

    List locationItemKeys = widget.cartInfo[0][6].keys

        .toList(); //Get the item ids for the first profiles so that it
can be used to get the restaurant info saved for it

    final List restaurantAddress =

        widget.cartInfo[0][6][locationItemKeys[0]][0][9].split(',');
//Split the restaurant address into seperate items in array

    final double restaurantLat =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][10]);

    final double restaurantLng =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][11]);

    final controllerMap = Completer<GoogleMapController>(); //Google
Maps Controller

    final double destinationLat = double.parse(destination[4]);
    final double destinationLng = double.parse(destination[5]);

}

return Column(children: [
    SizedBox(
        //Create google map with height 200px without any movement
        controls centered around the destination of delivery
        width: double.infinity,
        height: 200,
        child: LayoutBuilder(builder: (BuildContext context,
BoxConstraints constraints) {

```

```
        return GoogleMap(  
  
            myLocationEnabled: false,  
  
            compassEnabled: false,  
  
            mapToolbarEnabled: false,  
  
            zoomGesturesEnabled: true,  
  
            // hide location button  
  
            myLocationButtonEnabled: false,  
  
            mapType: MapType.normal,  
  
            zoomControlsEnabled: false,  
  
            rotateGesturesEnabled: false,  
  
            tiltGesturesEnabled: false,  
  
            // camera position  
  
            initialCameraPosition: CameraPosition(target:  
LatLng(destinationLat, destinationLng), zoom: 12),  
  
            onMapCreated: (GoogleMapController controller) {  
  
                controllerMap.complete(controller);  
  
            },  
  
            markers: getmarkers(restaurantAddress, restaurantLat,  
restaurantLng, destination, destinationLat, destinationLng));  
  
        )),  
  
    ),  
  
    const SizedBox(  
  
        height: 20,
```

```
        ),

        //Column with the locations of the destination and restaurant
using the colour coded markers as a key

        Padding(
            padding: const EdgeInsets.symmetric(horizontal: 40, vertical:
20),

            child: Row(
                children: [
                    const Icon(
                        Icons.location_on,
                        size: 30,
                        color: Color(0xffca458f),
                    ),
                    const SizedBox(
                        width: 30,
                    ),
                    Expanded(
                        child: Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                                Text(
                                    "Delivery Address",

```

```
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
,  
        const SizedBox(  
height: 10,  
,  
Text(  
"${destination[0]}, ${destination[1]}",  
style: Theme.of(context).textTheme.bodyText1,  
,  
Text(  
"${destination[2]}, ${destination[3]}",  
style: Theme.of(context).textTheme.bodyText1,  
,  
],  
)),  
],  
)),  
Padding(  
padding: const EdgeInsets.symmetric(horizontal: 40, vertical:  
20),  
child: Row(  
children: [  
]
```

```
const Icon(  
    Icons.location_on,  
    size: 30,  
    color: Color(0xff4133e3),  
,  
const SizedBox(  
    width: 30,  
,  
Expanded(  
    child: Column(  
        crossAxisAlignment: CrossAxisAlignment.start,  
        children: [  
            Text(  
                "Restaurant Address",  
                style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
,  
            const SizedBox(  
                height: 10,  
,  
            Text(  
,
```

```
"${widget.cartInfo[0][6][locationItemKeys[0]][0][3]}:  
${widget.cartInfo[0][6][locationItemKeys[0]][0][9]}",  
        style: Theme.of(context).textTheme.bodyText1,  
    ),  
],  
))  
],  
)),  
]);  
} else {  
    return const Text("Failed to get location");  
}  
} else {  
    return const Text("Getting Destination");  
}  
},  
,  
//Tipping section  
Padding(  
padding: const EdgeInsets.symmetric(vertical: 20),  
child: Divider(  
thickness: 2,
```

```
        color:  
Theme.of(context).textTheme.headline6?.color?.withOpacity(0.5),  
  
        indent: 40,  
  
        endIndent: 40,  
    )),  
  
Padding(  
  
padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 10),  
  
child: Text(  
  
"Tipping",  
  
style: Theme.of(context).textTheme.headline2,  
)),  
  
const SizedBox(  
  
height: 10,  
),  
  
LayoutBuilder(builder: (p0, p1) {  
  
subtotal = 0;  
  
for (int i = 0; i < widget.cartInfo.length; i++) {  
  
//For each profile, get the profile item total price and add it to the  
current subtotal  
  
subtotal = (subtotal * 100 + widget.cartInfo[0][7] * 100) / 100;  
}  
  
//Tipping buttons  
  
return Padding(  
)
```

```
padding: const EdgeInsets.symmetric(horizontal: 10),  
  
child: Column(children: [  
  
    Row(  
  
        //Row of preset percentage tips  
  
        children: [  
  
            Expanded(  
  
                //15% tip button  
  
                child: Padding(  
  
                    padding: const EdgeInsets.all(10),  
  
                    child: InkWell(  
  
                        onTap: () {  
  
                            //On button tap, change the selected tip to 15%  
                            //so that the button colour changes and set the tip price to 15% of the subtotal  
  
                            setState(() {  
  
                                selectedTip = "15%";  
  
                                tipPrice = double.parse((((subtotal * 100) *  
0.15) / 100).toStringAsFixed(2));  
  
                            });  
  
},  
  
            child: Container(  
  
                alignment: Alignment.center,  
  
                decoration: BoxDecoration(  
  
                    color:  
Theme.of(context).colorScheme.onSurface,
```

```
borderRadius: BorderRadius.circular(10),  
  
border: Border.all(  
  
    color: (selectedTip == "15%") ?  
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,  
  
    width: 2),  
  
boxShadow: [  
  
    BoxShadow(  
  
        color: Colors.black.withOpacity(0.01),  
  
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes  
position of shadow  
  
    ),  
  
]),  
  
child: AspectRatio(  
  
    aspectRatio: 1 / 1,  
  
    child: Column(  
  
        mainAxisAlignment:  
MainAxisAlignment.center,  
  
        children: [  
  
            Text(  
  
                "15%",  
  
                style:  
Theme.of(context).textTheme.headline3,
```

```

),
const SizedBox(
height: 5,
),
Text(
"+\${(((subtotal * 100) * 0.15) /
100).toStringAsFixed(2)}",
style:
Theme.of(context).textTheme.bodyText1,
)
],
)))),
Expanded(
//20% tip button
child: Padding(
padding: const EdgeInsets.all(10),
child: InkWell(
onTap: () {
setState(() {
//On button tap, change the selected tip to 20%
so that the button colour changes and set the tip price to 20% of the subtotal
selectedTip = "20%";
tipPrice = double.parse(((subtotal * 100) *
0.20) / 100).toStringAsFixed(2));
}

```

```
        });

    },
    child: Container(
      alignment: Alignment.center,
      decoration: BoxDecoration(
        color:
Theme.of(context).colorScheme.onSurface,
        borderRadius: BorderRadius.circular(10),
        border: Border.all(
          color: (selectedTip == "20%") ?
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,
          width: 2),
        boxShadow: [
          BoxShadow(
            color: Colors.black.withOpacity(0.01),
            spreadRadius: 2,
            blurRadius: 10,
            offset: const Offset(0, 10), // changes
position of shadow
          ),
        ],
        child: AspectRatio(
          aspectRatio: 1 / 1,
          child: Column(
```

```
        mainAxisAlignment:  
MainAxisAlignment.center,  
  
        children: [  
  
            Text(  
  
                "20%",  
  
                style:  
Theme.of(context).textTheme.headline3,  
  
            ),  
  
            const SizedBox(  
  
                height: 5,  
  
            ),  
  
            Text(  
  
                "+\${(((subtotal * 100) * 0.20) /  
100).toStringAsFixed(2)}",  
  
                style:  
Theme.of(context).textTheme.bodyText1,  
  
            )  
  
        ],  
  
    ))))),  
  
    Expanded(  
  
        //25% tip button  
  
        child: Padding(  
  
            padding: const EdgeInsets.all(10),  
  
            child: InkWell(  
  
               
```

```
        onTap: () {

            //On button tap, change the selected tip to 25%
            so that the button colour changes and set the tip price to 25% of the subtotal

            setState(() {

                selectedTip = "25%";

                tipPrice = double.parse(((subtotal * 100) *
0.25) / 100).toStringAsFixed(2));

            });

        },
        child: Container(
            alignment: Alignment.center,
            decoration: BoxDecoration(
                color:
Theme.of(context).colorScheme.onSurface,
                borderRadius: BorderRadius.circular(10),
                border: Border.all(
                    color: (selectedTip == "25%") ?
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,
                    width: 2),
                boxShadow: [
                    BoxShadow(
                        color: Colors.black.withOpacity(0.01),
                        spreadRadius: 2,
                        blurRadius: 10,
```

```
        offset: const Offset(0, 10), // changes  
position of shadow  
  
    ),  
  
    ]),  
  
    child: AspectRatio(  
  
        aspectRatio: 1 / 1,  
  
        child: Column(  
  
            mainAxisAlignment:  
MainAxisAlignment.center,  
  
            children: [  
  
                Text(  
  
                    "25%",  
  
                    style:  
Theme.of(context).textTheme.headline3,  
  
                ),  
  
                const SizedBox(  
  
                    height: 5,  
  
                ),  
  
                Text(  
  
                    "+E$((((subtotal * 100) * 0.25) /  
100).toStringAsFixed(2))",  
  
                    style:  
Theme.of(context).textTheme.bodyText1,  
  
                )  
            )  
        )  
    )  
);
```



```
    });

    },
    child: Container(
        alignment: Alignment.center,
        height: 60,
        decoration: BoxDecoration(
            color: Theme.of(context).colorScheme.onSurface,
            border: Border.all(
                color: (selectedTip == "custom") ?
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,
                width: 2),
            borderRadius: BorderRadius.circular(10),
            boxShadow: [
                BoxShadow(
                    color: Colors.black.withOpacity(0.01),
                    spreadRadius: 2,
                    blurRadius: 10,
                    offset: const Offset(0, 10), // changes
position of shadow
                ),
            ],
            child: (selectedTip == "custom") //If the selected
button is "custom" display text field that is in the currency format
                ? Padding(

```

```
padding: const EdgeInsets.symmetric(horizontal:  
40, vertical: 10),  
  
child: Row(  
  
children: [  
  
Text(  
  
"Tip:",  
  
style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
),  
  
const SizedBox(  
  
width: 10,  
  
),  
  
Expanded(  
  
child: TextFormField(  
  
onChanged: (value) {  
  
try {  
  
tipPrice =  
  
double.parseDouble(double.parse((customAmount.text.toString().split("£"))[1])).toStringAsFixed(2));  
  
} catch (e) {  
  
tipPrice = 0;  
  
}  
}
```

```

        },

        controller: customAmount,

        keyboardType: TextInputType.number,

        style:

Theme.of(context).textTheme.bodyText2,

        inputFormatters:

[CurrencyTextInputFormatter(symbol: '£')],


        decoration: (InputDecoration(

            hintText: "£3.21",

            contentPadding:

Theme.of(context).inputDecorationTheme.contentPadding,


            border:

Theme.of(context).inputDecorationTheme.border,


            focusedBorder:

Theme.of(context).inputDecorationTheme.focusedBorder,


            enabledBorder:

Theme.of(context).inputDecorationTheme.enabledBorder,


            floatingLabelBehavior:

FloatingLabelBehavior.never))),

        )

    ],
}

//If the custom price is not selected, show title
"custom tip amount" instead of the text field

: Text(


    "Custom Tip Amount",

```

```
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline3?.color),  
        ))),  
  
    Padding(  
  
        //No tip button  
  
        padding: const EdgeInsets.all(10),  
  
        child: InkWell(  
  
            onTap: () {  
  
                //On button press, change selected to be "none" so that  
it changes the highlight colour for it and set the tip price to be 0  
  
                setState(() {  
  
                    selectedTip = "none";  
  
                    tipPrice = 0;  
  
                });  
  
            },  
  
            child: Container(  
  
                alignment: Alignment.center,  
  
                height: 60,  
  
                decoration: BoxDecoration(  
  
                    color: Theme.of(context).colorScheme.onSurface,  
  
                    borderRadius: BorderRadius.circular(10),  
  
                    border: Border.all(  

```

```
        color: (selectedTip == "none") ?  
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,  
  
        width: 2),  
  
        boxShadow: [  
  
        BoxShadow(  
  
        color: Colors.black.withOpacity(0.01),  
  
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes  
position of shadow  
  
> ),  
  
    ]),  
  
    child: Text(  
  
    "No Tip",  
  
    style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline3?.color),  
  
    ))))  
  
]);  
},  
  
const SizedBox(  
  
height: 20,  
> ),  
  
Padding(  
)
```

```

padding: const EdgeInsets.symmetric(vertical: 20),

child: Divider(
  thickness: 2,
  color:
Theme.of(context).textTheme.headline6?.color?.withOpacity(0.5),
  indent: 40,
  endIndent: 40,
)),

LayoutBuilder(builder: ((p0, p1) {
  //Checkout final price section

  List priceItemKeys = widget.cartInfo[0][6].keys.toList(); //get the item
  ids for the first profile in the cart

  double total = (subtotal * 100 +
  double.parse(widget.cartInfo[0][6][priceItemKeys[0]][0][5]) * 100 + tipPrice *
  100) /
  100; //Total is equal to the subtotal + delivery price + tip

  return Padding(
    padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 20),
    child: Row(mainAxisAlignment: MainAxisAlignment.end, children: [
      Row(
        children: [
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [

```

```
    Text(
        "Subtotal",
        style: Theme.of(context).textTheme.headline6,
    ),
    Text(
        "Delivery Fee",
        style: Theme.of(context).textTheme.headline6,
    ),
    Text(
        "Tip",
        style: Theme.of(context).textTheme.headline6,
    ),
    const SizedBox(height: 20),
    Text(
        "Total",
        style:
Theme.of(context).textTheme.headline5?.copyWith(color:
Theme.of(context).primaryColor),
    )
],
),
const SizedBox(
    width: 50,
```

```

),
Column(
  mainAxisAlignment: MainAxisAlignment.end,
  children: [
    Text(
      "£${subtotal.toStringAsFixed(2)}",
      style: Theme.of(context).textTheme.bodyText2,
    ),
    Text(
      "£${widget.cartInfo[0][6][priceItemKeys[0]][0][5].toString()}",
      style: Theme.of(context).textTheme.bodyText2,
    ),
    Text(
      "£${tipPrice.toStringAsFixed(2)}",
      style: Theme.of(context).textTheme.bodyText2,
    ),
    const SizedBox(height: 20),
    Text("£${total.toStringAsFixed(2)}",
      style:
        Theme.of(context).textTheme.headline5?.copyWith(color:
        Theme.of(context).textTheme.headline1?.color)),
  ],
)

```

```
        ],
    ),
]);
})),  
  
Row(mainAxisAlignment: MainAxisAlignment.center, children: [
    Expanded(
        //Checkout button
        child: Padding(
            padding: const EdgeInsets.symmetric(vertical: 30, horizontal:
20),
            child: ElevatedButton(
                onPressed: () async {
                    //On button press get the delivery location and join the
destination address into a single string
                    List destination = await getDeliveryDestination();
                    String destinationAddress = [destination[0],
destination[1], destination[2], destination[3]].join(" ,");
                    Map orderCreated = await sendCart(
                        //Send the cart to the server
                        widget.cartInfo,
                        tipPrice.toString(),
                        destinationAddress,
                        destination[4].toString(),
                        destination[5].toString());
                }
            )
        )
    )
]);
```

```

        if (orderCreated["error"] == true) {

            //If there is an error go to the homepage and display
            snackbar with error message

            setState(() {

                Navigator.of(context).pop();

                Navigator.of(context).pop();

                ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("ERROR:
${orderCreated["message"]}")));

            });

        } else {

            //If there isn't an error try to clear the cart

            try {

                await SQLiteCartItems.clearCart();

                setState(() {

                    Navigator.of(context).pop();

                    ScaffoldMessenger.of(context).showSnackBar(const
                    SnackBar(content: Text("Successfully ordered.")));

                });

            }

            } catch (e) {

                //If there is an error clearing the cart display error

                setState(() {

                    Navigator.of(context).pop();

                    ScaffoldMessenger.of(context)

```

```

        .showSnackBar(SnackBar(content: Text("ERROR:
Successfully created order but failed to clear cart. \n $e")));
    });

}

}),

child: const Text("Complete Order")))

])

])));

}

}

```

Files - Server

orders.php

```

<?php
$db = "u352759660_m3uFj"; //database name
$dbuser = "u352759660_GDFIr"; //database username
$dbpassword = "F07&Ruvr;0G"; //database password
$dbhost = "localhost"; //database host

$return[ "error" ] = false;
$return[ "message" ] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["type"]);

if($val){

```

```

        if ($_POST["type"] == "add"){ //If the order query type is add
            $addval = isset($_POST["tip"]) && isset($_POST["restaurantid"])
&& isset($_POST["address"]) && isset($_POST["latitude"]) &&
isset($_POST["longitude"]) && isset($_POST["iteminfo"]) &&
isset($_POST["customiseinfo"]); //Check for if it contains the correct
variables
            if($addval){ //If it contains the correct information
                $tip = round((floatval($_POST["tip"])), 2); //Set the tip to
2dp
                $restaurantid = intval($_POST["restaurantid"]); //Set
restaurant id to an integer
                $address = $_POST["address"];
                $latitude = floatval($_POST["latitude"]);
                $longitude = floatval($_POST["longitude"]);
                $iteminfo = json_decode($_POST["iteminfo"], true); //Decode
the json string into a php array
                $customiseinfo = json_decode($_POST["customiseinfo"], true);
//Decode the json string into a php array
                $sqlorderinfo = "INSERT INTO orderinformation SET
restaurantid = '$restaurantid', tip = '$tip', deliveryaddress = '$address',
deliverylatitude = '$latitude', deliverylongitude = '$longitude', status =
'confirming'"; //Create the SQL insert query to add the order information
to the orderinformation table
                if($resorderinfo = mysqli_query($link, $sqlorderinfo)){ //If
the insert query was successful get the orderid (primary key) using the
most recent order id that fits under the same latitude and longitude
inputted
                    $sqlorderid = "SELECT orderid FROM orderinformation WHERE
deliverylatitude = '$latitude' AND deliverylongitude = '$longitude' ORDER
BY orderid DESC LIMIT 1";
                    if($orderidinfo = mysqli_query($link, $sqlorderid)){
                        $orderidrecord = mysqli_fetch_assoc($orderidinfo);
//Get the first result and save the orderid
                        $orderid = intval($orderidrecord["orderid"]);
                        for ($i = 0; $i < count($_POST["iteminfo"]); $i++){
//For each item in the list of items

                            $profileid = intval($iteminfo[$i][0]); //Set
profile id to an integer
                            $itemid = intval($iteminfo[$i][1]); //Set item id
to be an integer
                            $itemquantity = intval($iteminfo[$i][2]); //Set

```

```

item quantity to be an integer

    $sqlorderiteminfo = "INSERT INTO
orderiteminformation SET orderid = '$orderid', profileid = '$profileid',
itemid = '$itemid', quantity = '$itemquantity"'; //Create an SQL insert
query to add the item to the orderiteminformation info using orderid,
profileid and itemid as foreign keys

    if($sqlorderiteminfo = mysqli_query($link,
$sqlorderiteminfo)){ //If is successfully inserted the item information
into the table for the order
        $sqlorderitemid = "SELECT orderitemid FROM
orderiteminformation WHERE profileid = '$profileid' AND itemid = '$itemid'
ORDER BY orderitemid DESC LIMIT 1"; //Get the orderitemid (primary key)
from orderiteminformation
        if($sqlorderitemidinfo = mysqli_query($link,
$sqlorderitemid)){ //If the orderitemid is queried successfully
            $orderitemidrecord =
mysqli_fetch_assoc($sqlorderitemidinfo); //Get the first record
            $orderitemid =
intval($orderitemidrecord["orderitemid"]); // Store it under orderitemid as
an integer
            for ($j = 0; $j < count($customiseinfo);
$j++){ //For each item customise option
                if ($customiseinfo[$j][0] == $i){ //If
the profile is equal to the index of the item, insert it into the table
using the sql insert query
                    $customiseid =
intval($customiseinfo[$j][1]);
                    $quantity =
intval($customiseinfo[$j][2]);
                    $sqlorderitemoptioninfo = "INSERT
INTO ordercustomisediteminformation SET orderitemid = '$orderitemid',
customiseid = '$customiseid', quantity = '$quantity'";
                    if($sqlorderitemoptioninfo =
mysqli_query($link, $sqlorderitemoptioninfo)){
                        $return["message"] = "success";
                    } else{
                        $return["error"] = true;
                        $return["message"] = "failed to
save customised item";
                    }
                }
            }
        }
    }
}

```

```

                }
            }
        } else {
            $return["error"] = true;
            $return["message"] = "failed to verify item
info";

        }

    } else{
        $return["error"] = true;
        $return["message"] = "failed to save item info";
    }
}
}else{
    $return["error"] = true;
    $return["message"] = "failed to verify order info";
}
}else{
    $return["error"] = true;
    $return["message"] = "failed to save order";
}

} else{ //If the data of adding an order doesnt have the correct
data return error
    $return["error"] = true;
    $return["message"] = "data not specified";}

} else{ //If the order query type is not the specified type, return
error
    $return["error"] = true;
    $return["message"] = "order query type unknown";
}

}

else{$return["error"] = true;
    $return["message"] = "data not specified";}

mysqli_close($link); //close mysqli

```

```
header('Content-Type: application/json');
// tell browser that its a json data
echo json_encode($return);
//converting array to JSON string
?>
```

Summary

Prototype 3 completes the ordering process of the food delivery. With the additions of the cart and checkout, the user is now able to go from install all the way to getting their order. Although the order information page was not added, as predicted in the success criteria evaluated at the start, the order successfully goes through to the server and is permanently stored where it can be queried in the future

Evaluation

Development of prototype 3 was quite difficult, with the cart and checkout being relatively difficult due to it being very data driven. If I were to make this again, I would have built the apps using classes and objects. With the current app, it uses arrays, tables and lists in order to display and manage each restaurant with their apps and as a result it makes it harder to query, insert, update and delete data, using indexes instead of calling individual keys.

As well, a big reason I could not complete development with the orders being added was because I was using a trial of the map API which had a 6 month trial and with the trial coming to an end, it is important that the remaining documentation and errors are found before the app breaks.

Project Analysis

Final Testing

With the number of changes made over the many prototypes and changed files which had not had a success criteria, it was important that it worked as intended. At the beginning of the project, I had the thought that the quantity of success criteria was adequate to the number of testable elements but after analysing the success criteria, it was found that more needs to be added in order to check if everything is working as intended.

Incomplete Prototypes

With the size of the app, some success criteria were not able to be started, with some of the major features not being added. With more time and iterations to the project, there is a chance that these could have been completed.

Incomplete Success Criteria
2.2 - On the login/profile creation page, there should be buttons that allow you to continue with Google, Facebook or Apple.
2.3 - If the user cancels the continue with Google, Facebook or Apple, it should not create an account and instead bring them back to the profile creation page where the user can put in their email or login with it again
2.9 - Phone numbers should be 11 digits long and a minimum of 10 digits
2.11 - A card number should not be fully shown, only showing the last 4 digits unless it is being changed
2.12 - When logging into a device and creating a new profile, the account should either be able to be authenticated using a 2FA app or a 2FA message that is sent to the phone set up with the profile.
2.13 - When creating a profile, there should be a search bar that allows you to search foods you are allergic to
2.14 - The allergic foods search function should allow you to select foods to and show them in a list below the search bar
2.15 - When searching for allergic foods, the user should be able to select food categories like vegan or gluten-free
2.16 - Upon making a profile, the user should be able to select up to 10 dishes they like from a list of different dishes.
2.17 - When on the recommendation page, browse page, restaurant page and checkout, the user should be able to change profiles after which it should refresh the page and change depending on the user and the page content like the card information and the recommended restaurants.
2.18 - There should be a button that the user can click that allows them to see information about the current profile that is selected
2.19 - The user should be able to edit their profile details including their name and password

2.20 - The user should be able to add and remove payment cards from their profile
2.23 - Each profile should be able to access their favourites
2.24 - A user should be able to control how long their profile will stay logged in for when on a guest device. This will be done before generating a QR code for the profile to share.
2.25 - Profiles on a guest device will have a timer stating how long until it is logged out
3.3 - The user should be able to flip the sort from highest to lowest and vice versa for restaurants
3.6 - When filtering restaurants, it should show the user how many restaurants there are under each filter and how many will be shown when all the filters are used that are selected.
3.10 - The user should be able to sort food items using a button
3.11 - When sorting food items, it should allow the user to sort by recommended, rating or price
3.12 - The user should be able to flip the sort from highest to lowest and vice versa for food
3.13 - The user should be able to filter food using a button
3.14 - When filtering food, it should allow the user to filter by price range, delivery, pickup, delivery price, food category, and spice level
3.15 - When filtering food, it should show the user how many dishes there are under each filter and how many will be shown when all the filters are used that are selected.
3.16 - Before filtering food, there should be a button to confirm that they would like to filter the dishes.
3.17 - When the user has selected some food filters, the user should have a button they can click to clear the filters and have it show all the dishes
3.18 - If there are no dishes under a category filter, it should not show the filter
3.19 - When an item is recognised to be in the list of foods that a profile is allergic to, it should have an icon to indicate that it contains a food that cannot be eaten
3.20 - If a profile that has an allergy to a food in a dish or cannot eat a food within a dish tries to add a dish to their order, a popup will tell them that they are unable to add the food
3.21 - On the allergy popup that occurs when a user tries to add a dish that they are allergic to, it should include the ingredient(s) that it contains that caused the app to block them from ordering it and the quantity it contains

3.22 - The user should be able to bypass the popup by clicking a small secondary button and having to confirm the bypass
3.23 - When a user is looking at a restaurant, there should be a button to see the location and its opening hours
3.24 - When a user is checking the location of a restaurant, it should show a pin where the user is currently, where the restaurant is and the destination of where the food will be delivered, with a way to get directions to the restaurant.
3.25 - If there is no location of where the device is, it should only show the location of where the restaurant is and the destination, hiding the marker of where the device is
3.26 - When a user is on the recommendation page, it should never include foods that the profile is allergic to
4.1 - There should be a button from the recommendation page and browse page to scan QR codes
4.2 - When the QR code scanner is activated, it should automatically look for QR Codes
4.3 - If the QR code scanner recognises a QR code, it should align the image by the reference points on the code and read it
4.4 - When the correct QR code is scanned, a popup should show, indicating what it has recognised
4.5 - When an incorrect QR code that is either not for the app or is corrupted, it should indicate that it is an invalid QR code
4.6 - For each dish, there should be a button to generate a QR code which can be scanned by another phone to add the dish to the cart under the profile that is currently selected
4.7 - When a user selects to make a QR code, it should be encrypted and made
4.8 - QR codes should have data loss packets where if some of the QR code is hidden or does not scan properly, it can rely on the rest of the other data to put together the missing data
4.9 - When a QR code is scanned, it should be decrypted
4.10 - If a QR code contains profile data, the hash should be checked if it is the same as the one in the database
5.3 - If a user has bypassed the allergy warning that stops them from ordering foods they have been allergic to, an indicator will show that they have done this at the bottom of the page

5.4 - When completing the order, the app should ask the type of payment where you can either pay with one profile or split the order by the price of the food individually that each person will get (the equivalent of ordering for one but the delivery and tip split over the different profiles paying)

5.6 - After the food has been delivered, the app should ask the user to rate their food.

5.7 - When the app asks the user to rate the food, it should only ask them to rate the food that is associated to their profile

5.8 - When a user switches to their profile, it should ask them to rate any foods they ate previously

5.9 - If a user is logged into their profile from another device, it should sync ratings for foods and if they haven't rated a previous dish, ask them to rate it

5.10 - The status of the order should sync with all profiles on all devices they are signed in with

5.11 - Recommendations should be based on number of times ordered, favourites, rating of each dish and ratings of each restaurant

Prototype 4

This prototype was intended to be all about profile interaction and customisation. With each profile, the current use is just visual but it was intended to be useful, being able to have its own cards and favourite restaurant area. The following areas would be added:

- **Favourites section:** In this section, any restaurants that had been favorited would show up in this location being able to interact with. Although this was added in prototype 1A, during the remake of the app in prototype 1B, the location of the favourites page was yet to be determined so was never added. The favourites section would be easy to implement into the app since the code of the restaurant list could be iterated to check for if it is favourites only mode and then the file could be then called from the screen to just display the favourites
- **Orders Area:** With this section, the orders would be able to be viewed, getting live updates on the status of the order. As well, the order summary would be able to be clicked on and it would show the currently selected profile's order with their items. It would then give them the price paid by them. This would be relatively easy to implement, with each order already being stored on the server, it could be grabbed for the currently selected profile stored in shared preferences. The profile id could be sent to the server and by using the order.php file, and the type "Get". Using foreign keys, the

table's can be referenced to get the relevant information and it can then be unpacked on the app.

- **Settings Area:** In this area, the general app's settings could be viewed including the app theme, being able to switch between dark mode and light mode. There would also be a setting to see the status of the server. This would just ping a server with a basic php file which responds with true if it receives it and sends it back. If nothing is returned then it shows on the app that it is offline. Since people do not like having notifications from all apps, app notifications could be customised depending on its importance. For example, the user could disable coupon notifications but have order updates still show. This is quite basic to implement, with notifications being sorted under different categories. This is built into Android from Android Oreo (8.0)
- **Profile Settings Section:** The profile settings page would be used to change the information for a profile. With people forgetting passwords and cards expiring, it is important that people can change their profile information. On this page, the user could add and remove cards, change their first name, last name, email and password. For any profile changes, it would require the user to enter their current password before changing any details due to the profile potentially being on a device that is not associated with the profile. In order to change any details, it would need to perform verification with the server by running a hash check with the current password.

Prototype 5

Prototype 5 was intended to add onto prototype 4 with the ability to add ratings to each item. Each item would be able to be rated individually and the restaurant could be a restaurant as a whole. Depending on the profile currently selected, while on the profile page, if they had recently ordered, it would ask them to rate their order. This data would then be displayed for each restaurant and under each item as an average score. This data could also be used to sort the data since it could be used to filter; If a restaurant had a lot of highly rated reviews recently, it would show up higher when sorting by popularity.

Prototype 6

For prototype 6, it would have been the largest prototype out of all of them, where the recommendation system would be made. The recommendation system would use a neural network to build a judgement on the chance that a user would order an item. By using the rating of the user for an item and the restaurant as well as analysing other user's ordering patterns, a more accurate model could be made to determine which restaurant a user would want to order from. Tags could also be used to determine the general category of food and

restaurants they like to order. To make the recommendations even more accurate, a table could be made, storing the ingredients in each item. All of these data points help the neural network to improve. Since there is likely to only be a few people ordering from the app to start with, the neural network can be tested on a dataset generated. [Kaggle](#) is a good source for data, having extremely large datasets. The recommended restaurants could be displayed on the For You page within the app.

Prototype 7

Prototype 7 would complete the remaining parts of the app, this includes the homepage where many of the trailerd sections could be added together in a summary as well as the search mechanics. On the homepage, it would stitch the favourite restaurants into bubbles which could be selected depending on their distance from the user. It would also display the most recent order if it has been less than 1 week since ordering from them, allowing the user to see the status of the order from the first screen and rate the order if they so wish. The search can also be added to the homepage and the browse page using the same widget. Upon searching, it would break the inputted string into words and try to find the result most suited based on items, restaurant names, tags and locations.

Success Criteria Test Summary

Success Criteria	Justification	Test No.	Test	Test Type
1.1 - All inputs should be checked for SQL injection, checking for key terms that use SQL	This ensures that the database cannot be edited by the user	1	On the profile creation page, it should reject SQL statements for the forename field	Erroneous
		2	On the profile creation page, it should reject SQL statements for the surname field	Erroneous
		3	On the profile creation page, it should reject SQL statements for the email field	Erroneous
		4	On the profile creation page, it	Erroneous

			should reject SQL statements for the email confirm field	
	5		On the profile creation page, it should reject SQL statements for the password field	Erroneous
	6		On the profile creation page, it should reject SQL statements for the password confirm field	Erroneous
	7		On the profile login page, it should reject SQL statements for the email field	Erroneous
	8		On the profile login page, it should reject SQL statements for the password field	Erroneous
	9		On the location selection page, it should reject SQL statements for the address line 1 field	Erroneous
	10		On the location selection page, it should reject SQL statements for the address line 2 field	Erroneous
	11		On the location selection page, it should reject SQL statements for the city/county field	Erroneous
	12		On the location selection page, it should reject SQL statements for the postcode field	Erroneous
	13		On the checkout page, it should reject SQL statements for the custom tip field	Erroneous
1.2 - Passwords used to make and log into a profile should be	Passwords should use a hash to ensure if a database is hacked, the passwords are	14	Checking the database of the server, it should find that they are not readable	Normal
		15	Checking the local database, it should find that the passwords	Normal

kept secure by using a password hash	not readable		are not readable	
1.3 - The database should be in third-normal form and normalised	It reduces the amount of duplicate data, avoids data anomalies, ensures referential integrity, and simplifies data management.	16	Checking the database structure of the server, it should find that there is no duplicate data where unnecessary	Normal
		17	Checking the database structure of the local device, it should find that there is no duplicate data where unnecessary	Normal
(NEW) 2.26 - The user should be able to remove their profile from a device	This ensures that if a user doesn't want the other person to have their profile on the device, they can remove it.	18	There should be a button or gesture to remove a profile from the device	Normal
2.1 - When there is no profiles on the app, the user should be presented with the login/profile creation page	This means that the app will not break when it tries to recommend foods or restaurants, having some data to make recommendations and know any existing allergies.	19	When the user removes the last profile on the device, it should bring the user to the welcome page where they can login or create a new profile	Normal
2.4 - If the user closes the app or	This will mean that the database does	20	When the user force closes the app during the profile creation part, it should go back to the	Erroneous

goes back during profile creation, the app should not process the profile	not have any half made profiles without the correct information		main screen, and a profile should not be created	
		21	When the user clicks on the profile multiple times, it should only create one profile	Extreme
2.5 - When creating a profile using an email, it should only allow emails that contain an @ symbol	To ensure that emails are sent, they should validate that it isn't a plain string	22	An email must not be allowed containing no @ symbols	Erroneous
		23	An email should be allowed containing one @ symbol	Normal
		24	An email should not be allowed that is just @ symbols	Extreme
2.6 - First names and last names should only be accepted if they are under 50 characters each	This ensures that a name does not go over to another line.	25	A first name or last name should be allowed if 50 characters	Boundary
		26	A first name or last name should be allowed if within 50 characters	Normal
		27	A first name or last name should not be allowed if more than 50 characters	Erroneous
2.7 - Passwords made should be a minimum of 8 characters and a max of 99	This ensures that the password is secure since it cannot be quickly guessed	28	A password should be allowed if 8 characters	Boundary
		29	A password should be allowed if 99 characters	Boundary
		30	A password should be allowed if within 8 and 99 characters	Normal
		31	A password should not be allowed if under 8 characters	Erroneous
		32	A password should not be allowed if over 99 characters	Erroneous
(NEW) 2.27 -	This ensures that the user	33	If the exact same password is entered, it should be allowed	Normal

Passwords should be checked if they are the same	knows their password and is able to rewrite it.	34	If the incorrect password is entered, it should not be allowed	Erroneous
		35	If the password is entered with different capitals, it should display as invalid	Erroneous
		36	If extra characters are written around the correct password, it should be invalid	Erroneous
2.8 - Passwords should contain a letter and a number		37	If a password contains both a letter and a number, it should be allowed	Normal
		38	If a password only contains letters, it should be invalid	Erroneous
		39	If a password only contains numbers, it should be invalid	Erroneous
2.10 - When there is no profiles on the app, the user should be presented with the login/profile creation page	This means that the app will not break when it tries to recommend foods or restaurants, having some data to make recommendations and know any existing allergies.	40	When the user removes the last local profile from the device, it should automatically bring them to welcome screen	Normal
(NEW) 2.28 - When the user clicks create profile, it creates a profile on the cloud	This allows for the use of signing in on multiple devices	41	When a user creates a profile, it should send the information to the server and create a profile	Normal

2.21 - A user should be able to sign in with their profile using multiple devices	This means that if you have multiple devices or are signed in on a family/friends phone, you can still use it on another device	42	A user should be able to sign in to the same account from multiple devices	Normal
2.22 - Information of the profile should be synced to the cloud	If you are using multiple devices and change some details or get an order, you could see them on another device without needing to sign out	43	When the user favourites a restaurant on one device, it should sync the favourites with the other device	Normal
		44	When the user removes a favourites for a restaurant on one device, it should sync the favourites with the other device	Normal
		45	If the user favourites a restaurant on both devices at the same time, it should favourite the restaurant	Normal
3.1 - The user should be able to sort restaurants using a button	This will allow the user to change how the many restaurants are shown so that they can see the best option first that they are looking for	46	By default, there should be a sort method selected	Normal
		47	A user should not be able to select two sort methods at the same time	Erroneous
3.2 - When sorting restaurants, it should allow the user to sort by recommended, rating or distance				

3.4 - The user should be able to filter restaurants using a button	This will allow the user to reduce the amount of options and only see the relevant restaurants they are looking for	48	A user should be able to filter by favourites	Normal	
3.5 - When filtering restaurants, it should allow the user to filter by price range, delivery, pickup, delivery price, new		49	When filtering by favourites, it should only show restaurants that are favorited by the selected profile	Normal	
		50	A user should be able to filter by price range	Normal	
		51	A user should be able to change their max delivery fee slider	Normal	
		52	When a user changes the maximum delivery fee, it should show all restaurants in the nearby area with less than or equal to the max delivery fee	Normal	
		53	A user should be able to change their minimum order price slider	Normal	
		54	When a user changes the minimum order price, it should show all restaurants in the nearby area with less than or equal to the minimum order price	Normal	
		55	There should be a clear save button that once pressed, it should update the filters and sorting of the restaurants shown	Normal	

	restaurants remaining			
3.8 - When the user has selected some restaurant filters, the user should have a button they can click to clear the filters and have it show all the restaurants	This decreases the amount of time it takes to unselect the filters, with a single button to undo any changes	56	There should be a clear button where once pressed, it goes back to the default options	Normal
3.9 - When the user has selected a restaurant filter, they should be able to unselect a filter and change the settings	This means if the user wants to remove a filter because it is too refined, they can do it	57	For each filter, they should be editable and customised together	Normal
3.27 - A user should be able to favourite dishes and restaurants using a button	This allows the user to get back to foods/restaurants that they like and indicate to the app that it is a very liked thing	58	Each restaurant should have a favourite button which once pressed, it adds it to a list of favourites for the currently selected profile	Normal
5.1 - There	This means	59	There should be a way to remove	Normal

should be a button to remove an item from the order before it has been checked out.	that if you make a mistake or want to change it, you can		an item from the cart	
		60	If a user removes two items at the same time it should remove the correct items	Normal
5.2 - For each item, it should show which profile the item is for	This ensures that you know which item is for which person and removes the confusion	61	Each profile should have the correct items associated to it	Normal
5.5 - When checking out, there should be a way to input the amount you would like to tip	This helps out delivery drivers and encourages them to continue	62	There should be a way to input the tip amount	Normal
		63	If a user enters something other than numbers, it should not accept it	Erroneous
		64	If a user enters nothing, it should display the tip as 0	Erroneous
5.12 - The checkout and cart should display the prices of all the items as a total	This means that the user knows how much they will be paying	65	There should be text which displays the price total	Normal
		66	The price should update on item removal	Normal
		67	The price should update on item addition	Normal
6.1 - The user should be able to open the application using an app icon with the correct logo	This is useful to make sure that the user goes to the correct application on the phone	68	The app icon should be the icon of All Eat, using a unique logo to differentiate it	Normal

6.2 - By default, the user should be brought to the home page when the app is opened	This will allow for the user to quickly make an order with something they may like	69	The user should be brought to the homepage by default	Normal
--	--	----	---	--------

Testing

Test No. 1

By using an input formatter of only letters and numbers, it blocks the basic SQL injection statements.

```
inputFormatters: [ //Only allows the input of letters a-z and A-Z and @,.-

    FilteringTextInputFormatter.allow(
        RegExp('^[a-zA-Z0-9@,. ]+')
    ),
]
```

When the user tries to type

' or 1=1; drop table profile

It only displays the characters and numbers



For those which bypass the app and just use the URL, it is protected

```
$firstname = mysqli_real_escape_string($link, $firstname);
$lastname = mysqli_real_escape_string($link, $lastname);
$email = mysqli_real_escape_string($link, $email);
$hashpassword = mysqli_real_escape_string($link,
$hashpassword);
```

Test No. 2

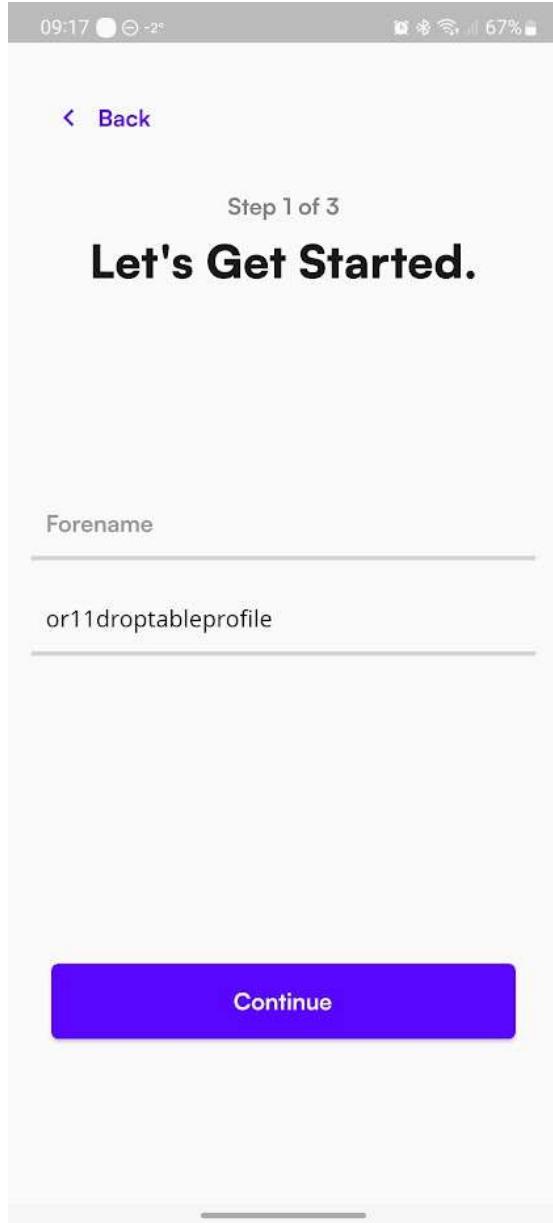
By using an input formatter of only letters and numbers, it blocks the basic SQL injection statements.

```
inputFormatters: [ //Only allows the input of letters a-z and A-Z and @, . -  
    FilteringTextInputFormatter.allow(  
        RegExp('[a-zA-Z0-9@,. -]'))  
    ],
```

When the user tries to type

' or 1=1; drop table profile

It only displays the characters and numbers



For those which bypass the app and just use the URL, it is protected

```
$firstname = mysqli_real_escape_string($link, $firstname);
$lastname = mysqli_real_escape_string($link, $lastname);
$email = mysqli_real_escape_string($link, $email);
$hashpassword = mysqli_real_escape_string($link,
$hashpassword);
```

Test No. 3

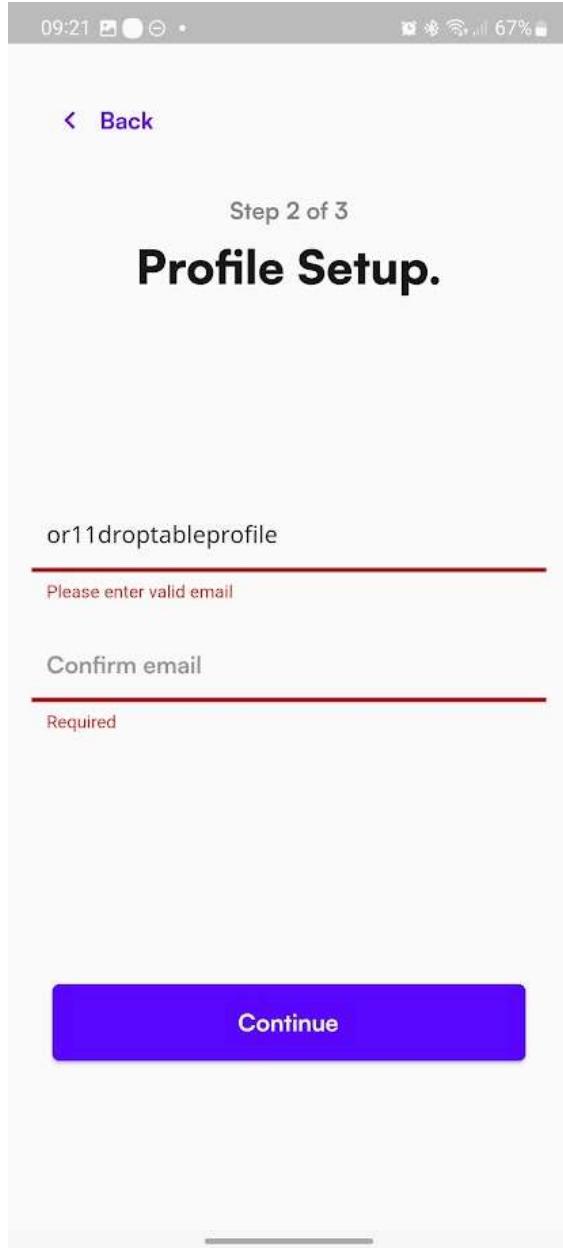
By using an input formatter of only letters, numbers and @ signs, it blocks the basic SQL injection statements.

```
inputFormatters: [ //Only allows the input of letters a-z and A-Z and @, .-
    FilteringTextInputFormatter.allow(
        RegExp('[a-zA-Z0-9@,. -]'))
    ],
]
```

When the user tries to type

' or 1=1; drop table profile

It only displays the characters and numbers



Not only does the SQL inject code not work within the field, but it also is not a valid email

For those which bypass the app and just use the URL, it is protected

```
$firstname = mysqli_real_escape_string($link, $firstname);
$lastname = mysqli_real_escape_string($link, $lastname);
$email = mysqli_real_escape_string($link, $email);
$hashpassword = mysqli_real_escape_string($link,
$hashpassword);
```

Test No. 4

With email confirmations, it does not need to be tested for SQL injection methods since it is not sent to the server, only the email field. This is because the confirm email field validates that the user has typed the correct email field

Test No. 5

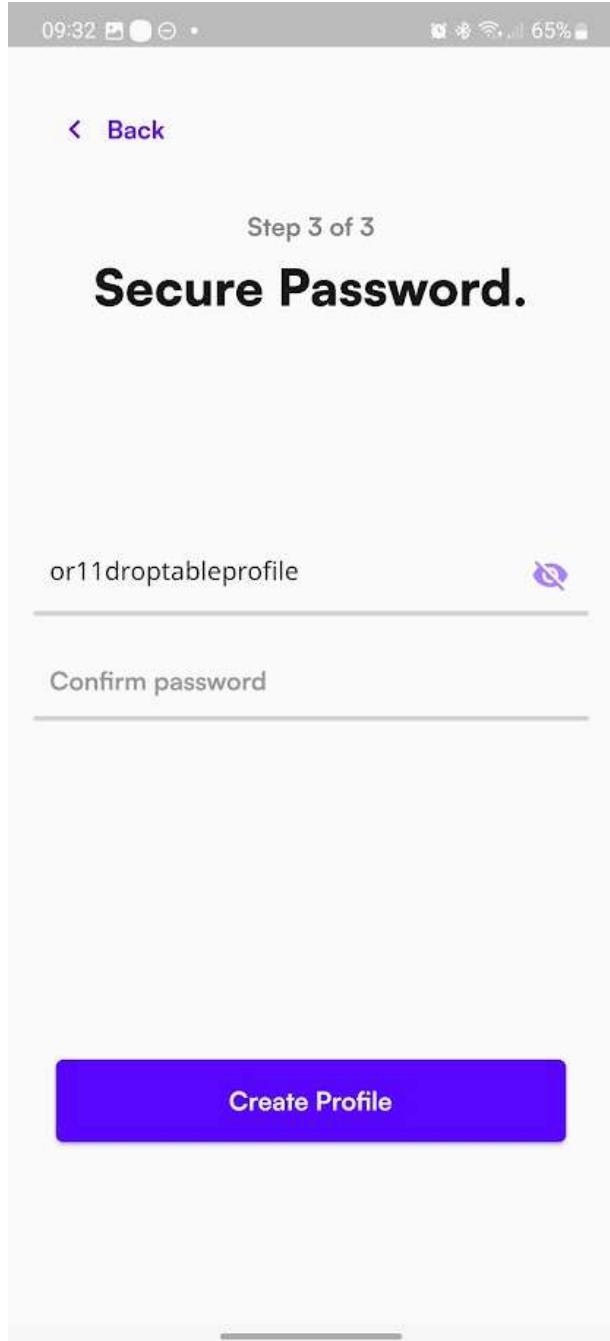
By using an input formatter of only letters and numbers, it blocks the basic SQL injection statements.

```
inputFormatters: [
    //Only allows the input of letters a-z and
    A-Z and 0-9 and @, .-$&!#?
    FilteringTextInputFormatter.allow(
        RegExp(r'[a-zA-Z0-9@#$&!#?]+')
    ),
]
```

When the user tries to type

' or 1=1; drop table profile

It only displays the characters and numbers



For those which bypass the app and just use the URL, it is protected

```
$firstname = mysqli_real_escape_string($link, $firstname);
$lastname = mysqli_real_escape_string($link, $lastname);
$email = mysqli_real_escape_string($link, $email);
$hashpassword = mysqli_real_escape_string($link,
$hashpassword);
```

Test No. 6

Similar to the email confirmation, the password confirmation just checks if it is equal to the password to check if the user has memorised the password and is able to type it without seeing it. The password confirmation field is never sent to the server

Test No. 7

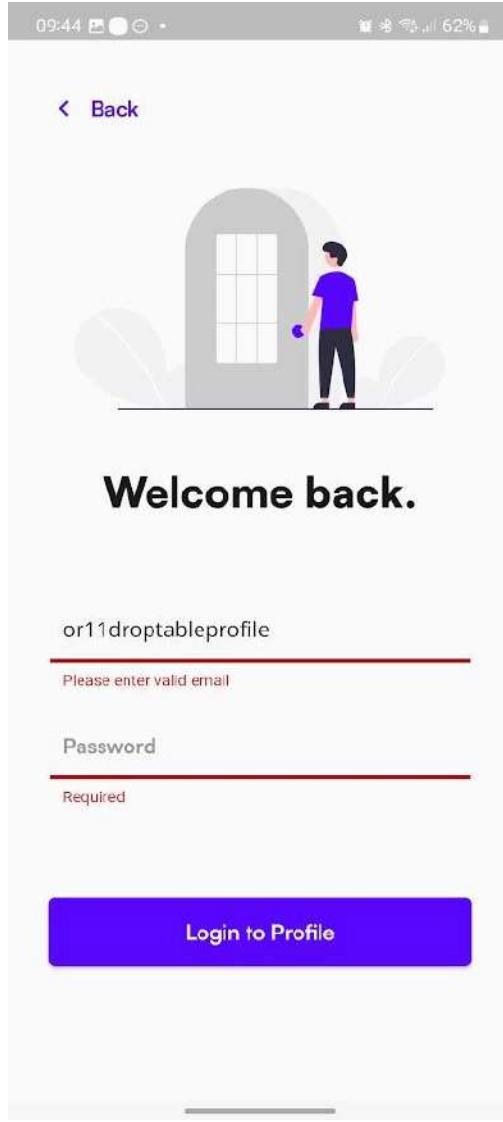
By using an input formatter of only letters, numbers and @ signs, it blocks the basic SQL injection statements.

```
inputFormatters: [
    //Only allows the input of letters a-z and A-Z and
    '@, . -'
    FilteringTextInputFormatter.allow(RegExp('[a-zA-Z0-
    9@,. -]'))
],
```

When the user tries to type

' or 1=1; drop table profile

It only displays the characters and numbers



Not only does the SQL inject code not work within the field, but it also is not a valid email

For those which bypass the app and just use the URL, it is not protected, without any validation. To fix this, I added the same preparation code.

```
$email = mysqli_real_escape_string($link, $_POST["email"]);
```

Test No. 8

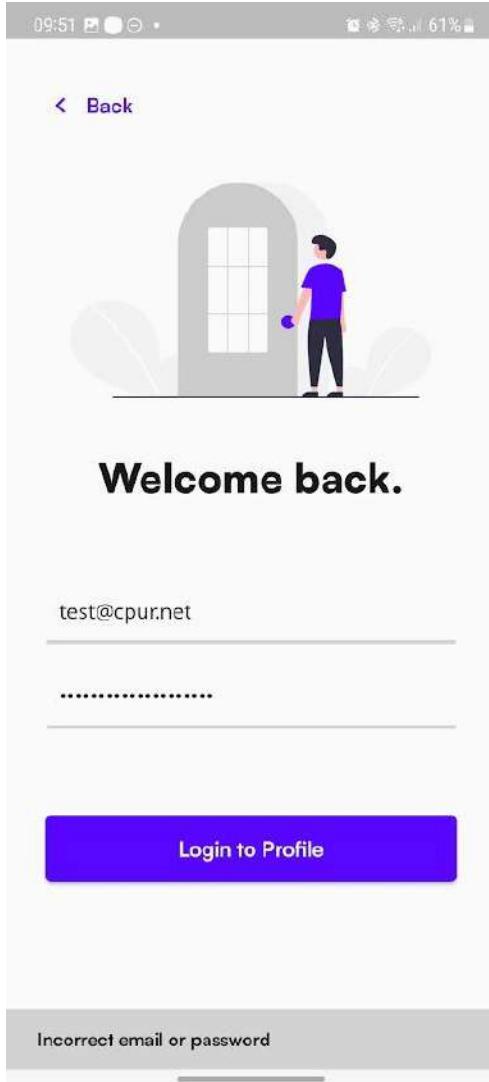
By using an input formatter of only letters and numbers, it blocks the basic SQL injection statements.

```
inputFormatters: [  
    //Only allows the input of letters a-z and  
    A-Z and 0-9 and @, .-$&!#?  
    FilteringTextInputFormatter.allow(  
        RegExp(r'[a-zA-Z0-9@#$&!#?]+'))  
],
```

When the user tries to type

' or 1=1; drop table profile

It only displays the characters and numbers



For those which bypass the app and just use the URL, it is not protected, without any validation. To fix this, I added the same preparation code.

```
$password = mysqli_real_escape_string($link, $_POST[ "password" ]);
```

Test No. 9

By using an input formatter of only letters and numbers and spaces and dashes, it blocks the basic SQL injection statements.

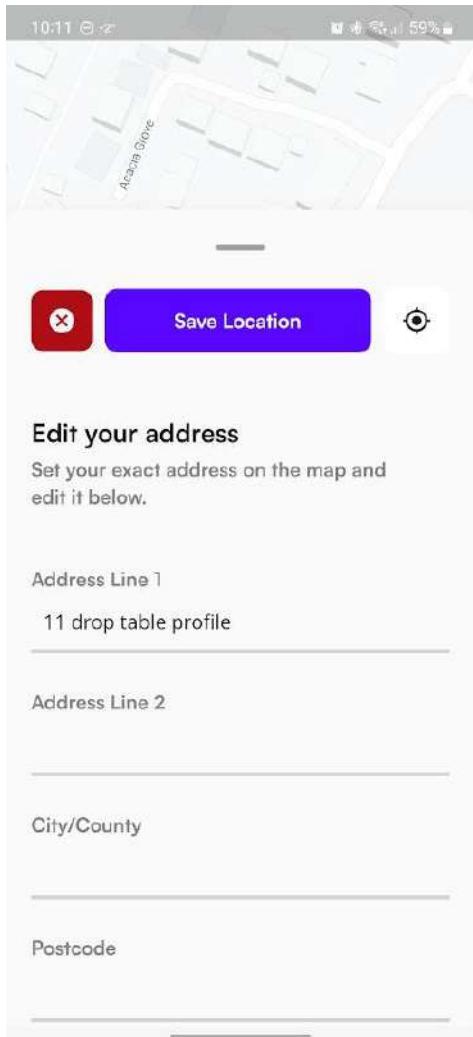
```
inputFormatters: [
```

```
//Only allows the input of  
letters a-z and A-Z and , - and spaces  
  
FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9,. -]+|\s'))  
],
```

When the user tries to type

' or 1=1; drop table profile

It only displays the letters and numbers



For those which bypass the app and just use the URL, it is not protected, without any validation.
To fix this, I prepared the statement for the orders.php file

```
$address = mysqli_real_escape_string($link, $_POST[ "address"]);
```

Test No. 10

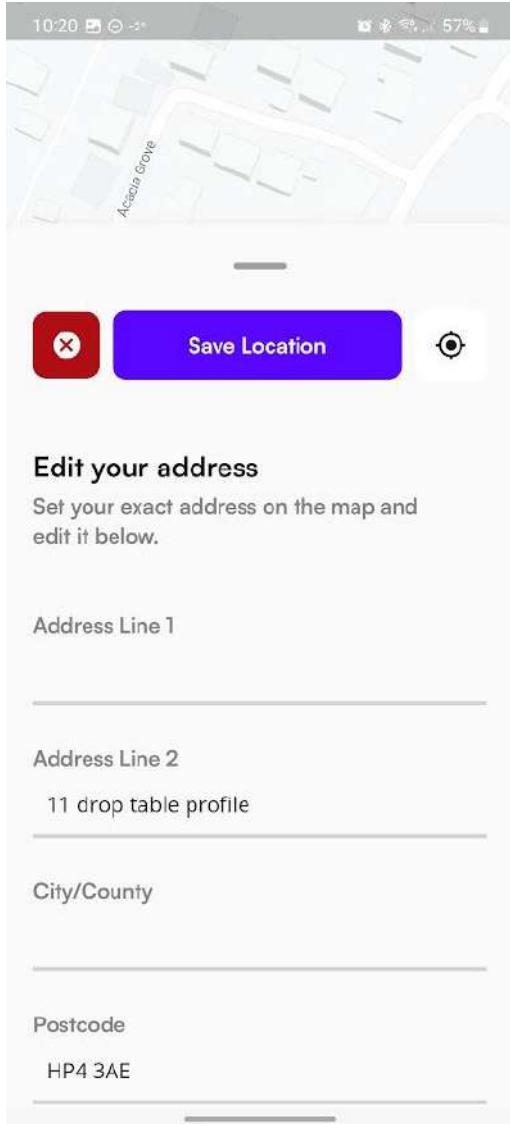
By using an input formatter of only letters and numbers and spaces and dashes, it blocks the basic SQL injection statements.

```
inputFormatters: [
    //Only allows the input of
    letters a-z and A-Z and , - and spaces
    FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9,. -]+|\s'))
],
```

When the user tries to type

' or 1=1; drop table profile

It only displays the letters and numbers



For those which bypass the app and just use the URL, it is now protected since in test no. 9, it was found that it was missing validation on the server. To fix this, I prepared the statement for the orders.php file

```
$address = mysqli_real_escape_string($link, $_POST["address"]);
```

Test No. 11

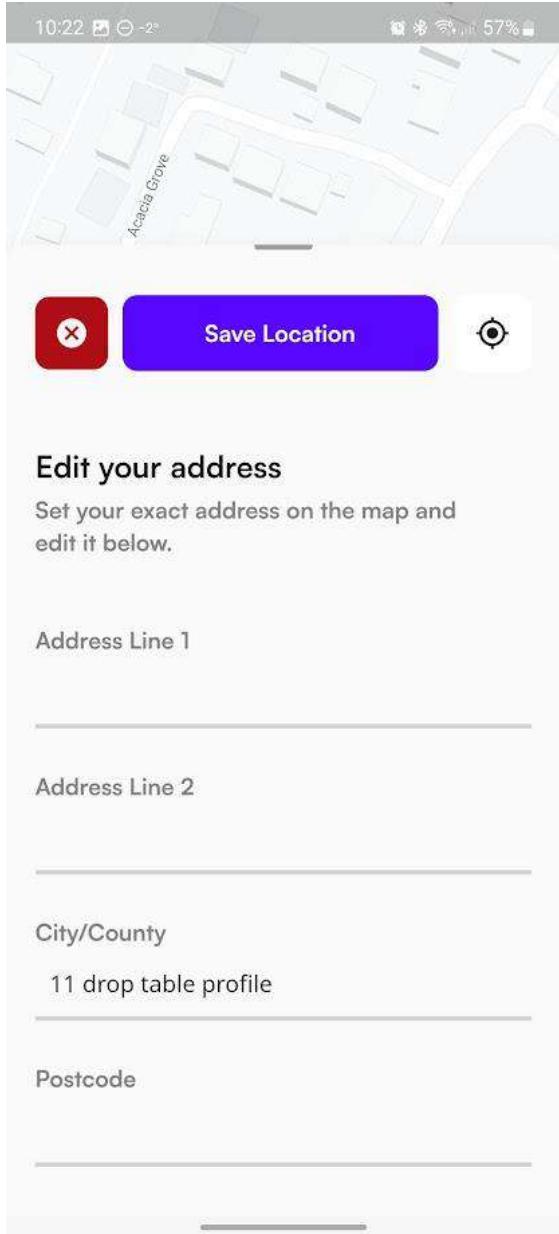
By using an input formatter of only letters and numbers and spaces and dashes, it blocks the basic SQL injection statements.

```
inputFormatters: [  
    //Only allows the input of  
    letters a-z and A-Z and , - and spaces  
  
    FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9,. -]+|\s'))  
],
```

When the user tries to type

' or 1=1; drop table profile

It only displays the letters and numbers



For those which bypass the app and just use the URL, it is now protected since in test no. 9, it was found that it was missing validation on the server. To fix this, I prepared the statement for the orders.php file

```
$address = mysqli_real_escape_string($link, $_POST["address"]);
```

Test No. 12

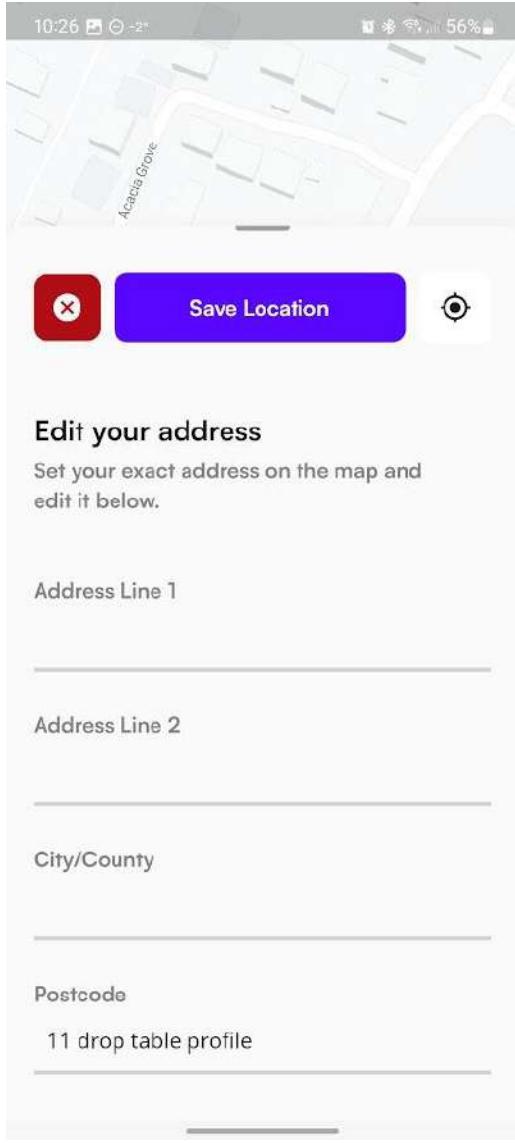
By using an input formatter of only letters and numbers and spaces and dashes, it blocks the basic SQL injection statements.

```
inputFormatters: [
    //Only allows the input of
    letters a-z and A-Z and , - and spaces
    FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9,. -]+|\s'))
],
```

When the user tries to type

' or 1=1; drop table profile

It only displays the letters and numbers

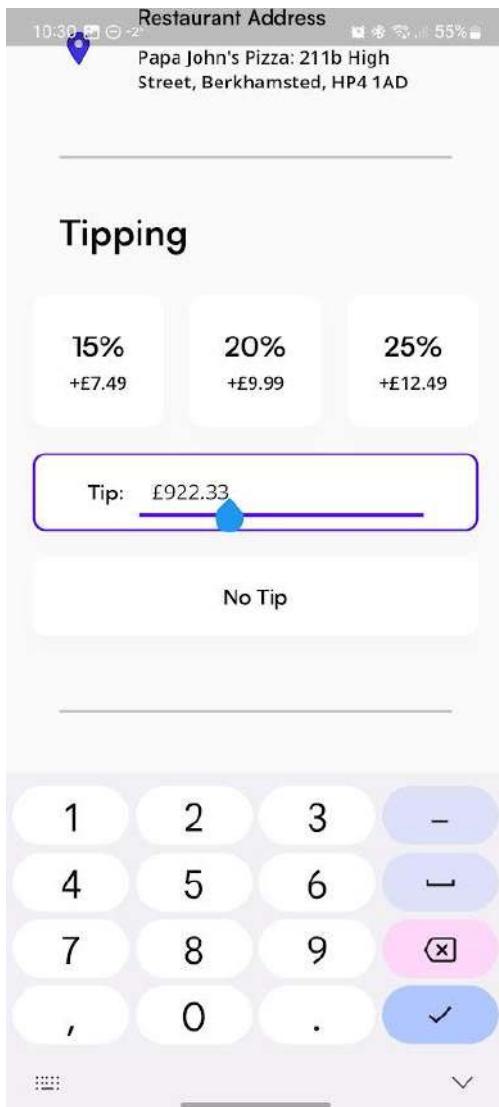


For those which bypass the app and just use the URL, it is now protected since in test no. 9, it was found that it was missing validation on the server. To fix this, I prepared the statement for the orders.php file

```
$address = mysqli_real_escape_string($link, $_POST["address"]);
```

Test No. 13

Since the tip amount uses a numerical keyboard, it is not possible to input letters. As well, if the user inputs anything other than numbers, it clears the tip field to null.



For those which bypass the app and just use the URL, it is not protected, without any validation.
To fix this, I prepared the statement for the orders.php file

```
$tip = round((floatval(mysqli_real_escape_string($link, $_POST["tip"]))),  
2); //Set the tip to 2dp
```

Test No. 14

Passwords stored on the server are hashed which means that cant be read

Password

```
$2y$10$0wHGKYOAXDLrjA/B5B1Euw7pKe8RLC16UOAynH3SG2...
$2y$10$3uELnq22vYzeID/tGTqNVO4KrFPfGMtOva1mQunFbNG...
$2y$10$0qHSiKMsCEGhA8htWVjazVunWxniolRDZMld8haHZad...
$2y$10$COERARg3L5eqD0SqmOTFHuOvM/J71lZBIX2dMI9htj...
$2y$10$7uCjJt3QLTIQ1PM45.eiveuib9343cu7hGC0Ox219zz...
$2y$10$vOWQ73A5.7fleUmp0Ey67eADN/xgET1ps0W28L5tb0Y...
$2y$10$snHBwLakPVoj8ZB9zLLoper1sjnqKndSyQ/Pl/yh/kb...
$2y$10$KR0IURHRNLnw.HuypGyu.lSQy3chEng8dHaN07avV...
$2y$10$zSwosE.LUY4VK7Gf7i9pHetdi1JlUf6KOpwujbzdl8...
$2y$10$1x3mG0dw/L8gt39/ED180ePxKAbeXPMXY/NH4wXtlzj...
$2y$10$hvyPW5JinXeAVI1DzyGW06cqjcL1RdQ14thEpZpf1...
$2y$10$vMyYM21Nmcoe74rEnSLmDuHb21A.0DD87/KUiBBEuGp...
$2y$10$shPg2a1rNPmOKEnJbFmQM.2lIS6pvqazJVZbaqswR6Q...
$2v$10$kv4Px2O1Otxuwo6CzDD14NvkvSRO63lBu39fnCXdl...
```

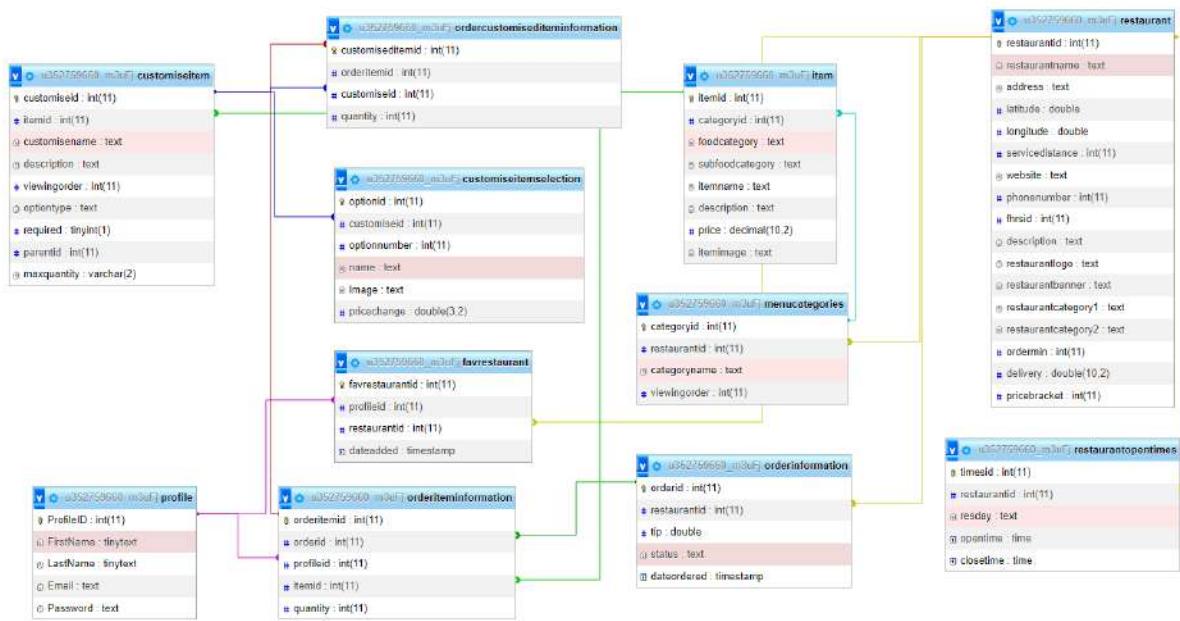
Test No. 15

Passwords on the client are encrypted

```
[{"id": 1, "profileId": 136, "firstname": "CPU", "lastname": "Admin", "email": "admin@cpu.net", "password": "1HQPO08516paAE8nkVi5/kLss+gcFUo/Ss3p07knSf5X21XGHUUFLnxF4/Hsp5naestu/XCJIMkZogc0pkVoQ--"}, ...]
```

Test No. 16

The database contains no duplicate data, using foreign keys to reference the tables



Test No. 17

The app uses foreign keys, and is in third-normal form

```
await database.execute("""
CREATE TABLE localprofiles(
    id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    profileid INT,
    firstname TEXT,
    lastname TEXT,
    email TEXT,
    password TEXT,
    selected TEXT,
    profilecolorred INT,
    profilecolorgreen INT,
    profilecolorblue INT,
    createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
)
""");
```

```

await database.execute("""
CREATE TABLE cartItems(
    cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    profileid INTEGER,
    itemid INT,
    customised TEXT,
    quantity INT,
    FOREIGN KEY (profileid) REFERENCES localprofiles(profileid)
)
""");

```

Test No. 18

This is a key function but it hasn't been implemented. Since the profile selection section uses a horizontal slide, in order to remove a profile, it is best to have a long press to remove. This should then make a popup to confirm removing a profile since there is a chance that a person may misclick and accidentally hold on a profile for too long instead of just quickly tapping.

```

onLongPress: () {
    //Remove profile from device
    showDialog(
        context: context,
        builder: (BuildContext context)
    => AlertDialog(
        backgroundColor:
            Theme.of(context).backgroundColor,
        title: Text(
            'Remove Profile',
            style:
                Theme.of(context)
                    .textTheme

```

```
.headline5

?.copyWith(color:
Theme.of(context).textTheme.headline1?.color),
),

content: Text(
'Are you sure you want
to remove this profile from the device',
style:
Theme.of(context).textTheme.bodyText2,
),

actions: <Widget>[
TextButton(
//Cancel button to
close the popup
onPressed: () =>
Navigator.pop(context, 'Cancel'),
child: const
Text('Cancel'),
),

TextButton(
//On confirm button
press delete profile from device and if there is no remaining profiles, go to
welcome screen
onPressed: () async {
SQLiteLocalProfiles.deleteProfile(profileInfo[index]["id"]);
}
)
]
```

```
Navigator.pop(context);

    var
availableProfiles = await SQLiteLocalProfiles.getDisplayProfilesList();

        if
(availableProfiles.isEmpty) {

Navigator.pop(context);

Navigator.of(context).push(
MaterialPageRoute(builder: (_) => const ProfileSetupWelcome()),

);
} else {
    //If there are
profiles remaining get the first profile in the database and set the selected
profile to the first profile

    var
newSelectedProfile = await SQLiteLocalProfiles.getFirstProfile();

        bool
checkSelected = await SetSelected.selectProfile(
newSelectedProfile[0]["profileid"],

newSelectedProfile[0]["firstname"],

newSelectedProfile[0]["lastname"],
```

```
newSelectedProfile[0]["email"]);

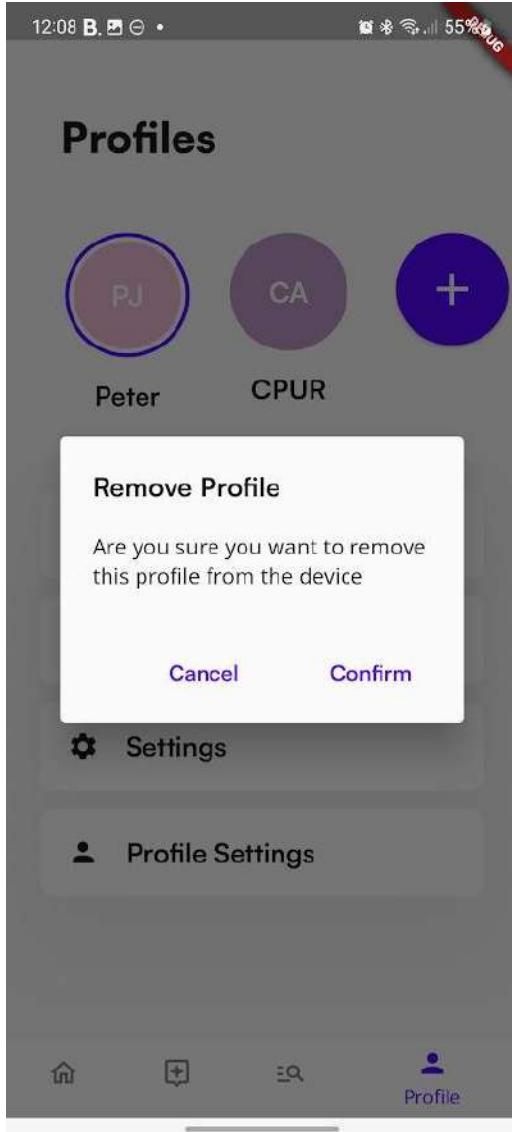
        if (checkSelected
== true) {

            setState(() {
});

        }

    }

    child: const
Text('Confirm'),
),
]);
},
```



Test No. 19

With the addition of the profile removal in test no. 18 , this test can be completed. When an app is first opened, it is verified using the setupverification.dart file

```
import 'package:alleat/screens/profilesetup/welcomeScreen.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/widgets/genericlocating.dart';
```

```
import 'package:alleat/widgets/navbar.dart';

import 'package:flutter/material.dart';

class SetupWrapper extends StatefulWidget {

  const SetupWrapper({Key? key}) : super(key: key);

  @override

  State<SetupWrapper> createState() => _SetupWrapperState();
}

class _SetupWrapperState extends State<SetupWrapper> {

  //Default setup not complete if something wrong happens

  bool setup = false;

  Future<bool> isSetupComplete() async {

    List<Map> profileInfo = await SQLiteLocalProfiles

      .getFirstProfile(); //Call Database for the first entry

    if (profileInfo.isEmpty) {

      //If the first entry is empty

      //Then setup is not complete (pass to build)

      return false;
    } else {

```

```
//If the first entry exists

//Setup is complete (pass to build)

return true;

}

}

@Override

Widget build(BuildContext context) => FutureBuilder<bool>(

  future: isSetupComplete(),

  builder: (context, snapshot) {

    if (!snapshot.hasData) {

      //While loading, go to loading page

      return const GenericLoading();

    }

    if (snapshot.hasError) {

      //If the app has an error, go to error page

      return const GenericLoading();

    } else {

      bool setup = snapshot.data ?? [] as bool; //Get data from Future

      if (setup == false) {

        //If setup is not complete

        return const ProfileSetupWelcome(); //Go to Setup page

      }

    }

  }

);
```

```
    } else if (setup == true) {

        //If setup is complete

        return const Navigation(); //Go to main page

    } else {

        //If there is an error

        return const GenericLoading(); // Go to error page

    }

}

},

);

}
```

When a profile is deleted, it is checked after and if there are no remaining profiles on the device, the user is brought to the welcome screen.

```
if (availableProfiles.isEmpty) {

}

Navigator.pop(context);

Navigator.pop(context);

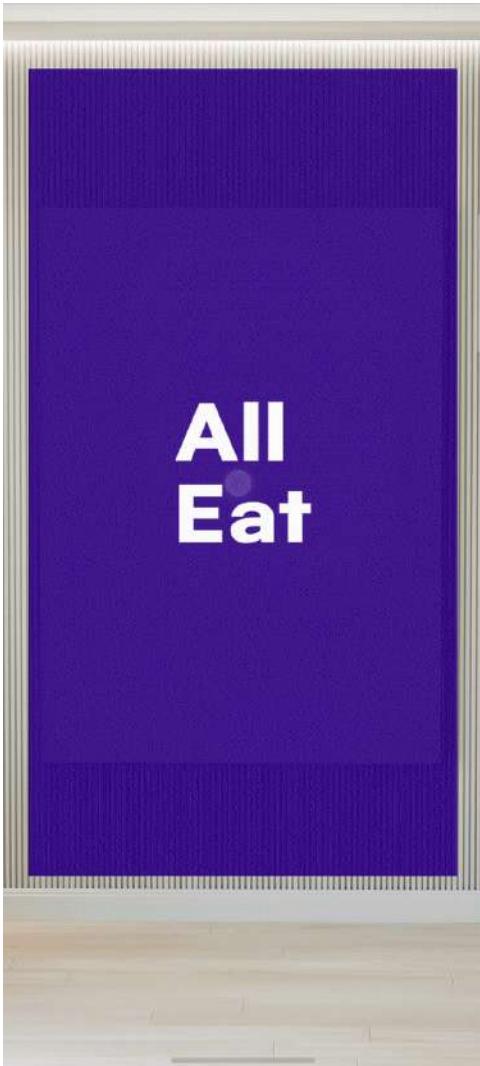
Navigator.of(context).push(

MaterialPageRoute(builder: (_) => const ProfileSetupWelcome()),
```

```
);
```

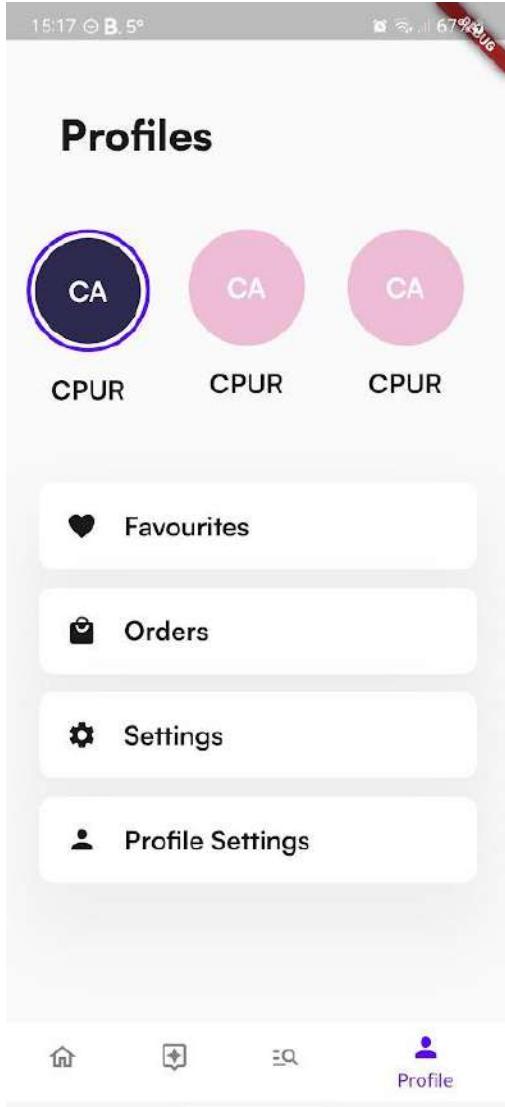
```
}
```

Test No. 20



Test No. 21

When the user clicks on the create profile or login button more than once, it is added multiple times.



To fix this, I will make it check if there is a profile already in the app with the same profile ID.

For the login, I added the code:

```
if (recievedServerData["message"]["exists"] == true) {  
    //If the profile is correct and exists  
    List importedProfile = recievedServerData["message"]["profile"];  
  
    var allProfiles =  
  
        await SQLiteLocalProfiles.getDisplayProfilesList(); //Get list of  
profiles currently saved and if it is already on the device do nothing
```

```
bool inLocalDatabase = false;

for (int i = 0; i < allProfiles.length; i++) {

    if (allProfiles[i]["profileid"] == importedProfile[0]) {

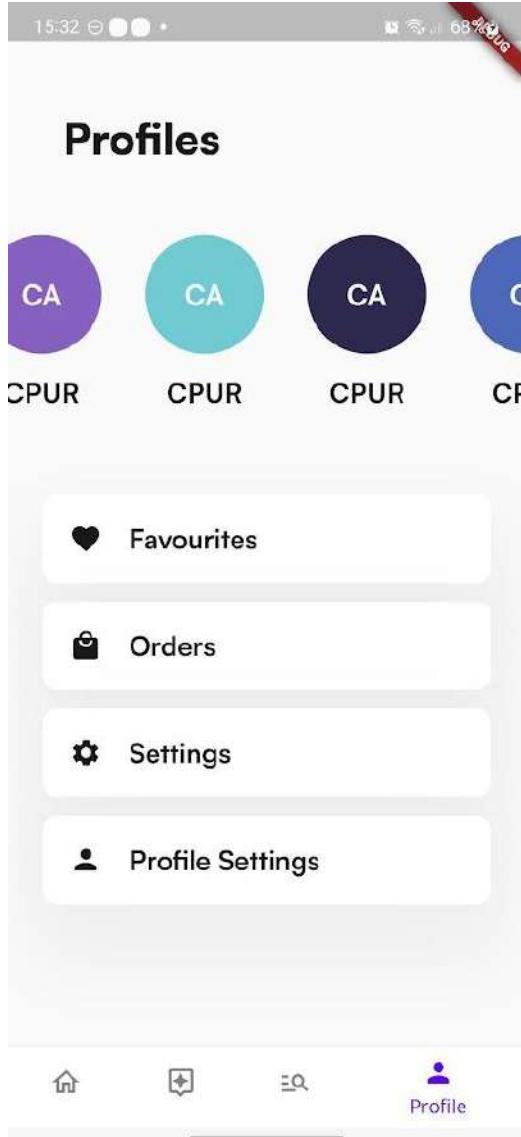
        inLocalDatabase = true;

    }

}

if (inLocalDatabase == false) {
```

This did not work.



After checking by logging in to the same account multiple times, it seems that it doesn't do anything.

After checking the code, it was found that checking if there was duplicates was already implemented but never worked

```
Future<void> _checkDuplicate() async {  
  
    List profileList = await SQLiteLocalProfiles.getProfiles();  
  
    bool alreadyLogged = false;  
  
    for (int i = 0; i < (profileList.length - 1); i++) {
```

```

    if (profileList[i]["email"].contains(email.text)) {

        alreadyLogged = true;

    }

}

if (alreadyLogged == false) {

    _loginUser();

} else {

    setState(() {

        ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content:
Text("Profile is already logged in.")));

    });

}

```

The problem is that it checks for if there are duplicate profiles but when it checks, the other profiles are also processing so when the button is pressed multiple times, it adds them multiple times.

```

I/flutter (20563): []
I/ViewRootImpl@ee11d6b[MainActivity](20563): ViewPostIme pointer 0
I/ViewRootImpl@ee11d6b[MainActivity](20563): ViewPostIme pointer 1
I/flutter (20563): Instance of 'Future<List<Map<String, Object?>>>'
I/flutter (20563): []
I/ViewRootImpl@ee11d6b[MainActivity](20563): ViewPostIme pointer 0
I/ViewRootImpl@ee11d6b[MainActivity](20563): ViewPostIme pointer 1
I/flutter (20563): Instance of 'Future<List<Map<String, Object?>>>'
I/flutter (20563): []
I/ViewRootImpl@ee11d6b[MainActivity](20563): ViewPostIme pointer 0

```

I tested the duplicate checking when doing separate profile login sessions and it is able to successfully detect it.

To fix all these issues, I created a variable called disableButton. When the button is pressed, it disables the button and when the validation is finished processing, it enables it.

```
ElevatedButton(  
    //submit button  
    style: Theme.of(context).elevatedButtonTheme.style,  
    onPressed: () {  
        if (disableButton == false) {  
            setState(() {  
                disableButton = true;  
            });  
            if (_formKey.currentState!.validate()) {  
                _checkDuplicate();  
            }  
        }  
    },  
    child: const Text('Login to Profile'),  
,
```

```
setState(() {  
    disableButton = false;  
    ScaffoldMessenger.of(context).showSnackBar(  

```

```
const SnackBar(content: Text('Successfully logged in.')),  
);  
  
Navigator.push(  
    context, //Go to main area  
    MaterialPageRoute(builder: (context) => const Navigation()));  
});
```

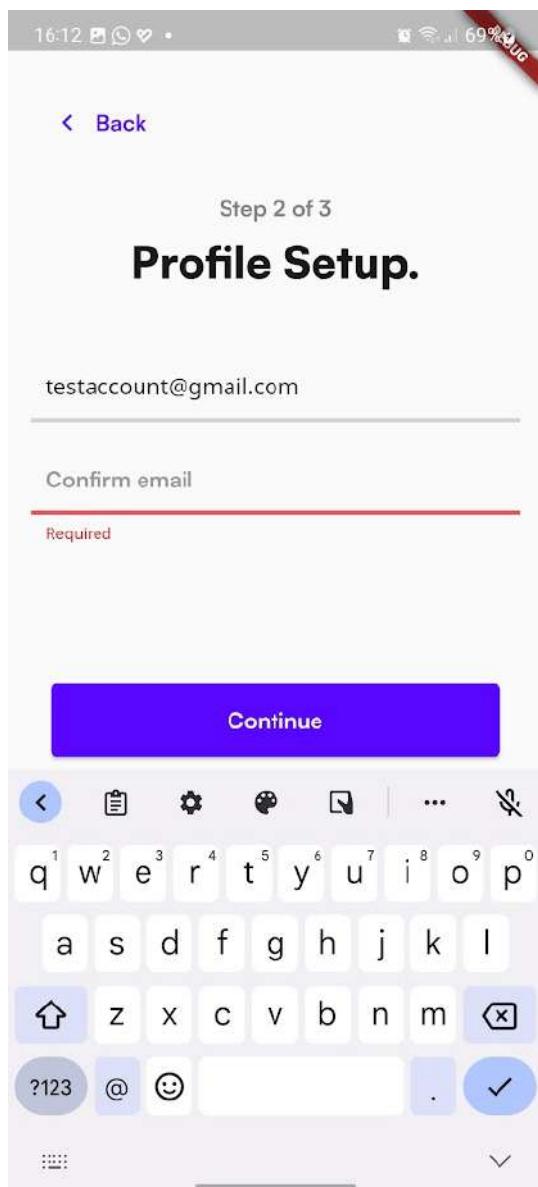
Test No. 22

The email is not valid if it doesn't contain an @ symbol



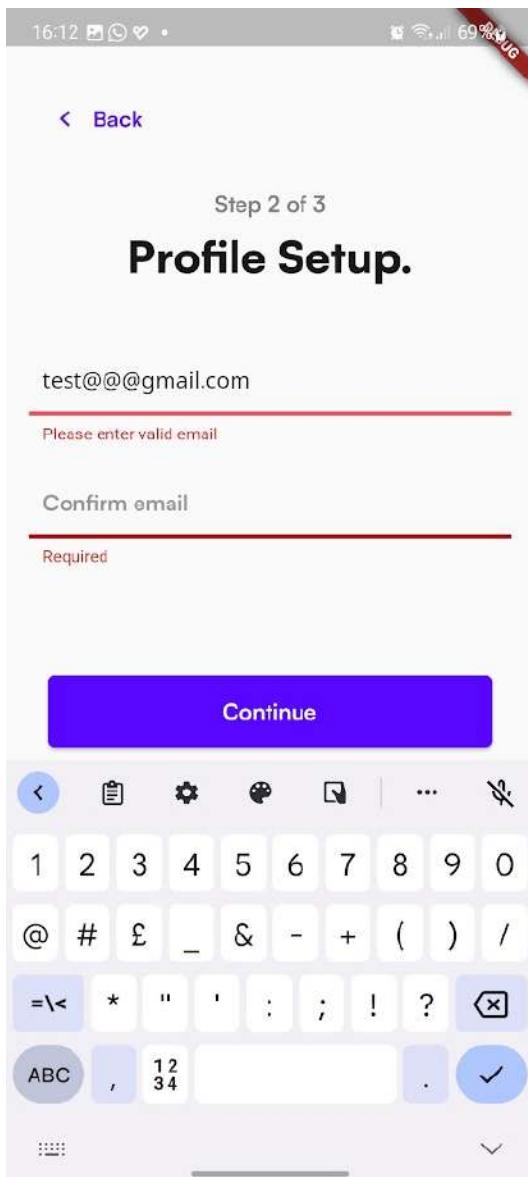
Test No. 23

It works when it is one @ symbol



Test No. 24

If there is more than one @ symbol, it is not valid



Test No. 25

The first name and last name are allowed to be 50 characters



Test No. 26

The first name and last name can be under 50 characters



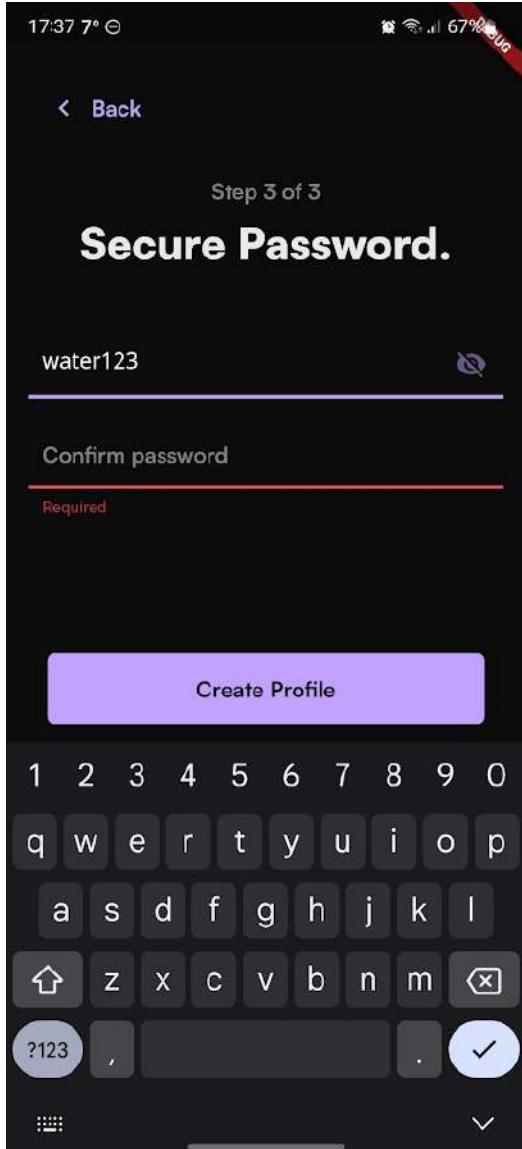
Test No. 27

The first name or last name can't be over 50 characters



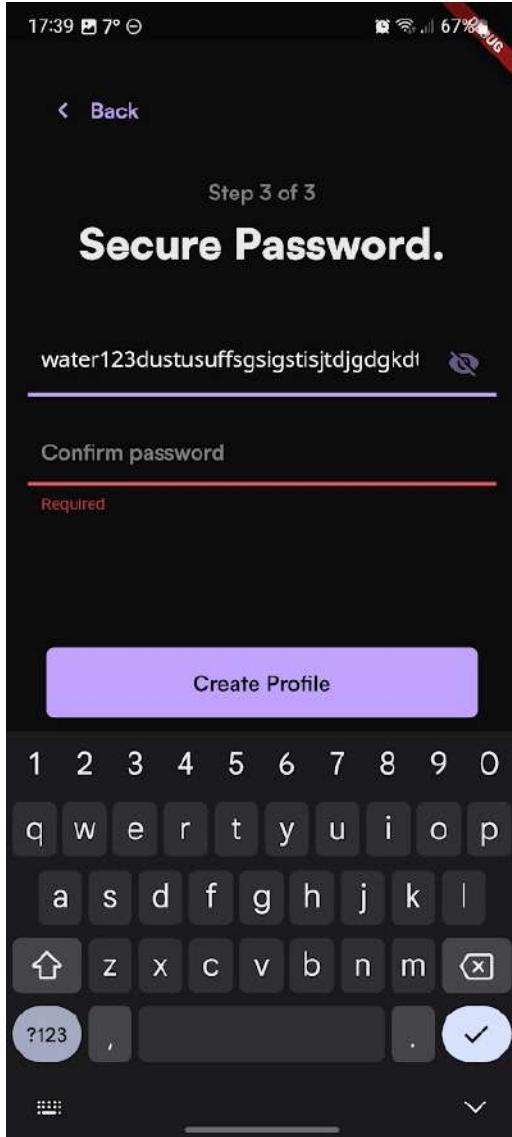
Test No. 28

It is allowed to be greater than 8 characters



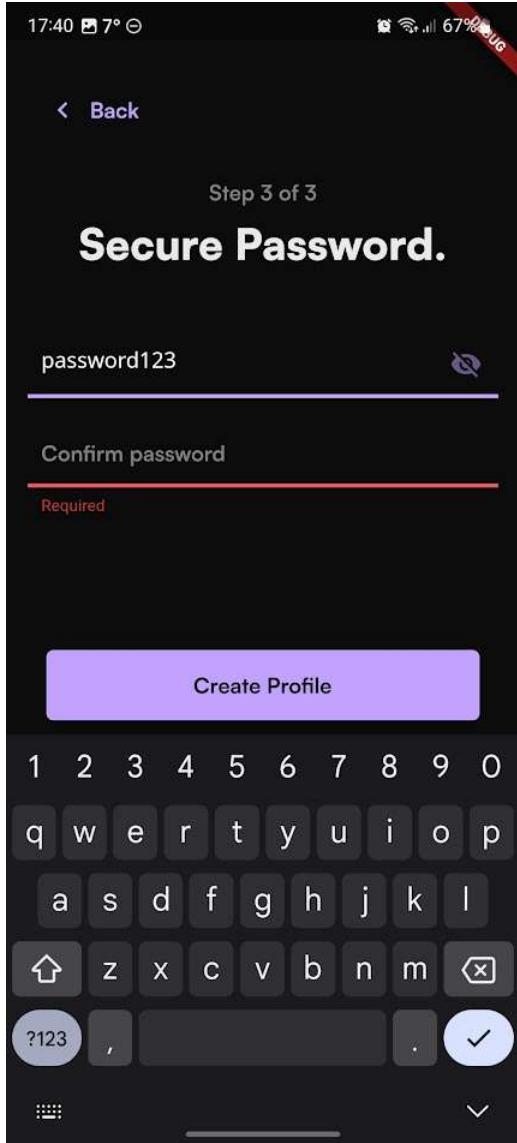
Test No. 29

It is allowed to be 99 characters



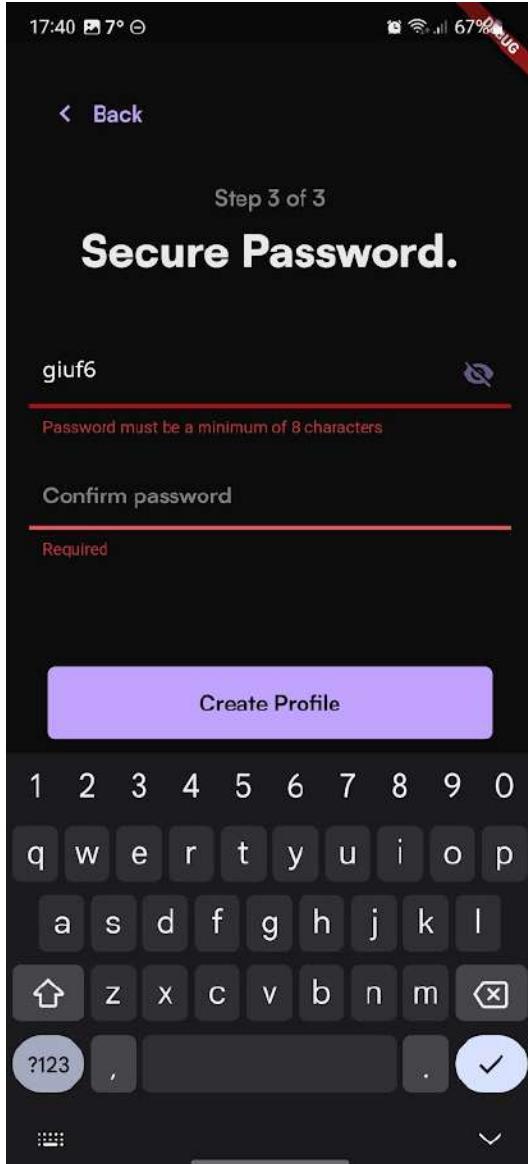
Test No. 30

It is allowed to fit between 8 and 99 characters



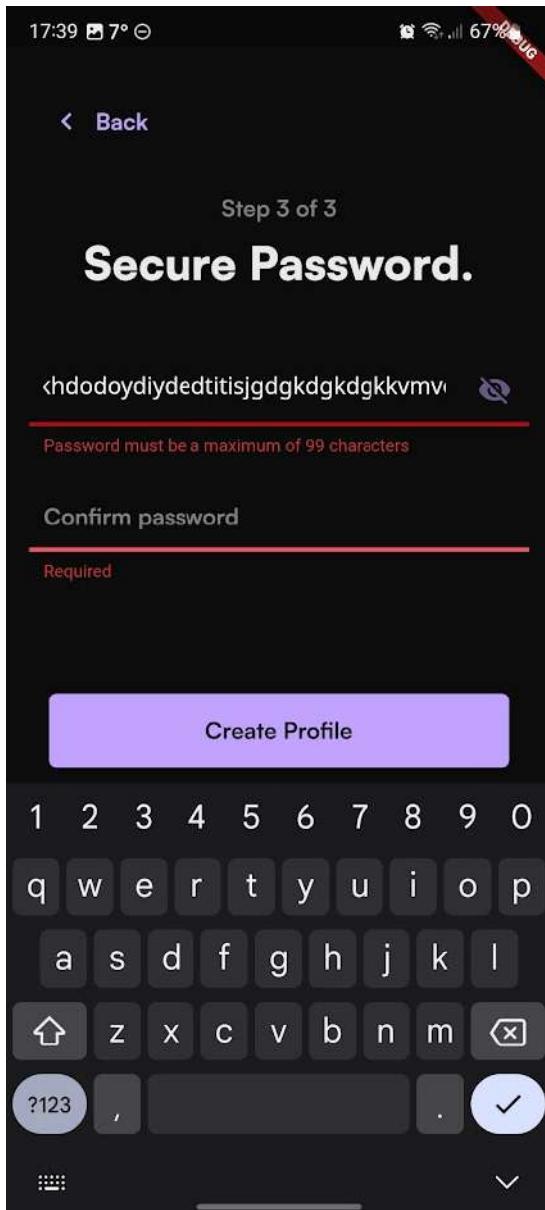
Test No. 31

It is not allowed if under 8 characters



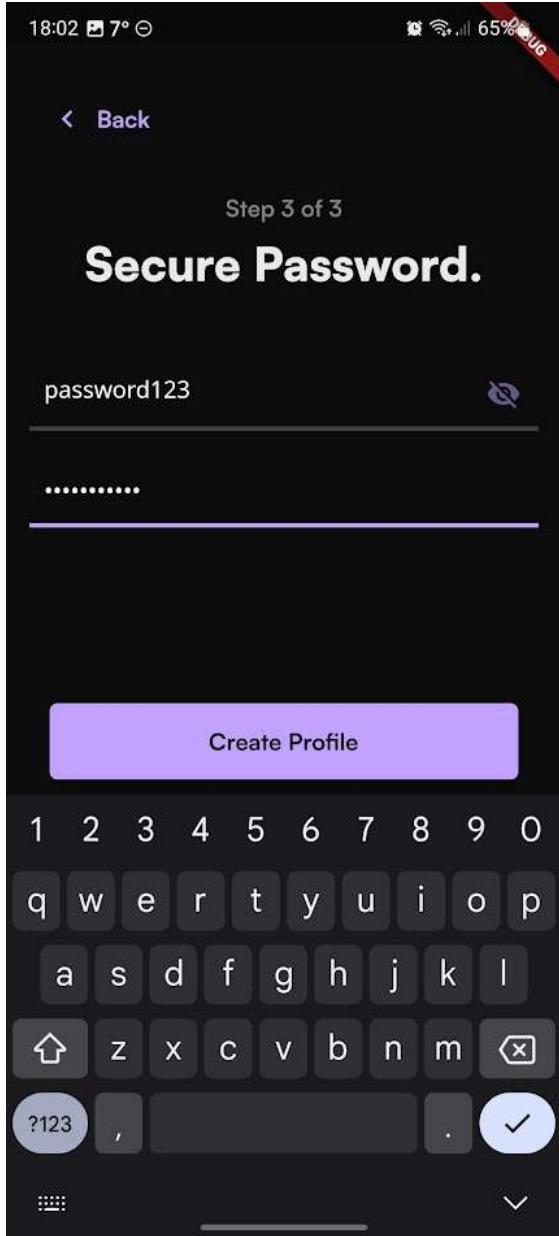
Test No. 32

Passwords are not allowed if greater than 99 characters



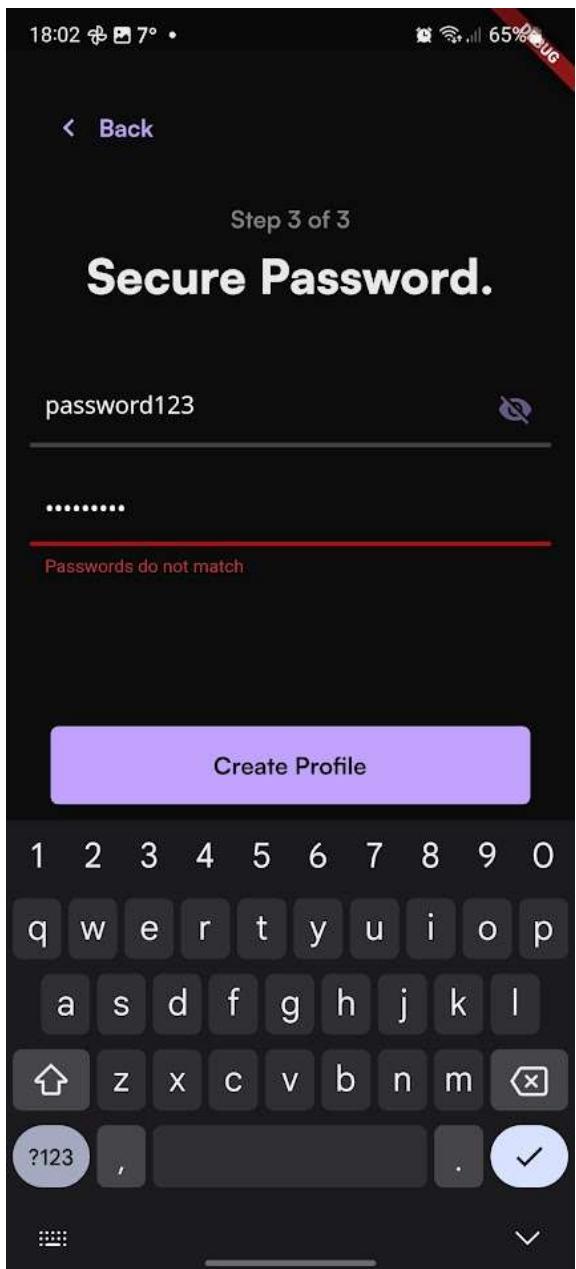
Test No. 33

When password123 is entered into the confirm password field, it is accepted



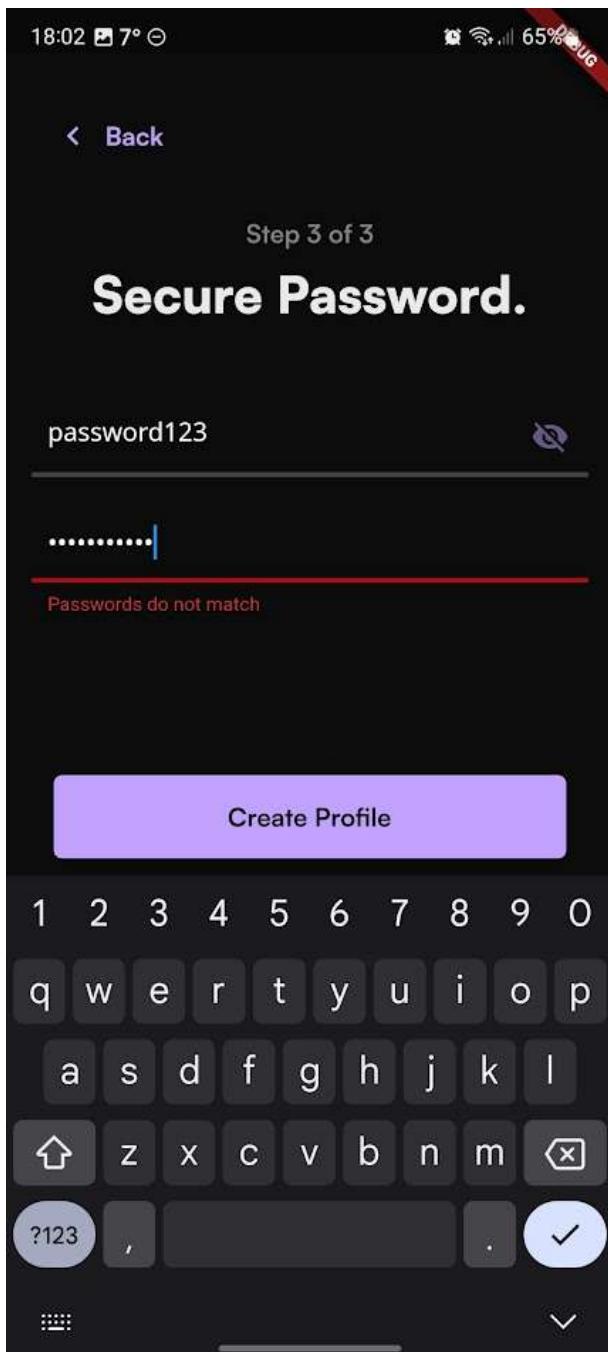
Test No. 34

When password is entered into the confirm password field it is not accepted



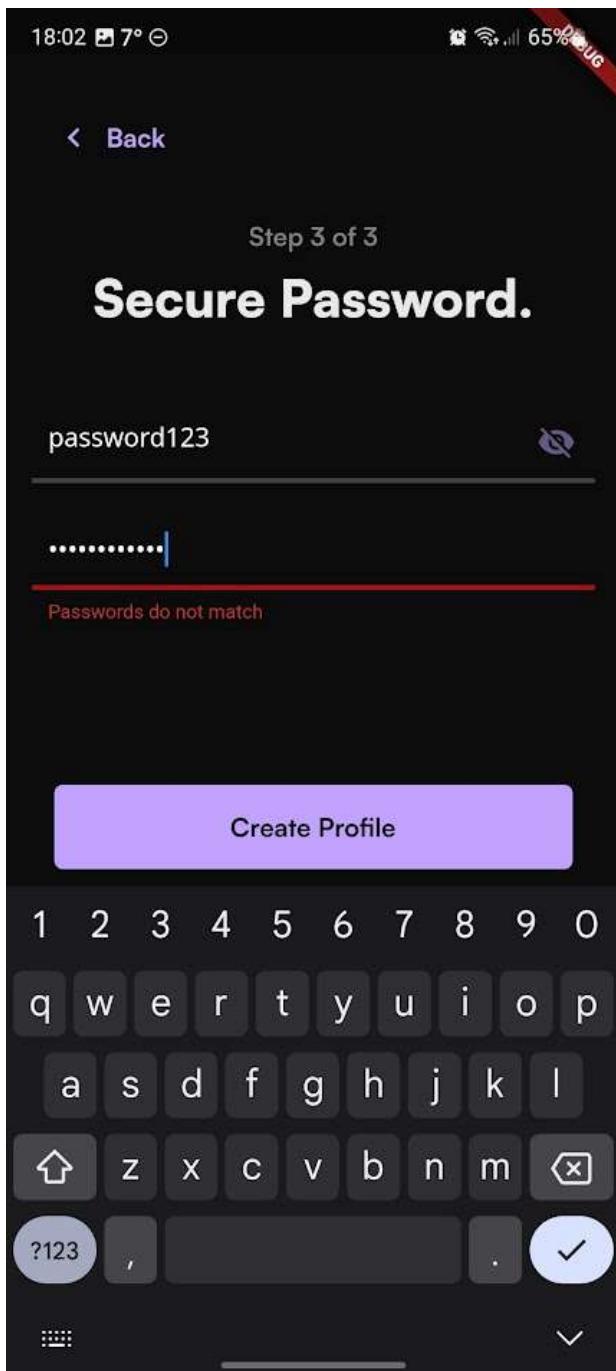
Test No. 35

When Password123 is entered it is displayed as invalid



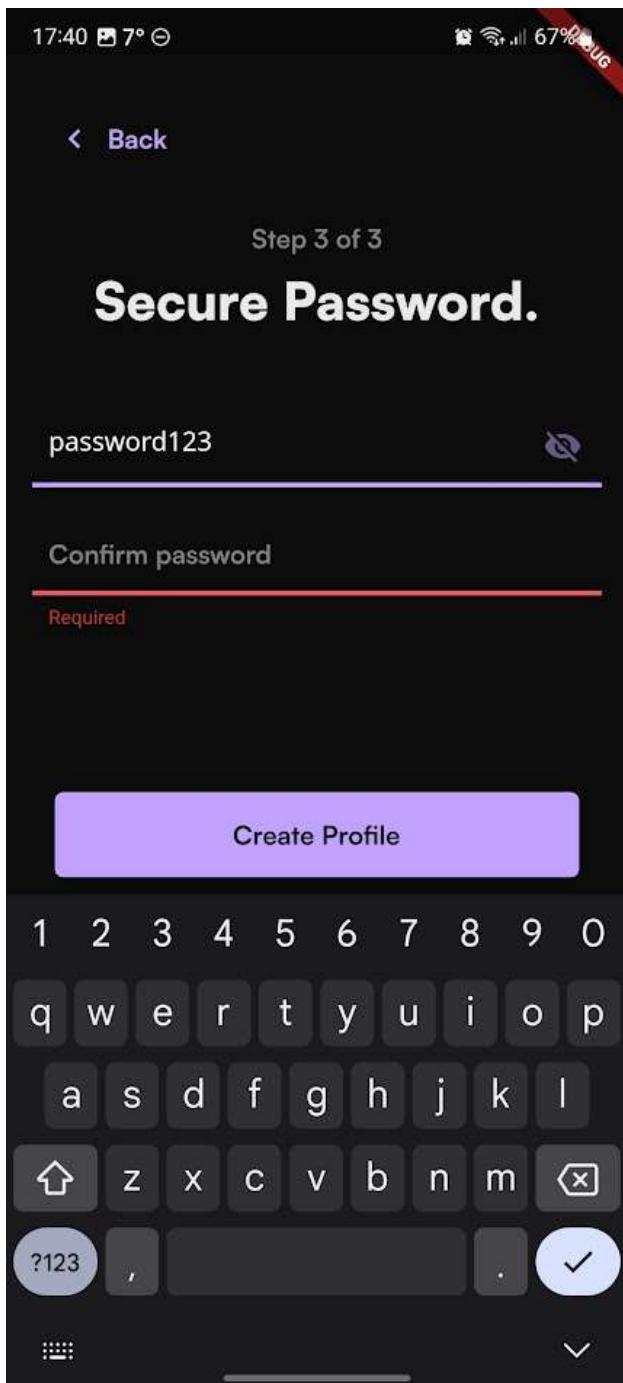
Test No. 36

When password1230 is entered into the confirm password field, it is invalid



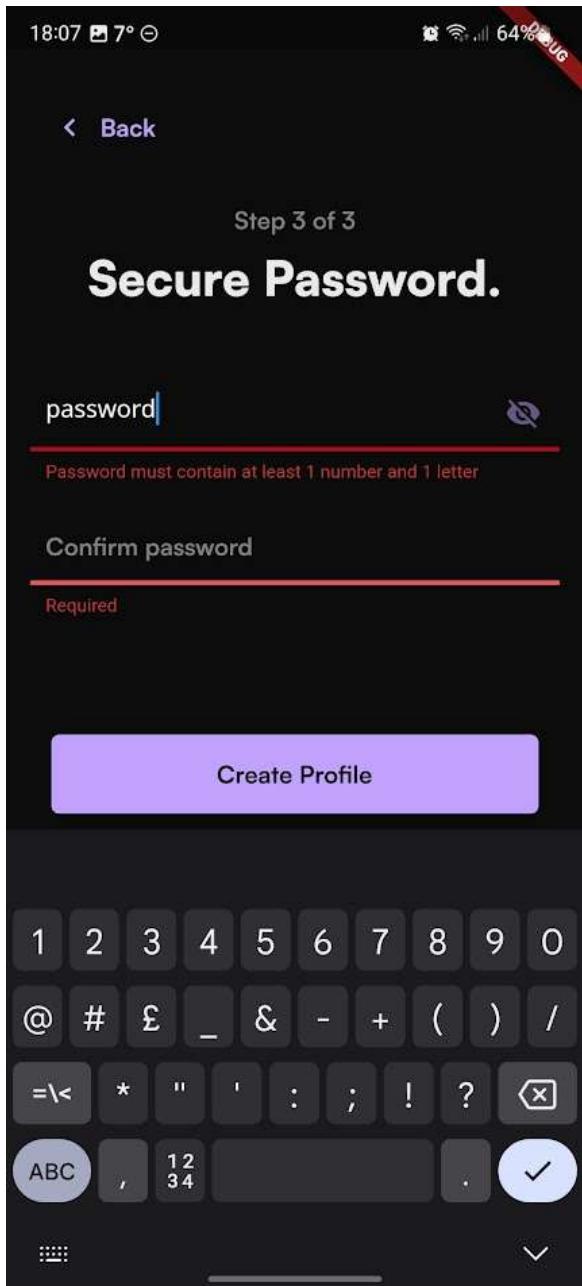
Test No. 37

If a password contains a letter and a number and is between 8 and 99 characters, it is accepted



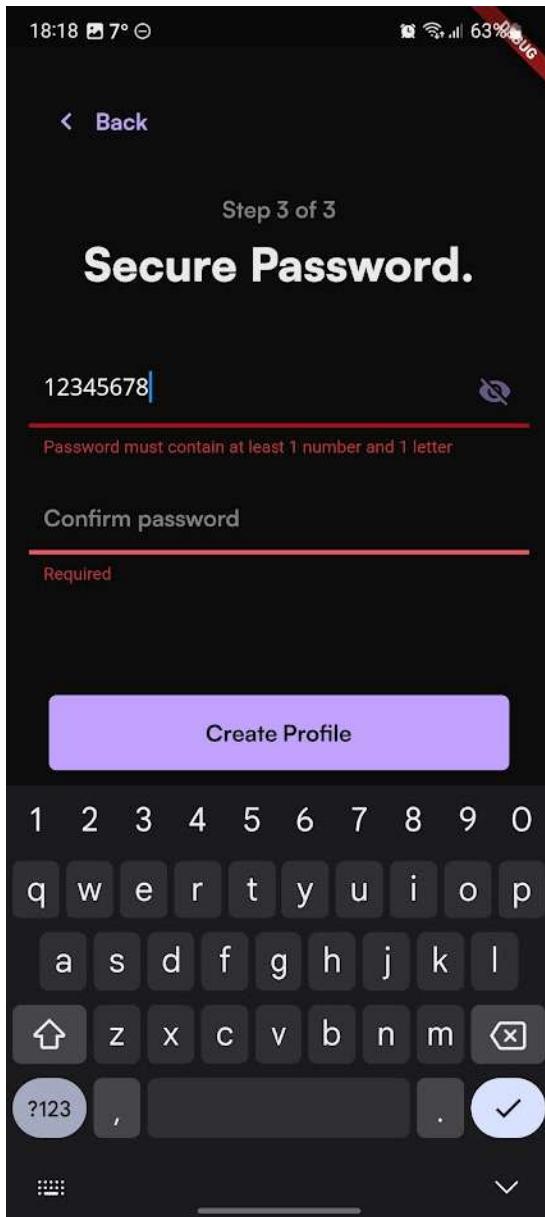
Test No. 38

If the password field is filled with only letters, it is shown as invalid



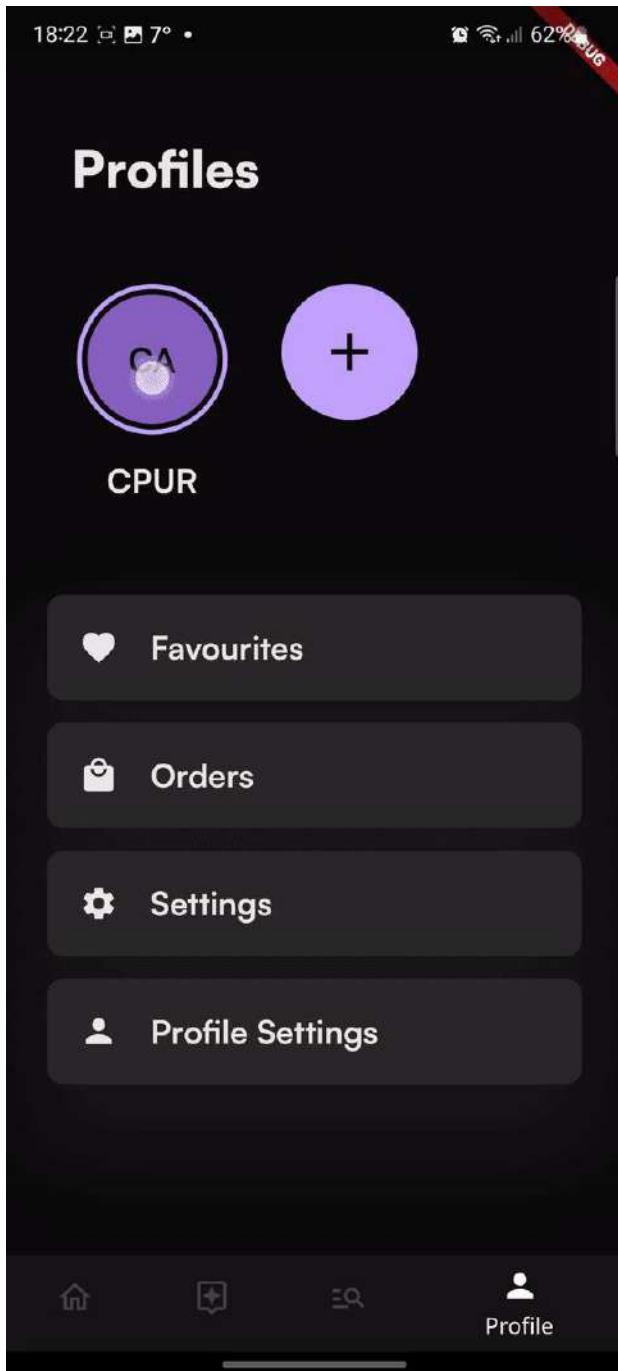
Test No. 39

If only numbers are entered into the password field, it shows as invalid



Test No. 40

The user is brought to the home screen (The user must long-press on the profile to remove it)



Test No. 41

By using the register.php file located on the server, it verifies the information and creates a unique profile

142 random

guy

randomguy@gmail.com

\$2y\$10\$KR0lUfRHRNLnw.HuypGyu.iSQy3chEng8dHaN07avV...

Test No. 42

By using an emulator and my personal phone, I was able to test multi-device uses.

Emulator	Personal
 A screenshot of a mobile application interface on an emulator. The top status bar shows the time as 6:56, signal strength, battery level at 57%, and a red 'DEBUG' ribbon. Below the status bar, there is a location icon, the word 'Location', and 'No Location Set'. A shopping cart icon is also present. The main area displays the text 'Welcome back' and 'CPUR Admin' in large, bold, black font. At the bottom, there is a navigation bar with icons for Home, Search, and Profile.	 A screenshot of the same mobile application interface on a personal device. The background is black. The top status bar shows the time as 18:59, signal strength, battery level at 57%, and a red 'DEBUG' ribbon. Below the status bar, there is a location icon, the word 'Location', and '63 Granville Rd'. A shopping cart icon is also present. The main area displays the text 'Welcome back' and 'CPUR Admin' in large, bold, white font. At the bottom, there is a navigation bar with icons for Home, Search, and Profile.

One bug I found was that due to it being run on Windows, it did not contain the correct file system meaning that when shared preferences was run, it would fail to save

I attempted to run the app on an Android 7 tablet but due to the age of the device, it was not able to sign in, with the following error occurring:

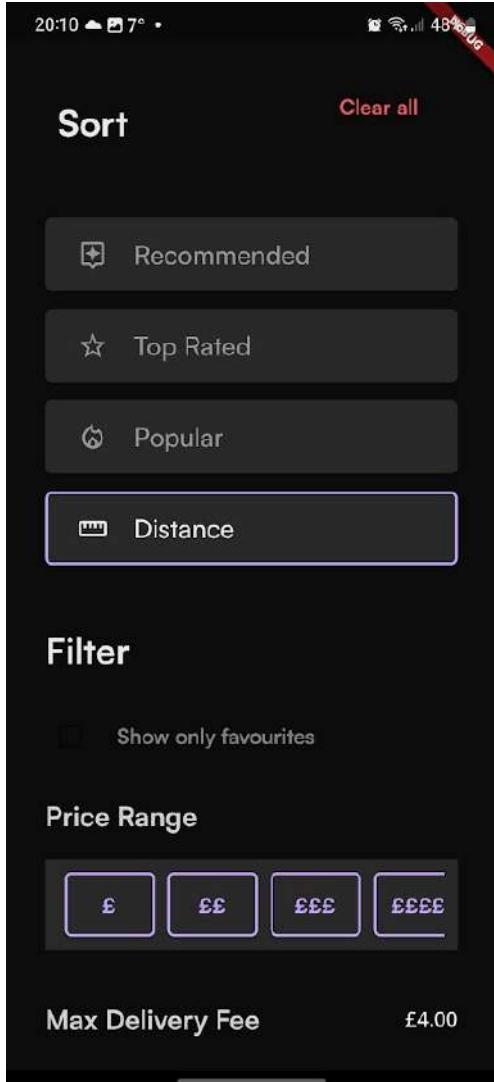
HandshakeException: Handshake error in client (OS Error: CERTIFICATE_VERIFY_FAILED: unable to get local issuer certificate(handshake.cc:359))

Without multiple devices, I am unable to complete the following tests:

- Test 42
- Test 43
- Test 44
- Test 45

Test No. 46

The default sort method is distance (it is the only sort method made)

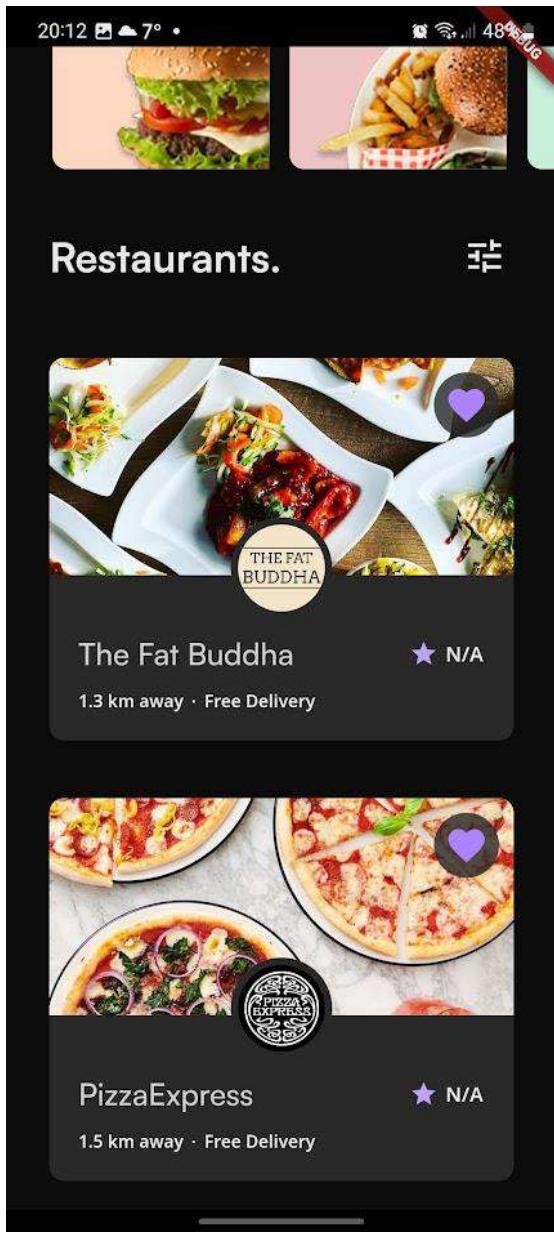


Test No. 47

Since the sort methods replace each other on click, it is not possible to have both selected at the same time.

Test No. 48

The user is able to filter by favourites



One thing found was that when the user goes to the filter and sort screen and saves the options, the navigation bar is removed. I looked at the code and found that it was going to the browse page. This means that it was opening a new instance of the main screen as just the browse page not the navigation. With the navigation, the screens are overlaid onto the main navigation bar. I switched the go to browse page to a pop which just removes the filter and sort page

```
ElevatedButton(  
    onPressed: () {  
        //Save button encodes the filter and sort dictionary and replaces  
        //the sharedpreference with the new value then closes the filter screen  
    },
```

```

        style: ElevatedButton.styleFrom(minimumSize: const
Size.fromHeight(50)),

        onPressed: () async {

            final prefs = await SharedPreferences.getInstance();

            String encodedCustomiseSelected =
json.encode(customiseSelected);

            await prefs.setString('filtersort', encodedCustomiseSelected);

            setState(() {

                Navigator.pop(context);

                Navigator.pop(context);

                Navigator.of(context).push(
                    MaterialPageRoute(builder: (_) => const Navigation()),

                );
            });

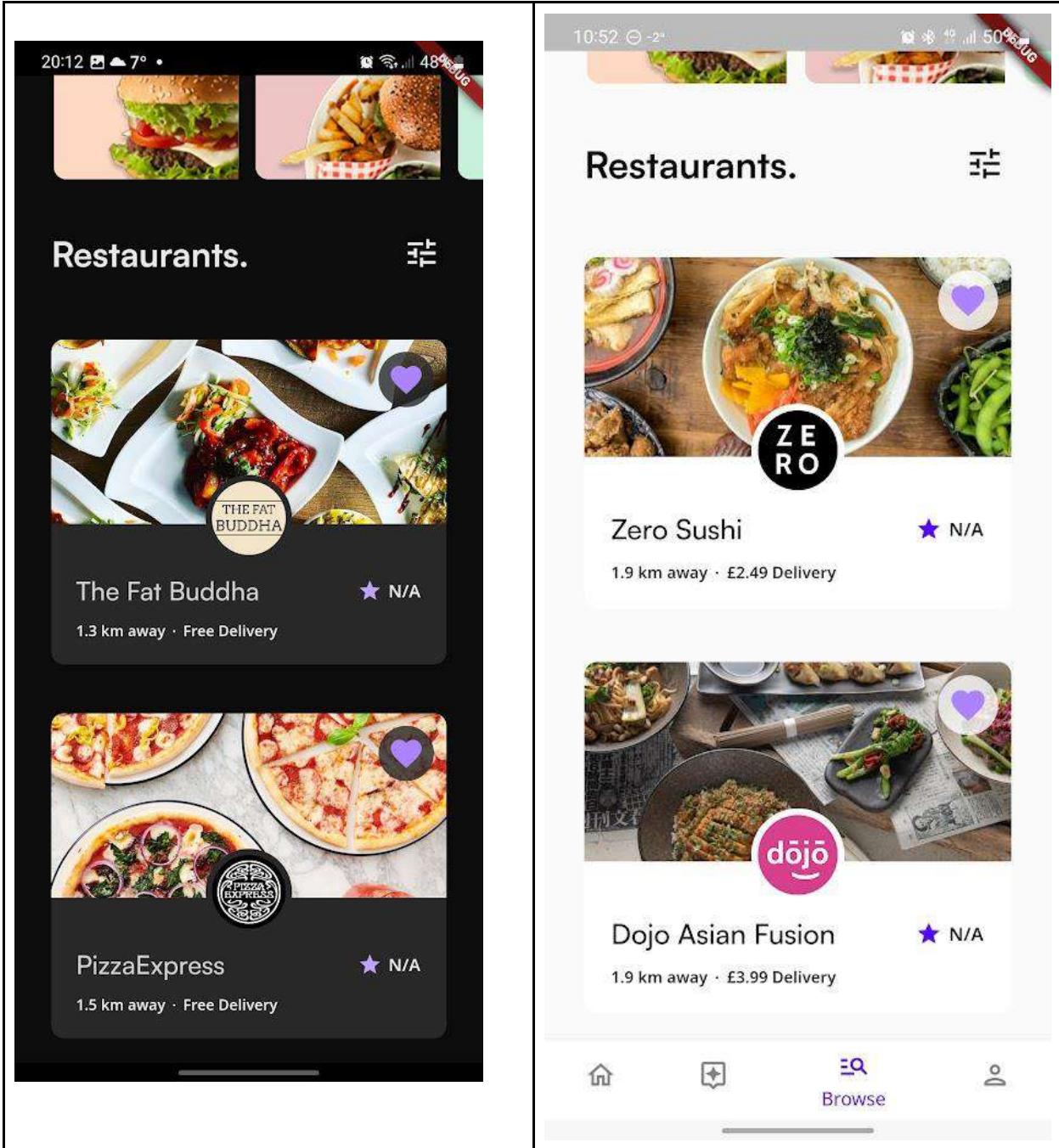
        },
        child: const Text("Save")
    )));

```

Test No. 49

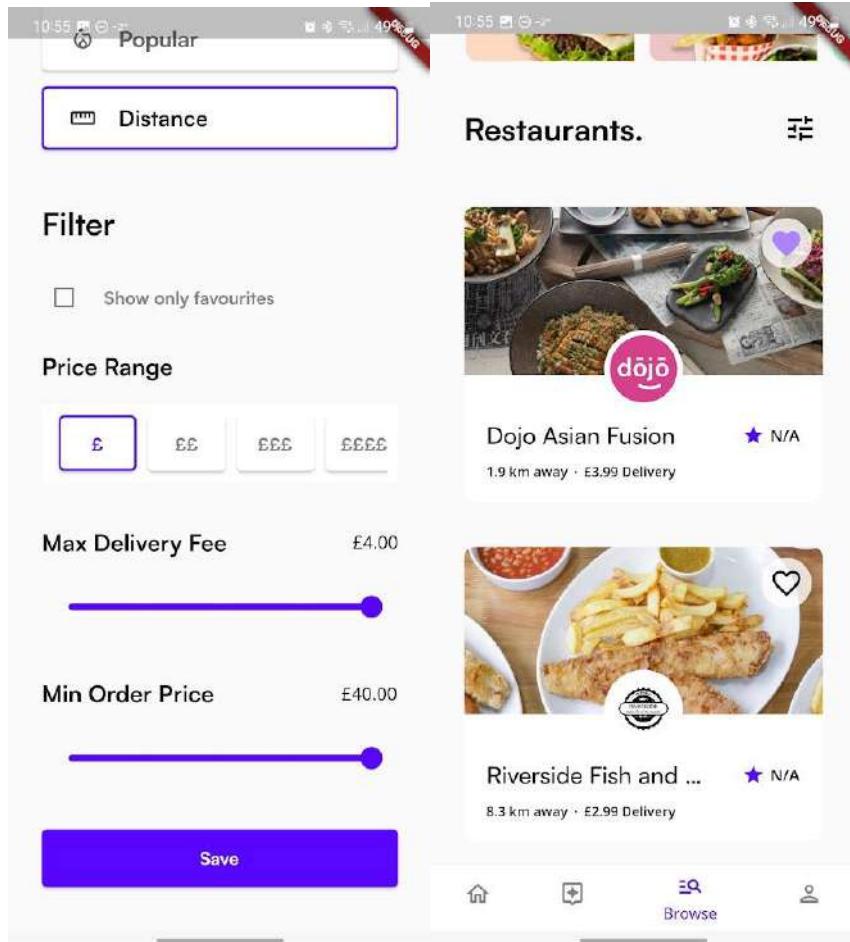
It only shows the favourites for the current profile

CPUR Admin	Peter John
------------	------------



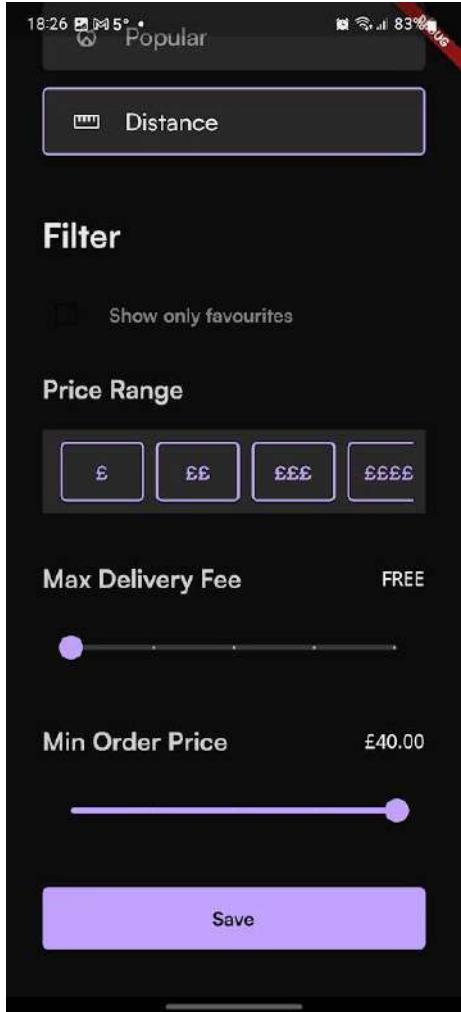
Test No. 50

Testing with a budget restaurant works correctly, showing only the cheap restaurants



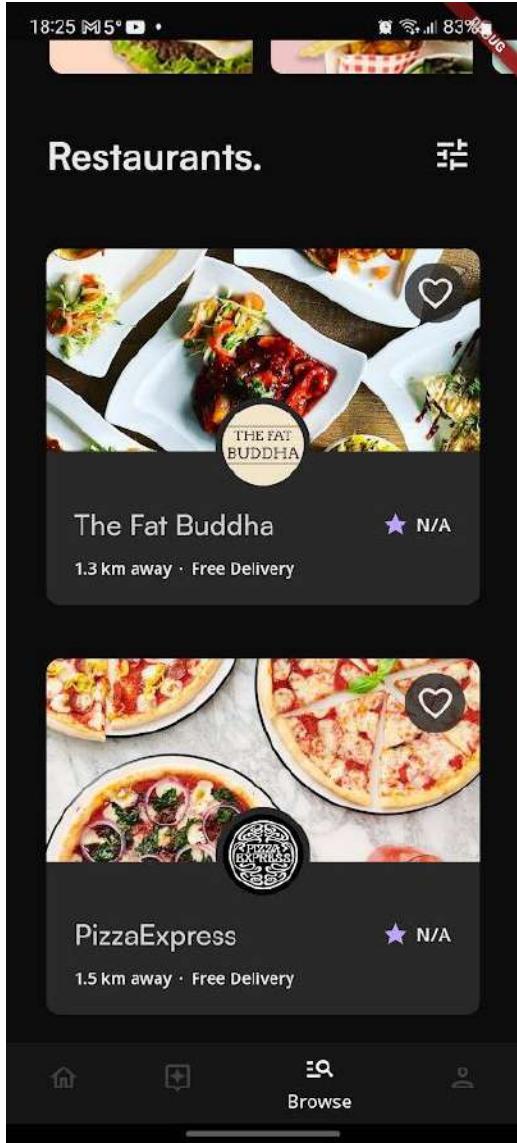
Test No. 51

The user is able to change the slider, with the price changing with the slider position



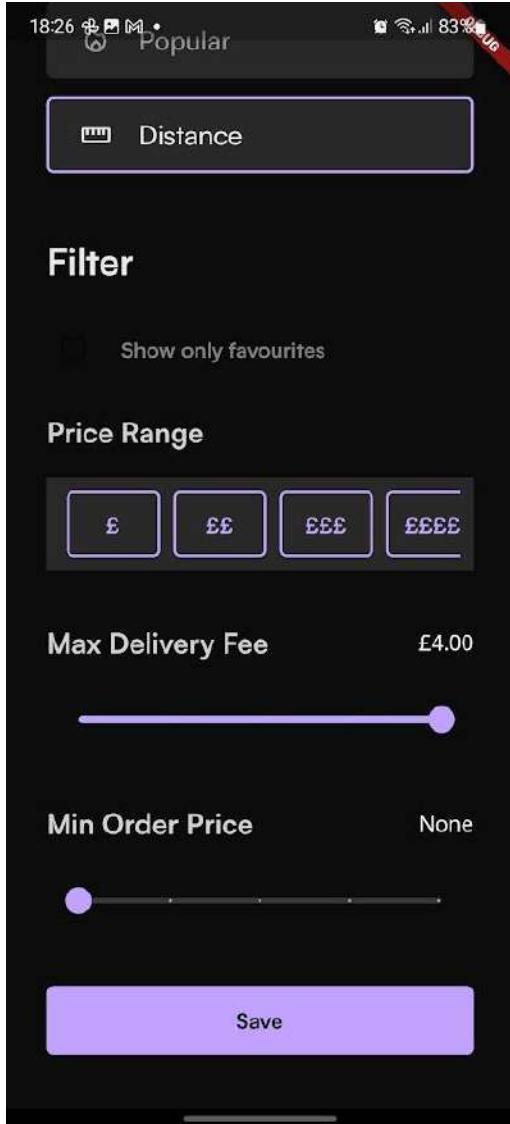
Test No. 52

When the user selected free delivery, it shows the restaurants with free delivery



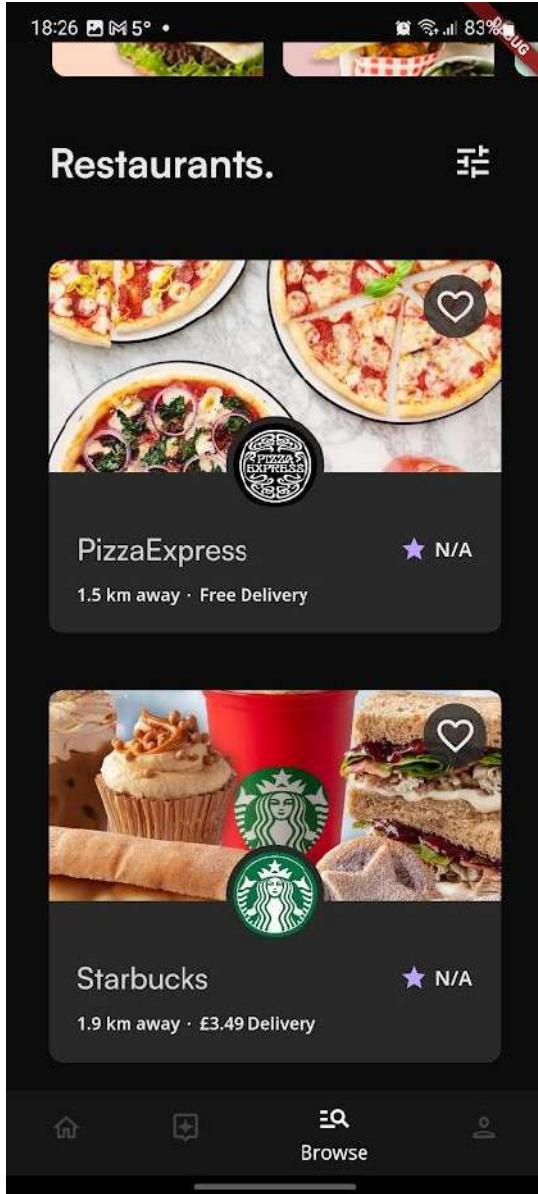
Test No. 53

The user is able to change their minimum order amount using a slider



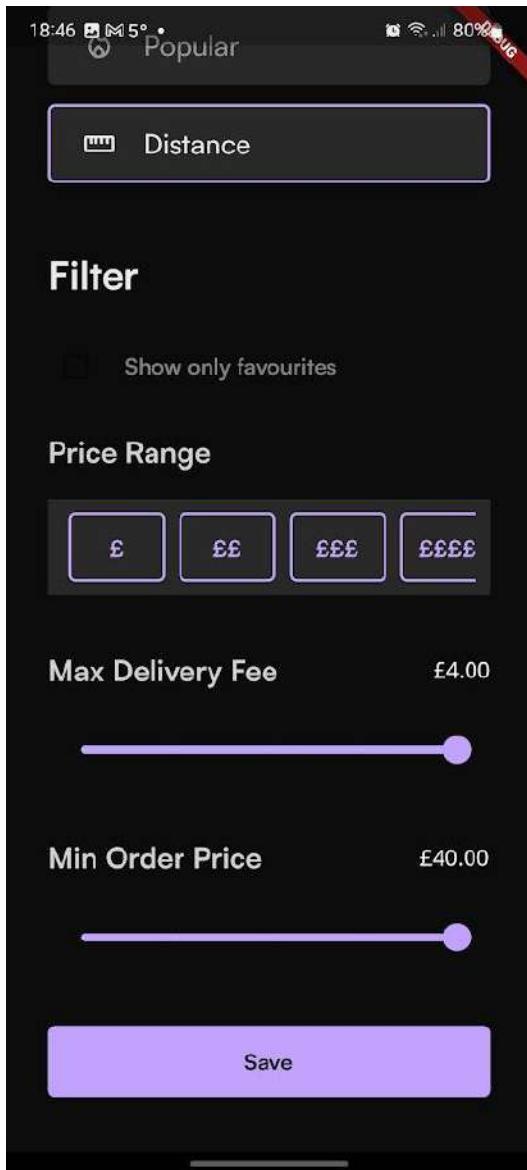
Test No. 54

When minimum order price is set to none, it only shows restaurants with no order minimum



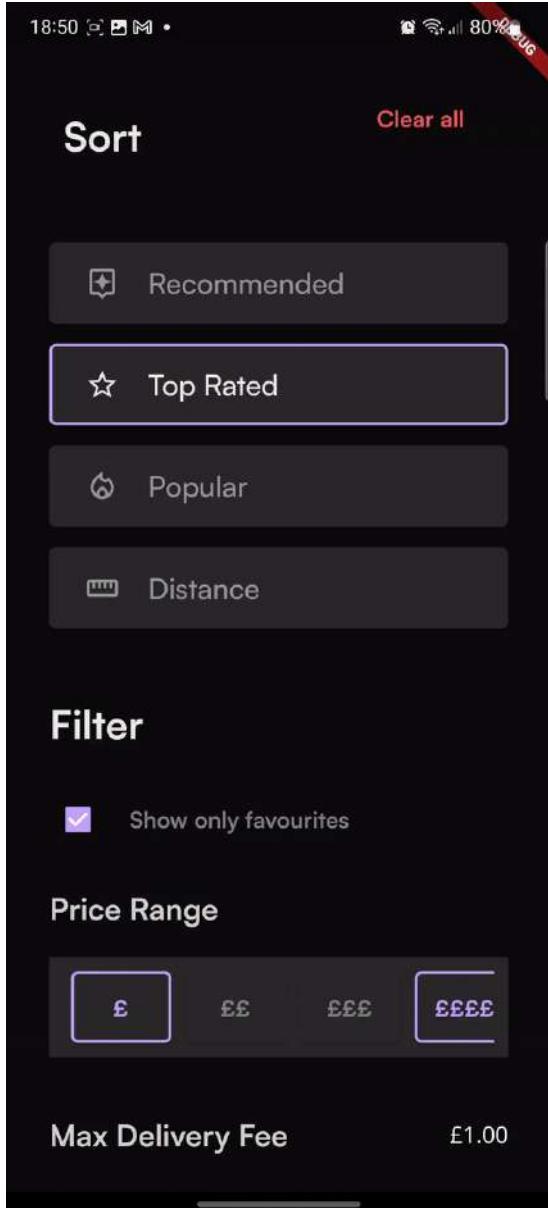
Test No. 55

At the bottom is a save button which reopens the navigation screen and saves the preferences.



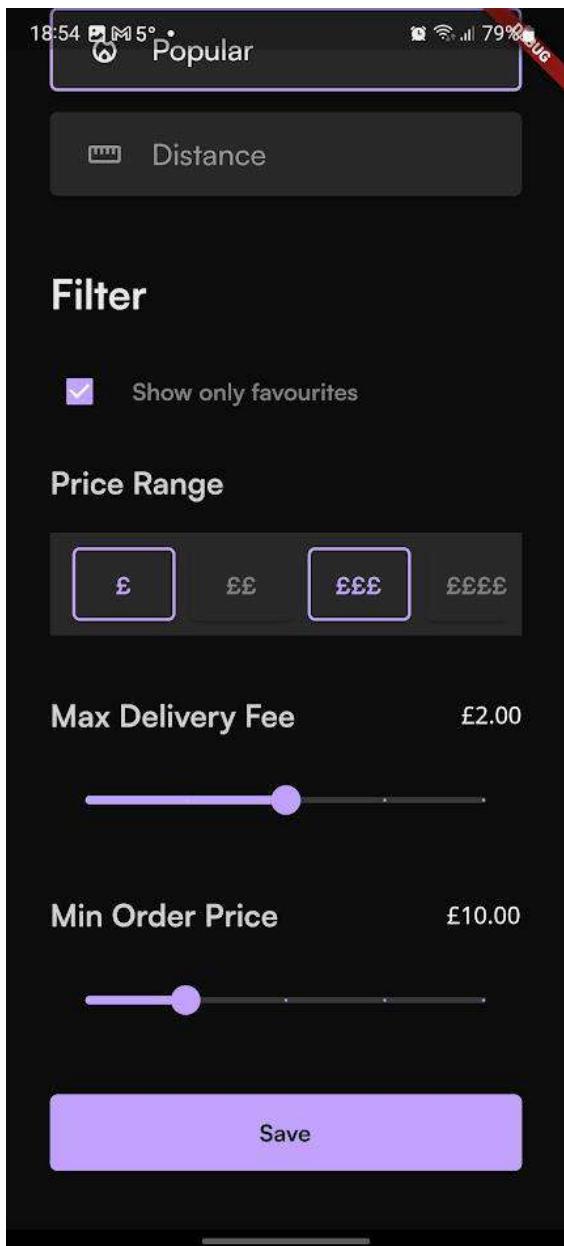
Test No. 56

There is a clear button which resets the filter and sort methods

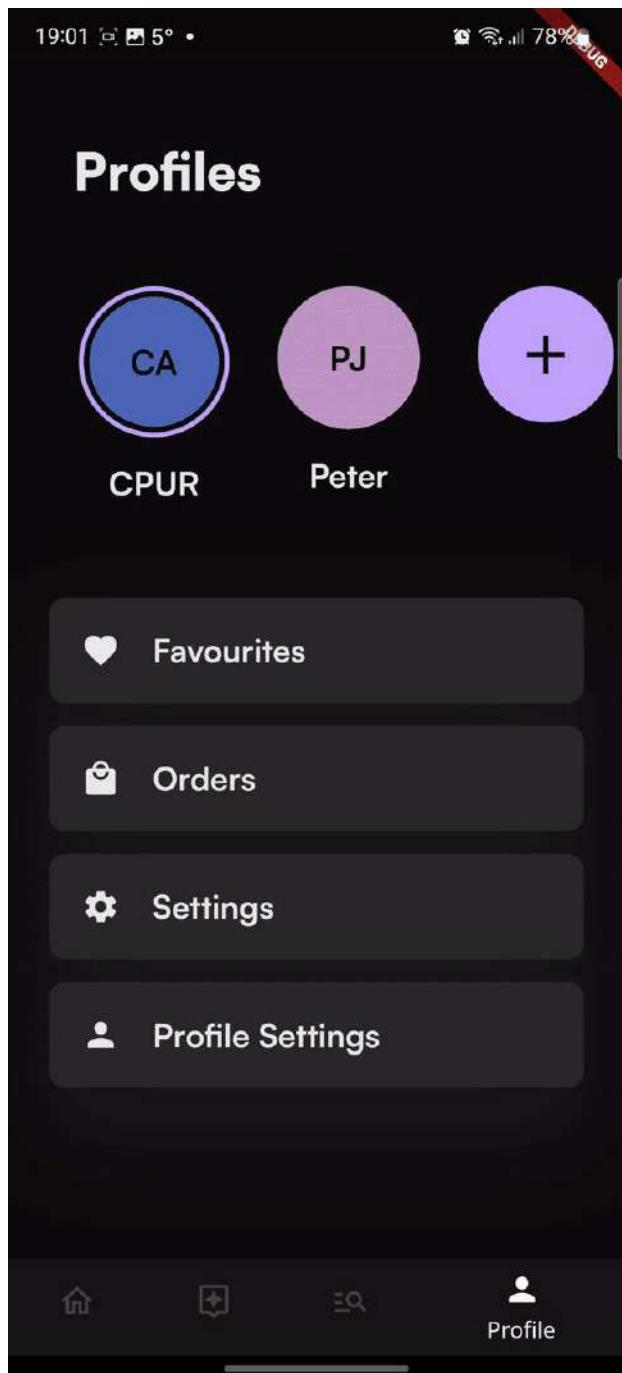


Test No. 57

Multiple filters can be used together



Test No. 58



Test No. 59

19:11 Apple & Pear Cawston Press 77%
Price: £7.25

1 House Dressing Dip
PizzaExpress
Price: £0.50

1 Tiramisu
PizzaExpress
Price: £6.25

1 San Pellegrino Limonata
PizzaExpress
Price: £2.80

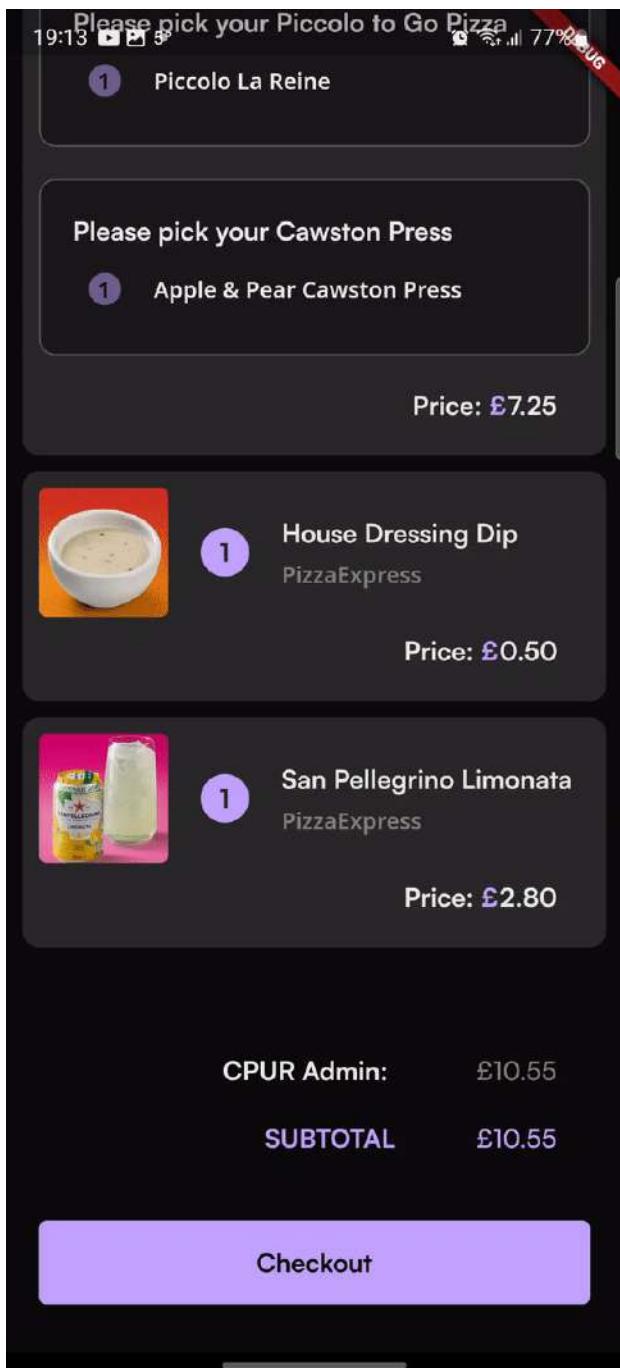
CPUR Admin: £16.80

SUBTOTAL £16.80

Checkout

Test No. 60

If a person removes two items, it removes the one that was removed first to ensure that that the person removes it correctly

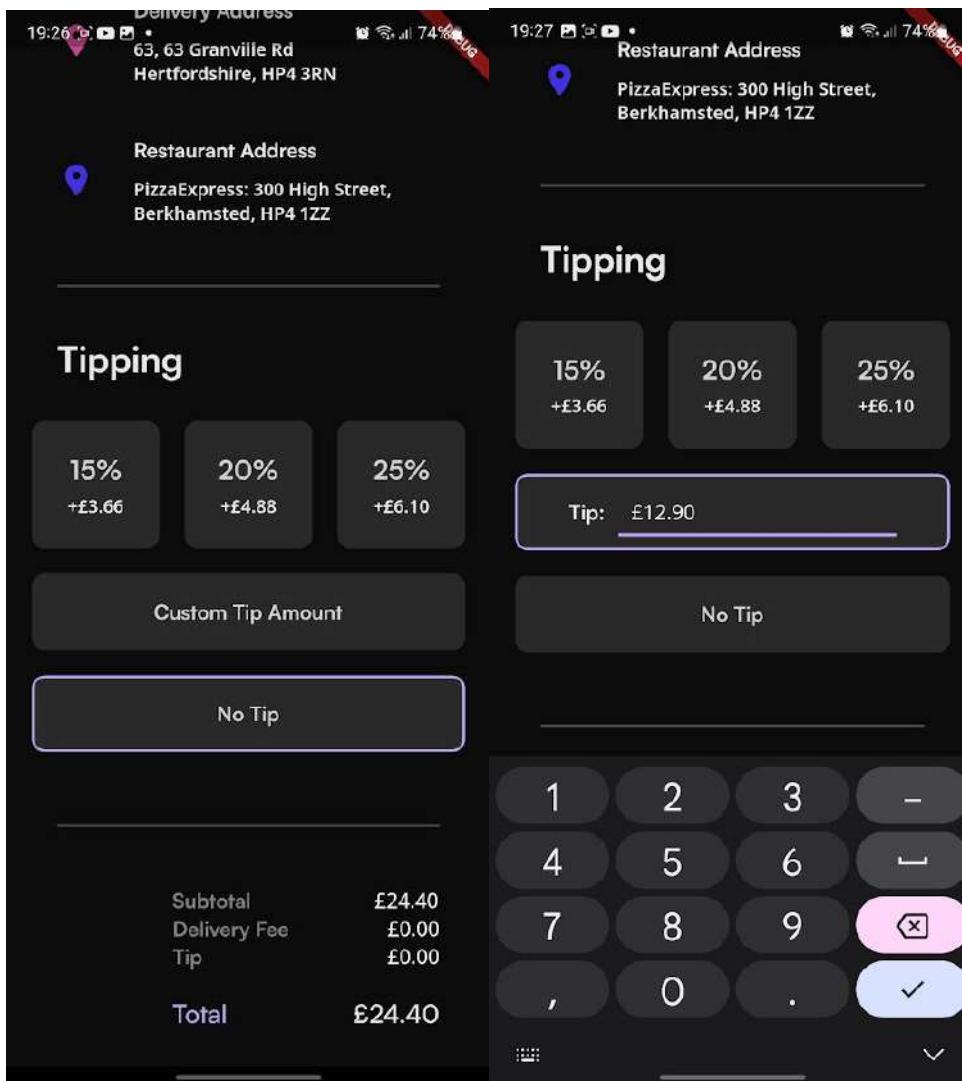


Test No. 61

Each item is successfully saved under the correct profile selected

Test No. 62

The user is able to both select a percentage as a tip or manually add a tip price.

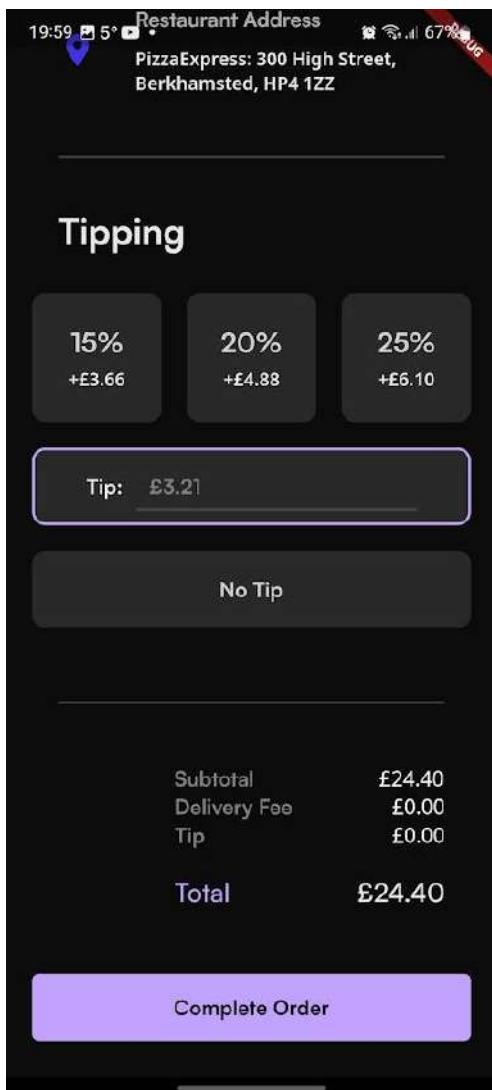


Test No. 63

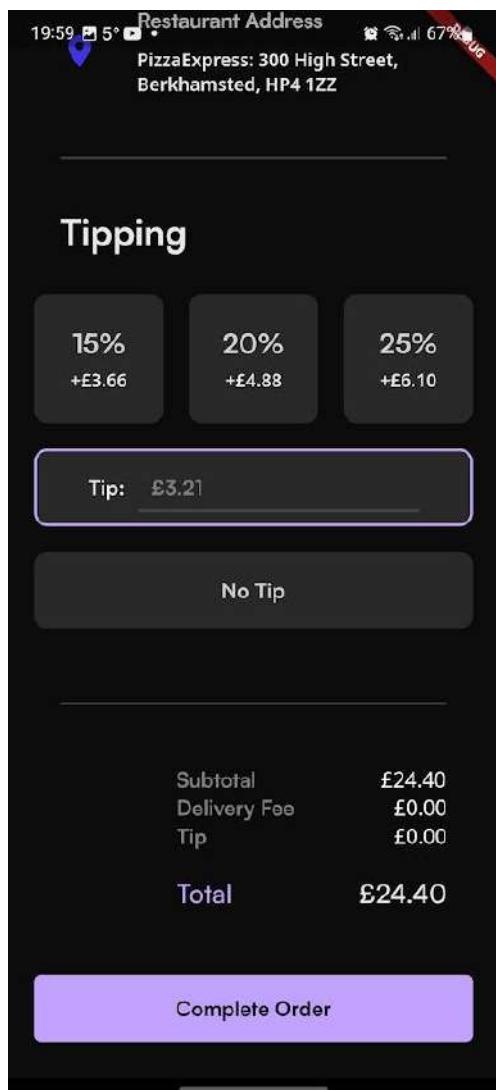
Since the user has a numerical keyboard, the only way to test this is by pasting text which when done, it does nothing since only numbers are done

Test No. 64

Example text is overlaid when there is no price entered in the custom tip amount



Test No. 65



Test No. 66

19:11 Apple & Pear Cawston Press 77%
Price: £7.25

1 House Dressing Dip
PizzaExpress
Price: £0.50

1 Tiramisu
PizzaExpress
Price: £6.25

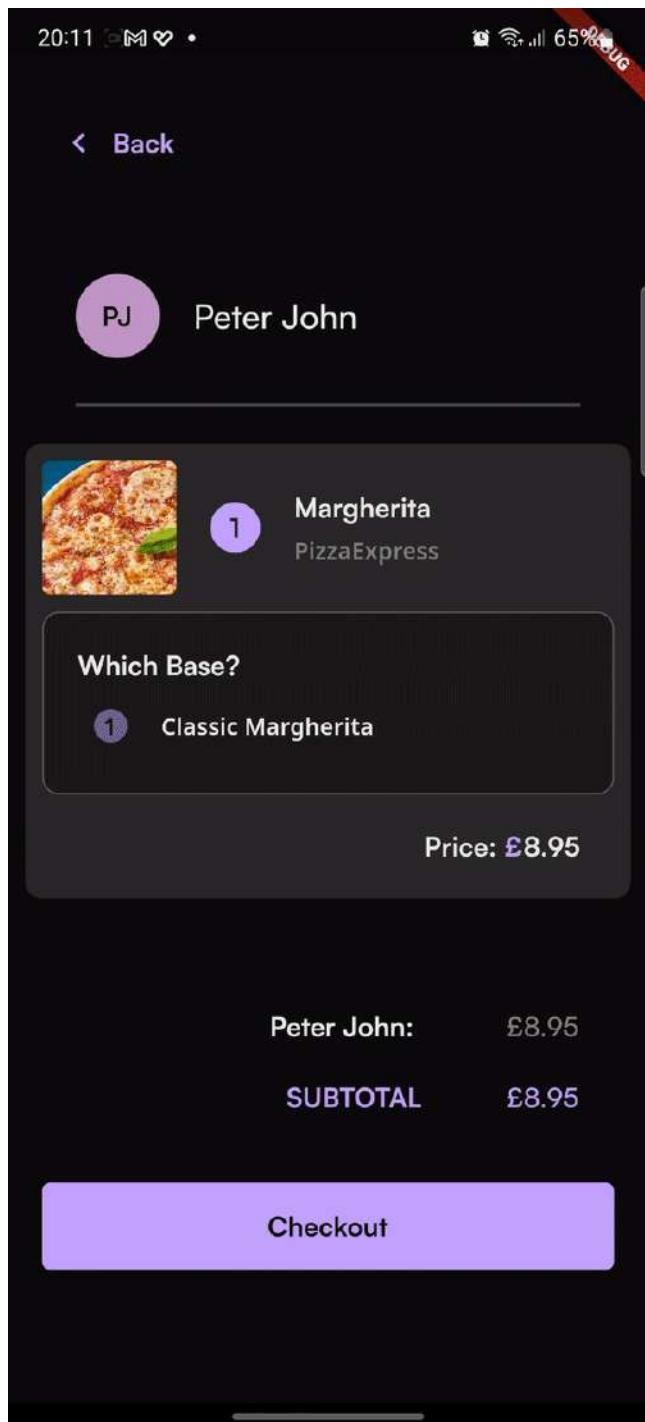
1 San Pellegrino Limonata
PizzaExpress
Price: £2.80

CPUR Admin: £16.80

SUBTOTAL £16.80

Checkout

Test No. 67



Test No. 68



Test No. 69



Usability testing

UI Analysis

Looking at the UI of the final app compared to the concept art, it seems that the design is almost identical. Thanks to Prototype 1B, I was able to remodel my app to fit a more user-centred design with a modern and clean design which is intuitive, allowing for a more icon based system.

In the future it would have been better if I stuck with a more Android design style since a lot of it used containers and unrestricted sizes meaning on larger screens, it would contain a lot of empty space. With the release of Android 12, it brought around major visual changes which I did not predict. I attempted to fix this by implementing dark mode and light mode depending on the device theme. Material Design 3 is an Android 12 feature that allows the user to customise their phone by changing their system colour scheme to the main colours of their background on their home screen. Some apps are beginning the transition to implementing this feature which I would like to do if I had more time. The following is an article that contains a more in-depth analysis of this

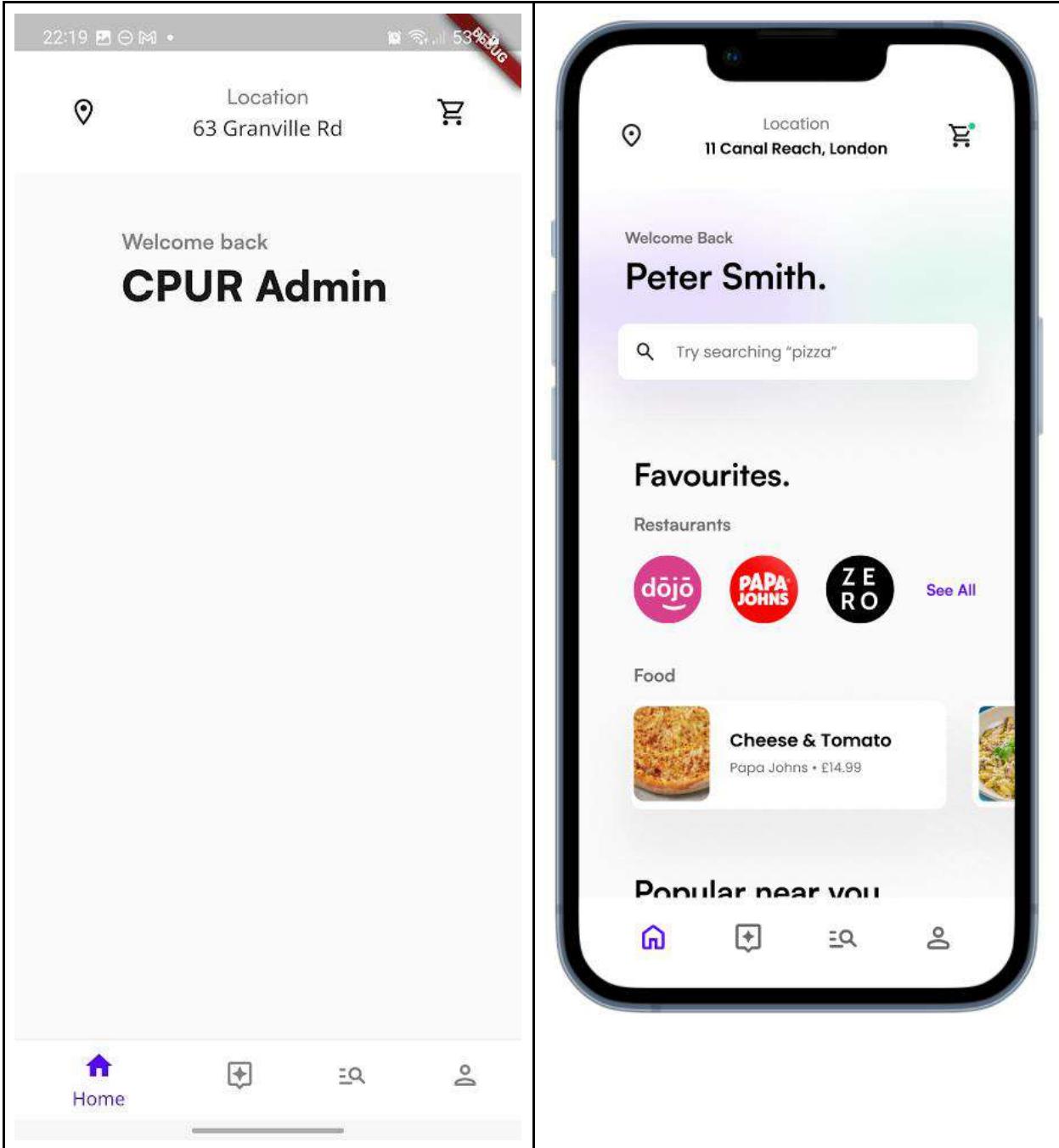
<https://m3.material.io/styles/color/dynamic-color/overview>

A comparison between the concept art and the final result is shown below

Homepage

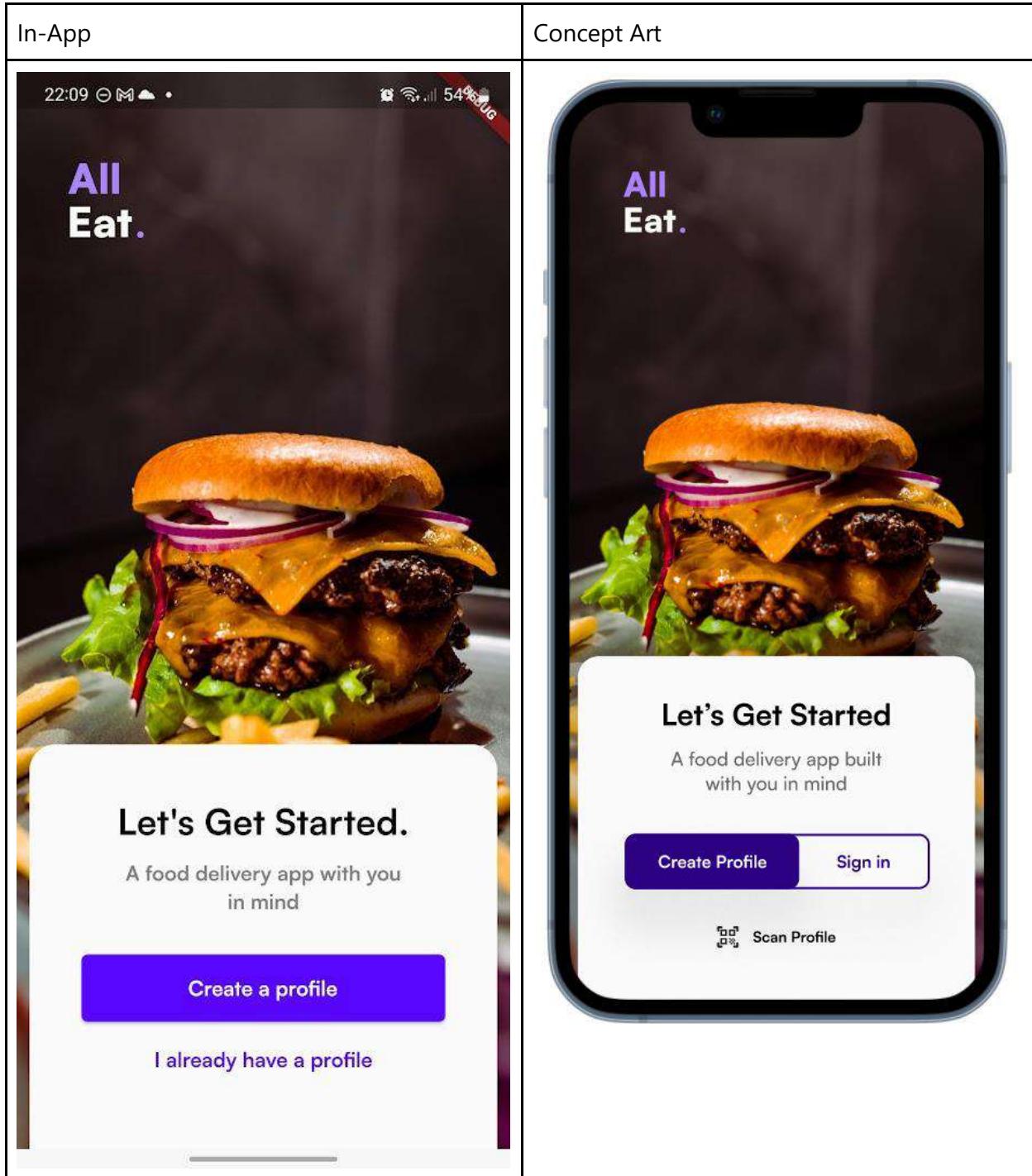
- I was unable to make the concept art main page due to the lack of time
- The navigation bottom bar has text in the app in order to help the user know what the page is and also helps those who are colour blind as it changes to show if it is selected

In-App	Concept art
--------	-------------



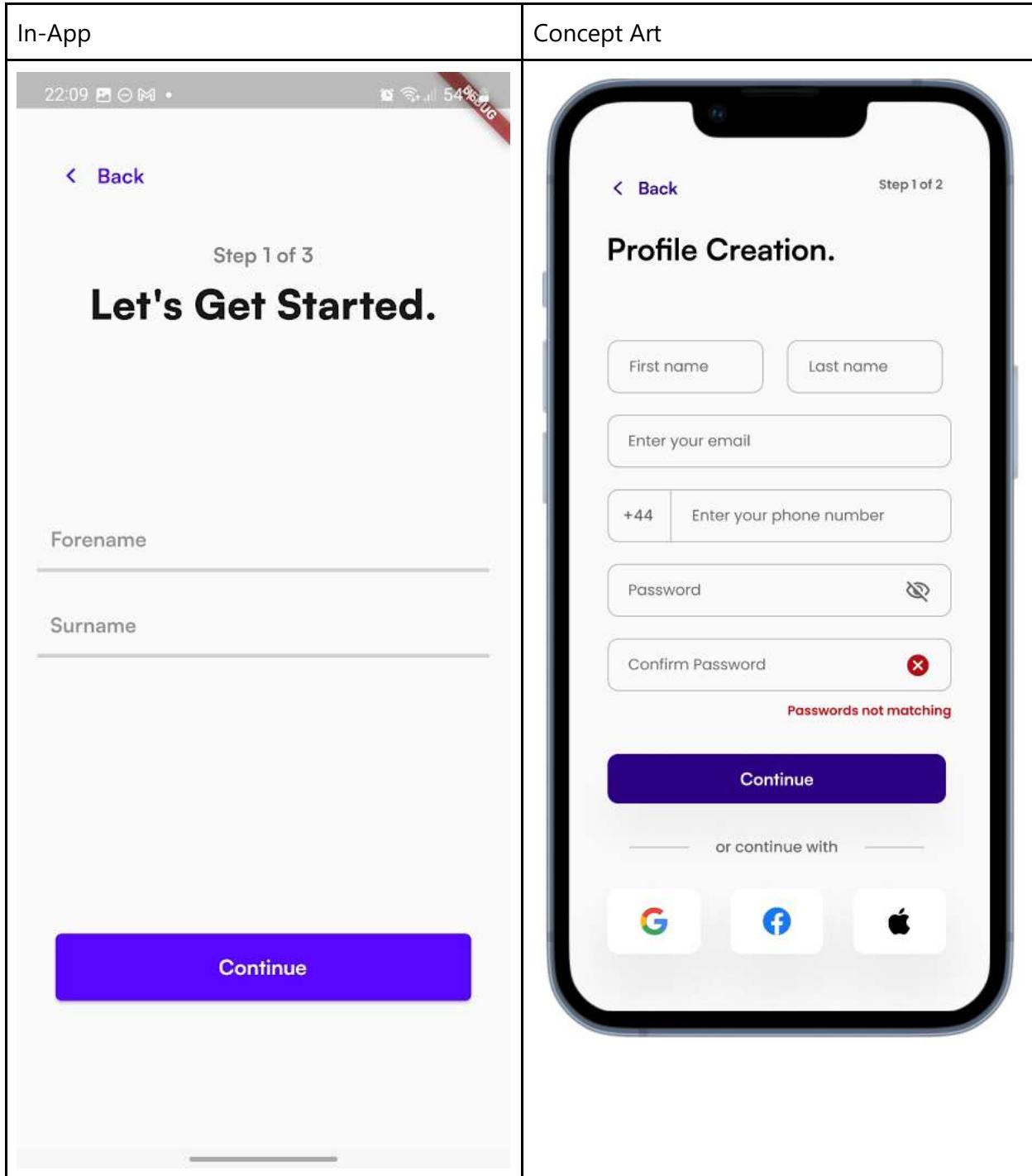
Welcome Page

- I was unable to put the buttons side-by-side due to the size of the phone being adaptive so on a small phone, it would be few letters per line and would overflow



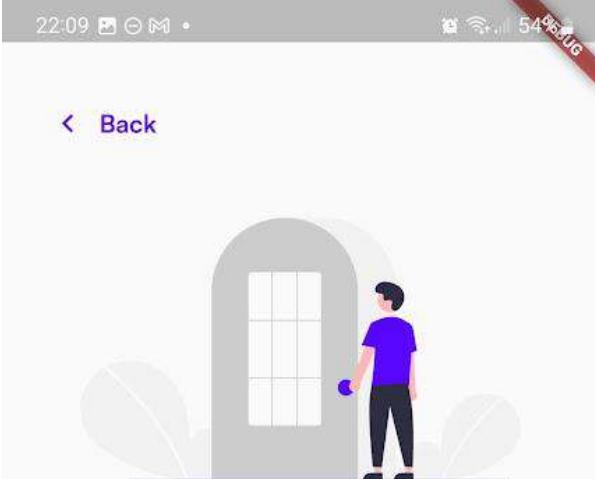
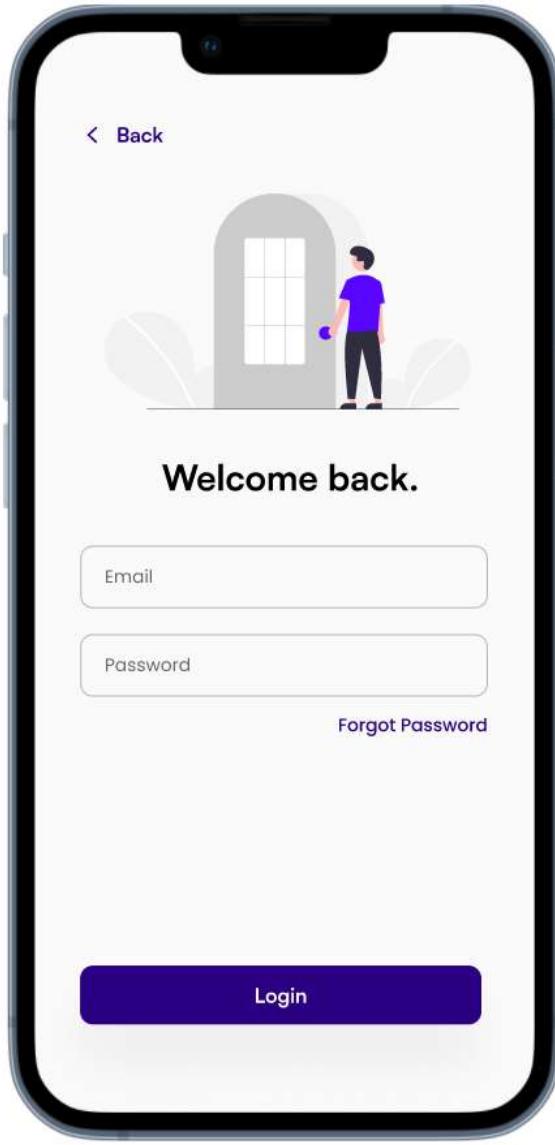
Profile Creation

- Once again, due to the size of a phone, it is not possible to fit two fields in one and having everything on one page makes it cramped



Login Page

- With the logistics behind the forgot password field, it is too much work to complete it and is lower priority compared to other parts of the app

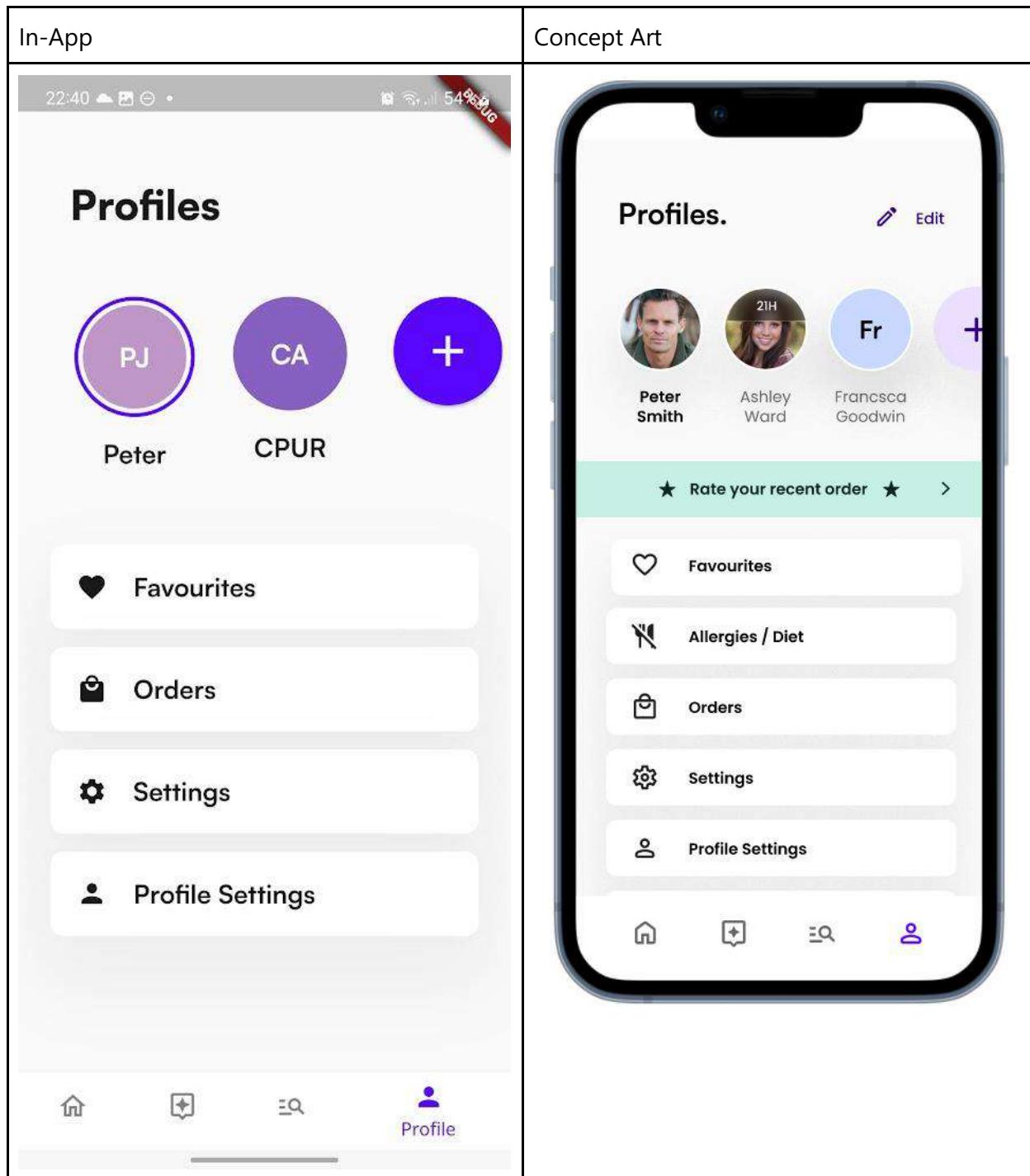
In-App	Concept Art
 <p>22:09 54% 54% Back</p> <p>Welcome back.</p> <p>Email <input type="text"/></p> <p>Password <input type="password"/></p> <p>Login to Profile</p>	 <p>Back</p> <p>Welcome back.</p> <p>Email <input type="text"/></p> <p>Password <input type="password"/></p> <p>Forgot Password</p> <p>Login</p>

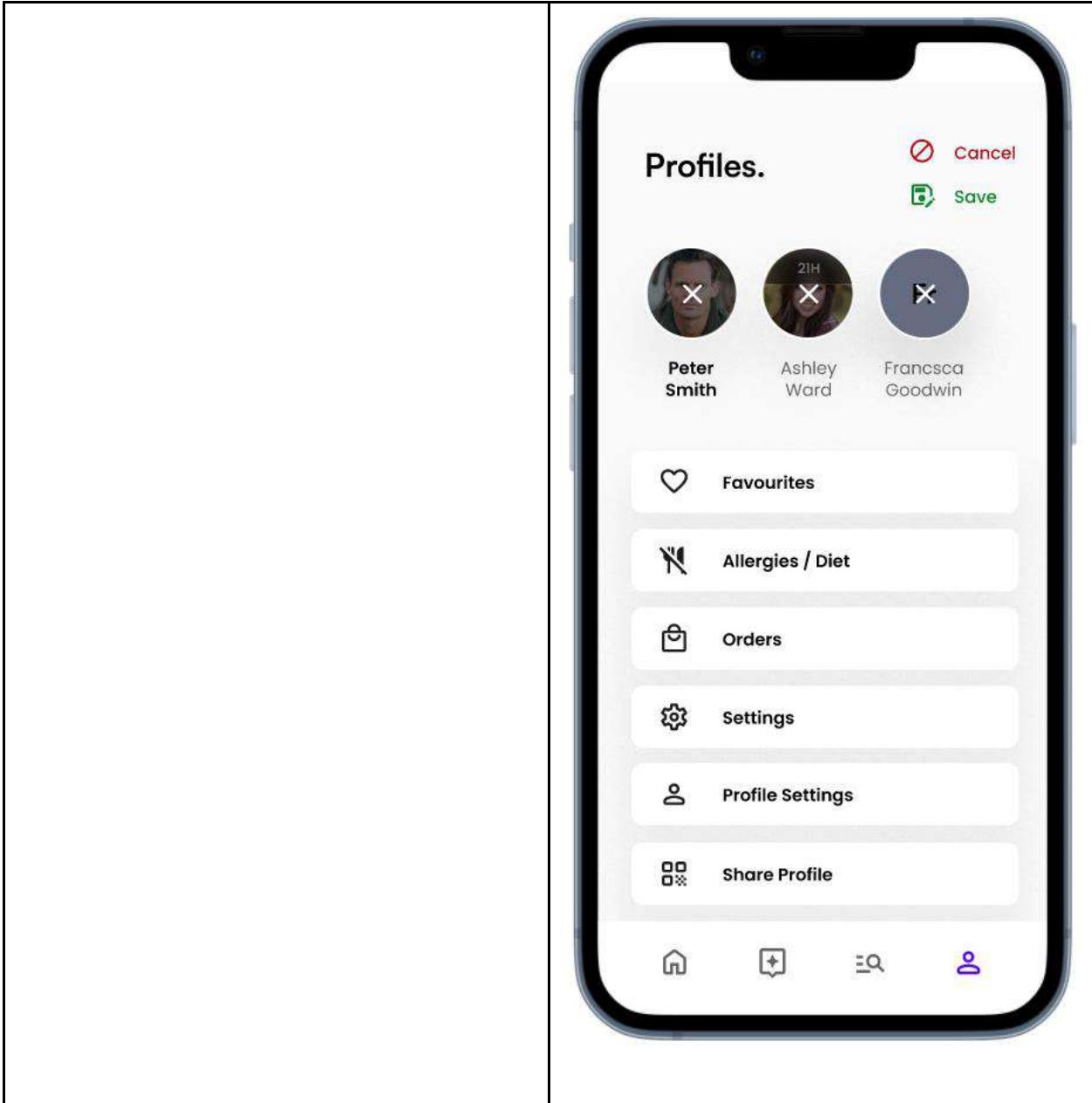
Profiles Page

- With the points of breaking with profile pictures, it was decided it should not be allowed. As well, to manage image upscaling and downscaling for number of sizes the profile can

be shown in, it was preferred to have a randomly selected profile colour for the profile icon

- I found that the edit button wasn't very useful since the only thing it did was highlight that each profile icon could be pressed to remove. In the final app, you tap to select the profile and long-press to remove it.

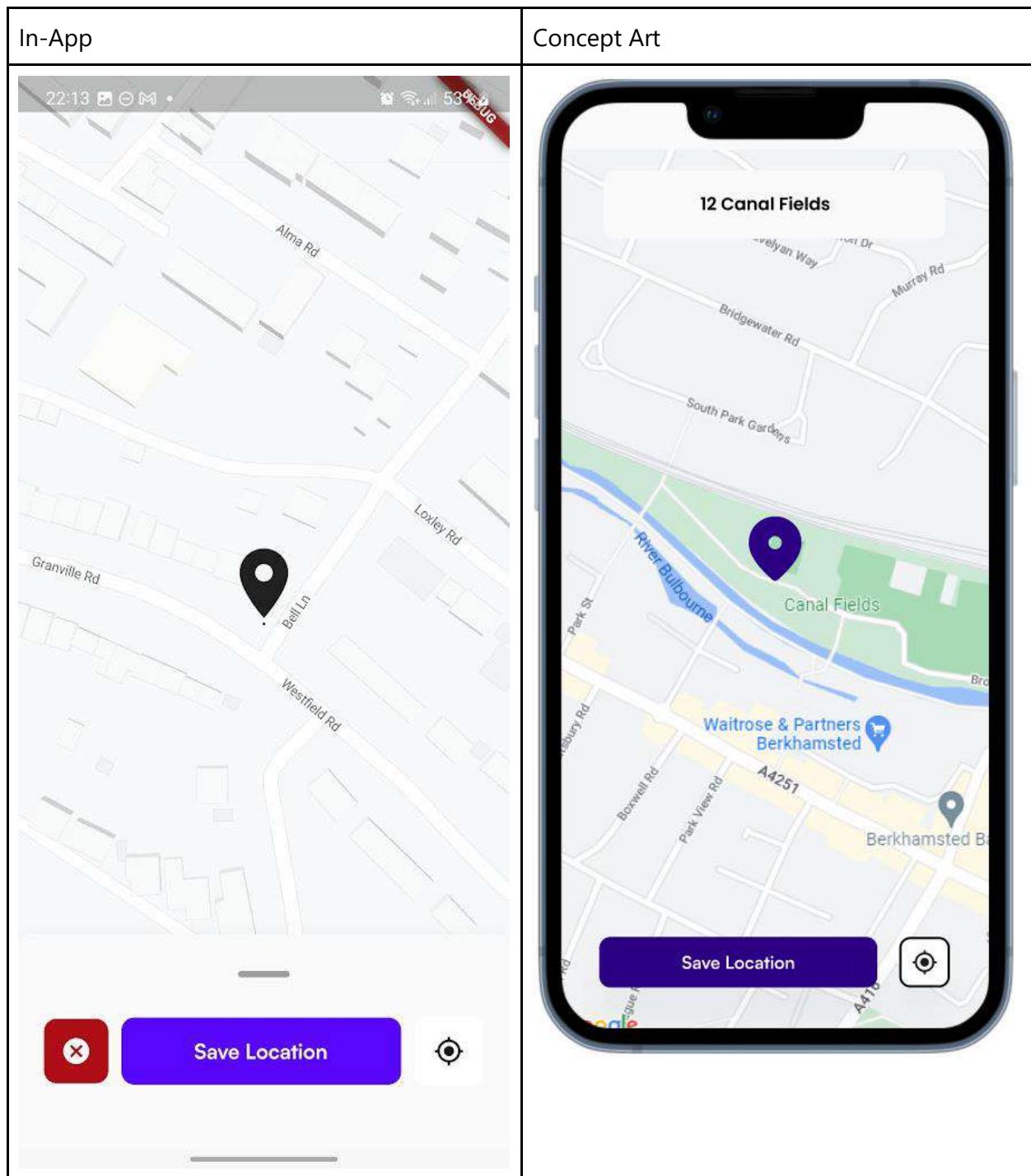




Location Selection

- Since the user can make mistakes and wants to revert their changes to their location, it is better to include a cancel button
- With the name at the top, I found that it was more useful to hide it and prioritise the map marker selection. If the user wishes to manually set the location, they can drag the

bottom panel to reveal the street name, county and postcode. This will allow for setting apartment numbers as well for dense residential or commercial areas

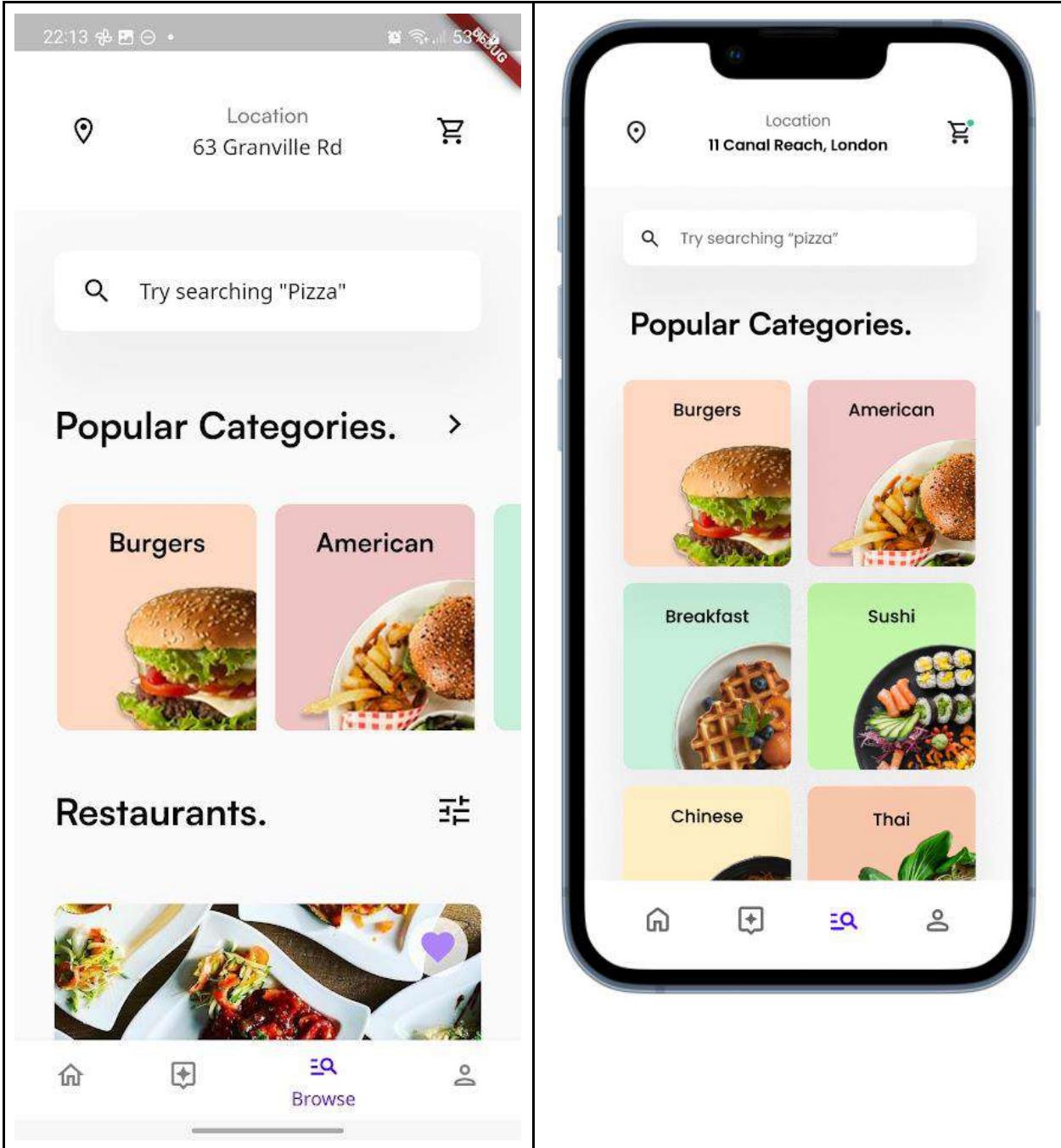


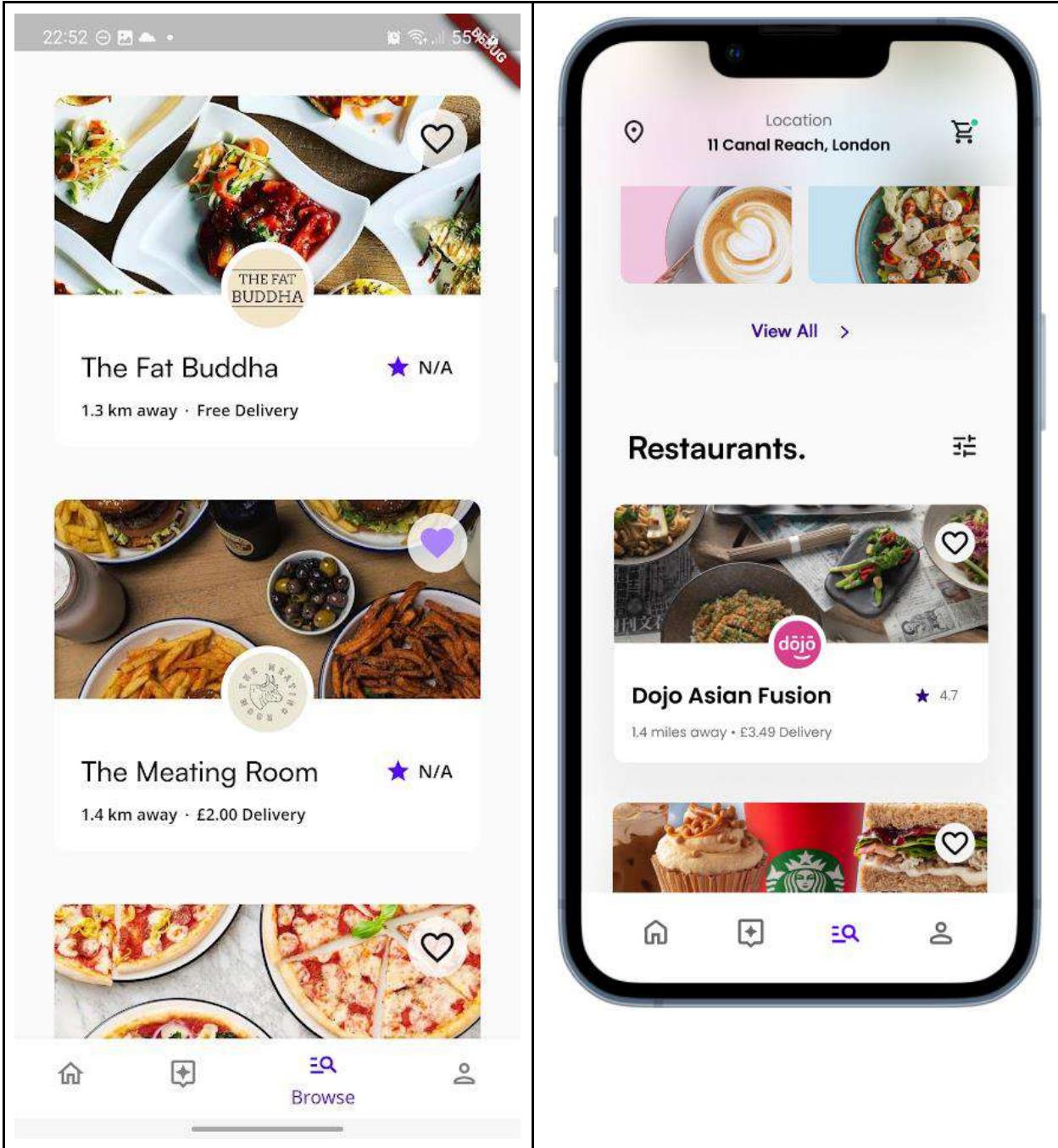
Browse Page

- The search bar was never completed so a container was substituted
- The categories don't go anywhere as well since they were never made
- The categories section was switched from a vertical list to a horizontal list in order to reduce the space taken. With such a long list of categories, there is a high chance the user will not see the restaurants list. If they want to see more categories, they can click on the right pointing arrow to get to a vertical list of categories formatted similar to the original browse page.
- For the favourite button, transparency was added just in case there was important information behind it that needs to be seen

In-App

Concept Art





Restaurant Page

- The restaurant information was never added since it was low priority
- Although intended to be added, the quick profile switcher was never added, only the space being left for it

- The remaining page is identical to the concept art

In-App	Concept Art
<p>Papa John's Pizza</p> <p>1.8 km away · £2.50 Delivery: £14 Order Minimum</p>	<p>Papa John's</p> <p>4.1 (12)</p> <p>Pizza • American • ££</p> <p>1.4 miles away • £3.49 Delivery • £20 Min. Order</p> <p>Currently ordering for</p> <p>Peter Smith</p> <p>Menu</p> <p>Pizza</p> <p>Cheese & Tomato £ 14.99</p> <p>The classic. Full of flavour, with our tomato sauce and mozzarella. Enjoy it pure ...</p>

Pizza

	All the Meats Our tomato sauce with mozzarella, pepperoni, pork sausage, crispy bacon, spicy	£ 17.99
	American Hot Our tomato sauce with mozzarella, slices of spicy pepperoni and jalapeño	£ 15.99
	BBQ Meat Feast BBQ sauce with mozzarella, spicy beef, sliced pepperoni, ham, pork sausage and	£ 17.99
	BBQ Chicken Classic Our tomato sauce with mozzarella, chargrilled chicken, crispy bacon, sliced	£ 16.99
	Cheese & Tomato The classic. Full of flavour, with our tomato sauce and mozzarella. Enjoy it pure	£ 14.99

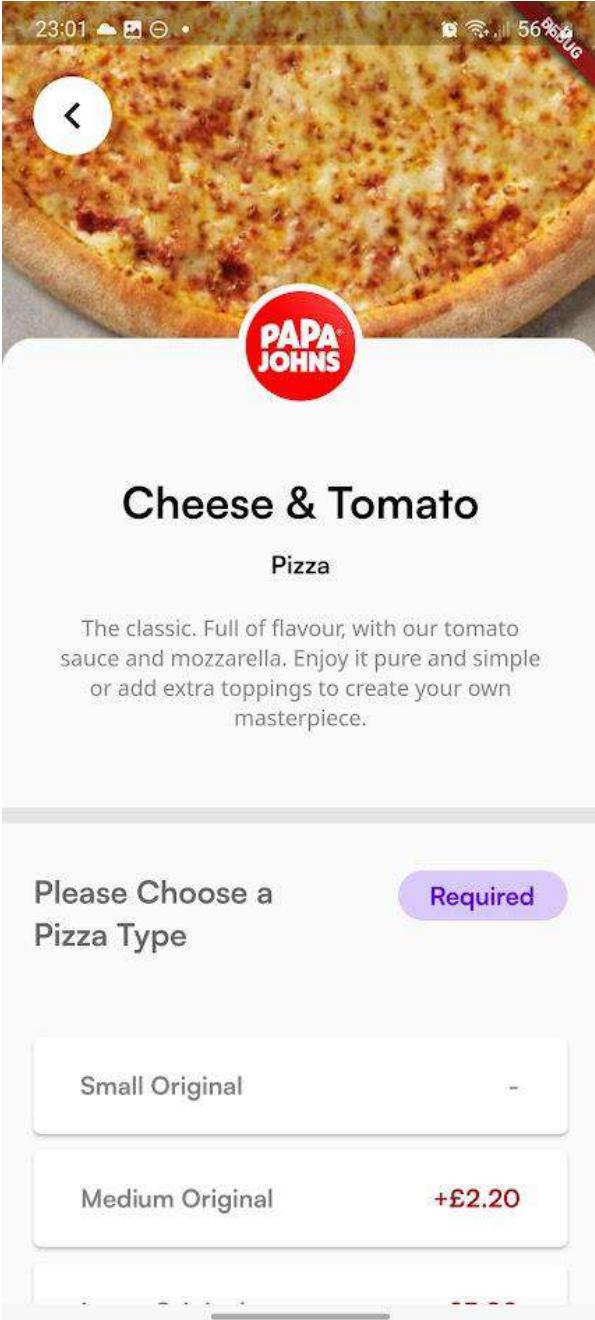
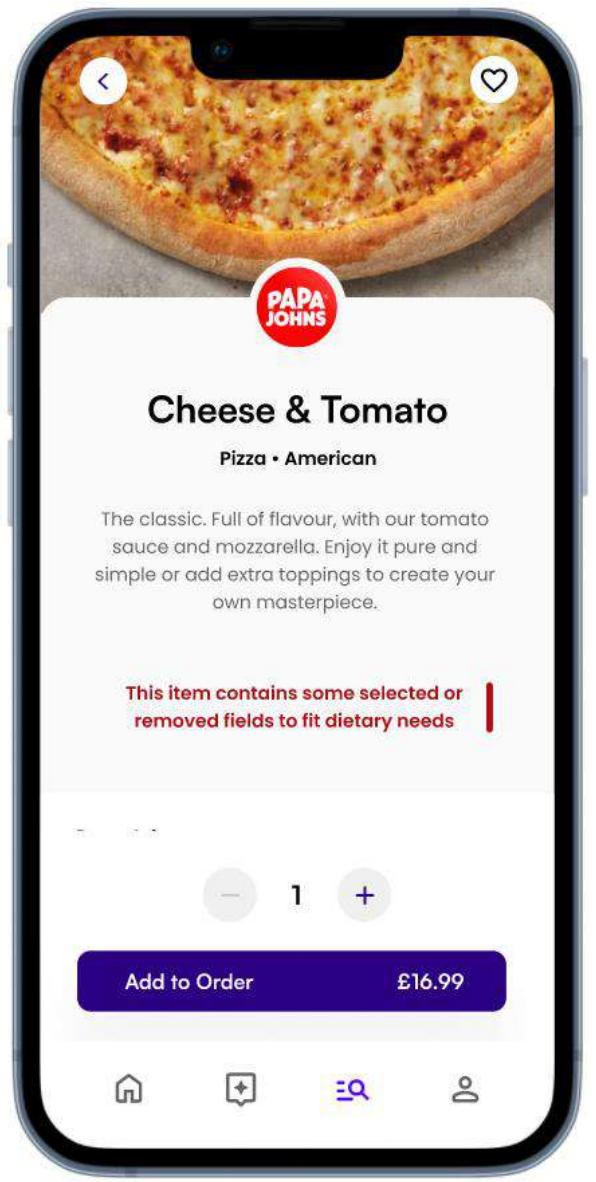
Papadias

	BBQ Chicken and Bacon Our new Italian flat-bread style papadia made from fresh dough with grilled	£ 6.49
	Italian Sausage & Pepperoni	

Item Page

- Without the allergies, the page looks slightly different
- Adding and removing has a more simplified design in order to have space for bigger hitboxes

- I was unable to make the overlaid price work so it was placed at the bottom. It works better by showing the price for one item above the quantity selector and the price for the quantity of items in the button

In-App	Concept Art
 <p>Cheese & Tomato</p> <p>Pizza</p> <p>The classic. Full of flavour, with our tomato sauce and mozzarella. Enjoy it pure and simple or add extra toppings to create your own masterpiece.</p> <p>Please Choose a Pizza Type</p> <p>Small Original</p> <p>Medium Original +£2.20</p> <p>Required</p>	 <p>Cheese & Tomato</p> <p>Pizza • American</p> <p>The classic. Full of flavour, with our tomato sauce and mozzarella. Enjoy it pure and simple or add extra toppings to create your own masterpiece.</p> <p>This item contains some selected or removed fields to fit dietary needs</p> <p>- 1 +</p> <p>Add to Order £16.99</p>

The image displays two side-by-side screenshots from a mobile application for ordering pizza.

Screenshot 1 (Left):

- Header:** "Please Choose a Pizza Type" (Required).
- Options:**
 - Small Original
 - Medium Original +£2.20** (Selected)
 - Large Original +£3.80
 - Medium Authentic Thin Crust +£2.20
 - Large Authentic Thin Crust +£3.80
 - Medium Stuffed Crust +£5.50
 - Large Stuffed Crust +£7.10
- Section:** "Defaults" (Optional).
 - Select which when...

Screenshot 2 (Right):

- Header:** Papa Johns (4.1 (12))
- Information:** Pizza • American • ££, 1.4 miles away, £3.49 Delivery, £20 Min. Order.
- Price:** £ 14.99
- Rating:** 4.8 (92)
- Section:** Select your crust. (Required)
 - Original
 - Authentic Thin Crust
 - Stuffed Crust +£2.99
- Quantity:** 1
- Action:** Add to Order £16.99
- Bottom Navigation:** Home, Search, Profile

23:08 Large Stuffed Crust +£7.10

Large Stuffed Crust +£7.10

Defaults Optional

Select which what you would like removed

1 Base Pizza Sauce

Mozzarella Cheese -

Dip Special Garlic -

Optional Optional

Select which what you would like added

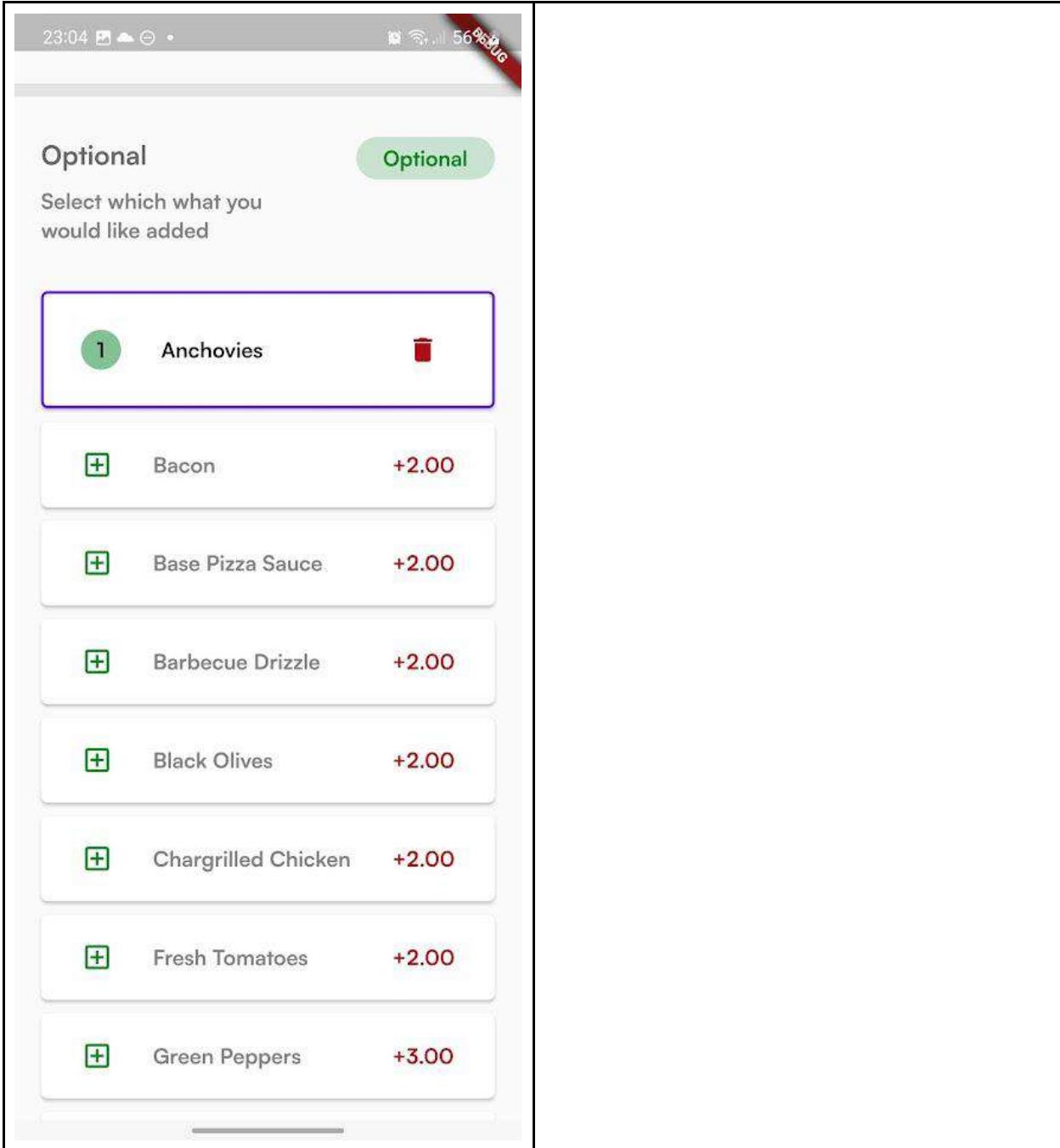
1 Anchovies

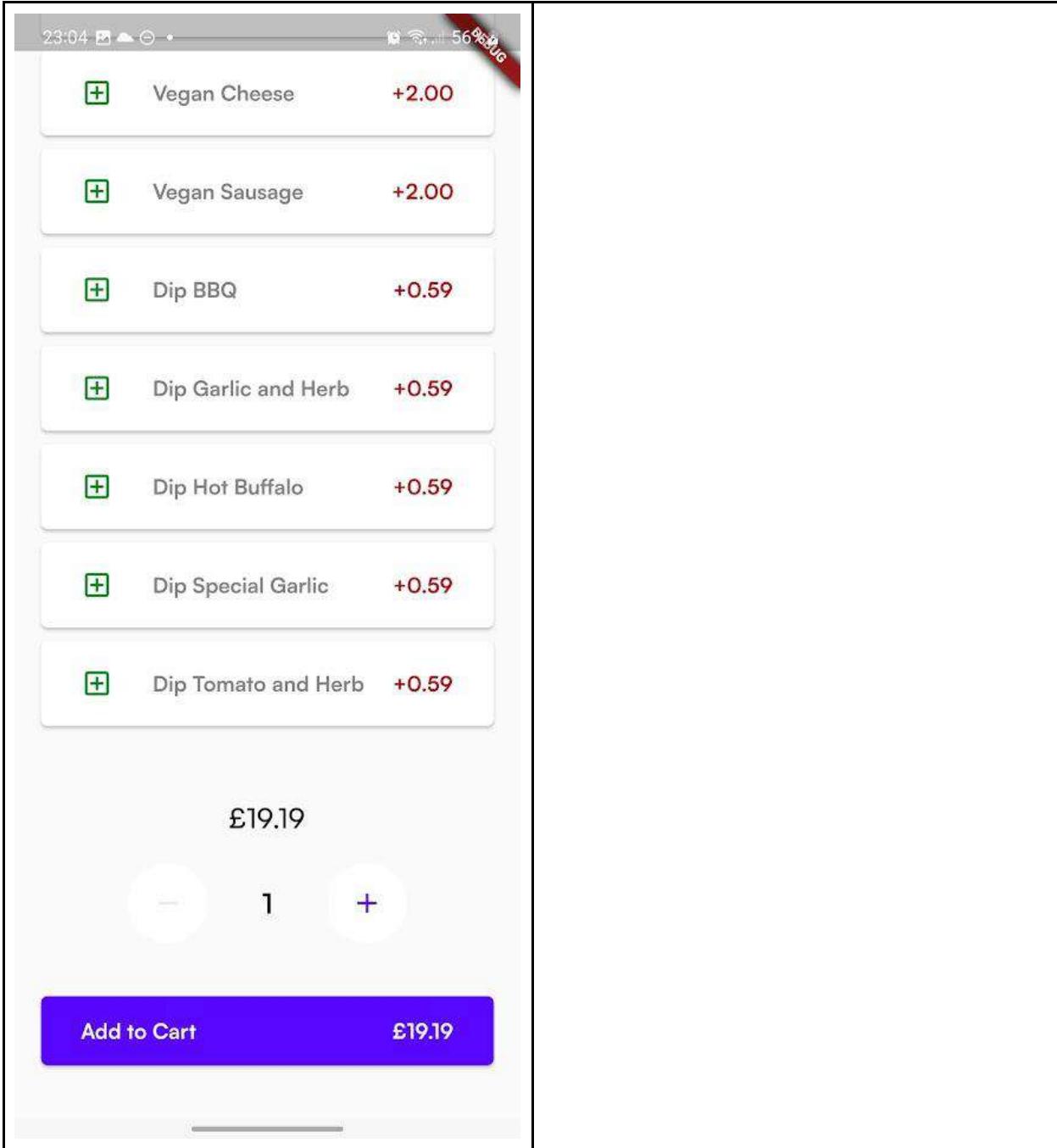
+£1.60

+£1.60

+£1.60

Add to Order £16.99





Cart

- When making the concept art, I forgot to display the quantity for each customised option
- Due to the size, each option has to be put on a separate line for each customised title

- Although unintuitive, the removal of items is done with a slide to remove gesture since it means the index can be stored within the element easily
- The delivery cannot be calculated in the cart since the way I test if one restaurant has been selected is within the button so it would not work the way the concept art intended

In-App	Concept Art
--------	-------------

22:17 53 DEBUG

< Back

CA CPUR Admin



1 Margherita
PizzaExpress

Which Base?

1 Romana Margherita

Price: £10.90

CPUR Admin: £10.90

SUBTOTAL £10.90

Checkout

< Back

Cart.

Edit

Peter Smith 2 items



1 Vegan Giardiniera
Pizza Express £ 13.45

Which Base? Romana

Would you like to add any extra toppings + Mushroom

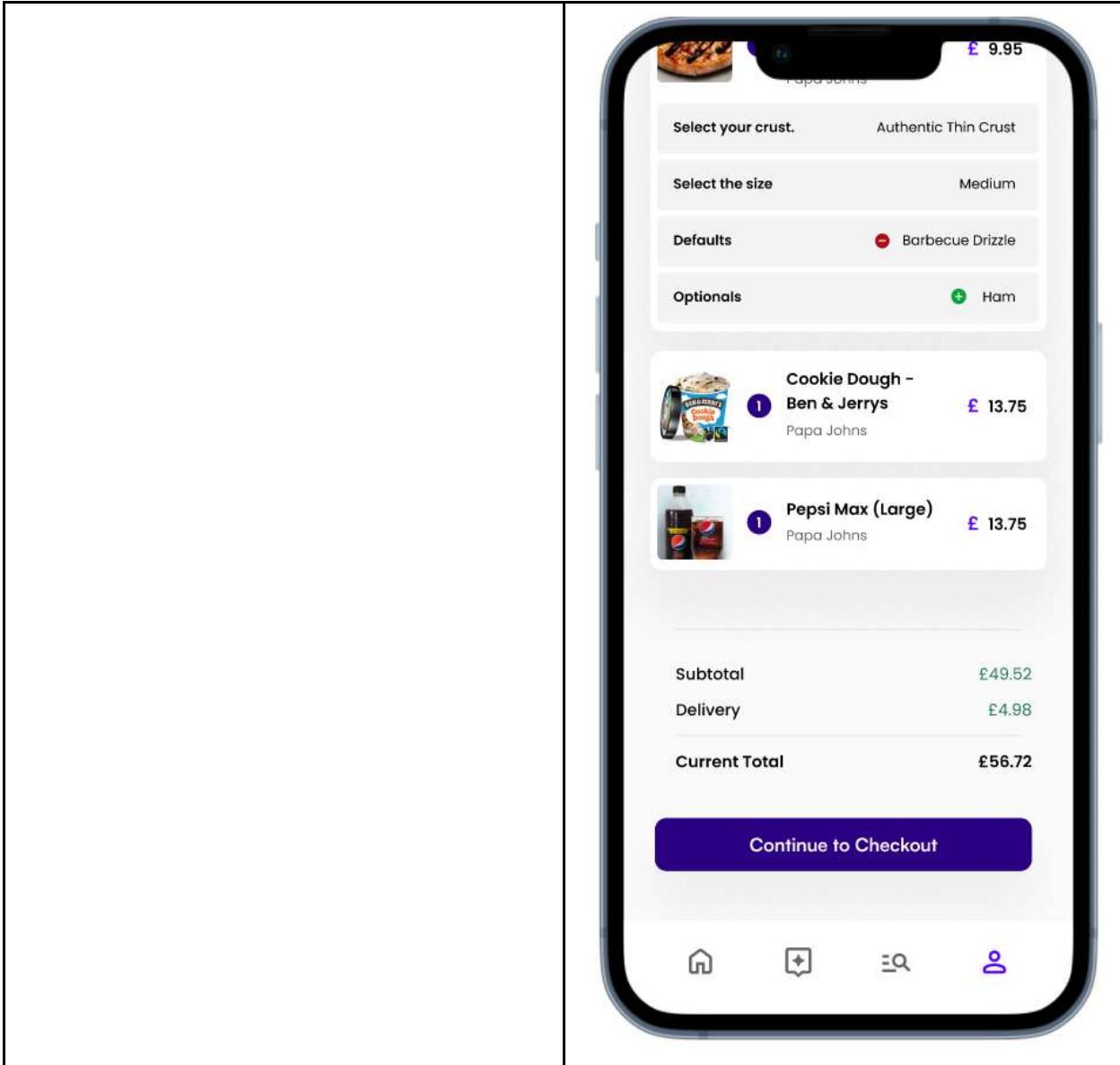


1 Pollo Pesto
Pizza Express £ 13.75

This order contains ingredients which Peter is not allowed to consume
MILK

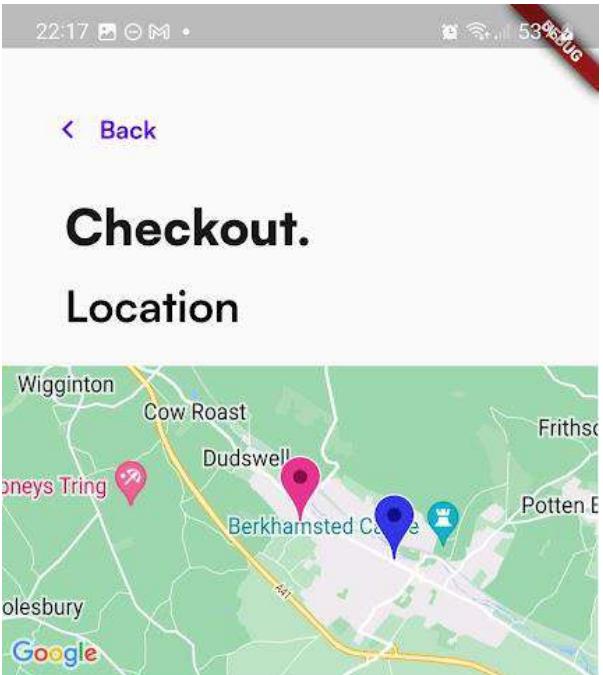
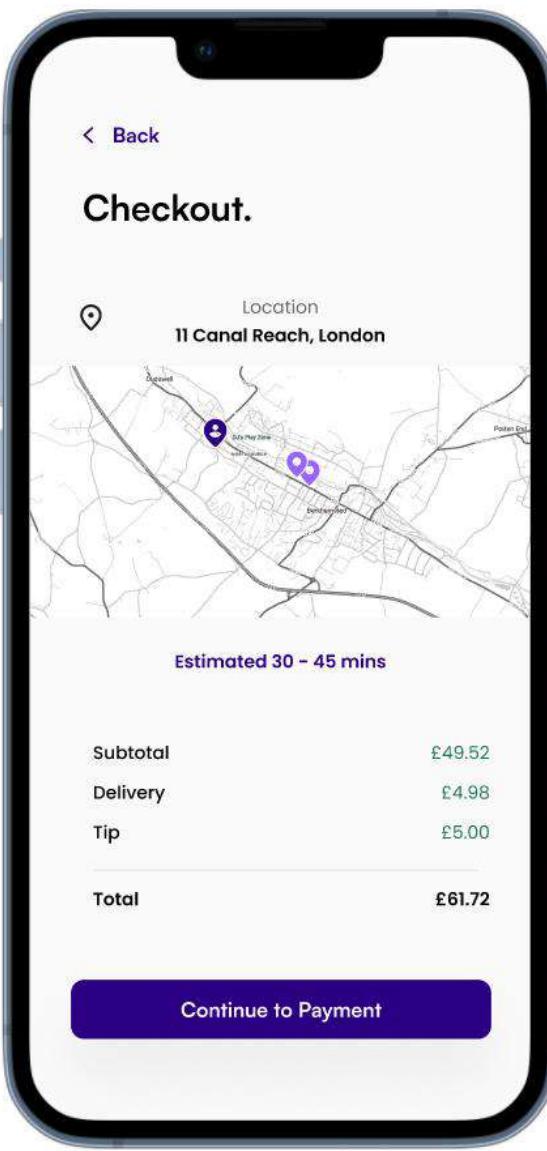
Ashley Ward 3 items

Home Search Profile



Checkout

- I tried to get custom maps working but could not get it working so I stuck with keys and just saying what each one was
- The way to get the tip was never shown in the concept art so in the real app, I made a multi-button system similar to the systems used in restaurants
- It doesn't go to the payment section since it would require calculations and storing sensitive personal data about the user.

In-App	Concept Art								
 <p>22:17 53% •</p> <p>< Back</p> <h2>Checkout.</h2> <h3>Location</h3> <p>Wigginton Cow Roast Dudswell Frithsden Potten End Berkhamsted Station A41 Google</p> <p>Delivery Address  63, 63 Granville Rd Hertfordshire, HP4 3RN</p> <p>Restaurant Address  PizzaExpress: 300 High Street, Berkhamsted, HP4 1ZZ</p> <hr/> <h3>Tipping</h3>	 <p>< Back</p> <h2>Checkout.</h2> <p>Location 11 Canal Reach, London</p> <p>Estimated 30 - 45 mins</p> <table> <tbody> <tr> <td>Subtotal</td> <td>£49.52</td> </tr> <tr> <td>Delivery</td> <td>£4.98</td> </tr> <tr> <td>Tip</td> <td>£5.00</td> </tr> <tr> <td>Total</td> <td>£61.72</td> </tr> </tbody> </table> <p>Continue to Payment</p>	Subtotal	£49.52	Delivery	£4.98	Tip	£5.00	Total	£61.72
Subtotal	£49.52								
Delivery	£4.98								
Tip	£5.00								
Total	£61.72								

22:17 M •

Restaurant Address 53 DEBUG

PizzaExpress: 300 High Street,
Berkhamsted, HP4 1ZZ

Tipping

15% +£1.64 20% +£2.18 25% +£2.73

Custom Tip Amount

No Tip

Subtotal	£10.90
Delivery Fee	£0.00
Tip	£0.00
Total	£10.90

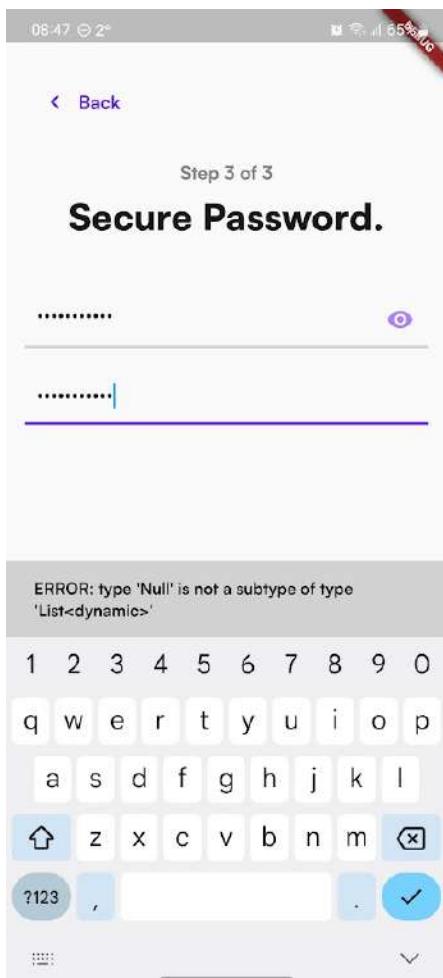
Complete Order

Testing

Black Box Testing

Issue 1

I gave my friend the app to try to find any issues and they were able to successfully break it. When typing an email with a zero in it, it would display an error.



What was interesting is that it would only happen when the email contained a 0 not any number.

Checking the email field, no discrepancies were found

```
inputFormatters: [
```

```

        //Only allows the input of letters a-z and A-Z and
0-9 and @,.-$&!#?

        FilteringTextInputFormatter.allow(RegExp(r'[a-zA-
Z0-9@#$&!#?]')))

    ],

validator: (password) {

    //Required field and uses emailvalidator package to
verify it is an email to simplify the code

    if (password == null || password.isEmpty) {

        return "Required";

    } else if (!password.contains(RegExp(r'[0-9]'))) ||
!password.contains(RegExp(r'[a-z]'))) {

        return "Password must contain at least 1 number
and 1 letter";

    } else if (password.length < 8) {

        return "Password must be a minimum of 8
characters";

    } else if (password.length > 99) {

        return "Password must be a maximum of 99
characters";

    }

    return null;

},

```

To find the source of the error, I removed the catch statement that output the error to the user in order to have the IDE identify the line that it occurs.

It was found that the following list was not a list

```
542 //try {  
543 List importedProfile = (recievedServerData["message"])[ "profile"];  
  
Exception has occurred. ×  
_TypeError (type 'Null' is not a subtype of type 'List<dynamic>')  
  
[1]: https://www.dartlang.org/tutorials/language/variables#variables
```

Printing the message received from the server resulted in the following being returned

```
{error: false, message: No matching profiles, hash:  
6wOHD0RldvIXqjNSPB5ZouQI//b7k4LPQMTq+6OLU/GAxgJ9ZM85nREGf3qRSknLuo863  
qLpYnBaDpHv9lCYK2tKX+QzsQCGf6SWLqXa8KmZHd9MuEmOi8EK1aOKa+nr, profile:  
null, exist: false, exists: false}
```

Looking at the register.php, the problem was easily identified

```
        }
    else{
        $return["error"] = false;
        $return["message"] = "Email or password incorrect";
        $return["exists"] = false;
        $return["profile"] = null;
    }
} else{
    $return["error"] = false;
    $return["message"] = "No matching profiles";
    $return["exists"] = false;
    $return["profile"] = null;
}
} else{
    $return["error"] = true;
    $return["message"] = "Profile does not exist";
}
```

The file was returning the error status as false when it should be true. As well, the error is actually caused by multiple profiles existing under that profile causing that error. The problem with the check done is that it only checks for if there is only 1 profile under that email which if there is 0 or multiple, returns "No matching profiles", implying that there is no profiles at all. To fix this, a check is done to see if there is more than one and returns an alternate statement.

The root source of the problem is not this though since it is a last chance error used to protect the app from crashing. The cause comes from the search done before it is inserted into the database

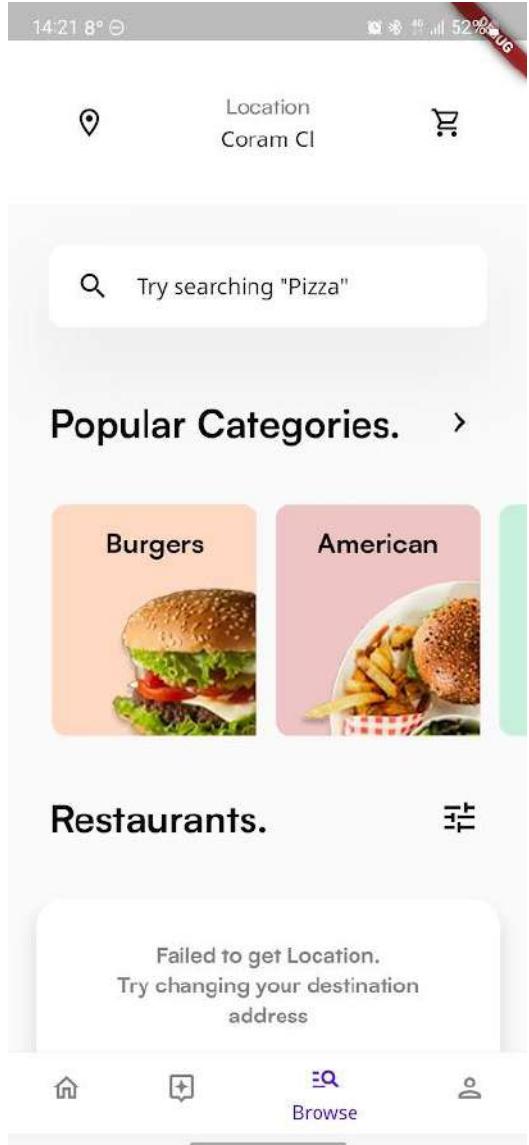
```
$sqlverify = "SELECT email FROM profile WHERE Email = '$email'";
$res = mysqli_query($link, $sqlverify);
if (mysqli_num_rows($result) > 0) {
    $return["exist"] = true;
}
else{
    $sql = "INSERT INTO profile SET
        FirstName = '$firstname'
```

Looking at line 3, we see that it is querying for \$result not \$res so it falls back to the else statement. What it should be is a statement checking if there are no profiles under the email and then performing the insert statement, with an else statement to catch any other issues.

With this fixed, I was forced to clear all profiles from the database.

Issue 2

When trying to get the location of the device, it would not save the automatic longitude and latitude locations if the person does not move the map. This is a problem since the user may see that the location is the place selected but it would cause an error, saying that it failed to get the location



For the user, they have no idea what is going on until they set the location again since the name of the location is saved but not the latitude and longitude (which isn't shown).

To fix this, I first decided to make it so that the location was being set to the map properly so I added the line

```
onMapCreated: (GoogleMapController controller) {  
    _controller.complete(controller);  
    controller.animateCamera(CameraUpdate.newCameraPosition(  
        CameraPosition(
```

```
        target: LatLng(savedPosition[0], savedPosition[1]),  
        zoom: 18,  
    ),  
);  
},
```

This had the added benefit of adding an animation zooming in slightly into the map similar to a magnifying glass

I then found the variables used to save the latitude and longitude values to see if the camera location was being saved in the right location

```
e11d6b[MainActivity](29004): ViewPostIME pointer 1  
): HERE CameraPosition(bearing: 0.0, target: LatLng(51.75256014979541, -0.5639424920082092), tilt: 0.0, zoom: 18.0).target
```

It seems that it was failing to save the original location because it was formatted incorrectly since when I did a ".toDouble()" on it, it saved correctly and displayed the location on the map correctly, with the restaurants displaying in the correct location.

White Box Testing

Since I have the knowledge of how the app works, I am able to break it a lot easier but many of these issues are extremely hard to fix

Issue 1

When the user changes networks or disconnects from the app, it will either crash or infinitely load. This is caused by a change in network IP address which means the connection to the server changes. During network switches while grabbing or loading data, it is not able to keep the connection. The way I would fix this in the future would be to have a unique identifiable ID which is saved on the server and the device when a query is made. Once the user reconnects to the internet, the device would require the server using this id and the server would attempt to provide the data again.

For more information, read the article below:

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-cifs/39a29276-cadf-41d3-b5f1-74facea48607

Issue 2

This one is unlikely but possible at an extremely large scale; If a user orders an item at the same time from the same profile, the database has a chance of completing an order in a staggered fashion. How the order.php file works is by inserting the table then running a query to get the primary key for that order by using the profile id and longitude and latitude. If this is the same, a duplicate order is made. There is a higher chance though that the device makes the order twice back to back and the server processes them in a stagger, using round robin. The way I could fix it is to use table locking and only unlock the table once the whole order is complete. With how it is currently set up, it breaks the ACID rule by having a chance of not completing all at once.

Issue 3

Using HTTP is easier to perform than HTTPS but it requires certificates and updates to be sustained and it is not supported well with Flutter if you are not using Firebase. With my app, it uses HTTP for data transfers. With the amount of management, I was unfortunately unable to do HTTPS but the password is encrypted. One bad thing is that it uses a static key instead of a dynamic key. I tried to do dynamic keys but was unable to get it working with php since it uses a different codebase.

The issue with HTTP compared to HTTPS is that it isn't encrypted, allowing for data interception where it can be read and tampered with.

This issue can cause issues in relation to the database returning data when a malicious user inputs data into a web-address. This can be relatively easy to do if the user knows the URL and is able to guess the user id which is incremental so can be guessed easily.

Stakeholder Judgement

Group 1: Experienced Customer - Sam R

"The app is quite impressive. I really enjoy the dark mode which is a very unique experience unlike Just Eat and Deliveroo. I very often order in the evening and hate how bright the app is on my eyes but with this, it looks nice.

The restaurant browse section looks good but it would have been nice if it had a way to choose other sort options other than distance but I understand that you did this all by yourself.

You should be proud of yourself, there is no way I could have made anything close to this."

Group 2: Customer with No Experience - Stevie T

"Your food delivery app is quite unique. I love the minimal design you have put into the app. I couldn't figure out how to clear my cart though which is annoying. I wish there was a way to remove some food from the cart from the cart because I accidentally added food from multiple restaurants to the cart. I didn't know there were things in the cart and now it says that 'multiple restaurant delivery isn't available'. Other than that I think the concept of having multiple profiles is nice.

Also I think it would be nice if I could do profiles without putting in my details because I don't know how much I trust the app to keep my information secure"

Group 3: Customer With Dietary Needs - Scott P

"Your app isn't really what I wanted but I guess it is fine. I was really hoping it could show me which foods I can't eat since other foods still haven't added those features. It is so annoying that they still haven't added that feature because it feels like people like me have been ignored and pushed aside to get the information ourselves.

Other than that, the app itself works as it should and I could not find any issues."

Group 4: Restaurant Manager - Paddy R

After trying to find Paddy, he was not working at the restaurant anymore so I was unable to ask him any post-development questions

Evaluation of Maintenance

The use of an iterative methodology impacted the maintainability of the app, as each prototype built upon the previous one, leading to less efficient coding. For example, the All Eat app became bulkier with each added feature, particularly in the cart and item pages. Improving the app's maintenance could be achieved by using objects and classes, each with its own functions and methods for managing data. If the app were to be remade, inheritance could be used to create a parent-child relationship between restaurants and items, making it easier to customize data. The code is well-documented with comments to explain complex data manipulation. Variable names are descriptive and use camel casing to distinguish between classes and variables. The emphasis with the code was to make it easy to understand, even with the use of indexes within indexes and dictionaries since maintenance should be the highest priority for an ongoing project.

Limitations

There are many limitations with the All Eat app, primarily caused by the limited time frame of the app and the high expectations I could do in a year. These include:

- The lack of security: With my limited knowledge, I was aware my app should be safe but I was not able to work out how to use HTTPS with Flutter. It did not help that Flutter was made in 2017 and is still getting major improvements to the language. The documentation of Flutter is improving all the time with even my app going through 2 major changes which forced me to update my previous code. The app does not run everything through an encryption layer meaning that people can intercept the data.
- Only one sort method: With the amount of data needed to make the "recommended" sort method, I was unable to make it. To do that, I would have needed to collect the order information, get the user rating of both the restaurant data and item data, get the user touch patterns and process them with other users information to create patterns

and recommend the correct items and restaurants. To make things worse, I would need to get a more extensive list of restaurants and get import their items due to the fact that I only have 2 restaurants available to order from. The remaining restaurants I have only contain the basic restaurant information without the items and their customise options.

- No allergy selection: The allergy section was also caused by lack of time since it would have been quite challenging to implement. Not only would the ingredient list have to be imported but a separate searchable menu would need to be made to get the user's p1838 allergies and then for each item shown, it would have to run checks to see if it would fit under the user's allergies. The other problem is also recommending the correct items since the recommendation system would have to be modified to also take allergies into account and recommend foods without that.
- No changeable profile information: Without changeable profile information, if a user changes their email address or wants to change their password, they are not able to. This is important since most people forget their passwords frequently.
- No social logins: It would have been great to include the social logins page to make logins quicker but to do that, I would likely need to use Firebase which would mean I would have to remake the whole profile system. The main reason I didn't make the app using firebase was because it doesn't support multi-user login system where multiple people could be logged in at the same time and integrate together for things like the orders section where multiple profiles create an order together. With the lack of this feature, I was forced to make a bespoke system.
- No forgot password system: The forgot password system would be made with the change password system in the profile settings page where a one-time code would be sent to the user, and they can use that to change their password.
- No customised homepage: The homepage was designed to be the main place people go to quickly get to the correct part of the app, with shortcuts to favourite restaurants, the most recent order and the search bar. This would require all the features mentioned to be already made so that it could query them.

Summary & Conclusion

The All Eat app is a food delivery app developed between February 2022 and February 2023. While originally planned to be made for people with allergies, the final project ended up being a multi-profile food delivery app.

Prototype 1A

The creation of prototype 1A was my first experience building a large database and using Flutter. Although unknown to me at the time, I was building the database in the wrong language, using SQLite; this error was in part the cause of me doing research of how databases work on mobile. I realised later after making it that my server was not stored locally and that it only needs to use SQL on the server.

Starting work on my app, I began by making the login system, first started making it using Firebase but I was unsuccessful in getting it working. What I had not found out then but found out later was that it was correct to not use it since for the purpose I was using the app for, multiprofile interactions and although it had lots of integrations with database, it would not well together with SQLite. During this time, I learnt how to use encryption and learning about AES 256. I also started work on the server building the profile table and the first login and register files. What I could not work out was how to use dynamic keys to encrypt the data so by the end of 2 weeks, I gave up and decided to go with a static key. The biggest problem was the amount of extra code needed to encrypt the data, so I decided to only encrypt the password. When making the login and registration system, I worked on the local database, where the profile is stored for when it is logged in. With the login and profile creation pages made, I added the feature which allowed for profile switching and profile removal.

Continuing work on prototype 1A, I made the location selector. I made the button act as an object where it could be called with anything, and it will act as its own. While doing this, I learnt how to use the device's sensors to get the location. By asking the user to accept using the device's GPS, the longitude and latitude of the device can be found and by using the geolocator package, it can be translator, it can be converted into an address. I also worked on the restaurant part of the app, querying the server for the table of information about each restaurant and displaying them. During this, I learned how to display images and how to send back large amounts of data.

Prototype 1B

Prototype 1B was all about redesigning the user interface and the code itself to be better. With the number of iterations needed to complete the final app, it was important to have a good understanding of what the app should look like so that some of the code can be reused in other parts of the app. I used Figma to develop the concept art of the app, using a variety of other food delivery apps like Uber Eats and Just Eat as inspiration. With such a text dense app, it was

important that it was easy to understand. How these apps were very interesting; they used icons when possible and tried to split content into multiple pages in order to increase whitespace. While making the concept art, I tried to stick with a simplified view, using blocks since I was aware that over-complication not only makes it harder to develop but also harder to interact with. I did some research using the Google Developer guide to building a good-looking app where I learnt how to use the built-in blocks designed for Flutter.

After making the concept art, I started the development for the app from scratch, using prototype 1A as inspiration since the functions had to be re-wrote in order to reduce the amount of repeated code. While it would have been nice if I kept it up after this, building the app was difficult and sometimes meant compromising between the app looking how the concept art looked and having it optimised. The problem with the redesign was that many of the features included in prototype 1A was not included due to it being in a different position and it not working with the redesigned app. If I were to redesign the app again, I would have stuck more to the Android look since with the look of the app, it is very weirdly done like the top navigation bar just being a top widget on the home screen.

Prototype 2

In prototype 2, the whole aim was to get the user from the main page to the cart. In prototype 1B, I was able to get the item page but it was mostly empty, only having the data forwarded from the main restaurant page.

To start with though, I worked on making the user interface for getting the delivery restaurant look better, using a map instead of a text-based destination. The main reason for doing this was to get the exact latitude and longitude for the destination so that the restaurants can be sorted by distance from the destination since getting the location from text can result in an unknown location to deliver to.

NOTE: Although not known at the time, I now know that the latitude and longitude can be found by using the Google Maps API with the address validation feature.

While making the location selection, I learnt how to make a map and use the google console, using an API to get details. Since people may have locations that cannot be defined by a 2D map, like multi-story apartments, I was forced to add a location changer so that once the user changes their location on the map, they can further refine the destination by dragging the bottom card up to reveal the list of text fields done in prototype 1b. This was disappointing to

see since I wanted to reduce the space but was the only way I could figure out that would allow for this much refinement.

With the location selection complete, the best thing to do was to use this location practically so I decided to do the filter and sort methods. This was relatively hard on the server-side, coming up with how to get multi-query methods working and learning how to merge methods together in SQL. For the sorting by distance, this was one of the trickiest things, learning how to determine how to get distances on a sphere. On the client side, there was no need to update anything since it displays it in the order that the server sends it. For the final part of the prototype, I made the customise item page. For this, I originally tried to have a separate file for each customise type, I was not able to get them working together because they were not inheriting and syncing changes with the main item file so I was forced to have them all in one file. This was not ideal since there was lots of item management and checks done, making the final file over 1000 lines long.

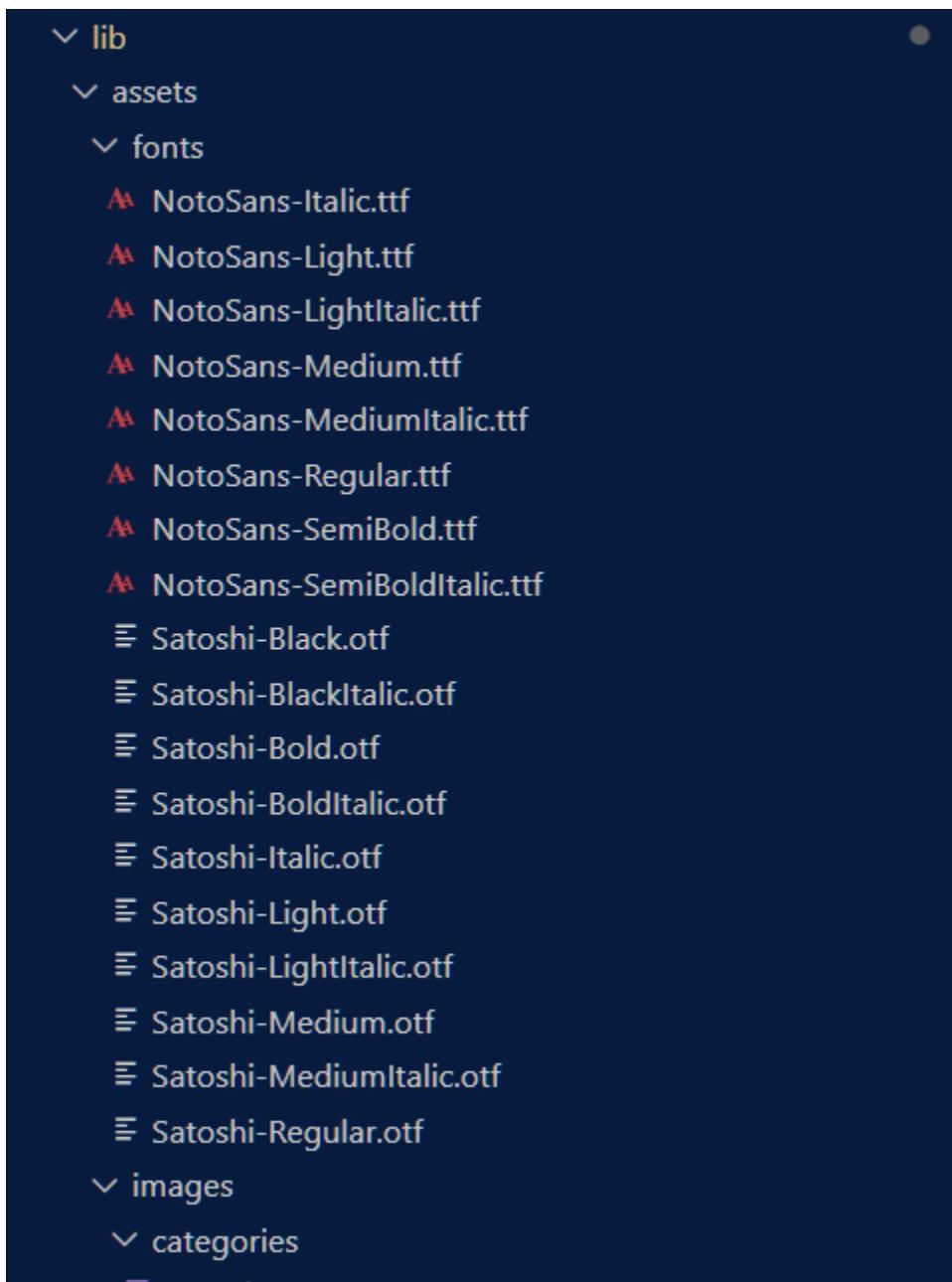
Prototype 3

Prototype 3 was the final developed prototype. In this prototype I worked on the cart and checkout system. This was quite tricky mainly because of the vast amount of data that needed to be managed. If I were to make this again, I would have built the apps using classes and objects. With the current app, it uses arrays, tables, and lists to display and manage each restaurant with their apps and as a result it makes it harder to query, insert, update and delete data, using indexes instead of calling individual keys. As well, a big reason I could not complete development with the orders being added was because I was using a trial of the map API which had a 6-month trial and with the trial coming to an end, it is important that the remaining documentation and errors are found before the app breaks

Final Code

Client App File Library

Structure



-  american-category.png
-  breakfast-category.png
-  burger-category.png
-  chinese-category.png
-  coffee-category.png
-  greek-category.png
-  sushi-category.png
-  thai-category.png
- ⌄ logo
 -  foregroundicon.png
 -  icon.png
 -  logodark.png
 -  splashscreen-android12-dark.png
 -  splashscreen-android12.png
- ⌄ screens
 - ⌄ existingsetup
 -  existingsetupscreenfood.jpg
 - ⌄ profilesetup
 -  login-illustration-dark.png
 -  login-illustration-light.png
 - ⌄ welcomescreen
 -  welcomescreenfood.jpg

```
✓ screens
  ✓ checkout
    cart.dart
    checkout.dart
  ✓ navigationscreens
    browse.dart
    foryou.dart
    homepage.dart
    profiles.dart
  ✓ profilesetup
    profilesetup_create.dart
    profilesetup_existing.dart
    profilesetup_login.dart
    welcomescreen.dart
  ✓ restaurant
    restaurant_customise.dart
    restaurant_main.dart
    filtersort.dart
    locationselection.dart
    setupverification.dart
  ✓ services
    cart_service.dart
    dataencryption.dart
    localprofiles_service.dart
    queryserver.dart
    setselected.dart
```

```
    ▼ theme
        ⚒ theme.dart
    ▼ widgets
        ▼ elements
            ⚒ browse_categories.dart
            ⚒ elements.dart
            ⚒ profiles_buttons.dart
            ⚒ profiles_selection.dart
            ⚒ search.dart
            ⚒ genericlocading.dart
            ⚒ navigationbar.dart
            ⚒ restaurants_list.dart
            ⚒ topbar.dart
        ⚒ main.dart
```

Files

main.dart

```
import 'package:alleat/screens/setupverification.dart';

import 'package:flutter/services.dart';

import 'package:alleat/theme/theme.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:alleat/theme/theme.dart' as globals;

void main() {

    //Start App

    runApp(
```

```
const MyApp(),  
);  
}  
  
class MyApp extends StatelessWidget {  
  
  const MyApp({super.key});  
  
  Future getTheme() async {  
  
    //Get theme preferences from theme.dart  
  
    final prefs = await SharedPreferences.getInstance(); //Load shared  
preferences as new instance  
  
    final int? appTheme = prefs.getInt('appTheme');  
  
    globals.appthemepreference = appTheme; //Save theme under app theme within a  
global instance  
  
  }  
  
  @override  
  
  Widget build(BuildContext context) {  
  
    SystemChrome.setPreferredOrientations([  
  
      //Force portrait mode  
      DeviceOrientation.portraitUp,  
  
      DeviceOrientation.portraitDown,  
    ]);  
  
    WidgetsFlutterBinding.ensureInitialized();  
  }  
}
```

```
getTheme();

switch (globals.appthemepreference) {

  case 1:

    return MaterialApp(
      //Create main app

      title: 'AllEat.',

      themeMode: ThemeMode.light,

      theme: ThemeClass.lightTheme,

      home: const SetupWrapper(), //SetupWrapper(), //Check if setup is
      complete for app (reference)

    );

  case 2:

    return MaterialApp(
      //Create main app

      title: 'AllEat.',

      themeMode: ThemeMode.dark,

      theme: ThemeClass.darkTheme,

      home: const SetupWrapper(), //SetupWrapper(), //Check if setup is
      complete for app (reference)

    );

}

return MaterialApp(
  //Create main app
```

```
        title: 'AllEat.',

        themeMode: ThemeMode.system,

        theme: ThemeClass.lightTheme,

        darkTheme: ThemeClass.darkTheme,

        home: const SetupWrapper(), //SetupWrapper(), //Check if setup is complete
for app (reference)

);

}

}
```

cart.dart

```
import 'package:alleat/screens/checkout/checkout.dart';

import 'package:alleat/services/cart_service.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/queryserver.dart';

import 'package:alleat/widgets/elements/elements.dart';

import 'package:flutter/material.dart';

import 'dart:convert';

class Cart extends StatefulWidget {

const Cart({super.key});
```

```
@override

State<Cart> createState() => _CartState();

}

class _CartState extends State<Cart> {

Future<Map> getCart() async {

    //Get cart data including restaurant info, profile info, item data, customise
titles for each item and the customised options performed on the item

    Map returnCartInfo = {"error": false, "message": "", "cartinfo": []}; //Set
returned cart info to be empty and there to be no errors

    List availableProfiles =

        [];//Each profile that is in the cart is stored in available profiles,
storing their basic information including profileid, firstname, lastname, profile
icon colour

    List cartProfileIDs = await SQLiteCartItems.getProfilesInCart(); //Get
profile ids that are in the cart

    List profileInfo = await SQLiteLocalProfiles.getProfiles(); //Get all info
stored for each profile within the local profiles database

    //Get profiles

    for (int i = 0; i < cartProfileIDs.length; i++) {

        //For each profile id that is in the available profiles

        for (int j = 0; j < profileInfo.length; j++) {

            //For each profile in the list of total profiles
```

```

        if (cartProfileIDs[i] == profileInfo[j]["profileid"]) {

            //If the id of the current available profile in the cart is equal to
            the currently searched profile, add its details to availableprofiles

            availableProfiles.add([
                profileInfo[j]["profileid"],
                profileInfo[j]["firstname"],
                profileInfo[j]["lastname"],
                profileInfo[j]["profilecolorred"],
                profileInfo[j]["profilecolorgreen"],
                profileInfo[j]["profilecolorblue"]
            ]);

        }
    }

}

for (int iProfile = 0; iProfile < availableProfiles.length; iProfile++) {

    //For each available profile

    Map tempItemInfo = {};

    List profileCart = await SQLiteCartItems.getProfileCart(
        availableProfiles[iProfile][0]); //Get list of items for profile.
    Returns in format {cartid, itemid, customised, quantity}

    for (int i = 0; i < profileCart.length; i++) {

        // For each item, add it as an index i
}

```

```

tempItemInfo[profileCart[i]["itemid"]] = [[], {}];

Map basicItemInfo = await
QueryServer.query("https://alleat.cpur.net/query/cartiteminfo.php", {

    "type": "item",

    "term": profileCart[i]["itemid"].toString()

}); //returns item id, item name, price, item image, restaurant id,
restaurant name, delivery price

if (basicItemInfo["error"] == true) {

    returnCartInfo["error"] = true;

    returnCartInfo["message"] = basicItemInfo["message"];

    return returnCartInfo;

} else {

    tempItemInfo[profileCart[i]["itemid"]][0].addAll([
        basicItemInfo["message"]["message"][1], //Item name
        basicItemInfo["message"]["message"][3], //Item image link
        basicItemInfo["message"]["message"][2], //Item price
        basicItemInfo["message"]["message"][5], //Restaurant name
        basicItemInfo["message"]["message"][4], //Restaurant id
        basicItemInfo["message"]["message"][6], //Delivery price
        profileCart[i]["quantity"], //Item quantity
        profileCart[i]["cartid"], //Cart ID
    ]);
}

```

```

        basicItemInfo["message"]["message"][7], //Minimum order price for
restaurant

        basicItemInfo["message"]["message"][8], //Restaurant address

        basicItemInfo["message"]["message"][9], //Restaurant location
latitude

        basicItemInfo["message"]["message"][10], //Restaurant location
longitude

        basicItemInfo["message"]["message"][0] //Item ID
    ]);

    Map customised = json.decode(profileCart[i]["customised"]);

    List customisedtitleids = customised.keys.toList(); //Each customise
title is stored here

    List customisedOptions = []; //Each unique option stored here

    Map updatedCustomised = {};//Data that will be sent back in the
iteminfo key, correctly formatted

    for (int i = 0; i < customisedtitleids.length; i++) {

        //For each customise title

        updatedCustomised[customisedtitleids[i]] = [[], []]; //Add customised
key to new Map

        for (int j = 0; j < customised[customisedtitleids[i]].length; j++) {

            //For each item in list of customised options for customise title

            if
(!customisedOptions.contains(customised[customisedtitleids[i]][j])) {

                //If the option is not in the list of options, add it to the list

```

```

updatedCustomised[customisedtitleids[i]][1].add([customised[customisedtitleids[i]][j], 1]);

        customisedOptions.add(customised[customisedtitleids[i]][j]);

    } else {

        for (int k = 0; k <
updatedCustomised[customisedtitleids[i]][1].length; k++) {

            //If there is more than one of an option add it to the quantity
eg. (192, 162, 192) = [192, 2]

            if (updatedCustomised[customisedtitleids[i]][1][k][0] ==
customised[customisedtitleids[i]][j]) {

                updatedCustomised[customisedtitleids[i]][1][k][1] += 1;

            }

        }

    }

}

try {

    if (customisedtitleids.isNotEmpty) {

        //If there is at least one customisable title get the details of
all the customise titles using the list of titles that have been gotten from the
keys of the map (each key is the title id)

        Map customisedTitlesInfo = await QueryServer.query(
            "https://alleat.cpur.net/query/cartiteminfo.php", {"type": "title", "term": json.encode(customisedtitleids)} );

```

```

    if (customisedTitlesInfo["error"] == true) {

        //If there is an error, return error=true with the message
        returned from the server and end future

        returnCartInfo["error"] = true;

        returnCartInfo["message"] = customisedTitlesInfo["message"];

        return returnCartInfo;

    } else {

        List customisedTitlesInfoFormatted =
        customisedTitlesInfo["message"]["message"];

        if (customisedOptions.isNotEmpty) {

            Map customisedOptionInfo = await QueryServer.query(
                "https://alleat.cpur.net/query/cartiteminfo.php", {"type": "option", "term": customisedOptions.toString()});

            if (customisedOptionInfo["error"] == true) {

                //If there is an error, return error=true with the message
                returned from the server and end future

                returnCartInfo["error"] = true;

                returnCartInfo["message"] = customisedOptionInfo["message"];

                return returnCartInfo;

            } else {

                List customisedOptionInfoFormatted =
                customisedOptionInfo["message"]["message"];
            }
        }
    }
}

```

```

        for (int i = 0; i < customisedtitleids.length; i++) {

            // For each customise title

            for (int j = 0; j < customisedTitlesInfoFormatted.length;
j++) {

                //For each item in list of returned from server info
about each title

                if (customisedTitlesInfoFormatted[j][0] ==
customisedtitleids[i]) {

                    // If it has the id of the title in the first index,
add the info to the composite info

                    updatedCustomised[customisedtitleids[i]][0]

                        .addAll([customisedTitlesInfoFormatted[j][1],
customisedTitlesInfoFormatted[j][2]]);

                }

            }

            for (int j = 0; j <
updatedCustomised[customisedtitleids[i]][1].length; j++) {

                //For each option in the list of options for each title

                for (int k = 0; k < customisedOptionInfoFormatted.length;
k++) {

                    //For each item in list of returned from server about
option info

                    if (customisedOptionInfoFormatted[k][0] ==
updatedCustomised[customisedtitleids[i]][1][j][0]) {

                        // If it has the id of the option in the first index,
add the info to the list

                        updatedCustomised[customisedtitleids[i]][1][j]

```

```
        .addAll([customisedOptionInfoFormatted[k][1],  
customisedOptionInfoFormatted[k][2]]);  
  
    }  
  
}  
  
}  
  
}  
  
tempItemInfo[profileCart[i]["itemid"]][1] =  
updatedCustomised;  
  
}  
  
}  
  
}  
  
}  
  
}  
  
}  
  
} catch (e) {  
  
    returnCartInfo["error"] = true;  
  
    returnCartInfo["message"] = "An error occurred while attempting to  
process the item information.";  
  
}  
  
}  
  
  
  
// Update item price to be the new customised price  
  
List customiseOptionsKeys =  
tempItemInfo[profileCart[i]["itemid"]][1].keys.toList();  
  
for (int iCust = 0; iCust < customiseOptionsKeys.length; iCust++) {  
  
    //For each customise title
```

```

        if
(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
"SELECT" ||

tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
"ADD") {

    //If it is either a selection or an add function add it to the item
price (multiplied by the quantity)

    for (int iCustOptions = 0;

        iCustOptions <
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1].length;

        iCustOptions++) {

        tempItemInfo[profileCart[i]["itemid"]][0][2] =
((double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2])) * 100 +

double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]]
][1][iCustOptions][3]) *

100 *

tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1][iCustOptions][1]) /

100)

        .toString();

    }

} else if
(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][0][1] ==
"REMOVE") {

```

```

        //If it is a remove function, remove from the item price (multiplied
by the quantity)

        for (int iCustOptions = 0;

                iCustOptions <
tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1].length;

                iCustOptions++) {

            tempItemInfo[profileCart[i]["itemid"]][0][2] =
((double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2]) * 100 -

double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]]
][1][iCustOptions][3]) *

                    100 *

tempItemInfo[profileCart[i]["itemid"]][1][customiseOptionsKeys[iCust]][1][iCustOptions][1]) /

                    100)

            .toString();

        }

    }

}

//Multiply price of each item by the quantity of the item

tempItemInfo[profileCart[i]["itemid"]][0][2] =

(double.parseDouble(tempItemInfo[profileCart[i]["itemid"]][0][2]) *

tempItemInfo[profileCart[i]["itemid"]][0][6]).toString();

}

```

```
returnCartInfo["cartinfo"].add([
    availableProfiles[iProfile][0],
    availableProfiles[iProfile][1],
    availableProfiles[iProfile][2],
    availableProfiles[iProfile][3],
    availableProfiles[iProfile][4],
    availableProfiles[iProfile][5],
    tempItemInfo
]); //Return the item information of profile with the profile information
}

return returnCartInfo;
}

//Remove Item from cart in local profiles table

Future<bool> removeItem(cartID) async {
try {
    await SQLiteCartItems.removeItem(cartID);
    return true;
} catch (e) {
    return false;
}
}
```

```
@override

Widget build(BuildContext context) {
  return Scaffold(
    body: SingleChildScrollView(
      child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
        const ScreenBackButton(),
        FutureBuilder<Map>(
          future: getCart(),
          builder: ((context, snapshot) {
            if (snapshot.hasData) {
              List cartInfo = [snapshot.data ?? []];
              if (cartInfo[0]["error"] == true) {
                //If there is an error getting the cart info display error box
              }
            }
            return Padding(
              padding: const EdgeInsets.only(left: 20, right: 20, top: 10,
              bottom: 10),
              child: Container(
                decoration: BoxDecoration(
                  borderRadius: const BorderRadius.all(Radius.circular(20)),
                  color: Theme.of(context).colorScheme.onSurface,
                  boxShadow: [

```

```
        BoxShadow(  
          color: Colors.black.withOpacity(0.1),  
          spreadRadius: 2,  
          blurRadius: 10,  
          offset: const Offset(0, 10), // changes position  
          of shadow  
        ),  
      )),  
    child: Column(children: [  
      Padding(  
        padding: const EdgeInsets.all(30),  
        child: SizedBox(  
          width: double.infinity,  
          child: Center(  
            child: Column(children: [  
              const Text(  
                "An error has occurred.",  
                textAlign: TextAlign.center,  
              ),  
              const SizedBox(  
                height: 20,  
              ),  
              Text(  
                "An error has occurred.",  
                textAlign: TextAlign.center,  
              ),  
            ]),  
          ),  
        ),  
      ),  
    ]),  
  ),  
);
```

```

        cartInfo[0]["message"],

        textAlign: TextAlign.center,
    ),
    ]))),,
)

] )));

} else {

if (cartInfo[0]["cartinfo"].isNotEmpty) {

//If there are profiles in the cart

return ListView.builder(
    physics: const NeverScrollableScrollPhysics(), //Dont allow
scrolling (Done by main page)

    scrollDirection: Axis.vertical,
    shrinkWrap: true,
    itemCount: cartInfo[0]["cartinfo"].length + 1, //For each
profile (add one for the total price)

    itemBuilder: (context, indexProfile) {

if (indexProfile != cartInfo[0]["cartinfo"].length) {

    List currentProfile =
cartInfo[0]["cartinfo"][[indexProfile]];

    List itemIDs = currentProfile[6].keys.toList();

    return Column(children: [
Container(
    //Contain the profile within a container

```

```
        margin: const EdgeInsets.symmetric(vertical: 10,  
horizontal: 20),  
  
        padding: const EdgeInsets.symmetric(horizontal:  
20, vertical: 10),  
  
        child: Row(  
  
            children: [  
  
                Container(  
  
                    // Create profile circle with first and  
last letter  
  
                    width: 50,  
  
                    height: 50,  
  
                    decoration: BoxDecoration(  
  
                        shape: BoxShape.circle,  
  
                        color:  
Color.fromRGBO(currentProfile[3], currentProfile[4], currentProfile[5], 1)),  
  
                    child: Align(  
  
                        alignment: Alignment.center,  
  
                        child:  
Text('${currentProfile[1][0]}${currentProfile[2][0]}',  
  
                            style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).backgroundColor))),  
  
                    const SizedBox(width: 20), //Profile  
firstname and lastname  
  
                Text(  
                    style:  
Theme.of(context).textTheme.bodyText1?.copyWith(color:  
Theme.of(context).colorScheme.onBackground))  
                ),  
            ],  
        ),  
    ),  
);
```

```
        "${currentProfile[1]}  
${currentProfile[2]}",  
  
        style:  
Theme.of(context).textTheme.headline5?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
        overflow: TextOverflow.fade,  
  
    )  
  
],  
  
)),  
  
Divider(  
  
thickness: 2,  
  
color:  
Theme.of(context).textTheme.headline6?.color?.withOpacity(0.5),  
  
indent: 40,  
  
endIndent: 40,  
  
),  
  
const SizedBox(  
  
height: 10,  
  
),  
  
ListView.builder(  
  
//Create a container for each profile containing  
the customised options and a summary of the item  
  
physics: const NeverScrollableScrollPhysics(),  
//Dont allow scrolling (Done by main page)  
  
scrollDirection: Axis.vertical,
```

```

        shrinkWrap: true,

        itemCount: itemIDs.length, //For each item

        itemBuilder: (context, indexItem) {

            List currentItem =
currentProfile[6][itemIDs[indexItem]];

            List itemCustomiseIDs =
currentItem[1].keys.toList();

            return Padding(
                padding: const
EdgeInsets.symmetric(vertical: 5, horizontal: 10),
                child: Dismissible(
                    //Create a slide to delete container
(dismissible)

                    key: Key(currentItem[0][7].toString()),

//Key is the cart id

                    onDismissed: (direction) async {

                        bool hasDeleted = await
removeItem(currentItem[0][7]); //Remove item from local cart database

                        if (hasDeleted) {

                            //If it deleted

cartInfo[0]["cartinfo"][indexProfile][6].remove(itemIDs[indexItem]); //removing
the item from the list

                        setState(() {

                            ScaffoldMessenger.of(context)

```

```
        .showSnackBar(const  
SnackBar(content: Text("Successfully deleted item."));  
  
    });  
  
} else {  
  
    setState(() {  
  
        ScaffoldMessenger.of(context)  
  
            .showSnackBar(const  
SnackBar(content: Text("Failed to deleted item. Please reopen the cart")));  
  
    });  
  
}  
  
},  
  
background: Container(  
  
    color:  
Theme.of(context).colorScheme.error,  
  
    child: Row(  
  
        mainAxisAlignment:  
MainAxisAlignment.spaceBetween,  
  
        children: [  
  
            Padding(  
  
                padding: const  
EdgeInsets.symmetric(horizontal: 30),  
  
                child: Icon(  
  
                    Icons.delete,  
  
                    color:  
Theme.of(context).colorScheme.onSurface,
```

```
        )),

        Padding(
            padding: const
EdgeInsets.symmetric(horizontal: 30),

            child: Icon(
Icons.delete,
color:
Theme.of(context).colorScheme.onSurface,
))

        ],
),
),
child: Container(
width: double.infinity,
decoration: BoxDecoration(
borderRadius: const
BorderRadius.all(Radius.circular(10)),
color:
Theme.of(context).colorScheme.onSurface),
padding: const
EdgeInsets.symmetric(vertical: 10, horizontal: 10),
child: Column(children: [
Row(
children: [
Container(

```

```
        width: 80,  
  
        height: 80,  
  
        decoration: BoxDecoration(  
  
          borderRadius: const  
BorderRadius.all(Radius.circular(5)),  
  
          image:  
  
DecorationImage(fit: BoxFit.cover, image:  
NetworkImage(currentItem[0][1].toString()))),  
  
(  
          ),  
  
          const SizedBox(  
  
            width: 20,  
  
(  
          ),  
  
          Container(  
  
            alignment:  
Alignment.center,  
  
            height: 30,  
  
            width: 30,  
  
            decoration:  
BoxDecoration(  
  
              color:  
Theme.of(context).primaryColor, borderRadius: BorderRadius.circular(30)),  
  
              child: Text(  
  
currentItem[0][6].toString(),
```

```
        style:  
Theme.of(context)  
  
            .textTheme  
  
            .headline6  
  
            ?.copyWith(color:  
Theme.of(context).colorScheme.onSurface),  
  
        )),  
  
        const SizedBox(  
  
            width: 20,  
  
,  
  
        Expanded(  
  
            child: Column(  
  
            crossAxisAlignment:  
CrossAxisAlignment.start,  
  
            children: [  
  
                Text(  
  
                    currentItem[0][0],  
  
                    style:  
Theme.of(context)  
  
                        .textTheme  
  
                        .headline6  
  
                        ?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
            ),
```

```

        const SizedBox(
            height: 5,
        ),
        Text(
            currentItem[0][3],
            style:
                Theme.of(context)
                    .textTheme
                    .bodyText1
                    ?.copyWith(color:
                        Theme.of(context).textTheme.headline6?.color),
        )
    ],
    )),
    ],
),
),
ListView.builder(
    //Create a container with the
    title of the option that has been customised with a list of options changed
    physics: const
NeverScrollableScrollPhysics(), //Dont allow scrolling (Done by main page)
    scrollDirection:
Axis.vertical,
    shrinkWrap: true,

```

```

        itemCount:
itemCustomiseIDs.length, //For each customise title

                itemBuilder: (context,
indexCustomise) {

                    List currentCustomiseTitle
= currentItem[1][itemCustomiseIDs[indexCustomise]];

                    if
(currentCustomiseTitle[0][1] == "SELECT" &&
currentCustomiseTitle[1].toList().length != 0) {

                        //If the customise title
type is select and there is at least one customised option for the title return
the customised options in black text

                    return Container(
                        padding: const
EdgeInsets.symmetric(vertical: 20, horizontal: 20),
                        margin: const
EdgeInsets.symmetric(horizontal: 0, vertical: 10),
                        decoration:
BoxDecoration(
                            color:
Theme.of(context).backgroundColor.withOpacity(0.5),
                            border:
Border.all(
                                color:
Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),
                            borderRadius:
BorderRadius.circular(10)),
                        child: Column(

```

```
mainAxisAlignment: MainAxisAlignment.start,  
  
crossAxisAlignment: CrossAxisAlignment.start,  
            children: [  
  
Text(currentCustomiseTitle[0][0],  
            textAlign:  
TextAlign.start,  
            style:  
Theme.of(context)  
  
.textTheme  
  
.headline6  
  
.copyWith(color: Theme.of(context).textTheme.headline1?.color)),  
            const  
SizedBox(height: 15),  
  
ListView.builder(  
            physics:  
const NeverScrollableScrollPhysics(),  
            shrinkWrap:  
true,  
            itemCount:  
currentCustomiseTitle[1].length, //For each customise option
```

```
itemBuilder: (context, indexOption) {  
    List    return Padding(  
        padding: const EdgeInsets.only(left: 10, bottom: 10),  
        child:  
        Row(children: [  
            CircleAvatar(  
                backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
                radius: 10,  
                child: Text(currentCustomiseOption[1].toString(),  
                    style: Theme.of(context)  
                        .textTheme  
                        .bodyText1  
                        ?.copyWith(color: Theme.of(context).backgroundColor)),  
            ),  
        ],  
    );  
},
```

```

        const

SizedBox(
    width: 20,
),
Expanded(
    child: Text(currentCustomiseOption[2].toString(),
        style: Theme.of(context).textTheme.bodyText1)
    ],
);
}),
]);
} else if
(currentCustomiseTitle[0][1] == "ADD" &&
currentCustomiseTitle[1].toList().length != 0) {
    //If the customise title
    type is add and there is at least one customised option for the title return the
    customised options in green text
    return Container(
        padding: const
        EdgeInsets.symmetric(vertical: 20, horizontal: 20),
        margin: const
        EdgeInsets.symmetric(horizontal: 0, vertical: 10),

```

```
decoration:  
BoxDecoration(  
  
    color:  
Theme.of(context).backgroundColor.withOpacity(0.5),  
  
    border:  
Border.all(  
  
        color:  
Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),  
  
    borderRadius:  
BorderRadius.circular(10)),  
  
    child: Column(  
  
mainAxisAlignment: MainAxisAlignment.start,  
  
crossAxisAlignment: CrossAxisAlignment.start,  
  
        children: [  
  
Text(currentCustomiseTitle[0][0],  
  
        textAlign:  
 TextAlign.start,  
  
        style:  
Theme.of(context)  
  
.textTheme  
  
.headline6  
  
? .copyWith(color: Theme.of(context).textTheme.headline1?.color)),
```

```
        const  
SizedBox(height: 15),  
  
ListView.builder(  
    physics:  
    const NeverScrollableScrollPhysics(),  
    shrinkWrap:  
    true,  
    itemCount:  
    currentCustomiseTitle[1].length, //For each customise option  
  
    itemBuilder: (context, indexOption) {  
        List  
        currentCustomiseOption = currentCustomiseTitle[1][indexOption];  
  
        return  
        Padding(  
  
            padding: const EdgeInsets.only(left: 10, bottom: 10),  
            child:  
            Row(children: [  
  
                CircleAvatar(  
  
                    backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
  
                    radius: 10,  
  
                    child: Text(currentCustomiseOption[1].toString()),  
            ]),  
    ),  
);
```

```
style: Theme.of(context)

.textTheme

.bodyText1

?.copyWith(color: Theme.of(context).backgroundColor)),

),

const

SizedBox(

width: 20,

),


Expanded(


child: Text(currentCustomiseOption[2].toString(),


style: Theme.of(context)

.textTheme

.bodyText1

?.copyWith(color: Theme.of(context).colorScheme.tertiary)))]

),
```

```

    );
}

]),

})};

} else if
(currentCustomiseTitle[0][1] == "REMOVE" &&

currentCustomiseTitle[1].toList().length != 0) {

    //If the customise title
    type is remove and there is at least one customised option for the title return
    the customised options in red text

    return Container(
        padding: const
        EdgeInsets.symmetric(vertical: 20, horizontal: 20),
        margin: const
        EdgeInsets.symmetric(horizontal: 0, vertical: 10),
        decoration:
        BoxDecoration(
            color:
            Theme.of(context).backgroundColor.withOpacity(0.5),
            border:
            Border.all(
                color:
                Theme.of(context).colorScheme.onBackground.withOpacity(0.3), width: 1),
            borderRadius:
            BorderRadius.circular(10)),
        child: Column(

```

```
mainAxisAlignment: MainAxisAlignment.start,  
  
crossAxisAlignment: CrossAxisAlignment.start,  
            children: [  
  
Text(currentCustomiseTitle[0][0],  
            textAlign:  
TextAlign.start,  
            style:  
Theme.of(context)  
  
.textTheme  
  
.headline6  
  
.copyWith(color: Theme.of(context).textTheme.headline1?.color)),  
            const  
SizedBox(height: 15),  
  
ListView.builder(  
            physics:  
const NeverScrollableScrollPhysics(),  
            shrinkWrap:  
true,  
            itemCount:  
currentCustomiseTitle[1].length, //For each customise option
```

```
itemBuilder: (context, indexOption) {  
    List    return Padding(  
        padding: const EdgeInsets.only(left: 10, bottom: 10),  
        child:  
        Row(children: [  
            CircleAvatar(  
                backgroundColor: Theme.of(context).primaryColor.withOpacity(0.5),  
                radius: 10,  
                child: Text(currentCustomiseOption[1].toString(),  
                    style: Theme.of(context)  
                        .textTheme  
                        .bodyText1  
                        ?.copyWith(color: Theme.of(context).backgroundColor)),  
            ),  
        ],  
    );  
},
```

```
const SizedBox(  
  width: 20,  
),  
  
Expanded(  
  
  child: Text(currentCustomiseOption[2].toString(),  
  
  style: Theme.of(context)  
  
.textTheme  
  
.bodyText1  
  
.copyWith(color: Theme.of(context).colorScheme.error)))  
],),  
);  
},  
]);  
});  
} else {  
  return const SizedBox(  
    height: 0,  
  );  
}  
}
```

```
        }),

        Padding(
            padding: const
EdgeInsets.symmetric(vertical: 10, horizontal: 20),

            child: Row(mainAxisAlignment:
MainAxisAlignment.end, children: [
                Text(
                    "Price: ", style: Theme.of(context)
                    .textTheme
                    .headline6
                    ?.copyWith(color:
Theme.of(context).textTheme.headline1?.color),
                ),
                Text(
                    "\u00a3", style:
Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).primaryColor),
                ),
                Text(
                    double.parse(currentItem[0][2]).toStringAsFixed(2),
                    style: Theme.of(context)
                    .textTheme
```

```

        .headline6

        ?.copyWith(color:
Theme.of(context).textTheme.headline1?.color),
    )

    ]))

    ]))));

    })

    ]);

} else {

    return LayoutBuilder(
//Calculate the price for each profile in the botttom
section of the cart

    builder: (p0, p1) {

        double subtotal = 0;

        bool isOneRestaurant = true;

        int tempRestaurantID = -1;

        for (int iProfile = 0; iProfile <
cartInfo[0]["cartinfo"].length; iProfile++) {

            //For each profile in the cart

            cartInfo[0]["cartinfo"][iProfile].add(0.00);
//Add total price to the end of the profile index in the cartInfo

            List itemPriceKeys =
cartInfo[0]["cartinfo"][iProfile][6].keys.toList(); //Create a list of keys for
each item

```

```

                for (int iItem = 0; iItem < itemPriceKeys.length;
iItem++) {

                    //For the first check, the default value is -1
so replace with the first restaurant id. For the remaining items check if it is
equal to the current restaurant id and if it isnt set OneRestaurant to false

                    if (tempRestaurantID == -1) {

                        tempRestaurantID =
int.parse(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][4]);

                    } else {

                        if
(int.parse(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][4]) !=
tempRestaurantID) {

                            isOneRestaurant = false;

                        }

                    }

                }

                //For each item

                cartInfo[0]["cartinfo"][iProfile][7] =
(cartInfo[0]["cartinfo"][iProfile][7] * 100 +

double.parseDouble(cartInfo[0]["cartinfo"][iProfile][6][itemPriceKeys[iItem]][0][2]) *
100) /

                    100; //Add price to the total price for the
profile

            }

        }

    }

}

```

```

        for (int iSubtotalProfile = 0; iSubtotalProfile <
cartInfo[0]["cartinfo"].length; iSubtotalProfile++) {

            subtotal = (subtotal * 100 +
cartInfo[0]["cartinfo"][iSubtotalProfile][7] * 100) /
100; //Add the total price for profile to
subtotal

    }

    return Column(children: [
        const SizedBox(height: 50),
        ListView.builder(
            physics: const
NeverScrollableScrollPhysics(),
            shrinkWrap: true,
            itemCount: cartInfo[0]["cartinfo"].length,
//For each profile

            itemBuilder: (context, iProfilePrice) {
                //For each profile, display the profile
total price

                return Padding(
                    padding: const
EdgeInsets.symmetric(horizontal: 40, vertical: 10),
                    child: Row(
                        mainAxisAlignment:
MainAxisAlignment.end,
                        children: [

```

```

        Text(
            "$${cartInfo[0]["cartinfo"][[iProfilePrice][1]}"
            ${cartInfo[0]["cartinfo"][[iProfilePrice][2]]}: ",
            style: Theme.of(context)
                .textTheme
                    .headline6
                    ?.copyWith(color:
Theme.of(context).textTheme.headline1?.color),
            ),
            const SizedBox(
                width: 50,
            ),
        ),
    ),
    Text("£${cartInfo[0]["cartinfo"][[iProfilePrice][7]].toStringAsFixed(2)}",
        style:
Theme.of(context).textTheme.headline6?.copyWith(fontWeight: FontWeight.w500))
    ],
));
}),
Padding(
    //Display subtotal
    padding: const
EdgeInsets.symmetric(horizontal: 40, vertical: 10),

```

```

        child: Row(mainAxisAlignment:
MainAxisAlignment.end, children: [
            Text("SUBTOTAL", style:
Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).primaryColor)),
            const SizedBox(
                width: 50,
            ),
            Text("£${subtotal.toStringAsFixed(2)}",
                style: Theme.of(context)
                    .textTheme
                    .headline6
                    ?.copyWith(fontWeight:
FontWeight.w500, color: Theme.of(context).primaryColor))
        ])),
        LayoutBuilder(
            //Checkout button with checks for multiple
            conditions
            builder: (p0, p1) {
                if (isOneRestaurant == true) {
                    //If there is only one restaurant being
                    ordered from
                    List itemKeys =
cartInfo[0]["cartinfo"][0][6].keys.toList(); // Get a list of item ids
                    try {

```

```

        if
(double.parseDouble(cartInfo[0]["cartinfo"][0][6][itemKeys[0]][0][8]) <= subtotal) {

            //If the subtotal is more than the
minimum order price for the restaurant return the

            return Padding(
padding: const
EdgeInsets.symmetric(horizontal: 20, vertical: 30),

child: Row(children: [
Expanded(
child: ElevatedButton(
onPressed: () {
Navigator.push(
context,
MaterialPageRoute(
builder:
(context) => Checkout(
cartInfo: cartInfo[0]["cartinfo"],
)));
}),


child: const
Text("Checkout")))
]);
} else {

```

```
// If it isn't more than the minimum  
order price, display the minimum order price in greyed out button  
  
return Padding(  
  
    padding: const  
EdgeInsets.symmetric(horizontal: 20, vertical: 30),  
  
    child: Row(children: [  
  
        Expanded(  
  
            child: Opacity(  
  
                opacity: 0.2,  
  
                child: ElevatedButton(  
  
                    style: ButtonStyle(  
  
                        backgroundColor:  
  
MaterialStatePropertyAll(Theme.of(context).colorScheme.onBackground)),  
  
                        onPressed: null,  
  
                        child: Text(  
  
                            "Minimum order  
£${cartInfo[0]["cartinfo"][0][6][itemKeys[0]][0][8]}",  
  
                            textAlign:  
TextAlign.center,  
  
                        style:  
Theme.of(context)  
  
.textTheme  
.headline6
```

```
? .copyWith(color: Theme.of(context).backgroundColor),  
        ))))  
    ]));  
}  
}  
} catch (e) {  
    // If it fails to check the price, return  
    that there is an error. This fixes when the first item is removed and app tries  
    to check if it is equal during the rebuild  
  
    return Padding(  
        padding: const  
        EdgeInsets.symmetric(horizontal: 20, vertical: 30),  
        child: Row(children: [  
            Expanded(  
                child: ElevatedButton(  
                    style:  
  
                    ButtonStyle(backgroundColor:  
                        MaterialStatePropertyAll(Theme.of(context).colorScheme.error)),  
                    onPressed: null,  
                    child: Text(  
                        "Unknown minimum cart  
price",  
                        TextAlign.center,  
                        style:
```

```
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).backgroundColor),  
))  
]);  
}  
} else {  
//Return that there is more than one  
restaurant in the cart  
return Padding(  
padding: const  
EdgeInsets.symmetric(horizontal: 20, vertical: 30),  
child: Row(children: [  
Expanded(  
child: Opacity(  
opacity: 0.2,  
child: ElevatedButton(  
style: ButtonStyle(  
backgroundColor:  
MaterialStatePropertyAll(Theme.of(context).colorScheme.onBackground)),  
onPressed: null,  
child: Text(  
"Multiple Restaurant  
Delivery Unavailable",
```

```
        textAlign:  
        TextAlign.center,  
  
        style:  
        Theme.of(context)  
  
            .textTheme  
            .headline6  
  
            ?.copyWith(color:  
Theme.of(context).backgroundColor),  
  
        ))))  
  
    ]));  
  
}  
  
},  
  
)  
  
]);  
  
},  
  
);  
  
};  
  
});  
  
}  
  
} else {  
  
    return Padding(  
  
        padding: const EdgeInsets.all(30),  
  
        child: SizedBox(  
  
            width: double.infinity,  
  
            child: Center(  
})
```

```
        child: Column(children: [
          Text(
            "The cart is empty",
            textAlign: TextAlign.center,
            style: Theme.of(context).textTheme.headline1,
          ),
          const SizedBox(
            height: 5,
          ),
          Text(
            "Try adding an item through the browse page.",
            textAlign: TextAlign.center,
            style: Theme.of(context).textTheme.bodyText1,
          ),
          const SizedBox(
            height: 20,
          ),
        ],
      );
    }
  }
} else {
```

```
        return Row(mainAxisAlignment: MainAxisAlignment.center, children: [  
  
            Padding(  
  
                padding: const EdgeInsets.all(50),  
                child: CircularProgressIndicator(  
  
                    color: Theme.of(context).primaryColor,  
                ))  
        ]);  
    }  
  
    })),  
],));  
}  
}  
}
```

checkout.dart

```
import 'dart:async';  
  
import 'package:alleat/services/cart_service.dart';  
  
import 'package:alleat/services/queryserver.dart';  
  
import 'package:alleat/widgets/elements/elements.dart';  
  
import 'package:flutter/material.dart';  
  
import 'package:shared_preferences/shared_preferences.dart';  
  
import 'package:google_maps_flutter/google_maps_flutter.dart';
```

```

import
'package:currency_text_input_formatter/currency_text_input_formatter.dart';

import 'dart:convert';

class Checkout extends StatefulWidget {

  final List cartInfo;

  const Checkout({Key? key, required this.cartInfo}) : super(key: key);

}

class _CheckoutState extends State<Checkout> {

  final Set<Marker> markers = {};//markers for google map

  String selectedTip = "none";//Percentage tip (15%, 20%, 25%, custom, none)

  double tipPrice = 0; //Tip price

  double subtotal = 0; //Subtotal for items

  static TextEditingController customAmount = TextEditingController(text:
"£0.00");//Custom price for tip (text field)

  Future<Map> sendCart(cart, tip, address, latitude, longitude) async {

    //Send cart to the server - formatted before sending

    int indexid = 0; //Index of the current item so that the customise 2D array
can link to the item 2D array

```

```

List iteminfo = []; //profile id, item id, item quantity

List customiseinfo = []; //index id, customise option id, option quantity

int restaurantid = 0; //restaurant id

for (int i = 0; i < cart.length; i++) {

    //For each profile

    int profileid = cart[i][0]; //Save the profile id as profileid

    List itemkeys = cart[i][6].keys.toList();

    for (int j = 0; j < itemkeys.length; j++) {

        //For each item

        List item = cart[i][6][itemkeys[j]];

        restaurantid = int.parse(item[0][4]);

        int itemid = int.parse(item[0][12]); //Save item id as itemid

        int itemQuantity = item[0][6];

        iteminfo.add([
            profileid,
            itemid,
            itemQuantity,
        ]);

        List customisekeys = item[1].keys.toList();

        for (int k = 0; k < customisekeys.length; k++) {

            //For each customise title

```

```

List customiseTitle = item[1][customisekeys[k]];

for (int l = 0; l < customiseTitle[1].length; l++) {

    // For each option

    List customiseOption = customiseTitle[1][l];

    int customiseOptionid = int.parse(customiseOption[0]); //Save option
    id as customiseoptionid

    int optionQuantity = customiseOption[1];

    customiseinfo.add([indexid, customiseOptionid, optionQuantity]);

}

}

indexid++; //Add one to index (incremental for each item)

}

}

var res = await QueryServer.query("https://alleat.cpur.net/query/orders.php",
{

    //Send data to orders.php . If there is an error, it returns back the error
    code

    "type": "add",

    "tip": tip,

    "address": address,

    "latitude": latitude,

    "longitude": longitude,

    "restaurantid": restaurantid.toString(),

    "iteminfo": json.encode(iteminfo),

```

```

    "customiseinfo": json.encode(customiseinfo),

});

return res; //return result (contains basic message either with error=true
and the error or error=false and success)

}

Future<List> getDeliveryDestination() async {

final prefs = await SharedPreferences.getInstance(); // Get saved location
from shared preferences

final double? savedLocationLat = prefs.getDouble('locationLatitude');

final double? savedLocationLng = prefs.getDouble('locationLongitude');

final List<String>? savedLocationText =
prefs.getStringList('locationPlacemark');

if (savedLocationLat == null || savedLocationLng == null || savedLocationText
== null) {

    //If either the latitude, longitude or the text is null return empty array
to ensure that it cannot return a partial result ACID

    return [];
}

} else {

    savedLocationText.addAll([savedLocationLat.toString(),
    savedLocationLng.toString()]);
}

return savedLocationText.toList();
}
}

```

```
Set<Marker> getmarkers(restaurantAddress, restaurantLat, restaurantLng,
destination, destinationLat, destinationLng) {

    //Create the two markers on the map with their positions

    //markers to place on map

    BitmapDescriptor restaurantMarkerIcon =
BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueBlue);

    BitmapDescriptor destinationMarkerIcon =
BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueRose);

    markers.add(Marker(
        //add first marker

        markerId: const MarkerId("Restaurant"),

        position: LatLng(restaurantLat, restaurantLng), //position of restaurant

        infoWindow: const InfoWindow(
            //popup info

            title: 'Restaurant',

        ),
        icon: restaurantMarkerIcon //Icon for restaurant
    ));

    markers.add(Marker(
        markerId: const MarkerId("Delivery Address"),

        position: LatLng(destinationLat, destinationLng), //position of delivery
address

        infoWindow: const InfoWindow(

```

```
//popup info

    title: 'Delivery Address',
),
icon: destinationMarkerIcon //Icon for delivery address
));

return markers;
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: SingleChildScrollView(
            child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
                const ScreenBackButton(),
                Padding(
                    padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 10),
                    child: Text(
                        "Checkout.",
                        style: Theme.of(context).textTheme.headline1,
                    )),
                Padding(
```

```

padding: const EdgeInsets.symmetric(horizontal: 40),

child: Text(
  "Location",
  style: Theme.of(context).textTheme.headline2,
),

const SizedBox(
  height: 20,
),

FutureBuilder<List>(
  future: getDeliveryDestination(), //Get the delivery location from shared
preferences

  builder: (context, snapshot) {

    if (snapshot.hasData) {

      List destination = snapshot.data ?? [];
      if (destination != []) {

        List locationItemKeys = widget.cartInfo[0][6].keys
          .toList(); //Get the item ids for the first profiles so that it
can be used to get the restaurant info saved for it

        final List restaurantAddress =
          widget.cartInfo[0][6][locationItemKeys[0]][0][9].split(',');
        //Split the restaurant address into seperate items in array

        final double restaurantLat =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][10]);

```

```
        final double restaurantLng =
double.parse(widget.cartInfo[0][6][locationItemKeys[0]][0][11]);

        final controllerMap = Completer<GoogleMapController>(); //Google
Maps Controller

        final double destinationLat = double.parse(destination[4]);

        final double destinationLng = double.parse(destination[5]);



return Column(children: [
    SizedBox(
        //Create google map with height 200px without any movement
        controls centered around the destination of delivery
        width: double.infinity,
        height: 200,
        child: LayoutBuilder(builder: (BuildContext context,
BoxConstraints constraints) {
            return GoogleMap(
                myLocationEnabled: false,
                compassEnabled: false,
                mapToolbarEnabled: false,
                zoomGesturesEnabled: true,
                // hide location button
                myLocationButtonEnabled: false,
                mapType: MapType.normal,
                zoomControlsEnabled: false,

```

```

        rotateGesturesEnabled: false,
        tiltGesturesEnabled: false,
        // camera position
        initialCameraPosition: CameraPosition(target:
LatLng(destinationLat, destinationLng), zoom: 12),
        onMapCreated: (GoogleMapController controller) {
            controllerMap.complete(controller);
        },
        markers: getmarkers(restaurantAddress, restaurantLat,
restaurantLng, destination, destinationLat, destinationLng));
    }),
    const SizedBox(
        height: 20,
    ),
    //Column with the locations of the destination and restaurant
using the colour coded markers as a key
    Padding(
        padding: const EdgeInsets.symmetric(horizontal: 40, vertical:
20),
        child: Row(
            children: [
            const Icon(
Icons.location_on,

```

```
        size: 30,  
  
        color: Color(0xffca458f),  
  
,  
  
const SizedBox(  
  
        width: 30,  
  
,  
  
Expanded(  
  
        child: Column(  
  
        crossAxisAlignment: CrossAxisAlignment.start,  
  
        children: [  
  
        Text(  
  
        "Delivery Address",  
  
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
,  
  
        const SizedBox(  
  
        height: 10,  
  
,  
  
        Text(  
  
        "${destination[0]}, ${destination[1]}",  
  
        style: Theme.of(context).textTheme.bodyText1,  
  
,
```

```
        Text(
            "${destination[2]}, ${destination[3]}",
            style: Theme.of(context).textTheme.bodyText1,
        ),
    ],
))
],
),
Padding(
    padding: const EdgeInsets.symmetric(horizontal: 40, vertical:
20),
    child: Row(
        children: [
            const Icon(
                Icons.location_on,
                size: 30,
                color: Color(0xff4133e3),
            ),
            const SizedBox(
                width: 30,
            ),
            Expanded(
                child: Column(

```

```
crossAxisAlignment: CrossAxisAlignment.start,  
children: [  
    Text(  
        "Restaurant Address",  
        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
,  
    const SizedBox(  
        height: 10,  
,  
    ),  
    Text(  
        "${widget.cartInfo[0][6][locationItemKeys[0]][0][3]}:  
${widget.cartInfo[0][6][locationItemKeys[0]][0][9]}",  
        style: Theme.of(context).textTheme.bodyText1,  
,  
    ],  
    ))  
,  
)),  
]);  
}  
else {  
    return const Text("Failed to get location");  
}
```

```
        }

    } else {

        return const Text("Getting Destination");

    }

},

),

//Tipping section

Padding(

    padding: const EdgeInsets.symmetric(vertical: 20),

    child: Divider(

        thickness: 2,

        color:

Theme.of(context).textTheme.headline6?.color?.withOpacity(0.5),

        indent: 40,

        endIndent: 40,

    )),


Padding(

    padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 10),

    child: Text(

        "Tipping",

        style: Theme.of(context).textTheme.headline2,

    )),


const SizedBox(
```

```
height: 10,  
)  
LayoutBuilder(builder: (p0, p1) {  
    subtotal = 0;  
    for (int i = 0; i < widget.cartInfo.length; i++) {  
        //For each profile, get the profile item total price and add it to the  
        current subtotal  
        subtotal = (subtotal * 100 + widget.cartInfo[0][7] * 100) / 100;  
    }  
    //Tipping buttons  
    return Padding(  
        padding: const EdgeInsets.symmetric(horizontal: 10),  
        child: Column(children: [  
            Row(  
                //Row of preset percentage tips  
                children: [  
                    Expanded(  
                        //15% tip button  
                        child: Padding(  
                            padding: const EdgeInsets.all(10),  
                            child: InkWell(  
                                onTap: () {
```

```
//On button tap, change the selected tip to 15%
so that the button colour changes and set the tip price to 15% of the subtotal

        setState(() {

            selectedTip = "15%";

            tipPrice = double.parse(((subtotal * 100) *
0.15) / 100).toStringAsFixed(2));

        });

    },
    child: Container(
        alignment: Alignment.center,
        decoration: BoxDecoration(
            color:
Theme.of(context).colorScheme.onSurface,
            borderRadius: BorderRadius.circular(10),
            border: Border.all(
                color: (selectedTip == "15%") ?
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,
                width: 2),
            boxShadow: [
                BoxShadow(
                    color: Colors.black.withOpacity(0.01),
                    spreadRadius: 2,
                    blurRadius: 10,
```

```
        offset: const Offset(0, 10), // changes  
position of shadow  
  
    ),  
  
    ]),  
  
    child: AspectRatio(  
  
        aspectRatio: 1 / 1,  
  
        child: Column(  
  
            mainAxisAlignment:  
MainAxisAlignment.center,  
  
            children: [  
  
                Text(  
  
                    "15%",  
  
                    style:  
Theme.of(context).textTheme.headline3,  
  
                ),  
  
                const SizedBox(  
  
                    height: 5,  
  
                ),  
  
                Text(  
  
                    "+E$((((subtotal * 100) * 0.15) /  
100).toStringAsFixed(2))",  
  
                    style:  
Theme.of(context).textTheme.bodyText1,  
  
                )  
            )  
        )  
    )  
);
```

```
        ],  
  
        ))))),  
  
        Expanded(  
  
            //20% tip button  
  
            child: Padding(  
  
                padding: const EdgeInsets.all(10),  
  
                child: InkWell(  
  
                    onTap: () {  
  
                        setState(() {  
  
                            //On button tap, change the selected tip to 20%  
                            //so that the button colour changes and set the tip price to 20% of the subtotal  
  
                            selectedTip = "20%";  
  
                            tipPrice = double.parse(((subtotal * 100) *  
0.20) / 100).toStringAsFixed(2));  
  
                    });  
  
                },  
  
                child: Container(  
  
                    alignment: Alignment.center,  
  
                    decoration: BoxDecoration(  
  
                        color:  
Theme.of(context).colorScheme.onSurface,  
  
                        borderRadius: BorderRadius.circular(10),  
  
                        border: Border.all(  

```

```
        color: (selectedTip == "20%") ?  
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,  
  
        width: 2),  
  
        boxShadow: [  
  
        BoxShadow(  
  
        color: Colors.black.withOpacity(0.01),  
  
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes  
position of shadow  
  
(  
),  
]),  
  
child: AspectRatio(  
  
        aspectRatio: 1 / 1,  
  
        child: Column(  
  
        mainAxisAlignment:  
MainAxisAlignment.center,  
  
        children: [  
  
        Text(  
  
        "20%",  
  
        style:  
Theme.of(context).textTheme.headline3,  
  
(  
),  
  
const SizedBox(  
)
```

```

        height: 5,
    ),
    Text(
        "+\$(((subtotal * 100) * 0.20) /
100).toStringAsFixed(2)}",
        style:
Theme.of(context).textTheme.bodyText1,
    )
],
)))),
Expanded(
//25% tip button
child: Padding(
padding: const EdgeInsets.all(10),
child: InkWell(
onTap: () {
//On button tap, change the selected tip to 25%
so that the button colour changes and set the tip price to 25% of the subtotal
setState(() {
selectedTip = "25%";
tipPrice = double.parse(((subtotal * 100) *
0.25) / 100).toStringAsFixed(2));
});
},

```

```
        child: Container(



            alignment: Alignment.center,



            decoration: BoxDecoration(



                color:



Theme.of(context).colorScheme.onSurface,



                borderRadius: BorderRadius.circular(10),



                border: Border.all(



                    color: (selectedTip == "25%") ?



Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,



                    width: 2),



                boxShadow: [



                    BoxShadow(



                        color: Colors.black.withOpacity(0.01),



                        spreadRadius: 2,



                        blurRadius: 10,



                        offset: const Offset(0, 10), // changes



position of shadow



                    ),



                ],



                child: AspectRatio(



                    aspectRatio: 1 / 1,



                    child: Column(



                        mainAxisSize:



MainAxisSize.alignment,



MainAxisAlignment.center,
```

```

        children: [
          Text(
            "25%",
            style: Theme.of(context).textTheme.headline3,
          ),
          const SizedBox(
            height: 5,
          ),
          Text(
            "+\${(((subtotal * 100) * 0.25) /
100).toStringAsFixed(2)}",
            style: Theme.of(context).textTheme.bodyText1,
          )
        ],
      ))))),],
),
Padding(
  //Custom tip price
  padding: const EdgeInsets.all(10),
  child: InkWell(
    onTap: () {

```

```
//On button press, change the selectedTip to be "custom"
so that it displays the custom tip price text field instead of the title

        setState(() {

            selectedTip = "custom";

            try {

                //Try to set the tip price to be the price inputted
                by the user but if it fails because it is null or a letter is typed, return it as
                0

                tipPrice =
double.parse((double.parse((customAmount.text.toString().split("£"))[1])).toStringAsFixed(2));

            } catch (e) {

                tipPrice = 0;

            }

        });

    },
    child: Container(
        alignment: Alignment.center,
        height: 60,
        decoration: BoxDecoration(
            color: Theme.of(context).colorScheme.onSurface,
            border: Border.all(
                color: (selectedTip == "custom") ?
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,

```

```
        width: 2),  
  
        borderRadius: BorderRadius.circular(10),  
  
        boxShadow: [  
  
            BoxShadow(  
  
                color: Colors.black.withOpacity(0.01),  
  
                spreadRadius: 2,  
  
                blurRadius: 10,  
  
                offset: const Offset(0, 10), // changes  
position of shadow  
  
  
        ]),  
  
        child: (selectedTip == "custom") //If the selected  
button is "custom" display text field that is in the currency format  
  
        ? Padding(  
  
            padding: const EdgeInsets.symmetric(horizontal:  
40, vertical: 10),  
  
            child: Row(  
  
                children: [  
  
                    Text(  
  
                        "Tip:",  
  
                        style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
                ),  

```

```
        const SizedBox(  
          width: 10,  
        ),  
        Expanded(  
          child: TextFormField(  
            onChanged: (value) {  
              try {  
                tipPrice =  
                  double.parse((double.parse((customAmount.text.toString().split("£"))[1])).toStringAsFixed(2));  
              } catch (e) {  
                tipPrice = 0;  
              }  
            },  
            controller: customAmount,  
            keyboardType: TextInputType.number,  
            style:  
              Theme.of(context).textTheme.bodyText2,  
            inputFormatters:  
              [CurrencyTextInputFormatter(symbol: '£')],  
            decoration: (InputDecoration(  
              hintText: "£3.21",
```

```

        contentPadding:
Theme.of(context).inputDecorationTheme.contentPadding,

        border:
Theme.of(context).inputDecorationTheme.border,
                focusedBorder:
Theme.of(context).inputDecorationTheme.focusedBorder,
                enabledBorder:
Theme.of(context).inputDecorationTheme.enabledBorder,
                floatingLabelBehavior:
FloatingLabelBehavior.never))),

        )
    ],
))

//If the custom price is not selected, show title
"custom tip amount" instead of the text field

: Text(
    "Custom Tip Amount",
    style:
Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).textTheme.headline3?.color),
    ))),
Padding(
    //No tip button

    padding: const EdgeInsets.all(10),
    child: InkWell(
        onTap: () {

```

```
//On button press, change selected to be "none" so that
it changes the highlight colour for it and set the tip price to be 0

        setState(() {

            selectedTip = "none";

            tipPrice = 0;

        });

    },

    child: Container(

        alignment: Alignment.center,

        height: 60,

        decoration: BoxDecoration(

            color: Theme.of(context).colorScheme.onSurface,

            borderRadius: BorderRadius.circular(10),

            border: Border.all(

                color: (selectedTip == "none") ?
Theme.of(context).primaryColor : Theme.of(context).colorScheme.onSurface,

                width: 2),

            boxShadow: [

                BoxShadow(

                    color: Colors.black.withOpacity(0.01),

                    spreadRadius: 2,

                    blurRadius: 10,

                    offset: const Offset(0, 10), // changes

position of shadow
```

```
        ),  
  
        []),  
  
        child: Text(  
  
            "No Tip",  
  
            style:  
Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline3?.color),  
  
        ))))  
  
    ]);  
  
},  
  
const SizedBox(  
  
    height: 20,  
  
,  
  
Padding(  
  
    padding: const EdgeInsets.symmetric(vertical: 20),  
  
    child: Divider(  
  
        thickness: 2,  
  
        color:  
Theme.of(context).textTheme.headline6?.color?.withOpacity(0.5),  
  
        indent: 40,  
  
        endIndent: 40,  
  
    )),  
  
LayoutBuilder(builder: ((p0, p1) {  
  
    //Checkout final price section  

```

```
    List priceItemKeys = widget.cartInfo[0][6].keys.toList(); //get the item  
ids for the first profile in the cart  
  
    double total = (subtotal * 100 +  
double.parse(widget.cartInfo[0][6][priceItemKeys[0]][0][5]) * 100 + tipPrice *  
100) /  
  
    100; //Total is equal to the subtotal + delivery price + tip  
  
    return Padding(  
  
        padding: const EdgeInsets.symmetric(horizontal: 40, vertical: 20),  
  
        child: Row(mainAxisAlignment: MainAxisAlignment.end, children: [  
  
            Row(  
  
                children: [  
  
                    Column(  
  
                        crossAxisAlignment: CrossAxisAlignment.start,  
                        children: [  
  
                            Text(  
  
                                "Subtotal",  
  
                                style: Theme.of(context).textTheme.headline6,  
                            ),  
  
                            Text(  
  
                                "Delivery Fee",  
  
                                style: Theme.of(context).textTheme.headline6,  
                            ),  
  
                            Text(  
  
                                "Tip",  
                            ),  
                        ],  
                    ),  
                ],  
            ),  
        ],  
    );
```

```
        style: Theme.of(context).textTheme.headline6,  
    ),  
  
    const SizedBox(height: 20),  
  
    Text(  
  
        "Total",  
  
        style:  
Theme.of(context).textTheme.headline5?.copyWith(color:  
Theme.of(context).primaryColor),  
  
    )  
  
,  
  
,  
  
const SizedBox(  
  
    width: 50,  
  
,  
  
Column(  
  
    mainAxisAlignment: MainAxisAlignment.end,  
  
    children: [  
  
    Text(  
  
        "£${subtotal.toStringAsFixed(2)}",  
  
        style: Theme.of(context).textTheme.bodyText2,  
  
    ),  
  
    Text(  
  
        "£${widget.cartInfo[0][6][priceItemKeys[0]][0][5].toString()}\\"
```

```

        style: Theme.of(context).textTheme.bodyText2,
    ),
    Text(
        "£${tipPrice.toStringAsFixed(2)}",
        style: Theme.of(context).textTheme.bodyText2,
    ),
    const SizedBox(height: 20),
    Text("£${total.toStringAsFixed(2)}",
        style:
Theme.of(context).textTheme.headline5?.copyWith(color:
Theme.of(context).textTheme.headline1?.color)),
    ],
)
],
),
]);
})),
Row(mainAxisAlignment: MainAxisAlignment.center, children: [
Expanded(
//Checkout button
child: Padding(
padding: const EdgeInsets.symmetric(vertical: 30, horizontal:
20),
child: ElevatedButton(

```

```

 onPressed: () async {

    //On button press get the delivery location and join the
destination address into a single string

    List destination = await getDeliveryDestination();

    String destinationAddress = [destination[0],
destination[1], destination[2], destination[3]].join(" ,");

    Map orderCreated = await sendCart(
        //Send the cart to the server

        widget.cartInfo,
        tipPrice.toString(),
        destinationAddress,
        destination[4].toString(),
        destination[5].toString());

    if (orderCreated["error"] == true) {

        //If there is an error go to the homepage and display
snackbar with error message

        setState(() {
            Navigator.of(context).pop();
            Navigator.of(context).pop();
        });

        ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("ERROR:
${orderCreated["message"]}")));
    });
}

} else {

```

```
//If there isn't an error try to clear the cart

try {

    await SQLiteCartItems.clearCart();

    setState(() {

        Navigator.of(context).pop();

        ScaffoldMessenger.of(context).showSnackBar(const
SnackBar(content: Text("Successfully ordered.")));

    });

} catch (e) {

    //If there is an error clearing the cart display error

    setState(() {

        Navigator.of(context).pop();

        ScaffoldMessenger.of(context)

            .showSnackBar(SnackBar(content: Text("ERROR:
Successfully created order but failed to clear cart. \n $e")));

    });

}

}

),

child: const Text("Complete Order")))

])

]));


}
```

```
}
```

browse.dart

```
import 'package:alleat/screens/filterSort.dart';

import 'package:alleat/widgets/elements/browse_categories.dart';

import 'package:alleat/widgets/elements/search.dart';

import 'package:alleat/widgets/restaurants_list.dart';

import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

class BrowsePage extends StatefulWidget {

  const BrowsePage({Key? key}) : super(key: key);

  @override
  State<BrowsePage> createState() => _BrowsePageState();
}

class _BrowsePageState extends State<BrowsePage> {

  @override
  Widget build(BuildContext context) {
    return SafeArea(
```

```
//Keep within screen area

child: Scaffold(

  body: SingleChildScrollView(

    child: Column(children: [

const MainAppBar(

  height: 150,

),

const SizedBox(height: 20),

const SearchBar(), //Search bar

const SizedBox(height: 40),

Padding(

  padding: const EdgeInsets.symmetric(horizontal: 30),

  child: Row(

    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    crossAxisAlignment: CrossAxisAlignment.baseline,
    textBaseline: TextBaseline.alphabetic,
    children: [
      //Display row with the title popular categories and an arrow
      //pointing right to go to all the categories

      Text(
        "Popular Categories.",
        style: Theme.of(context).textTheme.headline2,
      ),
      InkWell(

```

```
        child: Padding(  
  
            padding: const EdgeInsets.all(5),  
  
            child: Icon(  
  
                Icons.chevron_right,  
  
                size: 30,  
  
                color: Theme.of(context).colorScheme.onBackground,  
            )),  
  
            onTap: () => (Navigator.push( //Go to Categories Page  
  
                context,  
  
                MaterialPageRoute(  
  
                    builder: (context) => const CategoriesPage(),  
                ))))  
  
        ],  
  
    )),  
  
const SizedBox(height: 40),  
  
const Categories(), // Display horizontally scrolling categories  
  
const SizedBox(height: 40),  
  
Padding(  
  
    padding: const EdgeInsets.symmetric(horizontal: 30),  
  
    child: Row(  
  
        mainAxisAlignment: MainAxisAlignment.center,  
  
        mainAxisSize: MainAxisSize.spaceBetween,  
    ),
```

```
children: [  
    // Display title restaurants with a filter icon to go to filtering  
    options  
  
    Text(  
        "Restaurants.",  
        style: Theme.of(context).textTheme.headline2,  
    ),  
  
    InkWell(  
        child: Padding(  
            padding: const EdgeInsets.all(5),  
            child: Icon(  
                Icons.tune,  
                size: 30,  
                color: Theme.of(context).colorScheme.onBackground,  
            )),  
        onTap: () => (Navigator.push(  
            context,  
            MaterialPageRoute(  
                builder: (context) => const FilterSort(),  
            ))  
    ],  
),  
  
const SizedBox(height: 30),
```

```
        const RestaurantList(),
    ])));
}

}
```

foryou.dart

```
import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

class ForYouPage extends StatelessWidget {

    const ForYouPage({Key? key}) : super(key: key);

    @override

    Widget build(BuildContext context) {

        return Column(children: [

            const MainAppBar(
                height: 150,
            ),

            Center(
                child: Text(
                    'For You',

```

```
        style: Theme.of(context).textTheme.headline1,  
    ),  
)  
]);  
}  
}
```

homepage.dart

```
import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

class HomePage extends StatelessWidget {

  const HomePage({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    Future<List> getName() async {

      // Get the name of the profile that is selected from shared preferences

      final prefs = await SharedPreferences.getInstance();
```

```
final String? firstname = prefs.getString('firstname');

final String? lastname = prefs.getString('lastname');

return [firstname, lastname];

}

return Column(children: [

const MainAppBar(


//Display main app bar at the top


height: 150,


),


Padding(


padding: const EdgeInsets.only(left: 40, top: 40, right: 60, bottom:


80),


child: Column(


crossAxisAlignment: CrossAxisAlignment.start,


children: [


Text(


// Display welcome back


'Welcome back',


style: Theme.of(context).textTheme.headline6,


),


FutureBuilder<List>(


// run future to get the name from shared preferences
```

```
future: getName(),

builder: (context, snapshot) {

    if (snapshot.hasData) {

        //If the data is received

        List? name = snapshot.data; //Assign received data to name

        if (name![0] != null) {

            // If the data is not null

            return Text(

                // Display firstname and lastname

                ('${name[0]} ${name[1]}'),

                style: Theme.of(context).textTheme.headline1,

            );

        } else {

            // If the data is null or anything else

            return Text(

                // Display Profile Unknown

                'Profile Unknown.',

                style: Theme.of(context).textTheme.headline1,

            );

        }

    } else {

        // While waiting to get data or there is an error while
        getting the data, display loading profile
    }
}
```

```
        return Text(
            'Loading Profile...',
            style: Theme.of(context).textTheme.headline1,
        );
    },
),
],
))
]);
}
}
```

profiles.dart

```
import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/widgets/topbar.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

class HomePage extends StatelessWidget {

    const HomePage({Key? key}) : super(key: key);
```

```
@override

Widget build(BuildContext context) {

Future<List> getName() async {

    // Get the name of the profile that is selected from shared preferences

    final prefs = await SharedPreferences.getInstance();

    final String? firstname = prefs.getString('firstname');

    final String? lastname = prefs.getString('lastname');

    return [firstname, lastname];

}

return Column(children: [

const MainAppBar(

    //Display main app bar at the top

    height: 150,

),

Padding(

    padding: const EdgeInsets.only(left: 40, top: 40, right: 60, bottom:

80),

    child: Column(

        crossAxisAlignment: CrossAxisAlignment.start,

        children: [

            Text(
```

```
// Display welcome back

'Welcome back',
style: Theme.of(context).textTheme.headline6,
),

FutureBuilder<List>(
// run future to get the name from shared preferences

future: getName(),
builder: (context, snapshot) {

if (snapshot.hasData) {

//If the data is received

List? name = snapshot.data; //Assign received data to name

if (name![0] != null) {

// If the data is not null

return Text(
// Display firstname and lastname

('${name[0]} ${name[1]}'),
style: Theme.of(context).textTheme.headline1,
);

} else {

// If the data is null or anything else

return Text(
// Display Profile Unknown

```

```
        'Profile Unknown.',  
        style: Theme.of(context).textTheme.headline1,  
    );  
  
}  
  
} else {  
  
    // While waiting to get data or there is an error while  
    getting the data, display loading profile  
  
    return Text(  
  
        'Loading Profile...',  
        style: Theme.of(context).textTheme.headline1,  
    );  
  
}  
  
},  
,  
],  
))  
]);  
}  
}
```

profilesetup_create.dart

```
import 'package:alleat/services/dataencryption.dart';
```

```
import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/queryserver.dart';

import 'package:alleat/services/setselected.dart';

import 'package:alleat/widgets/elements/elements.dart';

import 'package:alleat/widgets/navigationbar.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:email_validator/email_validator.dart';




class AddProfileCreationPageName extends StatefulWidget {

  const AddProfileCreationPageName({Key? key}) : super(key: key);

  @override

  State<AddProfileCreationPageName> createState() =>

  _AddProfileCreationPageNameState();

}

class _AddProfileCreationPageNameState extends State<AddProfileCreationPageName>

{

  final _formKey = GlobalKey<FormState>();

  static TextEditingController firstnameText = TextEditingController();

  static TextEditingController lastnameText = TextEditingController();

  final data = [firstnameText.text = "", lastnameText.text = ""]; //Store the data to be reset if back button pressed
```

```
@override

Widget build(BuildContext context) {

  return Scaffold(
    body: CustomScrollView(slivers: [
      SliverFillRemaining(
        hasScrollBody: false,
        child: Column(crossAxisAlignment: CrossAxisAlignment.start,
          mainAxisAlignment: MainAxisAlignment.spaceBetween, children: <Widget>[
        Flexible(
          //Create flexible widgets to be able to resize with keyboard
          flex: 2,
          fit: FlexFit.loose,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              const ScreenBackButton(), //Back button to go to starting
              screen
              Padding(
                padding: const EdgeInsets.only(top: 20, left: 20, right:
                20, bottom: 5),
                child: Align(
                  alignment: Alignment.center,
                  child: Text(
                    "Step 1 of 3",

```

```
        style: Theme.of(context).textTheme.headline6,  
  
        textAlign: TextAlign.center,  
  
      )),  
  
    Padding(  
  
      padding: const EdgeInsets.only(bottom: 20, left: 20,  
right: 20),  
  
      child: Align(  
  
        alignment: Alignment.center,  
  
        child: Text(  
  
          "Let's Get Started.",  
  
          style: Theme.of(context).textTheme.headline1,  
  
          textAlign: TextAlign.center,  
  
        )),  
  
      const SizedBox(  
  
        height: 20,  
  
      )  
  
    ],  
  
  )),  
  
Flexible(  
  
  flex: 2,  
  
  fit: FlexFit.loose,  
  
  child: Column(crossAxisAlignment: CrossAxisAlignment.start,  
children: [
```

```
Form(
    key: _formKey,
    child: Column(
        children: [
            ListTile(
                title: TextFormField(
                    controller: firstnameText, //Form data lastname
collected and sent to database

                    keyboardType: TextInputType.name,
                    style: Theme.of(context).textTheme.bodyText2,
                    decoration: (InputDecoration(
                        hintText: "Forename",
                        contentPadding:
Theme.of(context).inputDecorationTheme.contentPadding,
                        border:
Theme.of(context).inputDecorationTheme.border,
                        focusedBorder:
Theme.of(context).inputDecorationTheme.focusedBorder,
                        enabledBorder:
Theme.of(context).inputDecorationTheme.enabledBorder,
                        floatingLabelBehavior:
FloatingLabelBehavior.never)),
                    inputFormatters: [
                        //Only allows the input of letters a-z and A-Z and
@, . -

```

```
        FilteringTextInputFormatter.allow(RegExp('[a-zA-Z0-9@,. -]'))  
    ],  
  
    validator: (firstname) {  
  
        //Required field  
  
        if (firstname == null || firstname.isEmpty) {  
  
            return "Required";  
  
        } else if (firstname.length > 50) {  
  
            return "Enter a name less than 50 characters";  
  
        }  
  
        return null;  
    },  
},  
)),  
const SizedBox(  
    height: 10,  
,  
ListTile(  
    title: TextFormField(  
        controller: lastnameText, //Form data lastname  
collected and sent to database  
        keyboardType: TextInputType.name,  
        style: Theme.of(context).textTheme.bodyText2,
```

```
decoration: (InputDecoration(  
  
        hintText: "Surname",  
  
        contentPadding:  
Theme.of(context).inputDecorationTheme.contentPadding,  
  
        border:  
Theme.of(context).inputDecorationTheme.border,  
  
        focusedBorder:  
Theme.of(context).inputDecorationTheme.focusedBorder,  
  
        enabledBorder:  
Theme.of(context).inputDecorationTheme.enabledBorder,  
  
        floatingLabelBehavior:  
FloatingLabelBehavior.never)),  
  
        inputFormatters: [  
  
            //Only allows the input of letters a-z and A-Z and  
@, . -  
  
            FilteringTextInputFormatter.allow(RegExp('^[a-zA-Z0-  
9@,. -]'))  
  
        ],  
  
        validator: (lastname) {  
  
            //Required field  
  
            if (lastname == null || lastname.isEmpty) {  
  
                return "Required";  
  
            } else if (lastname.length > 50) {  
  
                return "Enter a name less than 50 characters";  
  
            }  
        }  
    )
```

```
        return null;  
  
    },  
  
    )),  
  
    const SizedBox(  
  
        height: 50,  
  
    ),  
  
    ],  
  
))  
  
]),  
  
Flexible(  
  
    flex: 1,  
  
    fit: FlexFit.loose,  
  
    child: Column(  
  
        children: [  
  
            Padding(  
  
                padding: const EdgeInsets.only(  
  
                    left: 30,  
  
                    right: 30,  
  
                    bottom: 60,  
  
                ),  
  
                child: Center(  
  
                    child: SizedBox(  

```

```
        width: double.infinity,  
  
        child: ElevatedButton(  
  
            onPressed: () {  
  
                if (_formKey.currentState!.validate()) {  
  
                    //If fields have no errors  
  
                    Navigator.pop(context);  
  
                    Navigator.push(  
  
                        context,  
  
                        MaterialPageRoute(  
  
                            builder: (context) =>  
AddProfileCreationPageEmail(  
  
                                firstname:  
firstnameText.text,  
  
                                lastname:  
lastnameText.text,  
  
                                )));  
  
                } else {  
  
                    null;  
  
                }  
  
            },  
  
            child: const Text("Continue"))),  
  
        const SizedBox(  
  
            height: 50,  
  
        )
```

```

        ],
        ))
    ]));
}]);
}

class AddProfileCreationPageEmail extends StatefulWidget {

  const AddProfileCreationPageEmail({Key? key, this.firstname, this.lastname})
  //Get firstname and lastname from previous screen

  : super(key: key);

  final dynamic firstname;

  final dynamic lastname;

  @override
  State<AddProfileCreationPageEmail> createState() =>
  _AddProfileCreationPageEmailState();
}

class _AddProfileCreationPageEmailState extends
State<AddProfileCreationPageEmail> {

  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  static TextEditingController emailText = TextEditingController();

  static TextEditingController confirmemailText = TextEditingController();
}

```

```
dynamic data = [emailText.text = "", confirmemailText.text = ""];// Store  
email and confirm email to be reset if back button pressed  
  
@override  
  
Widget build(BuildContext context) {  
  
  return Scaffold(  
  
    body: CustomScrollView(slivers: [  
  
      SliverFillRemaining(  
  
        hasScrollBody: false,  
  
        child: Column(crossAxisAlignment: CrossAxisAlignment.start,  
mainAxisAlignment: MainAxisAlignment.spaceBetween, children: <Widget>[  
  
          Flexible(  
  
            flex: 2,  
  
            fit: FlexFit.loose,  
  
            child: Column(  
  
              crossAxisAlignment: CrossAxisAlignment.start,  
              children: [  
  
                ScreenBackButton(data: data), //If back button pressed, reset  
inputted data  
  
                Padding(  
  
                  padding: const EdgeInsets.only(top: 20, left: 20, right:  
20, bottom: 5),  
  
                  child: Align(  
  
                    alignment: Alignment.center,  
  
                    child: Text(  
                      text: "Reset",  
                      style: TextStyle(fontSize: 18, color: Colors.purple),  
                    ),  
                  ),  
                ),  
              ],  
            ),  
          ),  
        ),  
      ),  
    ],  
  ),  
);  
}
```

```
        "Step 2 of 3",

        style: Theme.of(context).textTheme.headline6,

        textAlign: TextAlign.center,
    ))),

Padding(
    padding: const EdgeInsets.only(bottom: 20, left: 20,
right: 20),

    child: Align(
        alignment: Alignment.center,
        child: Text(
            "Profile Setup.",

            style: Theme.of(context).textTheme.headline1,
            textAlign: TextAlign.center,
        )),
),
const SizedBox(
    height: 20,
)
],
),
Flexible(
    flex: 2,
    fit: FlexFit.loose,
```

```
        child: Column(crossAxisAlignment: CrossAxisAlignmentAlignment.start,
children: [
    Form(
        key: _formKey,
        child: Column(
            children: [
                ListTile(
                    title: TextFormField(
                        controller: emailText, //Form data lastname collected
and sent to database
                        keyboardType: TextInputType.emailAddress,
                        style: Theme.of(context).textTheme.bodyText2,
                        decoration: (InputDecoration(
                            hintText: "Email",
                            contentPadding:
Theme.of(context).inputDecorationTheme.contentPadding,
                            border:
Theme.of(context).inputDecorationTheme.border,
                            focusedBorder:
Theme.of(context).inputDecorationTheme.focusedBorder,
                            enabledBorder:
Theme.of(context).inputDecorationTheme.enabledBorder,
                            floatingLabelBehavior:
FloatingLabelBehavior.never)),
                inputFormatters: [

```

```

        //Only allows the input of letters a-z and A-Z and
@, . - FilteringTextInputFormatter.allow(RegExp('[a-zA-Z0-
9@,. - ]'))
],
validator: (email) {
    //Required field and uses emailvalidator package to
verify it is an email to simplify the code
    if (email == null || email.isEmpty) {
        return "Required";
    }
    if (EmailValidator.validate(email) == false) {
        return "Please enter valid email";
    }
    return null;
},
),
const SizedBox(
height: 10,
),
ListTile(
title: TextFormField(
controller: confirmemailText, //Form data lastname
collected and sent to database

```

```

        keyboardType: TextInputType.emailAddress,
        style: Theme.of(context).textTheme.bodyText2,
        decoration: (InputDecoration(
            hintText: "Confirm email",
            contentPadding:
                Theme.of(context).inputDecorationTheme.contentPadding,
            border:
                Theme.of(context).inputDecorationTheme.border,
            focusedBorder:
                Theme.of(context).inputDecorationTheme.focusedBorder,
            enabledBorder:
                Theme.of(context).inputDecorationTheme.enabledBorder,
            floatingLabelBehavior:
                FloatingLabelBehavior.never)),
        inputFormatters: [
            //Only allows the input of letters a-z and A-Z and
            @, . -
            FilteringTextInputFormatter.allow(RegExp('^[a-zA-Z0-
            9@,. -]'))
        ],
        validator: (confirmemail) {
            //Required field and uses emailvalidator package to
            verify it is an email to simplify the code
            if (confirmemail == null || confirmemail.isEmpty) {
                return "Required";
            } else if (confirmemail != emailText.text) {

```



```
        child: Center(


            child: SizedBox(


                width: double.infinity,


                child: ElevatedButton(


                    onPressed: () {




                        if (_formKey.currentState!.validate()) {



                            //If fields have no errors



                            Navigator.pop(context);




                            Navigator.push(


                                // On continue, export inputted


                                fields to the final screen


                                context,



                                MaterialPageRoute(


                                    builder: (context) =>



AddProfileCreationPagePassword(


                                email: emailText.text,


                                confirmemail:


confirmemailText.text,


                                firstname:


widget.firstname,


                                lastname: widget.lastname,


                                )));



} else {




null;
```

```
        },
        child: const Text("Continue")))))
    ],
))
])
));
}

}

}

class AddProfileCreationPagePassword extends StatefulWidget {
    const AddProfileCreationPagePassword({Key? key, this.email, this.confirmemail,
this.firstname, this.lastname}) : super(key: key);

    final dynamic email;

    final dynamic confirmemail;

    final dynamic firstname;

    final dynamic lastname;

    @override
    State<AddProfileCreationPagePassword> createState() =>
    _AddProfileCreationPagePasswordState();
}
```

```
class _AddProfileCreationPagePasswordState extends  
State<AddProfileCreationPagePassword> {  
  
    final GlobalKey<FormState> _formKey = GlobalKey<FormState>();  
  
    static TextEditingController passwordText = TextEditingController();  
  
    static TextEditingController confirmPasswordText = TextEditingController();  
  
    static dynamic encryptPassword;  
  
    dynamic data = [passwordText.text = "", confirmPasswordText.text = ""]; //If  
back button pressed, reset data  
  
    bool _passwordVisible = false;  
  
    @override  
  
    Widget build(BuildContext context) {  
  
        return Scaffold(  
  
            body: CustomScrollView(slivers: [  
  
                SliverFillRemaining(  
  
                    hasScrollBody: false,  
  
                    child: Column(crossAxisAlignment: CrossAxisAlignment.start,  
mainAxisAlignment: MainAxisAlignment.spaceBetween, children: <Widget>[  
  
                        Flexible(  
  
                            flex: 2,  
  
                            fit: FlexFit.loose,  
  
                            child: Column(  
  
                                crossAxisAlignment: CrossAxisAlignment.start,  
                                children: [  
  
                                    ScreenBackButton(data: data),  
]  
                ]  
            ]  
        )  
    )  
}
```

```
Padding(  
  padding: const EdgeInsets.only(top: 20, left: 20, right:  
 20, bottom: 5),  
  child: Align(  
    alignment: Alignment.center,  
    child: Text(  
      "Step 3 of 3",  
      style: Theme.of(context).textTheme.headline6,  
      textAlign: TextAlign.center,  
    )),  
  Padding(  
    padding: const EdgeInsets.only(bottom: 20, left: 20,  
right: 20),  
    child: Align(  
      alignment: Alignment.center,  
      child: Text(  
        "Secure Password.",  
        style: Theme.of(context).textTheme.headline1,  
        textAlign: TextAlign.center,  
      )),  
  const SizedBox(  
    height: 20,  
  )
```

```
        ],  
        ),  
        Flexible(  
            flex: 2,  
            fit: FlexFit.loose,  
            child: Column(crossAxisAlignment: CrossAxisAlignment.start,  
children: [  
            Form(  
                key: _formKey,  
                child: Column(  
                    children: [  
                        ListTile(  
                            title: TextFormField(  
                                controller: passwordText, //Form data lastname  
collected and sent to database  
                                keyboardType: TextInputType.visiblePassword,  
                                obscureText: !_passwordVisible,  
                                style: Theme.of(context).textTheme.bodyText2,  
                                decoration: (InputDecoration(  
                                    hintText: "Password",  
                                    contentPadding:  
Theme.of(context).inputDecorationTheme.contentPadding,  
                                    border:  
Theme.of(context).inputDecorationTheme.border,
```

```
        focusedBorder:  
Theme.of(context).inputDecorationTheme.focusedBorder,  
  
        enabledBorder:  
Theme.of(context).inputDecorationTheme.enabledBorder,  
  
        floatingLabelBehavior:  
FloatingLabelBehavior.never,  
  
        suffixIcon: IconButton(  
  
            //Button to toggle password visibility  
  
            icon: Icon(  
  
                _passwordVisible ? Icons.visibility_off :  
Icons.visibility,  
  
            color:  
Theme.of(context).primaryColor.withOpacity(0.5),  
  
>,  
  
            onPressed: () {  
  
                setState(() {  
  
                    _passwordVisible = !_passwordVisible;  
  
                });  
  
            },  
  
        ))),  
  
        autofocusHints: const [AutofillHints.newPassword],  
  
        inputFormatters: [  
  
            //Only allows the input of letters a-z and A-Z and  
0-9 and @, .-$&!#?  
        ]
```

```
        FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9@#$!#?]'))

    ],
    validator: (password) {
        //Required field and checks length of password if it is between 8 and 99 characters
        if (password == null || password.isEmpty) {
            return "Required";
        } else if (!password.contains(RegExp(r'[0-9]')) || !password.contains(RegExp(r'[a-z]'))) {
            return "Password must contain at least 1 number and 1 letter";
        } else if (password.length < 8) {
            return "Password must be a minimum of 8 characters";
        } else if (password.length > 99) {
            return "Password must be a maximum of 99 characters";
        }
        return null;
    },
}),
const SizedBox(
    height: 10,
),
```

```
        ListTile(  
  
            title: TextFormField(  
  
                controller: confirmPasswordText, //Form data lastname  
collected and sent to database  
  
                keyboardType: TextInputType.visiblePassword,  
  
                style: Theme.of(context).textTheme.bodyText2,  
  
                decoration: (InputDecoration(  
  
                    hintText: "Confirm password",  
  
                    contentPadding:  
Theme.of(context).inputDecorationTheme.contentPadding,  
  
                    border:  
Theme.of(context).inputDecorationTheme.border,  
  
                    focusedBorder:  
Theme.of(context).inputDecorationTheme.focusedBorder,  
  
                    enabledBorder:  
Theme.of(context).inputDecorationTheme.enabledBorder,  
  
                    floatingLabelBehavior: FloatingLabelBehavior.never,  
                )),  
  
                obscureText: true,  
  
                autofocus: const [AutofillHints.newPassword],  
  
                autovalidateMode: AutovalidateMode.onUserInteraction,  
  
                inputFormatters: [  
  
                    //Only allows the input of letters a-z and A-Z and  
@, . -
```

```
        FilteringTextInputFormatter.allow(RegExp('[a-zA-Z0-9@,. -]'))  
    ],  
  
    validator: (confirmPassword) {  
  
        //Required field and uses emailvalidator package to  
        verify it is an email to simplify the code  
  
        if (confirmPassword == null ||  
confirmPassword.isEmpty) {  
  
            return "Required";  
  
        } else if (confirmPassword != passwordText.text) {  
  
            return "Passwords do not match";  
  
        }  
  
        return null;  
  
    },  
  
}),  
  
const SizedBox(  
  
height: 50,  
  
) ,  
  
],  
  
))  
  
]),  
  
Flexible(  
  
flex: 1,  
  
fit: FlexFit.loose,
```

```
        child: Column(  
  
            children: [  
  
                Padding(  
  
                    padding: const EdgeInsets.only(  
  
                        left: 30,  
  
                        right: 30,  
  
                        bottom: 60,  
  
                    ),  
  
                    child: Center(  
  
                        child: SizedBox(  
  
                            width: double.infinity,  
  
                            child: ElevatedButton(  
  
                                onPressed: () {  
  
                                    if (_formKey.currentState!.validate()) {  
  
                                        //If fields have no errors  
  
                                        _createProfile();  
  
                                    } else {  
  
                                        null;  
  
                                    }  
  
                                },  
  
                                child: const Text("Create Profile")))))  
  
            ],
```

```
        ))  
    ]));  
}  
  
Future<void> _createProfile() async {  
    encryptPassword = await DataEncryption.encrypt(passwordText.text);  
  
    var receivedServerData = await  
QueryServer.query("https://alleat.cpur.net/query/register.php", {  
  
    //Send data to login.php on server with email and encrypted password  
  
    "firstname": widget.firstname,  
  
    "lastname": widget.lastname,  
  
    "email": widget.email,  
  
    "password": encryptPassword  
});  
  
if (receivedServerData["error"] == true) {  
  
    //If error, display failed to create profile and reset password fields  
  
    print(receivedServerData);  
  
    setState(() {  
  
        ScaffoldMessenger.of(context)  
  
            .showSnackBar(SnackBar(content: Text(receivedServerData["message"] +  
" : Failed to create profile. Please try again")));  
    });  
}
```

```
passwordText.text = "";

confirmPasswordText.text = "";

} else {

    //If the email already exists, reset password fields and bring user back to
the add user page

    if ((recievedServerData["message"])["exist"] == true) {

        passwordText.text = "";

        confirmPasswordText.text = "";

        setState(() {

            Navigator.pop(context);

            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(content: Text('Profile already exists')),
            );
        });
    } else {

        //try {
        print(recievedServerData["message"]);

        List importedProfile = (recievedServerData["message"]["profile"]);

        print(importedProfile);

        await SQLiteLocalProfiles.createProfile(importedProfile[0],
importedProfile[1], importedProfile[2], importedProfile[3], importedProfile[4]);

        bool trySelect = await SetSelected.selectProfile(
```

```
importedProfile[0], //Try to select profile

importedProfile[1],

importedProfile[2],

importedProfile[3]);

if (trySelect == false) {

    setState(() {

        ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content:
Text("Failed to select profile")));

    });

} else {

    //If is able to select profile, clear password fields and go to
navigation page (defaults to homepage)

    passwordText.text = "";

    confirmPasswordText.text = "";

    setState(() {

        Navigator.pop(context);

        Navigator.pop(context);

        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('Successfully created profile.')),
        );

        Navigator.push(context, MaterialPageRoute(builder: (context) => const
Navigation()));

    });

}

}
```

```
// } catch (e) {  
  
    // // If error, display error  
  
    // setState(() {  
  
        // ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:  
Text("ERROR: $e")));  
  
    // });  
  
    // }  
  
}  
  
}  
  
}
```

profilesetup_existing.dart

```
import 'package:alleat/screens/profilesetup/profilesetup_create.dart';  
  
import 'package:alleat/screens/profilesetup/profilesetup_login.dart';  
  
import 'package:flutter/material.dart';  
  
  
  
  
class ProfileSetupExisting extends StatefulWidget {  
  
    const ProfileSetupExisting({Key? key}) : super(key: key);  
  
  
  
  
    @override  
  
    State<StatefulWidget> createState() {  
  
    }
```

```
        return _ProfileSetupExisting();  
    }  
}  
  
class _ProfileSetupExisting extends State<ProfileSetupExisting> {  
  
    @override  
    Widget build(BuildContext context) {  
  
        return Scaffold(  
  
            //Create new screen  
  
            resizeToAvoidBottomInset: false, //Allow resize  
  
            body: Stack(children: [  
  
                Image.asset(  
  
                    //Fullscreen image of food  
  
                    'lib/assets/images/screens/existingsetup/existingsetupscreenfood.jpg',  
  
                    fit: BoxFit.cover,  
  
                    height: double.infinity,  
  
                    width: double.infinity,  
  
                    alignment: Alignment.center,  
  
                ),  
  
                Column(  
  
                    children: [  
  
                ],  
            ),  
        );  
    }  
}
```

```
Padding(  
    //Image of All Eat logo  
    padding: const EdgeInsets.only(  
        top: 50, left: 30, right: 30, bottom: 200),  
    child: Container(  
        width: 50,  
        height: 50,  
        decoration: BoxDecoration(  
            shape: BoxShape.circle,  
            color: Theme.of(context).backgroundColor,  
        ),  
        child: IconButton(  
            color: Theme.of(context).textTheme.headline1?.color,  
            icon: const Icon(Icons.arrow_back),  
            onPressed: () => Navigator.of(context).pop()))  
    ],  
),  
Column(  
    mainAxisAlignment: MainAxisAlignment.start,  
    mainAxisSize: MainAxisSize  
        .end, //Create content at the bottom of the screen  
    children: [
```

```
Padding(  
  
    padding: const EdgeInsets.only(left: 10, right: 10),  
  
    child: Container(  
  
        //Bottom container with login and register actions  
  
        decoration: BoxDecoration(  
  
            color: Theme.of(context).backgroundColor,  
  
            borderRadius: const BorderRadius.only(  
  
                topLeft: Radius.circular(20),  
  
                topRight: Radius.circular(  
  
                    20))), // Round the top corners of container  
  
        width: double.infinity,  
  
        padding: const EdgeInsets.only(left: 20, right: 20),  
  
        child: Column(children: [  
  
            const SizedBox(  
  
                height: 40,  
  
) ,  
  
            Text("Add New Profile.",  
  
                style: Theme.of(context).textTheme.headline2),  
  
            Padding(  
  
                padding: const EdgeInsets.only(  
  
                    left: 50, right: 50, top: 10, bottom: 10),  
  
                child: Text(  

```

```
    "Enhance your experience by using multiple profiles",

    style: Theme.of(context).textTheme.headline6,

    textAlign: TextAlign.center,

)),

const SizedBox(
    height: 20,
),

Padding(
    //Button actions

    padding: const EdgeInsets.only(left: 20, right: 20),

    child: Column(children: [
        SizedBox(
            width: double.infinity,
            child: ElevatedButton(
                style:
                    Theme.of(context).elevatedButtonTheme.style,
                onPressed: () => (Navigator.push( //go to profile
creation page

                context,
                MaterialPageRoute(
                    builder: (context) =>
                    const
AddProfileCreationPageName()))),

```

```
        child: const Text("Create a profile"))),  
  
        SizedBox(  
  
            width: 250,  
  
            child: TextButton(  
  
                style: Theme.of(context).textButtonTheme.style,  
  
                onPressed: () => (Navigator.push( // go to  
profile login page  
  
                    context,  
  
                    MaterialPageRoute(  
  
                        builder: (context) =>  
  
                            const AddProfileLoginPage()))),  
  
                child: const Align(  
  
                    alignment: Alignment.center,  
  
                    child: Text(  
  
                        "I already have a profile",  
  
                        textAlign: TextAlign.center,  
  
                    ))))  
  
            ]),  
  
        ),  
  
        const SizedBox(  
  
            height: 40,  
  
        ),  
  
    ]),
```

```
)  
],  
)  
]),  
);  
}  
}
```

profilesetup_login.dart

```
import 'package:alleat/services/dataencryption.dart';  
  
import 'package:alleat/services/localprofiles_service.dart';  
  
import 'package:alleat/services/queryserver.dart';  
  
import 'package:alleat/services/setselected.dart';  
  
import 'package:alleat/widgets/elements/elements.dart';  
  
import 'package:alleat/widgets/navigationbar.dart';  
  
import 'package:flutter/material.dart';  
  
import 'package:email_validator/email_validator.dart';  
  
import 'package:flutter/services.dart';  
  
import 'dart:async';  
  
  
class AddProfileLoginPage extends StatefulWidget {
```

```

const AddProfileLoginPage({Key? key}) : super(key: key);

}

@Override
State<AddProfileLoginPage> createState() => _AddProfileLoginPageState();

}

class _AddProfileLoginPageState extends State<AddProfileLoginPage> {

  bool disableButton = false;

  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  static TextEditingController email = TextEditingController(); //Create text
  controllers to allow for dynamic variable for form fields

  static TextEditingController password = TextEditingController();

  static dynamic encryptPassword;

  late dynamic profileInfoImport;

  Future<void> _loginUser() async {

    encryptPassword = await DataEncryption.encrypt(password.text); //Encrypt
    password

    var receivedServerData = await
    QueryServer.query("https://alleat.cpur.net/query/login.php", {

      //Send data to login.php on server with email and encrypted password. It
      checks if the credentials are correct and returns exists if it is valid

      "email": email.text,
      "password": encryptPassword,
    });
  }
}

```

```
});  
  
if (recievedServerData["error"] == true) {  
    //If there is an error, clear password and display error from server  
  
    setState(() {  
  
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:  
Text(recievedServerData["message"] + " : Failed to login. Please try again")));  
  
    });  
  
    password.text = "";  
  
} else {  
  
    if (recievedServerData["message"]["exists"] == true) {  
  
        //If ther profile is correct and exists  
  
        List importedProfile = recievedServerData["message"]["profile"];  
  
        bool trySelect = await SetSelected.selectProfile(  
  
            importedProfile[0], //Try select profile  
  
            importedProfile[1],  
  
            importedProfile[2],  
  
            importedProfile[3]);  
  
        if (trySelect == false) {  
  
            // If the profile fails to select display error  
  
            setState(() {  
  
                disableButton = false;  
            });  
        }  
    }  
}
```

```
        ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content:  
Text("Failed to select profile")));  
  
    });  
  
} else {  
  
    // If succeeds to select profile  
  
    try {  
  
        await SQLiteLocalProfiles.createProfile(  
  
            //Create profile in database  
  
            importedProfile[0],  
  
            importedProfile[1],  
  
            importedProfile[2],  
  
            importedProfile[3],  
  
            importedProfile[4]);  
  
        email.text = ""; //Clear email and password  
  
        password.text = "";  
  
        setState(() {  
  
            disableButton = false;  
  
            ScaffoldMessenger.of(context).showSnackBar(  
  
                const SnackBar(content: Text('Successfully logged in.')),  
  
            );  
  
            Navigator.push(  
  
                context, //Go to main area  
  
                MaterialPageRoute(builder: (context) => const Navigation()));  
    }  
}
```

```
});  
  
} catch (e) {  
  
    //If there is an error, display that there was an error  
  
    setState(() {  
  
        disableButton = false;  
  
        ScaffoldMessenger.of(context).showSnackBar(  
  
            const SnackBar(content: Text('Failed to save profile on  
device.')),  
  
        );  
  
    });  
  
}  
  
}  
  
} else {  
  
    // If the password or email is incorrect, display incorrect email or  
password  
  
    setState(() {  
  
        disableButton = false;  
  
        ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content:  
Text("Incorrect email or password")));  
  
    });  
  
}  
  
}
```

```
Future<void> _checkDuplicate() async {

    List profileList = await SQLiteLocalProfiles.getProfiles();

    bool alreadyLogged = false;

    for (int i = 0; i < (profileList.length - 1); i++) {

        if (profileList[i]["email"] == email.text) {

            alreadyLogged = true;

        }

    }

    if (alreadyLogged == false) {

        _loginUser();

    } else {

        setState(() {

            ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content:
Text("Profile is already logged in.")));

        });

    }

}

@Override

Widget build(BuildContext context) {

    var brightness = MediaQuery.of(context).platformBrightness;

    bool darkModeOn = brightness == Brightness.dark;

    return Scaffold(
```

```
body: CustomScrollView(slivers: [  
    SliverFillRemaining(  
        hasScrollBody: false,  
        child: SingleChildScrollView(  
            child: Column(  
                crossAxisAlignment: CrossAxisAlignment.start,  
                mainAxisAlignment: MainAxisAlignment.spaceBetween,  
                children: <Widget>[  
                    Column(crossAxisAlignment: CrossAxisAlignment.start, children: [  
                        const ScreenBackButton(),  
                        Padding(  
                            padding: const EdgeInsets.only(left: 50, right: 50, top: 30),  
                            child: Image.asset((darkModeOn)  
                                ? 'lib/assets/images/screens/profilesetup/login-  
                                illustration-dark.png'  
                                : 'lib/assets/images/screens/profilesetup/login-  
                                illustration-light.png')),  
                        const SizedBox(  
                            height: 20,  
                        ),  
                        Padding(  
                            padding: const EdgeInsets.all(20),  
                            child: Align(alignment: Alignment.center, child:  
                                Text("Welcome back.", style: Theme.of(context).textTheme.headline1))),  
                ]  
            )  
        )  
    )  
]
```

```
const SizedBox(  
    height: 20,  
,  
Padding(  
    padding: const EdgeInsets.all(15),  
    child: Form(  
        key: _formKey,  
        child: SingleChildScrollView(  
            child: Column(  
                children: [  
                    ListTile(  
                        title: TextFormField(  
                            controller: email, //Form data lastname collected and  
sent to database  
                            keyboardType: TextInputType.emailAddress,  
                            style: Theme.of(context).textTheme.bodyText2,  
                            decoration: (InputDecoration(  
                                hintText: "Email",  
                                contentPadding:  
Theme.of(context).inputDecorationTheme.contentPadding,  
                                border:  
Theme.of(context).inputDecorationTheme.border,  
                                focusedBorder:  
Theme.of(context).inputDecorationTheme.focusedBorder,
```

```
        enabledBorder:  
Theme.of(context).inputDecorationTheme.enabledBorder,  
  
        floatingLabelBehavior:  
FloatingLabelBehavior.never)),  
  
        inputFormatters: [  
  
            //Only allows the input of letters a-z and A-Z and  
@, . -  
  
            FilteringTextInputFormatter.allow(RegExp('^[a-zA-Z0-  
9@,. -]'))  
  
        ],  
  
        validator: (email) {  
  
            //Required field and uses emailvalidator package to  
verify it is an email to simplify the code  
  
            if (email == null || email.isEmpty) {  
  
                return "Required";  
  
            }  
  
            if (EmailValidator.validate(email) == false) {  
  
                return "Please enter valid email";  
  
            }  
  
            return null;  
  
        },  
    ),  
    const SizedBox(  
  
height: 10,
```

```
 ),  
  
 ListTile(  
  
   title: TextFormField(  
  
     keyboardType: TextInputType.visiblePassword,  
  
     style: Theme.of(context).textTheme.bodyText2,  
  
     decoration: (InputDecoration(  
  
       hintText: "Password",  
  
       contentPadding:  
Theme.of(context).inputDecorationTheme.contentPadding,  
  
       border:  
Theme.of(context).inputDecorationTheme.border,  
  
       focusedBorder:  
Theme.of(context).inputDecorationTheme.focusedBorder,  
  
       enabledBorder:  
Theme.of(context).inputDecorationTheme.enabledBorder,  
  
       floatingLabelBehavior:  
FloatingLabelBehavior.never)),  
  
     inputFormatters: [  
  
       //Password cannot use " or ' in order to prevent  
SQL injection  
  
       FilteringTextInputFormatter.allow(RegExp('[a-zA-  
Z0-9!@#%^&*(),.?:{}|<>]'))  
  
     ],  
  
     obscureText: true, //Password not visible  
  
     controller: password, //Password copied and checked  
by confirm password
```

```
validator: (password) {  
  
    //Must be a minimum of 8 characters and contain a  
letter and number to make sure there is variety and make it harder to guess. Must  
be under 99 characters so that it reduces processing time on the system  
  
    if (password == null ||  
        password.isEmpty ||  
        password.length < 8 ||  
        password.length > 99 ||  
        !password.contains(RegExp(r'[0-9]')) ||  
        !password.contains(RegExp(r'[a-z]'))){  
  
        return "Required";  
  
    }  
  
    return null;  
  
},  
),  
),  
const SizedBox(height: 50), //Gap  
],  
),  
),  
),  
)  
]),
```

```
Padding(  
    padding: const EdgeInsets.only(  
        left: 30,  
        right: 30,  
        bottom: 60,  
    ),  
    child: Center(  
        child: SizedBox(  
            width: double.infinity,  
            height: 52,  
            child: ElevatedButton(  
                //submit button  
                style: Theme.of(context).elevatedButtonTheme.style,  
                onPressed: () {  
                    if (disableButton == false) {  
                        setState(() {  
                            disableButton = true;  
                        });  
                    if (_formKey.currentState!.validate()) {  
                        _checkDuplicate();  
                    }  
                }  
            )  
        )  
    )  
);
```

```
        child: const Text('Login to Profile'),  
        ),  
      ))),  
    ],  
  )))  
 ]));  
}  
}  
}
```

welcomescreen.dart

```
import 'package:alleat/screens/profilesetup/profilesetup_create.dart';

import 'package:alleat/screens/profilesetup/profilesetup_login.dart';

import 'package:flutter/material.dart';

class ProfileSetupWelcome extends StatefulWidget {

    const ProfileSetupWelcome({Key? key}) : super(key: key);

    @override

    State<StatefulWidget> createState() {

        return _ProfileSetupWelcome();
    }
}
```

```
    }

}

class _ProfileSetupWelcome extends State<ProfileSetupWelcome> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(
      //Create new screen

      resizeToAvoidBottomInset: false, //Allow resize

      body: Stack(children: [
        Image.asset(
          //Fullscreen image of food
          'lib/assets/images/screens/welcomeScreen/welcomeScreenFood.jpg',
          fit: BoxFit.cover,
          height: double.infinity,
          width: double.infinity,
          alignment: Alignment.center,
        ),
        Column(
          children: [
            Padding(
```

```
//Image of All Eat logo

padding: const EdgeInsets.only(
    top: 70, left: 40, right: 40, bottom: 200),
child: Image.asset(
    'lib/assets/images/logo/logodark.png',
    width: 58,
    height: 58,
))

],
),
Column(
crossAxisAlignment: CrossAxisAlignment.start,
mainAxisAlignment: MainAxisAlignment
.end, //Create content at the bottom of the screen
children: [
Padding(
padding: const EdgeInsets.only(left: 10, right: 10),
child: Container(
//Bottom container with login and register actions
decoration: BoxDecoration(
color: Theme.of(context).backgroundColor,
borderRadius: const BorderRadius.only(

```

```
        topLeft: Radius.circular(20),  
  
        topRight: Radius.circular(  
            20)), // Round the top corners of container  
  
        width: double.infinity,  
  
        padding: const EdgeInsets.only(left: 20, right: 20),  
  
        child: Column(children: [  
  
            const SizedBox(  
  
                height: 40,  
  
  
            Text("Let's Get Started.",  
  
                style: Theme.of(context).textTheme.headline2),  
  
            Padding(  
  
                padding: const EdgeInsets.only(  
  
                    left: 50, right: 50, top: 10, bottom: 10),  
  
                child: Text(  
  
                    "A food delivery app with you in mind",  
  
                    style: Theme.of(context).textTheme.headline6,  
  
                    textAlign: TextAlign.center,  
  
                )),  
  
            const SizedBox(  
  
                height: 20,  
  
            ),
```

```
Padding(  
    //Button actions  
    padding: const EdgeInsets.only(left: 20, right: 20),  
    child: Column(children: [  
        SizedBox(  
            width: double.infinity,  
            child: ElevatedButton(  
                style:  
                    Theme.of(context).elevatedButtonTheme.style,  
                onPressed: () => (Navigator.push( //Button to go  
                    to profile creation page  
                    context,  
                    MaterialPageRoute(  
                        builder: (context) =>  
                            const  
                            AddProfileCreationPageName()))),  
                child: const Text("Create a profile"))),  
        SizedBox(  
            width: 250,  
            child: TextButton(  
                style: Theme.of(context).textButtonTheme.style,  
                onPressed: () => (Navigator.push( //Text button  
                    to go to login page  
                    context,
```

```
        MaterialPageRoute(  
          builder: (context) =>  
            const AddProfileLoginPage()))),  
  
        child: const Align(  
          alignment: Alignment.center,  
          child: Text(  
            "I already have a profile",  
            textAlign: TextAlign.center,  
          ))))  
      ],  
    ),  
    const SizedBox(  
      height: 40,  
    ),  
  ],  
))  
],  
)  
]);  
);  
}  
}  
}
```

restaurant_customise.dart

```
import 'dart:math';

import 'package:alleat/services/cart_service.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

class RestaurantItemCustomisePage extends StatefulWidget {

    final String itemid;

    final String foodcategory;

    final String subfoodcategory;

    final String itemname;

    final String description;

    final String price;

    final String itemimage;

    final String reslogo;

    const RestaurantItemCustomisePage(
        {Key?

            key, //Get the items from the restaurant main page when an item is
            clicked

            required this.reslogo,
```

```

    required this.itemid,
    required this.foodcategory,
    required this.subfoodcategory,
    required this.itemname,
    required this.description,
    required this.price,
    required this.itemimage})
    : super(key: key);

}

@Override
State<RestaurantItemCustomisePage> createState() =>
    _RestaurantItemCustomisePageState();
}

class _RestaurantItemCustomisePageState
    extends State<RestaurantItemCustomisePage> {
    int quantity = 1; //default quantity 1
    Map customisedOptions = {};//changes to the item go in here in a dictionary
    bool beenMade = false; //Used to only build customisedOptions once
    double changingPrice = 0; //How much is the original price affected by
    customised options
    List requiredFields = [];//Which customsie options are required to be filled
    late double quantityPrice = double.parse(widget.price);
}

```

```
late double finalSinglePrice = double.parse(widget.price);
```

```
Future<bool> addToCart() async { //Add item with cutomsie options to db
```

```
try {
```

```
    String customisedOptionsEncoded = json.encode(customisedOptions); //Encode
```

```
Dictionary into a string
```

```
    bool isAdded = await SQLiteCartItem.addCartItem(
```

```
        int.parse(widget.itemid), customisedOptionsEncoded, quantity);
```

```
    if (isAdded){
```

```
        return true;
```

```
    }
```

```
    else{
```

```
        return true; //If no error, return true
```

```
    }
```

```
} catch (e) {
```

```
    return false; //If error, return false
```

```
}
```

```
}
```

```
Future<Map> getCustomiseOptions() async {
```

```
try {
```

```
    String phpurl = "https://alleat.cpur.net/query/itemcustomise.php";
```

```
    var res =
```

```
    await http.post(Uri.parse(phurl), body: {"itemid": widget.itemid});

    if (res.statusCode == 200) {

        //If sends successfully

        var data = json.decode(res.body); //Decode to array

        if (data["error"]) {

            Map error = {

                "error": true,

                "message": "Server Error: ${data["message"]}",

                "customise": "[]"

            } //Send blank list of customise data

            ;

            return error;

        //If fails to perform query

    } else {

        Map success = {

            "error": false,

            "message": "",

            "customise": data["customiseitem"]

        } //Send back returned data

        ;

        return success;

    }

}
```

```
        } else {

            Map error = {

                "error": true,

                "message": "Error $e: Please try again",

                "customise": "[]"

            } //Send blank list of customise data

            ;

            return error;

        }

    } catch (e) {

        Map error = {

            "error": true,

            "message": "Unexpected Error: $e",

            "customise": "[]"

        } //Send blank list of customise data

        ;

        return error;

    }

}

@Override

Widget build(BuildContext context) {
```

```
return Scaffold(  
    body: SingleChildScrollView(  
        child: Column(children: [  
            Stack(children: [  
                //Display item image at bottom of stack  
                Container(  
                    width: MediaQuery.of(context).size.width,  
                    height: 240,  
                    decoration: BoxDecoration(  
                        image: DecorationImage(  
                            fit: BoxFit.fitWidth,  
                            image: NetworkImage(widget.itemimage),  
                        ),  
                    ),  
                ),  
                Align(  
                    //Display rounded container at bottom of image to respresent  
                    //overhanging image  
                    alignment: Alignment.bottomCenter,  
                    child: Container(  
                        margin: const EdgeInsets.only(top: 215),  
                        width: double.infinity,  
                        height: 30,  
                    ),  
                ),  
            ],  
        ),  
    ),  
);
```

```
decoration: BoxDecoration(  
    color: Theme.of(context).backgroundColor,  
    borderRadius:  
        const BorderRadius.vertical(top: Radius.circular(20)),  
)  
  
Align(  
    // Display background color as outline of restaurant logo,  
    overlapping the image background  
    alignment: Alignment.bottomCenter,  
    child: Container(  
        margin: const EdgeInsets.only(top: 180),  
        width: 80,  
        height: 80,  
        decoration: BoxDecoration(  
            shape: BoxShape.circle,  
            color: Theme.of(context).backgroundColor,  
        )),  
    Align(  
        // Display circle restaurant logo  
        alignment: Alignment.bottomCenter,  
        child: Container(  
            margin: const EdgeInsets.only(top: 185),  
            width: 70,
```

```
height: 70,  
  
decoration: BoxDecoration(  
  
    image: DecorationImage(  
  
        fit: BoxFit.cover,  
  
        image: NetworkImage(widget.reslogo.toString()),  
  
    ),  
  
    shape: BoxShape.circle,  
  
    color: Theme.of(context).colorScheme.onSurface,  
  
)),  
  
SafeArea(  
  
    // Display back button in a circle  
  
    child: InkWell(  
  
        onTap: () => Navigator.of(context).pop(),  
  
        child: Container(  
  
            margin: const EdgeInsets.only(top: 20, left: 20),  
  
            width: 50,  
  
            height: 50,  
  
            decoration: BoxDecoration(  
  
                shape: BoxShape.circle,  
  
                color: Theme.of(context).colorScheme.onSurface,  
  
            ),  
  
            child: Icon(  
  
        ),  
  
    ),  
  
),
```

```
        Icons.chevron_left,  
        color: Theme.of(context).colorScheme.onBackground,  
        size: 35,  
,  
)),  
]),  
Padding(  
    // Item name  
    padding:  
        const EdgeInsets.only(top: 40, left: 30, right: 30, bottom: 10),  
    child: Text(  
        widget.itemname,  
        textAlign: TextAlign.center,  
        style: Theme.of(context).textTheme.headline2,  
Row(mainAxisAlignment: MainAxisAlignment.center, children: [  
    //Display restuarant category(ies)  
    Text(widget.foodcategory, //Must have food category  
        textAlign: TextAlign.center,  
        style: Theme.of(context)  
            .textTheme  
            .headline6!
```

```
.copyWith(color: Theme.of(context).textTheme.headline1!.color)),  
  
LayoutBuilder(builder: (context, constraints) {  
  
if (widget.subfoodcategory != "") {  
  
//If there is a subcategory, display it with a dot next to it  
  
//If there is a sub food category, show it  
  
return Text(" · ${widget.subfoodcategory}",  
  
textAlign: TextAlign.center,  
  
style: Theme.of(context).textTheme.headline6!.copyWith(  
  
color: Theme.of(context).textTheme.headline1!.color));  
  
} else {  
  
// If there is no sub-category, dont display anything  
  
//If there isnt a sub-food category, dont show it  
  
return const Text("");  
  
}  
  
})  
  
]),  
  
Padding(  
  
//Display item description  
  
padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 20),  
  
child: Text(  
  
widget.description,  
  
textAlign: TextAlign.center,
```

```
style: Theme.of(context).textTheme.bodyText1!.copyWith(
    color: Theme.of(context).textTheme.headline6!.color,
    fontWeight: FontWeight.w400),
),
),
FutureBuilder<Map>(
    future: getCustomiseOptions(), //Get customise options from server
    builder: (context, snapshot) {
        if (snapshot.hasData) {
            Map customiseData = snapshot.data ?? [] as Map;
            if (customiseData["error"] == true) {
                //Check if there was an error getting the data from the server
                return Container(
                    width: double.infinity,
                    padding:
                        const EdgeInsets.symmetric(vertical: 30, horizontal: 20),
                    decoration: BoxDecoration(
                        borderRadius: const BorderRadius.all(Radius.circular(10)),
                        color: Theme.of(context).colorScheme.onSurface),
                    margin:
                        const EdgeInsets.symmetric(vertical: 5, horizontal: 30),
                    child: Text(

```

```

        customiseData["message"], //Return that there was an error

        textAlign: TextAlign.center,

        style: Theme.of(context).textTheme.headline6,

    ),

);

} else {

    if (beenMade == false) { //If the customised dictionary has not
been made

        for (int i = 0; i < customiseData["customise"].length; i++) {
// For each customise option, add it as key with an empty list as the value

        customisedOptions[customiseData["customise"][[i][0]]] = [];

    }

    beenMade = true; //Set has been built to true

}

}

//If there was not an error getting the data

if (customiseData["customise"].length == 0) {

    //if there is no customise options, return nothing

    return const SizedBox(

        height: 1,

    );

} else {

    //If there are customise options, return customise options

```

```

return ListView.builder(
  physics:
    const NeverScrollableScrollPhysics(), //Disable
scrolling. Scroll with whole page

  scrollDirection: Axis.vertical,
  shrinkWrap: true,
  itemCount: (customiseData[
    "customise"] //For each customise section, build
    .length),
  itemBuilder: ((context, index) {
    if (customiseData["customise"][index][4] == "1" &&
      !requiredFields.contains(
        customiseData["customise"][index][0])) { //For
each section marked as 1 (required), add customise id to required list. Check if
it has not been added before.

      requiredFields
        .add(customiseData["customise"][index][0]);
    }
  return Column( //Customise section
    children: [
      Container( //Thick border above each section to
separate each section.

        width: double.infinity,
        height: 10,

```

```
        color: Theme.of(context)

        .colorScheme

        .onBackground

        .withOpacity(0.1),

    ),

LayoutBuilder(builder: ((context, constraints) {

    if (customiseData["customise"][index][3] ==

        "SELECT") { //If the section is marked as a
select widget

    return Column(
        children: [
            Padding(
                padding: const EdgeInsets.only(
                    left: 20,
                    right: 20,
                    top: 30,
                    bottom: 10),
                child: Row( //Top row containing the title,
description and container indicating if it is required or optional
                    crossAxisAlignment:
CrossAxisAlignment.start,
                    children: [
                        Expanded( //Go to next line if too
long
```

```
        child: Column(  
  
          mainAxisAlignment:  
            MainAxisAlignment  
              .start,  
  
          children: [  
  
            Text( //Customise section title  
              customiseData["customise"]  
                [index][1]  
  
                .toString(),  
  
              style: Theme.of(context)  
                .textTheme  
                  .headline5,  
  
              overflow:  
                TextOverflow.visible,  
  
            ),  
  
            const SizedBox(  
              height: 10,  
  
            ), //Customise section  
description  
  
            Text(  
              customiseData["customise"]  
                [index][2],  
  
              style: Theme.of(context)
```

```
        .textTheme  
        .headline5,  
  
        overflow:  
        TextOverflow.visible,  
  
    )  
  
]),  
  
const SizedBox(width: 30),  
  
LayoutBuilder(builder:  
  
((context, constraints) {  
  
if (customiseData["customise"]  
  
[index][4] ==  
  
"0") { //If the customise  
section is not required, display container in green with text "optional"  
  
return Container(  
  
decoration: BoxDecoration(  
  
color: Theme.of(context)  
  
.colorScheme  
  
.tertiary  
  
.withOpacity(0.2),  
  
borderRadius:  
  
BorderRadius.circular(  
  
20)),  
  
padding: const EdgeInsets
```

```
        .symmetric(
            horizontal: 20,
            vertical: 5),
        child: Text(
            "Optional",
            style: Theme.of(context)
                .textTheme
                .headline6
            ?.copyWith(
                color: Theme.of(
                    context)
                .colorScheme
                .tertiary),
        ),
    );
}

} else if (customiseData[
    "customise"][index][4] ==
    "1") { //If the customise
section is marked as required, display container in primary colour (purple) with
text "required"

return Container(
    decoration: BoxDecoration(
        color: Theme.of(context)
```

```
        .primaryColor  
        .withOpacity(0.2),  
  
        borderRadius:  
        BorderRadius.circular(  
            20)),  
  
        padding: const EdgeInsets  
        .symmetric(  
            horizontal: 20,  
            vertical: 5),  
  
        child: Text(  
            "Required",  
            style: Theme.of(context)  
            .textTheme  
            .headline6  
            ?.copyWith(  
                color: Theme.of(  
                    context)  
                .primaryColor),  
            ),  
        );  
  
    } else { //If the section is not  
marked as 0 or 1, display it as and error with a red container and text "ERROR"  
    return Container(  
       
```

```
decoration: BoxDecoration(  
    color: Theme.of(context)  
        .colorScheme  
        .error  
        .withOpacity(0.2),  
    borderRadius:  
        BorderRadius.circular(  
            20)),  
    padding: const EdgeInsets  
        .symmetric(  
            horizontal: 20,  
            vertical: 5),  
    child: Text(  
        "ERROR",  
        style: Theme.of(context)  
            .textTheme  
            .headline6  
            ?.copyWith(  
                color: Theme.of(  
                    context)  
                    .colorScheme  
                    .error),
```

```
        ),  
        );  
    }  
})  
],  
,  
),  
ListView.builder( //For each option on the  
customise section  
    physics:  
        const NeverScrollableScrollPhysics(),  
//Disable scrolling. Scroll with whole page  
    scrollDirection: Axis.vertical,  
    shrinkWrap: true,  
    itemCount: customiseData["customise"]  
        [index][7]  
        .length,  
    itemBuilder: ((context, index2) {  
        return Padding(  
            padding: const  
EdgeInsets.symmetric(  
                horizontal: 20, vertical: 5),  
            child: ElevatedButton( //Create a  
button for the option  
                style: ButtonStyle(  

```

```

        side:
(customisedOptions[customiseData["customise"][[index][0]].contains(
customiseData["customise"]
                                [index][7]
                                [index2][0])))

//If the option is in the customisedOptions (selected) then display with purple
border otherwise don't have a border

?

MaterialStateProperty.all(BorderSide(
                                width: 2,
                                color:
Theme.of(context)
                                .primaryColor))

                                : null,
                                alignment:
                                Alignment.centerLeft,
                                padding:
MaterialStateProperty.all(const EdgeInsets.symmetric(horizontal: 30, vertical:
20)),

                                textStyle:
(customisedOptions[customiseData["customise"][[index][0]].contains(customiseData["
customise"][[index][7][index2][0]])) //If the option is in the customisedOptions
(selected), display with brighter text

?

MaterialStateProperty.all(Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).textTheme.headline1?.color))

```

```
:  
MaterialStateProperty.all(Theme.of(context).textTheme.headline6),  
                                backgroundColor:  
MaterialStateProperty.all(Theme.of(context).colorScheme.onSurface)),  
onPressed: () {  
  
    if  
(customisedOptions[customiseData["customise"]][index][0])  
  
.length <  
  
int.parse(  
  
customiseData["customise"]  
  
[index]  
  
[6]) &&  
  
!customisedOptions[  
  
customiseData["customise"]  
  
[index][0]]  
  
.contains(  
  
customiseData["customise"]  
  
[index][7]  
  
[index2][0]))  
{ //If the length is less than the max amount selected and it is not in the  
customisedoptions list add it to the list and add to the changed price  
  
setState(() {
```

```
        customisedOptions[  
            customiseData[  
                "customise"]  
                    [index][0]]  
                .add(customiseData[  
                    "customise"]  
                        [index][7]  
                    [index2][0]);  
            changingPrice += double  
                .parse(customiseData[  
                    "customise"]  
                        [index][7]  
                    [index2][3]);  
            });  
        } else if (customisedOptions[  
            customiseData[  
                "customise"]  
                    [index][0]]  
                .contains(customiseData[  
                    "customise"]  
                        [index][7])
```

```

        [index2][0])) { //If
the option is in the customisedoptions list, remove from the list and remove to
the changed price

        setState(() {
            customisedOptions[
                customiseData[
                    "customise"]
                [index][0]]
            .remove(customiseData[
                "customise"]
                [index][
                    7][index2][0]
                .toString());
            changingPrice -= double
                .parse(customiseData[
                    "customise"]
                [index][7]
                [index2][3]);
        });
    }
},

```

```
        child: Row(mainAxisAlignment:  
MainAxisAlignment.spaceBetween, children: [ //Within button, display a row  
containing the option text and the price change  
  
        Expanded(  
  
            child: Text(  
  
                customiseData["customise"]  
  
                [index][7][index2][1],  
  
                style: (customisedOptions[  
  
customiseData["customise"]  
  
                [index]  
  
                [0]]  
  
.contains(customiseData["customise"]  
  
                [index][7]  
  
                [index2][0]))  
//If the option is in the customisedOptions (selected), display with brighter  
text  
  
            ? Theme.of(context)  
  
                .textTheme  
  
                .headline6  
  
                ?.copyWith(  
  
                    color:  
Theme.of(context)  
  
                    .textTheme
```

```

        .headline1

        ?.color)

        : Theme.of(context)

        .textTheme

        .headline6,

        )),

LayoutBuilder(
    builder: ((context,
constraints) {

    if (customiseData[

"customise"]

        [index][7]

        [index2][3] !=

        "0.00") { //If the
price change is not nothing, display the price change in red

    return Text(
        "+\${customiseData["customise"][index][7][index2][3]}",
        style: Theme.of(
            context)

        .textTheme

        .headline6

        ?.copyWith(

```

```
        color:  
Theme.of(  
  
context)  
  
.colorScheme  
  
        .error),  
  
);  
  
} else { //If the price  
change in nothing, display dash in grey  
  
return Text(  
  
" - ",  
  
style: Theme.of(  
  
context)  
  
.textTheme  
  
.headline6  
  
? .copyWith(  
  
color:  
Theme.of(  
  

```

```
.withOpacity(  
    0.5)),  
);  
}  
});  
]);  
}),  
const SizedBox(height: 50)  
],  
);  
} else if (customiseData["customise"][index][3] ==  
"ADD") {  
return Column(  
children: [  
Padding(  
padding: const EdgeInsets.only(  
left: 20,  
right: 20,  
top: 30,  
bottom: 10),  
child: Row( //Top row containing the title,  
description and container indicating if it is required or optional
```

```
crossAxisAlignment:  
    CrossAxisAlignment.start,  
  children: [  
    Expanded( //Go to next line if too  
long  
      child: Column(  
        crossAxisAlignment:  
          CrossAxisAlignment  
            .start,  
        children: [  
          Text( //Customise section title  
            customiseData["customise"]  
              [index][1]  
              .toString(),  
            style: Theme.of(context)  
              .textTheme  
                .headline5,  
            overflow:  
              TextOverflow.visible,  
        ),  
        const SizedBox(  
          height: 10,
```

```
        ), //Customise section  
description  
  
        Text(  
  
            customiseData["customise"]  
  
            [index][2],  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline6,  
  
            overflow:  
  
                TextOverflow.visible,  
  
            ),  
  
            const SizedBox(  
  
                height: 20,  
  
            ),  
  
            ])),  
  
            const SizedBox(width: 30),  
  
            LayoutBuilder(builder:  
  
                ((context, constraints) {  
  
                    if (customiseData["customise"]  
  
                        [index][4] ==  
  
                        "0") { //If the customise  
section is not required, display container in green with text "optional"  
  
                    return Container(  
            
```

```
decoration: BoxDecoration(  
    color: Theme.of(context)  
        .colorScheme  
        .tertiary  
        .withOpacity(0.2),  
    borderRadius:  
        BorderRadius.circular(  
            20)),  
    padding: const EdgeInsets  
        .symmetric(  
            horizontal: 20,  
            vertical: 5),  
    child: Text(  
        "Optional",  
        style: Theme.of(context)  
            .textTheme  
            .headline6  
            ?.copyWith(  
                color: Theme.of(  
                    context)  
                    .colorScheme  
                    .tertiary),
```

```
        ),  
    );  
  
} else if (customiseData[  
    "customise"][[index][4] ==  
    "1"]) { //If the customise  
section is marked as required, display container in primary colour (purple) with  
text "required"  
  
    return Container(  
        decoration: BoxDecoration(  
            color: Theme.of(context)  
                .primaryColor  
                .withOpacity(0.2),  
            borderRadius:  
                BorderRadius.circular(  
                    20)),  
        padding: const EdgeInsets  
            .symmetric(  
                horizontal: 20,  
                vertical: 5),  
        child: Text(  
            "Required",  
            style: Theme.of(context)  
                .textTheme
```

```
        .headline6

        ?.copyWith(
            color: Theme.of(
                context)

            .primaryColor),

        ),
    );

} else { //If the section is not
marked as 0 or 1, display it as and error with a red container and text "ERROR"

    return Container(
        decoration: BoxDecoration(
            color: Theme.of(context)
                .colorScheme
                .error
                .withOpacity(0.2),

        borderRadius:
            BorderRadius.circular(
                20)),

        padding: const EdgeInsets
            .symmetric(
                horizontal: 20,
                vertical: 5),

        child: Text(

```

```
        "ERROR",

        style: Theme.of(context)

            .textTheme

            .headline6

            ?.copyWith(

                color: Theme.of(

                    context)

                .colorScheme

                .error),

            ),

        );

    }

}

}())

]),

),

ListView.builder( //For each option on the
customise section

physics:

const NeverScrollableScrollPhysics(),

//Disable scrolling. Scroll with whole page

scrollDirection: Axis.vertical,

shrinkWrap: true,

itemCount: customiseData["customise"]
```

```
[index][7]

        .length,

        itemBuilder: ((context, index2) {

            return Padding(
                padding:
                    const EdgeInsets.symmetric(
                        horizontal: 20,
                        vertical: 5),
                child: ElevatedButton( //Create a
button for the option

                    style: ButtonStyle(
                        side:
                            (customisedOptions[customiseData["customise"]][index][0]).contains(customiseData["customise"][index][7][index2][0])) //If the option is in the customisedOptions (selected) then display with purple border otherwise don't have a border
                            ?
MaterialStateProperty.all(BorderSide(
                            width: 2,
                            color:
                                Theme.of(context)
                                    .primaryColor))
                            : null,
                    alignment:
                        Alignment.centerLeft,
```

```
padding:  
MaterialStateProperty.all(  
    const  
    EdgeInsets.symmetric(  
        horizontal: 30,  
        vertical: 20)),  
    textStyle:  
(customisedOptions[customiseData["customise"][[index][0]]].contains(customiseData["  
customise"][[index][7][index2][0]])) //If the option is in the customisedOptions  
(selected), display with brighter text  
?  
MaterialStateProperty.all(Theme.of(context).textTheme.headline6?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color))  
:  
MaterialStateProperty.all(Theme.of(context).textTheme.headline6),  
    backgroundColor:  
MaterialStateProperty.all(Theme.of(context).colorScheme.onSurface)),  
    onPressed: () {  
        if  
(customisedOptions[customiseData["customise"][[index][0]]]  
        .length <  
int.parse(customiseData[  
    "customise"]  
    [index][6])) &&  
    !customisedOptions[
```

```
customiseData["customise"]  
[index][0]]  
.contains(  
  
customiseData["customise"]  
[index][7]  
[index2])) {  
//If the length is less than the max amount selected and it is not in the  
customisedoptions list add it to the list and add to the changed price  
setState(() {  
customisedOptions[  
customiseData[  
"customise"]  
[index][0]]  
.add(customiseData[  
"customise"]  
[index][7]  
[index2][0]);  
changingPrice += double  
.parse(customiseData[  
"customise"]
```

```

        [index][7]

        [index2][3]));

    });

    },

    child: LayoutBuilder(builder:

        ((context, constraints) {

            int count = 0;

            for (int i = 0;

                i <

                customisedOptions[

                    customiseData[

"customise"]

                    [

                    index][0]]


.length;

                    i++) { //Count the number
of times, the customise option is mentioned in the customised list. This will get
the quantity.

            if (customisedOptions[

                customiseData[

"customise"]

                    [

```

```
index][0]][i] ==  
customiseData[  
    "customise"]  
[index][7]  
[index2][0])) {  
    count += 1;  
}  
}  
  
if (count == 0) { //If the  
option is not in the list (not selected)  
  
    return Row(  
        mainAxisAlignment:  
            MainAxisAlignment  
            .spaceBetween,  
        children: [ //Display row  
with icon, customise option text and price  
  
Icon(  
    Icons  
  
.add_box_outlined,  
        color: Theme.of(  
            context)  
        .colorScheme  
        .tertiary,  
    )
```

```
        ),  
  
        const SizedBox(  
            width: 30),  
  
        Expanded(  
            child: Text(  
                customiseData[  
  
"customise"]  
                    [index][7]  
                    [index2][1],  
                    style:  
(customisedOptions[  
  
customiseData["customise"][index]  
                    [0]]  
  
.contains(customiseData["customise"][index][7]  
  
[index2]  
                    [0]))  
//If the option is in the list, display it in a brighter text  
?  
Theme.of(context)  
            .textTheme  
            .headline6
```

```
? .copyWith(  
    color:  
Theme.of(context)  
  
.textTheme  
  
.headline1  
  
? .color)  
  
:  
Theme.of(context)  
  

```

```

        index2][3] !=

                "0.00") { //If
the price change is not nothing, display the price change in red

                return Text(


" + ${customiseData["customise"][index][7][index2][3]}",

                style: Theme.of(
context)

                .textTheme
                .headline6
                ?.copyWith(
color:

Theme.of(context)

.colorScheme

.error),

);

} else { //If the
price change is nothing, display dash in grey

                return Text(

" - ",

style: Theme.of(
context)

                .textTheme

```

```
        .headline6

        ?.copyWith(
            color:
Theme.of(context)

.colorScheme

.onBackground

.withOpacity(0.5)),

);

}

}))

]);

} else { //If the option is in
the list

return Row( //Display row
with the quantity, text and a delete button

mainAxisAlignment:
MainAxisAlignment

.spaceBetween,

children: [
CircleAvatar(
radius: 15,
backgroundColor:
```

```
Theme.of(  
    context)  
    .colorScheme  
    .tertiary  
    .withOpacity(  
        0.5),  
    child: Text(  
        count  
        .toString(),  
        style: Theme.of(  
            context)  
            .textTheme  
            .headline6  
            ?.copyWith(  
                color:  
Theme.of(context)  
  
.textTheme  
  
.headline1  
  
?.color)),  
,  
const SizedBox(  
)
```

```
        width: 30,  
    ),  
    Expanded(  
        child: Text(  
            customiseData[  
  
"customise"]  
                [index][7]  
                [index2][1],  
                style:  
(customisedOptions[  
  
customiseData["customise"][index]  
                [0]]  
  
.contains(customiseData["customise"][index][7]  
  
[index2]  
                [0]))  
//If the option is in the customisedOptions list, display the text brighter  
?  
Theme.of(context)  
        .textTheme  
        .headline6  
        ?.copyWith(  
    )
```

```
        color:  
Theme.of(context)  
  
.textTheme  
  
.headline1  
  
?.color)  
  
:  
Theme.of(context)  
  
.textTheme  
.headline6,  
)),  
const SizedBox(  
width: 20),  
 IconButton( //Delete  
button  
icon: const Icon(  
Icons.delete),  
splashRadius: 15,  
color: Theme.of(  
context)  
.colorScheme  
.error,
```

```
onPressed: () { //On
remove button pressed, remove from the list and remove from changed price.

        setState(() {

customisedOptions[

customiseData["customise"][index]

[0]]


.remove(customiseData["customise"][index][7]

[

index2][0]

.toString());



changingPrice -=

double.parse(customiseData["customise"]

[

index][7]

[

index2][3]);



});,



},
```

```
        ),  
        ]);  
  
    }  
});  
  
},  
const SizedBox(height: 50)  
],  
);  
}  
else if (customiseData["customise"][index][3] ==  
"REMOVE") {  
return Column(  
children: [  
Padding(  
padding: const EdgeInsets.only(  
left: 20,  
right: 20,  
top: 30,  
bottom: 10),  
child: Row( //Top row containing the title,  
description and container indicating if it is required or optional  
crossAxisAlignment:  
CrossAxisAlignment.start,
```

```
        children: [  
  
            Expanded( //Go to next line if too  
long  
  
                child: Column(  
  
                    mainAxisAlignment:  
  
                        CrossAxisAlignment.start  
  
                    .start,  
  
                    children: [  
  
                        Text( //Customise section title  
  
                            customiseData["customise"]  
  
                            [index][1]  
  
                            .toString(),  
  
                            style: Theme.of(context)  
  
                                .textTheme  
  
                                    .headline5,  
  
                            overflow:  
  
                                TextOverflow.visible,  
  
                            ),  
  
                        const SizedBox(  
  
                            height: 10,  
  
                        ), //Customise section  
description  
  
                        Text(  
)]  
);
```

```
customiseData["customise"]  
[index][2],  
style: Theme.of(context)  
.textTheme  
.headline6,  
overflow:  
TextOverflow.visible,  
),  
const SizedBox(  
height: 20,  
),  
]),  
const SizedBox(width: 30),  
LayoutBuilder(builder:  
((context, constraints) {  
if (customiseData["customise"]  
[index][4] ==  
"0") { //If the customise  
section is not required, display container in green with text "optional"  
return Container(  
decoration: BoxDecoration(  
color: Theme.of(context)  
.colorScheme
```

```
        .tertiary

        .withOpacity(0.2),

        borderRadius:

        BorderRadius.circular(
            20)),

padding: const EdgeInsets

        .symmetric(
            horizontal: 20,
            vertical: 5),

child: Text(
    "Optional",
    style: Theme.of(context)

        .textTheme
        .headline6

        ?.copyWith(
            color: Theme.of(
                context)

        .colorScheme
        .tertiary),
        ),
    );
}

} else if (customiseData[
```

```
"customise"][[index][4] ==  
    "1") { //If the customise  
section is marked as required, display container in primary colour (purple) with  
text "required"  
  
        return Container(  
  
            decoration: BoxDecoration(  
  
                color: Theme.of(context)  
  
                    .primaryColor  
  
                    .withOpacity(0.2),  
  
                borderRadius:  
  
                    BorderRadius.circular(  
  
                        20)),  
  
            padding: const EdgeInsets  
  
                .symmetric(  
  
                    horizontal: 20,  
  
                    vertical: 5),  
  
            child: Text(  
  
                "Required",  
  
                style: Theme.of(context)  
  
                    .textTheme  
  
                    .headline6  
  
                    ?.copyWith(  
  
                        color: Theme.of(  
)
```

```
        context)

        .primaryColor),

    ),

);

} else { //If the section is not
marked as 0 or 1, display it as and error with a red container and text "ERROR"

return Container(
    decoration: BoxDecoration(
        color: Theme.of(context)
            .colorScheme
            .error
            .withOpacity(0.2),
        borderRadius:
            BorderRadius.circular(
                20)),
    padding: const EdgeInsets
        .symmetric(
            horizontal: 20,
            vertical: 5),
    child: Text(
        "ERROR",
        style: Theme.of(context)
            .textTheme
```

```
        .headline6

        ?.copyWith(
            color: Theme.of(
                context)

            .colorScheme

            .error),

        ),
    );
}

}))

],
),
ListView.builder( //For each option on the
customise section

physics:

const NeverScrollableScrollPhysics(),
//Disable scrolling. Scroll with whole page

scrollDirection: Axis.vertical,
shrinkWrap: true,
itemCount: customiseData["customise"]

[index][7]

.length,
itemBuilder: ((context, index2) {
```

```
        return Padding(  
  
            padding:  
  
                const EdgeInsets.symmetric(  
  
                    horizontal: 20,  
  
                    vertical: 5),  
  
                    child: ElevatedButton( //Create a  
button for the option  
  
                        style: ButtonStyle(  
  
                            side:  
(customisedOptions[customiseData["customise"][[index][0]].contains(customiseData["  
customise"][[index][7][index2][0]])) //If the option is in the customisedOptions  
(selected) then display with purple border otherwise don't have a border  
  
?  
MaterialStateProperty.all(BorderSide(  
  
                                width: 2,  
  
                                color:  
Theme.of(context)  
  
.primaryColor))  
  
: null,  
  
alignment:  
Alignment.centerLeft,  
  
padding:  
MaterialStateProperty.all(  
  
const  
EdgeInsets.symmetric(  
  
horizontal: 30,
```

```

vertical: 20)),


        textStyle:
(customisedOptions[customiseData["customise"][[index][0]].contains(customiseData["customise"][[index][7][index2][0]])) //If the option is in the customisedOptions (selected), display with brighter text


?


MaterialStateProperty.all(Theme.of(context).textTheme.headline6?.copyWith(color:
Theme.of(context).textTheme.headline1?.color))

:


MaterialStateProperty.all(Theme.of(context).textTheme.headline6),


backgroundColor:
MaterialStateProperty.all(Theme.of(context).colorScheme.onSurface)),


onPressed: () {




if
(customisedOptions[customiseData["customise"][[index][0]]


.length <




int.parse(customiseData[



"customise"]


[index][6]) &&
!customisedOptions[


customiseData["customise"]


[index][0]]


.contains(


customiseData["customise"]


]
```

```
[index][7]

        [index2])) {
//If the length is less than the max amount selected and it is not in the
customisedoptions list add it to the list and add to the changed price

        setState(() {

            customisedOptions[
                customiseData[

"customise"]

                [index][0]]

            .add(customiseData[

"customise"]

                [index][7]

                [index2][0]));

            changingPrice -= double

            .parse(customiseData[

"customise"]

                [index][7]

                [index2][3]));

        });

    },

```

```

        child: LayoutBuilder(builder:

            ((context, constraints) {

                int count = 0;

                for (int i = 0;

                     i <

                     customisedOptions[

                         customiseData[

"customise"]

                         [

                             index][0]]]

                     .length;

                     i++)) { //Count the number
of times, the customise option is mentioned in the customised list. This will get
the quantity.

                if (customisedOptions[

                    customiseData[

"customise"]

                    [

                        index][0]][i] ==

                    customiseData[

"customise"]

                    [index][7]

                    [index2][0])) {

```

```

        count += 1;

    }

}

if (count == 0) { //If the
option is not in the list (not selected)

    return Row(
        mainAxisAlignment:
MainAxisAlignment
        .spaceBetween,
        children: [ //Display row
with icon, customise option text and price

Icon(
Icons

.disabled_by_default_outlined,
        color: Theme.of(
context)
        .colorScheme
        .error,
),
const SizedBox(
        width: 30),
Expanded(
        child: Text(

```

```
customiseData[  
  
"customise"]  
  
[index][7]  
  
[index2][1],  
  
style:  
(customisedOptions[  
  
customiseData["customise"][index]  
  
[0]]  
  
.contains(customiseData["customise"][index][7]  
  
[index2]  
  
[0]))  
//If the option is in the list, display it in a brighter text  
  
?  
Theme.of(context)  
  
.textTheme  
  
.headline6  
  
? .copyWith(  
  
color:  
Theme.of(context)  
  

```

```
? .color)

:

Theme.of(context)

    .textTheme

    .headline6,

)),

const SizedBox(

    width: 20),

LayoutBuilder(builder:

    ((p0, p1) {

        if (customiseData[

"customise"]

[

index][7]

[

index2][3] !=

"0.00") { //If

the price change is not nothing, display the price change in red

            return Text(


" -


${customiseData["customise"][[index][7][index2][3]]}",

style: Theme.of(
```

```
        context)

        .textTheme

        .headline6

        ?.copyWith(

            color:

Theme.of(context)

        .colorScheme

        .tertiary),

    );
}

} else { //If the
price change is nothing, display dash in grey

return Text(

    "-",

    style: Theme.of(
        context)

        .textTheme

        .headline6

        ?.copyWith(

            color:

Theme.of(context)

        .colorScheme
```

```
.onBackground

.withOpacity(0.5)),

);

}

}))

]);

} else { //If the option is in
the list

return Row( //Display row
with the quantity, text and a delete button

mainAxisAlignment:

MainAxisAlignment

.spaceBetween,

children: [

CircleAvatar(

radius: 15,

backgroundColor:

Theme.of(
context)

.colorScheme

.error

.withOpacity(
```

```
        0.5),  
  
        child: Text(  
  
            count  
  
                .toString(),  
  
            style: Theme.of(  
  
                context)  
  
.textTheme  
  
.headline6  
  
?.copyWith(  
  
    color:  
Theme.of(context)  
  

```

```
"customise"]  
[index][7]  
[index2][1],  
style:  
(customisedOptions[  
  
customiseData["customise"][index]  
[0]]  
  
.contains(customiseData["customise"][index][7]  
[index2]  
[0]))  
//If the option is in the customisedOptions list, display the text brighter  
?  
Theme.of(context)  
.  
.textTheme  
.headline6  
?.copyWith(  
color:  
Theme.of(context)  
  
.textTheme  
  
.headline1
```

```
? .color)

:

Theme.of(context)

    .textTheme

    .headline6,

)),

const SizedBox(

    width: 20),

IconButton(

    icon: const Icon(
        Icons.delete),

    splashRadius: 15,

    color: Theme.of(
        context)

    .colorScheme

    .error,

    onPressed: () { //On
remove button pressed, remove from the list and add from changed price.

        setState(() {

customisedOptions[

customiseData["customise"][index]
```

```
[0]]  
  
.remove(customiseData["customise"][index][7]  
[  
  
index2][0]  
  
.toString());  
changingPrice +=  
  
double.parseDouble(customiseData["customise"]  
[  
index][7]  
[  
  
index2][3]);  
});  
},  
),  
]);  
}  
})),  
));  
})),
```

```
        const SizedBox(height: 50)

    ],
);

} else { //If there is an unknown customise type
(not SELECT, ADD or REMOVE) dispaly container with the text "Unknown Customise
Option"

return Container(
    width: double.infinity,
    padding: const EdgeInsets.symmetric(
        vertical: 30, horizontal: 20),
    decoration: BoxDecoration(
        borderRadius: const BorderRadius.all(
            Radius.circular(10)),
        color: Theme.of(context)
            .colorScheme
            .onSurface),
    margin: const EdgeInsets.symmetric(
        vertical: 5, horizontal: 30),
    child: Text(
        "Unknown Customise Option", //Return that
there was an error

        textAlign: TextAlign.center,
        style:
            Theme.of(context).textTheme.headline6,
```

```

        ),
    );
}

})))
],
);
})));
}

}

} else {

    //While loading, return progress indicator

    return const LinearProgressIndicator(
        color: Color(0xff4100C4), backgroundColor: Color(0xffEBE0FF));
}

}),
//Add to cart and quantity

LayoutBuilder(builder: ((context, constraints) {
    finalSinglePrice =
        ((double.parse(widget.price) * 100) + (changingPrice * 100)) / 100;
    //In order to do correct calculations with a double type, multiply all by 100 to
    get integer and divide back to get it in pence and pounds

    quantityPrice = finalSinglePrice * quantity; //To get the quantity price
    multiply by quantity
}

```

```
return Column(crossAxisAlignment: CrossAxisAlignment.center, children: [  
    Text("£${finalSinglePrice.toStringAsFixed(2)}", //Display individual  
    price by 2dp.  
    style: Theme.of(context).textTheme.headline4),  
    Padding(  
        padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 20),  
        child: Row( //Display row with the quantity and add and subtract  
        buttons  
            mainAxisAlignment: MainAxisAlignment.center,  
            children: [  
                CircleAvatar( //Remove from quantity  
                    radius: 30,  
                    backgroundColor: Theme.of(context).colorScheme.onSurface,  
                    child: IconButton(  
                        icon: const Icon(Icons.remove),  
                        iconSize: 25,  
                        color: (quantity == 1) //if the quantity is 1 (unable to  
                        subtract from quantity), display at 0.1 opacity  
                        ? Theme.of(context).primaryColor.withOpacity(0.1)  
                        : Theme.of(context).primaryColor,  
                        onPressed: () {  
                            if (quantity > 1) { //If the quantity is greater than  
                            1, allow for removing 1 from quantity  
                        }  
                    )  
                )  
            ]  
        )  
    )  
];
```

```
        setState(() {
            quantity -= 1;
        });
    }
},
),
const SizedBox(
    width: 30,
),
SizedBox( //Quanity text with fixed width
    width: 30,
    child: Text(
        quantity.toString(),
        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.headline3,
    )),
const SizedBox(
    width: 30,
),
CircleAvatar( //Add from quantity
    radius: 30,
    backgroundColor: Theme.of(context).colorScheme.onSurface,
```

```
        child: IconButton(


            icon: const Icon(Icons.add),


            iconSize: 25,


            color: (quantity == 99)//if the quantity is 99 (unable to
add to quantity), display at 0.1 opacity


            ? Theme.of(context).primaryColor.withOpacity(0.1)


            : Theme.of(context).primaryColor,


            onPressed: () {

                if (quantity < 99) {//If the quantity is less than 99,
allow for adding 1 to quantity. Max 99 to stop overloading restaurants.


                setState(() {

                    quantity += 1;

                });

            }

        ),

    )),

],

)),

Padding(


padding: const EdgeInsets.all(20),


child: Row(children: [


Expanded(child:


LayoutBuilder(builder: ((context, constraints) { //Cart
```

```
        bool tempCheckRequiredFilled = true; //Check if required is
fullfilled

            for (int i = 0; i < requiredFields.length; i++) { //For each
customise section that is marked as required, check if each customisedOption are
not empty. If it is empty, change tempCheckRequiredFilled to false. Disables add
to cart button

                if (customisedOptions[requiredFields[i]].isEmpty) {

                    tempCheckRequiredFilled = false;

                }

            }

        if (tempCheckRequiredFilled == true) { //if required fields are
filled

            return ElevatedButton(
                onPressed: () async { //On press add to cart (add to
database)

                    bool isAddedToCart = await addToCart();

                    if (isAddedToCart == true) { //If is successfully adds
to cart, display success

                        setState(() {

                            ScaffoldMessenger.of(context).showSnackBar(
                                const SnackBar(
                                    content:
                                        Text("Successfully added to Cart")));
                        });

                    } else { //If fails to add to cart, display failed
                }
            );
        }
    }
}
```

```

        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                    content: Text("Failed to add to Cart")));
        });
    },
    child: Row( //Display row with text "add to cart" and the
price for the quantity
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
            const Text("Add to Cart"),
            Text("£${quantityPrice.toStringAsFixed(2)})")
        )));
} else {
    return Opacity( //Display add to cart button with low opacity
and disable onPressed
        opacity: 0.2,
        child: ElevatedButton(
            style: ButtonStyle(
                backgroundColor: MaterialStatePropertyAll(
                    Theme.of(context)
                        .colorScheme

```

```
        .onBackground())),
    onPressed: () {},
    child: Row(
      mainAxisAlignment:
        MainAxisAlignment.spaceBetween,
      children: [
        const Text("Add to Cart"),
        Text("£${quantityPrice.toStringAsFixed(2)})")
      ]));
  }
}
),
],
const SizedBox(height: 20),
];
})
]
));
}
}
}
```

restaurant_main.dart

```
import 'package:alleat/screens/restaurant/restaurant_customise.dart';
```

```
import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert' as convert;

import 'package:shared_preferences/shared_preferences.dart';

class RestaurantMain extends StatefulWidget {

    final String resid;

    final String resname;

    final String reslogo;

    final String resbanner;

    final String resdistance;

    final String resdelivery;

    final String resordermin;

    const RestaurantMain(
        //Get restaurant info from the restaurant list widget (passed from the
        container)

        {

        Key? key,
        required this.resid,
        required this.resname,
        required this.reslogo,
        required this.resbanner,
```

```

        required this.resdistance,
        required this.resdelivery,
        required this.resordermin,
    }) : super(key: key);

}

@Override
State<RestaurantMain> createState() => _RestaurantMainState();
}

class _RestaurantMainState extends State<RestaurantMain> {

Future<List> getMenuCategories() async {
    String phpurl =
        "https://alleat.cpur.net/query/restaurantmenucategories.php"; //Get the
    list of menu categories associated with the restaurant id

    try {
        var res = await http.post(Uri.parse(phpurl), body: {
            "restaurantid": widget.resid,
        });
        if (res.statusCode == 200) {
            //If successfully sent
            var data = convert.json.decode(res.body); //Decode to array
            if (data["error"]) {
                //If failed to query
            }
        }
    }
}
}

```

```
List error = [  
    {  
        "error": "true",  
        "restaurantcategories": "[]"  
    } //Return no menu categories  
];  
  
return error;  
}  
else {  
    List listdata = [data];  
  
    return listdata;  
}  
}  
else {  
    List error = [  
        {"error": "true", "restaurantcategories": "[]"}  
    ];  
  
    return error;  
}  
}  
}  
catch (e) {  
    List<Map<String, String>> error = [  
        {"error": "true", "restaurantcategories": "[]"}  
    ];  
  
    return error;  
}
```

```
        }

    }

Future<String> favouriteRestaurant(restaurantID, action) async {

    String phpurl =

        "https://alleat.cpur.net/query/favouriterestaurant.php"; //Action of
favourite/unfavourite restaurant by id (Unused in this version)

    final prefs = await SharedPreferences.getInstance();

    final String? email = prefs.getString('email');




try {

    var res = await http.post(Uri.parse(phpurl), body: {

        "action": action.toString(),

        "profileemail": email.toString(),

        "restaurantid": restaurantID,

    });

    if (res.statusCode == 200) {

        var data = convert.json.decode(res.body); //Decode to array

        if (data["error"]) {

            return "failed";

        } else {

            getFavourites();

            return "success";
    }
}
}
```

```

    }

} else {

    return "failed";
}

}

} catch (e) {

    return "failed";
}

}

}

Future<List> getFavourites() async {

String phpurl =

    "https://alleat.cpur.net/query/favouriterestaurantlist.php"; //Get
favourite restaurant list associated with email of profile (unused in this
version)

final prefs = await SharedPreferences.getInstance();

final String? email = prefs.getString('email');

try {

    var res = await http.post(Uri.parse(phpurl), body: {
        "profileemail": email.toString(),
    });

    if (res.statusCode == 200) {

        var data = convert.json.decode(res.body); //Decode to array

        if (data["error"]) {

```

```
List error = [  
    {"error": "true", "favouriterestaurants": "[]"}  
];  
  
return error;  
  
} else {  
  
    List listdata = [data];  
  
    return listdata;  
}  
  
}  
  
} else {  
  
    List error = [  
        {"error": "true", "favouriterestaurants": "[]"}  
    ];  
  
    return error;  
}  
  
}  
  
} catch (e) {  
  
    List<Map<String, String>> error = [  
        {"error": "true", "favouriterestaurants": "[]"}  
    ];  
  
    return error;  
}  
  
}
```

```
Future<List> getMenuItems(categoryid) async {

    String phpurl =
        "https://alleat.cpur.net/query/restaurantmenuitems.php"; //Get list of
    menu items, with the information associated with it. Grabs using the category id
    it falls under

    try {

        var res = await http.post(Uri.parse(phpurl), body: {
            "categoryid": categoryid,
        });

        if (res.statusCode == 200) {

            //If fails to send data

            var data = convert.json.decode(res.body); //Decode to array

            if (data["error"]) {

                //If fails the query

                List error = [
                    {"error": "true", "menuitems": "[]"} //Send nothing
                ];
            }

            return error;
        } else {

            List listdata = [
                data
            ]; //Send back the list of items associated with the category

            return listdata;
        }
    }
}
```

```
        }

    } else {

        List error = [
            {"error": "true", "menuitems": "[]"}
        ];

        return error;
    }

} catch (e) {

    List error = [
        {"error": "true", "menuitems": "[]"}
    ];

    return error;
}

}

@Override

Widget build(BuildContext context) {

    return Scaffold(
        body: SingleChildScrollView(
            //Make page scrollable
            child:
                Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
```

```
Stack(children: [  
    Container(  
        width: MediaQuery.of(context).size.width,  
        height: 240,  
        decoration: BoxDecoration(  
            image: DecorationImage(  
                fit: BoxFit.fitWidth,  
                image: NetworkImage(widget.resbanner),  
            ),  
        ),  
    ),  
    Align(  
        alignment: Alignment.bottomCenter,  
        child: Container(  
            margin: const EdgeInsets.only(top: 215),  
            width: double.infinity,  
            height: 30,  
            decoration: BoxDecoration(  
                color: Theme.of(context).backgroundColor,  
                borderRadius:  
                    const BorderRadius.vertical(top: Radius.circular(20)),  
            )),  
    ),  
],
```

```
Align(
  alignment: Alignment.bottomCenter,
  child: Container(
    margin: const EdgeInsets.only(top: 180),
    width: 80,
    height: 80,
    decoration: BoxDecoration(
      shape: BoxShape.circle,
      color: Theme.of(context).backgroundColor,
    )),
  Align(
    alignment: Alignment.bottomCenter,
    child: Container(
      margin: const EdgeInsets.only(top: 185),
      width: 70,
      height: 70,
      decoration: BoxDecoration(
        image: DecorationImage(
          fit: BoxFit.cover,
          image: NetworkImage(widget.reslogo.toString()),
        ),
        shape: BoxShape.circle,
```

```
        color: Theme.of(context).colorScheme.onSurface,  
    )),  
  
SafeArea(  
  
    child: InkWell(  
  
        onTap: () => Navigator.of(context).pop(),  
  
        child: Container(  
  
            margin: const EdgeInsets.only(top: 20, left: 20),  
  
            width: 50,  
  
            height: 50,  
  
            decoration: BoxDecoration(  
  
                shape: BoxShape.circle,  
  
                color: Theme.of(context).colorScheme.onSurface,  
            ),  
  
            child: Icon(  
  
                Icons.chevron_left,  
  
                color: Theme.of(context).colorScheme.onBackground,  
  
                size: 35,  
            ),  
        )),  
  
Align(  
  
    alignment: Alignment.topRight,  
  
    child: SafeArea(  
       
```

```
        child: InkWell(
```

```
            child: Container(
```

```
                margin: const EdgeInsets.only(top: 20, right: 80),
```

```
                width: 45,
```

```
                height: 45,
```

```
                decoration: BoxDecoration(
```

```
                    shape: BoxShape.circle,
```

```
                    color: Theme.of(context).colorScheme.onSurface.withOpacity(0.8),
```

```
                ),
```

```
                child: Icon(
```

```
                    Icons.info_outline,
```

```
                    color: Theme.of(context).colorScheme.onBackground,
```

```
                    size: 30,
```

```
                ),
```

```
            ))),
```

```
        FutureBuilder<List>(<pre>
```

```
            //Get favourites list from future
```

```
            future: getFavourites(),
```

```
            builder: ((context, snapshot) {
```

```
                if (!snapshot.hasData) {
```

```
                    //While no data received, show loading bar
```

```
                    return SafeArea(
```

```
        child: Align(  
  
            alignment: Alignment.topRight,  
  
            child: Container(  
  
                margin: const EdgeInsets.only(top: 20, right: 20),  
  
                width: 45,  
  
                height: 45,  
  
                decoration: BoxDecoration(  
  
                    shape: BoxShape.circle,  
  
                    color: Theme.of(context)  
  
                        .colorScheme  
  
                        .onSurface  
  
                        .withOpacity(0.8),  
  
  
                child: Icon(  
  
                    Icons.cached,  
  
                    color: Theme.of(context).colorScheme.onBackground,  
  
                ))));  
  
    } else if (snapshot.hasError) {  
  
        return SafeArea(  
  
            child: Align(  
  
                alignment: Alignment.topRight,  
  
                child: Container(  

```

```
        margin: const EdgeInsets.only(top: 20, right: 20),  
  
        width: 45,  
  
        height: 45,  
  
        decoration: BoxDecoration(  
  
            shape: BoxShape.circle,  
  
            color: Theme.of(context)  
  
                .colorScheme  
  
                .onSurface  
  
                .withOpacity(0.8),  
  
>),  
  
        child: Icon(  
  
            Icons.error,  
  
            color: Theme.of(context).colorScheme.error,  
  
>>>));  
  
>} else {  
  
    //If data received  
  
    List restaurantFavourites = snapshot.data ?? [];  
  
    if (restaurantFavourites[0]["error"] == true) {  
  
        return SafeArea(  
  
            child: Align(  
  
                alignment: Alignment.topRight,  
  
                child: Container(  

```

```
        margin: const EdgeInsets.only(top: 20, right: 20),  
  
        width: 45,  
  
        height: 45,  
  
        decoration: BoxDecoration(  
  
            shape: BoxShape.circle,  
  
            color: Theme.of(context)  
  
                .colorScheme  
  
                .onSurface  
  
                .withOpacity(0.8),  
  
>),  
  
        child: Icon(  
  
            Icons.error,  
  
            color: Theme.of(context).colorScheme.error,  
  
        ))));  
  
    } else {  
  
        return SafeArea(  
  
            child: Align(  
  
                alignment: Alignment.topRight,  
  
                child: InkWell(  
  
                    onTap: () {  
  
                        if (restaurantFavourites[0]["restaurantids"]  
  
                            .contains(widget.resid)) {  
  
                                Navigator.push(context,  
  
                                    MaterialPageRoute(builder:  
  
                                        (context) =>  
  
                                            RestaurantDetailsScreen(  
  
                                                restaurantFavourites[0],  
  
                                                widget.resid  
  
                                            ));  
  
                            }  
  
                        }  
  
                    }  
  
                )  
  
            )  
  
        );  
  
    }  
  
}
```

```
        favouriteRestaurant(  
  
            widget.resid.toString(), "unfavourite");  
  
            setState(() {  
  
                getFavourites();  
  
            });  
  
        } else {  
  
            favouriteRestaurant(  
  
                widget.resid.toString(), "favourite");  
  
            setState(() {  
  
                getFavourites();  
  
            });  
  
        }  
  
    },  
  
    child: Container(  
  
        margin:  
  
            const EdgeInsets.only(top: 20, right: 20),  
  
        width: 45,  
  
        height: 45,  
  
        decoration: BoxDecoration(  
  
            shape: BoxShape.circle,  
  
            color: Theme.of(context)  
  
.colorScheme
```

```
        .onSurface
        .withOpacity(0.8),
    ),
    child: restaurantFavourites[0]
        [ "restaurantids"]
        .contains(widget.resid)
    ? Icon(Icons.favorite,
        size: 30, // ? = favourited
        color: Theme.of(context)
            .colorScheme
            .secondary)
    : Icon(
        // : = unfavourited
        Icons.favorite_border_outlined,
        size: 30,
        color: Theme.of(context)
            .colorScheme
            .onBackground))));}

}
})
]))
],
]) ,
```

```
Padding(  
  padding: const EdgeInsets.symmetric(vertical: 20, horizontal: 30),  
  child: Column(children: [  
    Row(  
      mainAxisAlignment: MainAxisAlignment.spaceBetween,  
      children: [  
        Flexible(  
          child: Container(  
            padding: const EdgeInsets.only(right: 13.0),  
            child: Text(  
              widget.resname,  
              overflow: TextOverflow.ellipsis,  
              style: Theme.of(context).textTheme.headline3,  
            )),  
        Row(  
          children: [  
            Icon(  
              Icons.star,  
              size: 20,  
              color: Theme.of(context).primaryColor,  
            ),  
            const SizedBox(  
              width: 10,  
            ),  
            Flexible(  
              child: Container(  
                padding: const EdgeInsets.only(left: 10),  
                child: Text(  
                  widget.resname,  
                  overflow: TextOverflow.ellipsis,  
                  style: Theme.of(context).textTheme.headline3,  
                ),  
              ),  
            ),  
          ],  
        ),  
      ],  
    ),  
  ],  
),
```

```
        width: 5,  
    ),  
  
    Text(  
  
        "N/A",  
  
        style: Theme.of(context).textTheme.bodyText1,  
    )  
],  
,  
],  
),  
  
const SizedBox(height: 10),  
  
Align(  
  
    alignment: Alignment.topLeft,  
  
    child: Wrap(  
  
        children: [  
  
            Text("${widget.resdistance} km away",  
  
                style: Theme.of(context).textTheme.bodyText1),  
  
            const SizedBox(  
  
                width: 5,  
            ),  
  
            Text(  
  
                "...",  
        ],  
    ),  
),
```

```
        style: Theme.of(context).textTheme.bodyText1,  
    ),  
  
    const SizedBox(  
        width: 5,  
    ),  
  
    Text(  
        (widget.resdelivery == "0.00")  
            ? "Free Delivery"  
            : "£${widget.resdelivery} Delivery",  
        style: Theme.of(context).textTheme.bodyText1),  
  
    Text(  
        ".",  
        style: Theme.of(context).textTheme.bodyText1,  
    ),  
  
    const SizedBox(  
        width: 5,  
    ),  
  
    Text(  
        (widget.resordermin == "0")  
            ? "No Minimum Order"  
            : "£${widget.resordermin} Order Minimum",  
        style: Theme.of(context).textTheme.bodyText1),  
    ),
```

```
        ],  
  
        )),  
  
    ])),  
  
FutureBuilder<List>(  
  
    //Checks for updates in the restaurant menu category  
  
    future: getMenuCategories(),  
  
    builder: ((context, snapshot) {  
  
        if (!snapshot.hasData) {  
  
            //If no data has been received, show loading bar  
  
            return LinearProgressIndicator(  
  
                color: Theme.of(context).primaryColor,  
  
                backgroundColor: const Color.fromARGB(0, 235, 224, 255));  
  
        }  
  
        if (snapshot.hasError) {  
  
            //If there is an error, grabbing data, show error  
  
            return const Text("An Error occurred. Please try again");  
  
        } else {  
  
            //If the data has been received  
  
            List restaurantcategories = snapshot.data ??  
  
            []; //Categories data associated with restaurantcategories  
  
            if (restaurantcategories[0]["error"] == true) {  
  
                return Text(  
                    "An Error occurred. Please try again");  
            } else {  
  
                return ListView.builder(  
                    itemCount: restaurantcategories.length,  
                    itemBuilder: (context, index) {  
                        return Card(  
                            child: Column(  
                                children: [  
                                    Text(restaurantcategories[index]["category"]));  
                                ]));  
                    });  
            }  
        }  
    })  
);
```

```
        "Failed to get restaurant categories",

        style: Theme.of(context).textTheme.headline6,

    );

} else {

    return ListView.builder(

        //For each category

        physics:

            const NeverScrollableScrollPhysics(), //Disable
scrolling. Scroll with whole page

        scrollDirection: Axis.vertical,

        shrinkWrap: true,

        itemCount: restaurantcategories[0]["restaurantcategories"]

            .length, //For each restaurant

        itemBuilder: (context, index) {

            return LayoutBuilder(builder: (context, constraints) {

                if (restaurantcategories[0]["restaurantcategories"]

                    .isEmpty) {

                    //If there is no categories, display menu unavailable

                    return const Text("Menu unavailable");

                } else {

                    //If there are categories

                    return Column(children: [

```

//Add text of category name

```
Padding(  
  padding: const EdgeInsets.only(  
    left: 30, top: 50, right: 10, bottom: 10),  
  child: Text(  
    restaurantcategories[0]  
    ["restaurantcategories"][index][0],  
    style: Theme.of(context).textTheme.headline3,  
,  
  FutureBuilder<List>(  
    //For each category, use a future to get the list  
of items  
    future: getMenuItems(restaurantcategories[0]  
    ["restaurantcategories"][index][1]),  
    builder: (context, snapshot) {  
      if (!snapshot.hasData) {  
        //While no data received, show loading bar  
        return LinearProgressIndicator(  
          color: Theme.of(context).primaryColor,  
          backgroundColor: const Color.fromARGB(  
            0, 235, 224, 255));  
      }  
      if (snapshot.hasError) {
```

```

        //If there is an error, show failed to get
items

        return const Text("Failed to get items");

    } else {

        //List of items for category

        List restaurantItems = snapshot.data ??

[]; //Get data from Future

        if (restaurantItems[0]["menuitems"])

            .isEmpty) {

        return Container(
            width: double.infinity,
            padding: const EdgeInsets.symmetric(
                vertical: 30, horizontal: 20),
            decoration: BoxDecoration(
                borderRadius:
                    const BorderRadius.all(
                        Radius.circular(10)),
                color: Theme.of(context)
                    .colorScheme
                    .onSurface),
            margin: const EdgeInsets.symmetric(
                vertical: 5, horizontal: 30),

```

```

        child: Text(
            "Oh no! There are no items found under
this category.",
            textAlign: TextAlign.center,
            style: Theme.of(context)
                .textTheme
                .headline6,
        ),
    );
} else {
    return ListView.builder(
        physics:
            const
NeverScrollableScrollPhysics(),
        scrollDirection: Axis.vertical,
        shrinkWrap: true,
        itemCount: (restaurantitems[
            0] //For each item, create
a container
            [ "menuitems" ]
            .length), //For each restaurant
        itemBuilder: (context, index) {
            return LayoutBuilder(builder:
                (context, constraints) {

```

```
// If there is items to display

return InkWell(
    //Clickable items

    onTap: () {
        Navigator.push(
            context,
            MaterialPageRoute(
                builder: (context) =>
                    RestaurantItemCustomisePage(
                        reslogo: widget
                            .reslogo,
                        itemid:
                        restaurantitems[0]["menuitems"]
                            [index][0],
                        foodcategory:
                        restaurantitems[0]
                            ["menuitems"]
                            [index][1],
                        subfoodcategory:
                        restaurantitems[0]
                            ["menuitems"]
                            [index][2],
                    ),
            ),
        );
    },
);
```

```
        itemname:  
restaurantitems[0]  
  
["menuitems"]  
            [index][3],  
  
        description:  
restaurantitems[0]  
  
["menuitems"][index][4],  
            price:  
restaurantitems[0]["menuitems"][index][5],  
  
        itemimage:  
restaurantitems[0]["menuitems"][index][6])));  
  
},  
  
        child: Container(  
  
            //Create clickable  
container  
  
            width: double.infinity,  
  
            margin:  
                const EdgeInsets.only(  
  
                    left: 10,  
  
                    right: 10,  
  
                    top: 10,  
  
                    bottom: 10),  
  
            child: Row(  
        
```

```
//Create a row containing
item image, name and price

children: [
  Container(
    width: 80,
    height: 80,
    decoration: BoxDecoration(
      borderRadius: BorderRadius.all(
        Radius.circular(5)),
      image: DecorationImage(
        fit: BoxFit.cover,
        image: NetworkImage(restaurantitems[0]["menuitems"]
          [index][6]
          .toString()))),

```

```
        ),  
  
        const SizedBox(  
            width: 15),  
  
        Expanded(  
            child: Column(  
                mainAxisSize:  
  
MainAxisAlignment  
                .start,  
  
                crossAxisAlignment:  
  
CrossAxisAlignment  
                .start,  
  
                children: [  
                    Text(  
  
restaurantitems[0]  
                    [  
  
"menuitems"]  
                    [  
  
index][3]  
  
.toString(),
```

```
        textAlign:  
          TextAlign  
            .start,  
        style:  
Theme.of(  
  
context)  
        .textTheme  
          .headline6!  
        .copyWith(  
          color:  
Theme.of(context)  
  
.colorScheme  
  
.onBackground),  
      ),  
      Text(  
  
restaurantItems[0]  
      [  
        "menuitems"]  
      [  
index][4]
```

```
.toString(),  
    maxLines: 3,  
    style:  
Theme.of(  
  
context)  
    .textTheme  
    .bodyText2!  
    .copyWith(  
        color:  
Theme.of(context)  
  
.textTheme  
  
.headline5!  
  
.color,  
  
fontSize:  
14),  
    overflow:  
TextOverflow  
    .clip,  
)  
)
```

```
        ],
        ),
        ),
        const SizedBox(
            width: 15),
        Row(
            children: [
            Text(
                "\u20ac",
                style: Theme.of(
                    context)
                    .textTheme
                    .headline6!
                    .copyWith(
                        color:
                Theme.of(context)
                    .primaryColor),
            ),
            const SizedBox(
                width: 2),
            Text(
                restaurantitems[0]["menuitems"]
```

```
[  
  
index]  
[5]  
  
.toString(),  
style:  
Theme.of(  
  
context)  
.textTheme  
.headline6!  
.copyWith(  
color:  
Theme.of(context)  
  
.colorScheme  
  
.onBackground))  
],  
,  
const SizedBox(  
width: 10),  
],  
));
```

filtersort.dart

```
import 'package:alleat/widgets/navbar.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:flutter/material.dart';

import 'dart:convert';
```

```
class FilterSort extends StatefulWidget {

  const FilterSort({super.key});

  @override

  State<FilterSort> createState() => _FilterSortState();
}

class _FilterSortState extends State<FilterSort> {

  List sortOptions = [
    //Sort options [Visual button text, icon, option saved in customiseSelected]
    ["Recommended", Icons.assistant_outlined, "default"],
    ["Top Rated", Icons.star_border_outlined, "top"],
    ["Popular", Icons.local_fire_department_outlined, "hot"],
    ["Distance", Icons.straighten_outlined, "distance"]
  ];

  List priceRangeOptions = [
    //Price range options [Visual Text, Option Saved in customiseSelected]
    ["£", 1],
    ["££", 2],
    ["£££", 3],
    ["££££", 4]
  ];
}
```

```

bool isChecked = false; //Favourites selector

double _currentMaxDeliveryFeeValue = 4;

double _currentMinOrderPriceValue = 40;

String? encodedCustomiseSelected;

Map customiseSelected = {

    //initial filters and sort method

    "sort": "distance",

    "favourite": false,

    "price": [1, 2, 3, 4],

    "maxDelivery": 4.0,

    "minOrder": 40.0

};

@Override

void initState() {

    //On initiation, load filters from shared preferences

    super.initState();

    _loadFilter();

}

_loadFilter() async {

    SharedPreferences prefs = await SharedPreferences.getInstance(); //get from
shared preferences

```

```

setState(() {

    encodedCustomiseSelected = prefs.getString('filtersort');

});

if (encodedCustomiseSelected != null) {

    //If the filter and sort methods have been previously saved

    customiseSelected = json.decode(encodedCustomiseSelected.toString());
//Decode customise and split and replace default values

    _currentMaxDeliveryFeeValue = customiseSelected["maxDelivery"];

    _currentMinOrderPriceValue = customiseSelected["minOrder"];

    isChecked = customiseSelected["favourite"];

} else {

    //If the filter and sort methods have not been previously saved

    customiseSelected = {

        //Set default values

        "sort": "distance",

        "favourite": false,

        "price": [1, 2, 3, 4],

        "maxDelivery": 4.0,

        "minOrder": 40.0

    };

    _currentMaxDeliveryFeeValue = 4; //Change sliders to default values

    _currentMinOrderPriceValue = 40;

}

```

```
}
```



```
@override
```



```
Widget build(BuildContext context) {
```



```
    return Scaffold(
```



```
        body: SingleChildScrollView(
```



```
            child: Column(crossAxisAlignment: CrossAxisAlignment.start, children:
```



```
[
```



```
    Padding(
```



```
        padding: const EdgeInsets.only(left: 40, right: 20, top: 50),
```



```
        child: Row(
```



```
            //Row with the text sort and a button to clear filters and reset
```



```
            sort
```



```
            crossAxisAlignment: CrossAxisAlignment.end,
```



```
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
```



```
            children: [
```



```
                Text("Sort", style: Theme.of(context).textTheme.headline2),
```



```
                LayoutBuilder(builder: (context, constraints) {
```



```
                    //If there have been any changes to the filters and sort
```



```
                    methods, highlight clear all text
```



```
                    if (customiseSelected["sort"] != "default" ||
```



```
                        customiseSelected["favourite"] != false ||
```



```
                        !customiseSelected["price"].contains(1) ||
```



```
                        !customiseSelected["price"].contains(2) ||
```

```

    !customiseSelected["price"].contains(3) ||
    !customiseSelected["price"].contains(4) ||
    customiseSelected["maxDelivery"] != 4.0 ||
    customiseSelected["minOrder"] != 40.0) {

  return InkWell(
    onTap: () {
      //On tap, reset to defaults
      setState(() {
        customiseSelected = {
          "sort": "distance",
          "favourite": false,
          "price": [1, 2, 3, 4],
          "maxDelivery": 4.0,
          "minOrder": 40.0
        };
        _currentMaxDeliveryFeeValue = 4;
        _currentMinOrderPriceValue = 40;
        isChecked = false;
      });
    },
    child: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 20),

```

```

        child: Text(
            "Clear all",
            style:
Theme.of(context).textTheme.headline6!.copyWith(color:
Theme.of(context).colorScheme.error),
        )));
    } else {
        return InkWell(
            onTap: () {
                setState(() {
                    customiseSelected = {
                        "sort": "distance",
                        "favourite": false,
                        "price": [1, 2, 3, 4],
                        "maxDelivery": 4.0,
                        "minOrder": 40.0
                    };
                    _currentMaxDeliveryFeeValue = 4;
                    _currentMinOrderPriceValue = 40;
                });
            },
        ),
        child: Padding(
            padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 20),

```

```
        child: Text(  
  
            "Clear all",  
  
            style:  
Theme.of(context).textTheme.headline6!.copyWith(color:  
Theme.of(context).colorScheme.error.withOpacity(0.1)),  
  
        )),  
  
    }  
  
)  
  
]),  
  
Padding(  
  
padding: const EdgeInsets.symmetric(horizontal: 30),  
  
child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [  
  
    const SizedBox(  
  
        height: 20,  
    ),  
  
    ListView.builder(  
  
        //For each sort method  
  
        physics: const NeverScrollableScrollPhysics(),  
  
        scrollDirection: Axis.vertical,  
  
        shrinkWrap: true,  
  
        itemCount: sortOptions.length,  
  
        itemBuilder: (context, index) {  
  
            return Padding(  

```

```
padding: const EdgeInsets.symmetric(vertical: 7),  
  
child: ElevatedButton(  
  
    style: ElevatedButton.styleFrom(  
  
        side: (customiseSelected["sort"] ==  
  
                sortOptions[index][2]) //If the sort method  
is the same as the button, display with border  
  
            ? BorderSide(color:  
Theme.of(context).primaryColor, width: 2)  
  
            : BorderSide.none,  
  
        backgroundColor:  
Theme.of(context).colorScheme.onSurface,  
  
        padding: const EdgeInsets.symmetric(vertical: 15,  
horizontal: 25)),  
  
onPressed: () {  
  
    //On press, overwrite sort method with new sort  
method  
  
    setState(() {  
  
        customiseSelected["sort"] = sortOptions[index][2];  
  
    });  
  
},  
  
child: Row(  
  
    //Row with the icon and text for sort method grabbed  
from the list  
  
    children: [  
  
        Icon(
```

```
        sortOptions[index][1],  
  
        color: customiseSelected["sort"] ==  
sortOptions[index][2]  
  
        ?  
Theme.of(context).textTheme.headline1?.color  
  
        :  
Theme.of(context).textTheme.headline5?.color,  
  
(  
),  
  
const SizedBox(width: 20),  
  
Flexible(  
  
    child: Text(  
  
        sortOptions[index][0],  
  
        style:  
Theme.of(context).textTheme.headline5!.copyWith(  
  
            color: customiseSelected["sort"] ==  
sortOptions[index][2]  
  
            ?  
Theme.of(context).textTheme.headline1?.color  
  
            :  
Theme.of(context).textTheme.headline5?.color,  
  
(  
),  
  
))  
  
],  
));  
},  
});
```

```
const SizedBox(  
    height: 40,  
)  
  
Text("Filter", style: Theme.of(context).textTheme.headline2),  
  
const SizedBox(  
    height: 20,  
)  
  
Row(  
    children: [  
        Flexible(  
            child: CheckboxListTile(  
                //Checkbox with text  
                title: Text(  
                    "Show only favourites",  
                    style: Theme.of(context).textTheme.headline6,  
                ),  
                checkColor: Colors.white,  
                activeColor: Theme.of(context).primaryColor,  
                contentPadding: const EdgeInsets.all(0),  
                value: isChecked,  
                controlAffinity: ListTileControlAffinity.leading,  
                onChanged: (newValue) {  
  
    }  
}]  
)
```

```
        setState(() {
            isChecked = !isChecked;
        });

        if (isChecked) {
            customiseSelected["favourite"] = true;
        } else {
            customiseSelected["favourite"] = false;
        }
    },
),
],
),
const SizedBox(
    height: 20,
),
Text("Price Range", style: Theme.of(context).textTheme.headline3),
const SizedBox(
    height: 20,
),
Container(
    //Container with a list of price brackets
    padding: const EdgeInsets.symmetric(horizontal: 10),
```

```
color: Theme.of(context).colorScheme.onSurface,  
  
height: 70,  
  
child: ListView.builder(  
  
    scrollDirection: Axis.horizontal,  
  
    shrinkWrap: true,  
  
    itemCount: priceRangeOptions.length, //For each restaurant  
  
    itemBuilder: (context, index) {  
  
        //For each price bracket  
  
        return Padding(  
  
            padding: const EdgeInsets.symmetric(horizontal: 5,  
vertical: 10),  
  
            child: SizedBox(  
  
                width: 70,  
  
                child: ElevatedButton(  
  
                    style: ElevatedButton.styleFrom(  
  
                        side: customiseSelected["price"]  
  
.contains(priceRangeOptions[index][1]) //If customiseSelected has the price  
bracket, display with border  
  
                        ? BorderSide(color:  
Theme.of(context).primaryColor, width: 2)  
  
                        : BorderSide.none,  
  
                        backgroundColor:  
Theme.of(context).colorScheme.onSurface,
```

```

padding: const EdgeInsets.symmetric(vertical:
15, horizontal: 10)),

onPressed: () {

    // If customiseSelected has the price bracket,
remove otherwise add it.

    if
(customiseSelected["price"].contains(priceRangeOptions[index][1])) {

        setState(() {

customiseSelected["price"].remove(priceRangeOptions[index][1]);

    });

} else {

        setState(() {

customiseSelected["price"].add(priceRangeOptions[index][1]);

    });

}

},
child: Text(
//Display text with price bracket text (£, ££,
£££, ££££)

priceRangeOptions[index][0],


style:
Theme.of(context).textTheme.headline6!.copyWith(


color:
customiseSelected["price"].contains(priceRangeOptions[index][1])

```

```

        ? Theme.of(context).primaryColor
        :
Theme.of(context).textTheme.headline6?.color,
        ),
        ),
        )));

}),

const SizedBox(
    height: 40,
),
Row(mainAxisAlignment: MainAxisAlignment.spaceBetween, children: [
    //Row with title and the current soldier status. If price is 0,
display as free

    Text("Max Delivery Fee", style:
Theme.of(context).textTheme.headline3),
    LayoutBuilder(builder: (context, constraints) {
        if (_currentMaxDeliveryFeeValue == 0) {
            return const Text("FREE");
        } else {
            return Text("£${_currentMaxDeliveryFeeValue}0");
        }
    })
]),
const SizedBox(

```

```
        height: 20,  
    ),  
  
    Slider(  
  
        value: _currentMaxDeliveryFeeValue,  
  
        max: 4,  
  
        min: 0,  
  
        thumbColor: Theme.of(context).primaryColor,  
  
        activeColor: Theme.of(context).primaryColor,  
  
        inactiveColor:  
Theme.of(context).colorScheme.onBackground.withOpacity(0.2),  
  
        divisions: 4,  
  
        onChanged: (double value) {  
  
            //On change, update filter and the slider with new value  
  
            setState(() {  
  
                _currentMaxDeliveryFeeValue = value;  
  
            });  
  
            customiseSelected["maxDelivery"] = _currentMaxDeliveryFeeValue;  
  
        },  
    ),  
  
    const SizedBox(  
  
        height: 40,  
    ),  
  
    Row(mainAxisAlignment: MainAxisAlignment.spaceBetween, children: [
```

```

        //Row with title and the current soldier status. If price is 0,
display as None

        Text("Min Order Price", style:
Theme.of(context).textTheme.headline3),

        LayoutBuilder(builder: (context, constraints) {

            if (_currentMinOrderPriceValue == 0) {

                return const Text("None");

            } else {

                return Text("£${_currentMinOrderPriceValue}0");

            }

        })

    ],
const SizedBox(
    height: 20,
),
Slider(
    value: _currentMinOrderPriceValue,
    max: 40,
    min: 0,
    thumbColor: Theme.of(context).primaryColor,
    activeColor: Theme.of(context).primaryColor,
    inactiveColor:
Theme.of(context).colorScheme.onBackground.withOpacity(0.2),
    divisions: 4,

```

```

onChanged: (double value) {

    setState(() {
        _currentMinOrderPriceValue = value;
    });

    customiseSelected["minOrder"] = _currentMinOrderPriceValue;
},
),
const SizedBox(
    height: 40,
),
ElevatedButton(
    //Save button encodes the filter and sort dictionary and replaces
    the sharedpreference with the new value then closes the filter screen
    style: ElevatedButton.styleFrom(minimumSize: const
Size.fromHeight(50)),

    onPressed: () async {
        final prefs = await SharedPreferences.getInstance();

        String encodedCustomiseSelected =
json.encode(customiseSelected);

        await prefs.setString('filtersort', encodedCustomiseSelected);

        setState(() {
            Navigator.pop(context);
            Navigator.pop(context);
            Navigator.of(context).push(

```

```
        MaterialPageRoute(builder: (_) => const Navigation()),  
    );  
});  
},  
child: const Text("Save"))  
]),  
const SizedBox(  
height: 40,  
),  
]));  
}  
}  
}
```

locationselection.dart

```
import 'dart:async';  
  
import 'package:alleat/services/cart_service.dart';  
  
import 'package:alleat/widgets/genericlocading.dart';  
  
import 'package:alleat/widgets/navigationbar.dart';  
  
import 'package:flutter/services.dart';  
  
import 'package:google_maps_flutter/google_maps_flutter.dart';  
  
import 'package:flutter/material.dart';
```

```
import 'package:map_picker/map_picker.dart';

import 'package:geocoding/geocoding.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:geolocator/geolocator.dart';



class SelectLocation extends StatefulWidget {

    const SelectLocation({super.key});

    @override

    State<SelectLocation> createState() => _SelectLocationState();
}

class _SelectLocationState extends State<SelectLocation> {

    final _controller = Completer<GoogleMapController>();

    static TextEditingController addresslineone = TextEditingController();

    static TextEditingController addresslinetwo = TextEditingController();

    static TextEditingController postcode = TextEditingController();

    static TextEditingController city = TextEditingController();

    MapPickerController mapPickerController = MapPickerController();

    late var cameraPosition = const CameraPosition(target: LatLng(0, 0), zoom: 20);

    Future<List> getSavedPosition() async {
```

```

//Try to get saved location and current location

final prefs = await SharedPreferences.getInstance(); // Get saved location
from shared preferences

final double? savedLocationLat = prefs.getDouble('locationLatitude');

final double? savedLocationLng = prefs.getDouble('locationLongitude');

final List<String>? savedLocationText =
prefs.getStringList('locationPlacemark');

if (savedLocationLat != null && savedLocationLng != null && savedLocationText
!= null) {

    //If not null returned, return the value

    addresslineone = TextEditingController(text: savedLocationText[0]);

    addresslinetwo = TextEditingController(text: savedLocationText[1]);

    city = TextEditingController(text: savedLocationText[2]);

    postcode = TextEditingController(text: savedLocationText[3]);

    return [savedLocationLat, savedLocationLng];

} else {

    //If null returned from saved location, get the approximate location

    return getCurrentLocation();

}

}

Future<bool> saveLocation() async {

```

```
try {

    await SQLiteCartItems.clearCart();

    final prefs = await SharedPreferences.getInstance();

    await prefs.setDouble('locationLatitude',
cameraPosition.target.latitude.toDouble());

    await prefs.setDouble('locationLongitude',
cameraPosition.target.longitude.toDouble());

    await prefs.setStringList('locationPlacemark',
<String>[addresslineone.text, addresslinetwo.text, city.text, postcode.text]);




    return true;
}

} catch (e) {
    return false;
}

}

Future<List> getCurrentLocation() async {

    bool serviceEnabled;

    LocationPermission permission;

    // Test if location services are enabled.

    serviceEnabled = await Geolocator.isLocationServiceEnabled();

    if (!serviceEnabled) {

        await getPlacemark([false, 51.509865, -0.118092]);

```

```

        return [51.509865, -0.118092]; //London lat long

    }

    // Check if the location is denied

    permission = await Geolocator.checkPermission();

    if (permission == LocationPermission.denied) {

        permission = await Geolocator.requestPermission();

        if (permission == LocationPermission.denied) {

            await getPlacemark([false, 51.509865, -0.118092]);

            return [51.509865, -0.118092]; //London lat long

        }

    }

    if (permission == LocationPermission.deniedForever) {

        // Permissions are denied forever, handle appropriately.

        await getPlacemark([false, 51.509865, -0.118092]);

        return [51.509865, -0.118092]; //London lat long

    }

}

// Access the current possition of the device

Position locationDevice = await Geolocator.getCurrentPosition();

await getPlacemark([false, locationDevice.latitude,
locationDevice.longitude]);

```

```
        return [locationDevice.latitude, locationDevice.longitude];  
    }  
  
    Future<void> getPlacemark(placemarkLocation) async {  
        try {  
            switch (placemarkLocation[0]) {  
                case true:  
                    List<Placemark> placemarks = await placemarkFromCoordinates(  
                        cameraPosition.target.latitude,  
                        cameraPosition.target.longitude,  
                    );  
                    addresslineone = TextEditingController(text: placemarks.first.name);  
                    addresslinetwo = TextEditingController(text: placemarks.first.street);  
                    postcode = TextEditingController(text: placemarks.first.postalCode);  
                    if ((placemarksWithoutAddress.first.locality) == "") {  
                        city = TextEditingController(text:  
                            placemarks.first.subAdministrativeArea);  
                    } else {  
                        city = TextEditingController(text: placemarks.first.locality);  
                    }  
                    break;  
                case false:  
            }  
        } catch (e) {}  
    }  
}
```

```
        List<Placemark> placemarks = await
placemarkFromCoordinates(placemarkLocation[1], placemarkLocation[2]);

        addresslineone = TextEditingController(text: placemarks.first.name);

        addresslinetwo = TextEditingController(text: placemarks.first.street);

        postcode = TextEditingController(text: placemarks.first.postalCode);

        if ((placemarkLocation.first.locality) == "") {

            city = TextEditingController(text:
placemarkLocation.first.subAdministrativeArea);

        } else {

            city = TextEditingController(text: placemarks.first.locality);

        }

        break;

    }

} catch (e) {

    textController.text = "";

}

}

var textController = TextEditingController();

@Override

Widget build(BuildContext context) {

    return Scaffold(

        body: FutureBuilder<List>(  
}
```

```
future: getSavedPosition(),  
  
builder: (context, snapshot) {  
  
  if (!snapshot.hasData) {  
  
    return const GenericLoading();  
  
  }  
  
  if (snapshot.hasData) {  
  
    var savedPosition = snapshot.data ?? [];  
  
    CameraPosition cameraPosition = CameraPosition(target:  
LatLng(savedPosition[0], savedPosition[1]), zoom: 19);  
  
    return Stack(  
  
      alignment: Alignment.topCenter,  
  
      children: [  
  
        LayoutBuilder(builder: (BuildContext context, BoxConstraints  
constraints) {  
  
          return MapPicker(  
  
            // pass icon widget  
  
            iconWidget: const Icon(  
  
              Icons.location_on,  
  
              size: 65,  
  
            ),  
  
            //add map picker controller  
  
            mapPickerController: mapPickerController,  
  
            child: GoogleMap(  

```

```
    myLocationEnabled: false,  
  
    zoomControlsEnabled: false,  
    // hide location button  
    myLocationButtonEnabled: true,  
    mapType: MapType.normal,  
  
    rotateGesturesEnabled: false,  
    tiltGesturesEnabled: false,  
    // camera position  
    initialCameraPosition: cameraPosition,  
    onMapCreated: (GoogleMapController controller) {  
        _controller.complete(controller);  
        controller.animateCamera(CameraUpdate.newCameraPosition(  
            CameraPosition(  
                target: LatLng(savedPosition[0], savedPosition[1]),  
                zoom: 18,  
            ),  
        ));  
    },  
    onCameraMoveStarted: () {  
        // notify map is moving
```

```

    try {

        mapPickerController.mapMoving!();

    } catch (e) {

        Navigator.pop(context);

    }

},
onCameraMove: (cameraPosition) {

    try {

        //If the pointer position is invalid, dont try to move
the map.

        this.cameraPosition = cameraPosition;

    } catch (e) {

        cameraPosition = cameraPosition;

    }

},
onCameraIdle: () async {

    // notify map stopped moving

    try {

        // if there is an error getting the placemark return
null

        mapPickerController.mapFinishedMoving!();

        //get address name from camera position

        getPlacemark([true]);

    }
}

```

```
        } catch (e) {
            return;
        }
    },
));
}),
DraggableScrollableSheet(
    //Dragable bottom bar
    initialChildSize: 0.2,
    snap: true,
    minChildSize: 0.2,
    maxChildSize: 0.8,
    builder: ((context, scrollController) {
        return Container(
            //Setting bar attributes
            decoration: BoxDecoration(
                borderRadius: const BorderRadius.vertical(top: Radius.circular(10)),
                color: Theme.of(context).backgroundColor,
                boxShadow: const [BoxShadow(spreadRadius: 2, blurRadius: 10, color: Color.fromARGB(8, 0, 0, 0))]),
            child: ListView.builder(
                controller: scrollController,
```

```
        itemCount: 1,  
  
        itemBuilder: ((context, index) {  
  
            return Column(  
  
                children: [  
  
                    Container(  
  
                        width: 40, height: 5, decoration:  
BoxDecoration(color: Colors.grey, borderRadius: BorderRadius.circular(5))),  
  
                    Padding(  
  
                        padding: const  
EdgeInsets.symmetric(vertical: 30, horizontal: 20),  
  
                        child: Row(  
  
                            children: [  
  
                                InkWell(  
  
                                    child: Container(  
  
                                        width: 50,  
  
                                        height: 50,  
  
                                        decoration: BoxDecoration(  
  
                                            borderRadius:  
BorderRadius.circular(10), color: Theme.of(context).colorScheme.error),  
  
                                    child: Icon(  
  
                                        Icons.cancel,  
  
                                        color:  
Theme.of(context).backgroundColor,  
  
                                )),  
                            ]  
                        )  
                    )  
                ]  
            )  
        )  
    )  
);
```

```
        onTap: () {

            setState(() {
                Navigator.pop(context);

            });
        },
    ),
    const SizedBox(width: 10),
    Expanded(
        child: InkWell(
            //Save Location button

            onTap: () {
                showDialog<String>(
                    //Display popup to confirm

                    context: context,
                    builder: (BuildContext
context) => AlertDialog(
                    backgroundColor:
Theme.of(context).backgroundColor,
                    title: Text(
                        'Change Saved
Destination',
                    style:
Theme.of(context)

                    .textTheme

```

```
        .headline5

        ?.copyWith(color:
Theme.of(context).textTheme.headline1?.color),

        ),

        content: Text(
            'Changing your delivery
destination will clear any items currently saved in the cart.',

        style:
Theme.of(context).textTheme.bodyText2,

        ),

        actions: <Widget>[
            TextButton(
                //Cancel button to
close the popup bring user back to the location page

                onPressed: () =>
Navigator.pop(context, 'Cancel'),


                child: const
Text('Cancel'),


            ),


            TextButton(
                //Ok button to save
the location

                onPressed: (() async
{


                bool
hasSavedLocation = await saveLocation(); //Try to save

```

```
        if  
  
(hasSavedLocation == true) {  
  
    //If the location  
    has saved to shared preferences reopen the navigation page  
  
    setState(() {  
  
Navigator.pop(context);  
  
Navigator.pop(context);  
  
Navigator.of(context).push(  
  
MaterialPageRoute(builder: (_) => const Navigation()),  
  
);  
  
});  
  
} else {  
  
    //If it fails to  
    save  
  
    setState(() {  
  
        //Display  
        failed to update  
  
    });  
  
    Navigator.pop(context);  
  
    ScaffoldMessenger.of(context)
```

```
.showSnackBar(const SnackBar(content: Text("Failed to update location")));  
    });  
  
}  
  
},  
child: const  
Text('OK'),  
,  
],  
,  
);  
});  
child: Container(  
decoration:  
  
BoxDecoration(borderRadius: BorderRadius.circular(10), color:  
Theme.of(context).primaryColor),  
padding: const  
EdgeInsets.symmetric(vertical: 15, horizontal: 30),  
child: Text(  
"Save Location",  
textAlign:  
 TextAlign.center,  
style: Theme.of(context)  
.textTheme
```

```
        .headline6!
        .copyWith(color:
Theme.of(context).colorScheme.onSurface),
),
))),
const SizedBox(
width: 10,
),
InkWell(
//Current location button
onTap: () async {
// Get the current location of the
device
List currentLocation = await
getCurrentLocation();

// Move the camera to the current
location
final GoogleMapController
controller = await _controller.future;

controller.animateCamera(CameraUpdate.newCameraPosition(
CameraPosition(
target:
LatLng(currentLocation[0], currentLocation[1]),
zoom: 18,
```

```
        ),

        )),
    },
    child: Container(
        decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(10),
            color: Theme.of(context).colorScheme.onSurface,
            padding: const EdgeInsets.all(15),
            child: Icon(
                Icons.my_location,
                color: Theme.of(context).textTheme.headline1!.color,
            )));
    )
],),
Container(
    padding: const EdgeInsets.all(20),
    alignment: Alignment.bottomLeft,
    child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
        Text("Edit your address", style: Theme.of(context).textTheme.headline3),
    ]));
}
```

```
Padding(  
  padding: const EdgeInsets.only(top:  
  5, right: 20, bottom: 30),  
  
  child: Text("Set your exact address  
on the map and edit it below."),  
  
  style:  
Theme.of(context).textTheme.headline6)),  
  
Padding(  
  padding: const  
EdgeInsets.symmetric(vertical: 15),  
  
  child: Column(crossAxisAlignment:  
CrossAxisAlignment.start, children: [  
  
  Text(  
    "Address Line 1",  
  
    style:  
Theme.of(context).textTheme.headline6,  
  
,  
  
  TextFormField(  
    controller: addresslineone,  
  
    style:  
Theme.of(context).textTheme.bodyText2,  
  
    inputFormatters: [  
  
      //Only allows the input of  
letters a-z and A-Z and , - and spaces  
  
      FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9,. -]+\|\s'))
```

```
        ],  
        )  
    ])),  
    Padding(  
        padding: const  
EdgeInsets.symmetric(vertical: 15),  
        child: Column(crossAxisAlignment:  
CrossAxisAlignment.start, children: [  
        Text(  
            "Address Line 2",  
            style:  
Theme.of(context).textTheme.headline6,  
        ),  
        TextFormField(  
            controller: addresslinetwo,  
            style:  
Theme.of(context).textTheme.bodyText2,  
            inputFormatters: [  
                //Only allows the input of  
letters a-z and A-Z and ,.- and space  
FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9,.-]+|\s'))  
            ],  
        )  
    ])),
```

```
Padding(  
    padding: const  
EdgeInsets.symmetric(vertical: 15),  
  
    child: Column(crossAxisAlignment:  
CrossAxisAlignment.start, children: [  
  
    Text(  
  
        "City/County",  
  
        style:  
Theme.of(context).textTheme.headline6,  
  
    ),  
  
    TextFormField(  
  
        controller: city,  
  
        style:  
Theme.of(context).textTheme.bodyText2,  
  
        inputFormatters: [  
  
            //Only allows the input of  
letters a-z and A-Z and ,.- and space  
  
FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9,.-]+|\s'))  
  
        ],  
  
    )  
  
    ])),  
  
Padding(  
    padding: const  
EdgeInsets.symmetric(vertical: 15),
```

```
        child: Column(crossAxisAlignment:  
CrossAxisAlignment.start, children: [  
  
        Text(  
  
            "Postcode",  
  
            style:  
Theme.of(context).textTheme.headline6,  
  
        ),  
  
        TextFormField(  
  
            controller: postcode,  
  
            style:  
Theme.of(context).textTheme.bodyText2,  
  
            inputFormatters: [  
  
                //Only allows the input of  
letters a-z and A-Z and ,.- and space  
  
FilteringTextInputFormatter.allow(RegExp(r'[a-zA-Z0-9,.-]+|\s'))  
  
            ],  
  
        )  
  
    ])),  
  
]),  
  
);  
  
}));  
  
}),  
  
});
```

```
        ],
    );
} else {
    return const GenericLoading();
}
},
));
}
}
```

setupverification.dart

```
import 'package:alleat/screens/profilesetup/welcomeScreen.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/widgets/genericlocading.dart';

import 'package:alleat/widgets/navigationbar.dart';

import 'package:flutter/material.dart';

class SetupWrapper extends StatefulWidget {

    const SetupWrapper({Key? key}) : super(key: key);

    @override
```

```
State<SetupWrapper> createState() => _SetupWrapperState();  
}  
  
class _SetupWrapperState extends State<SetupWrapper> {  
  
  //Default setup not complete if something wrong happens  
  
  bool setup = false;  
  
  
  
  Future<bool> isSetupComplete() async {  
  
    List<Map> profileInfo = await SQLiteLocalProfiles  
  
        .getFirstProfile(); //Call Database for the first entry  
  
    if (profileInfo.isEmpty) {  
  
      //If the first entry is empty  
  
      //Then setup is not complete (pass to build)  
  
      return false;  
  
    } else {  
  
      //If the first entry exists  
  
      //Setup is complete (pass to build)  
  
      return true;  
  
    }  
  
  }  
  
  @override
```

```
Widget build(BuildContext context) => FutureBuilder<bool>(

    future: isSetupComplete(),

    builder: (context, snapshot) {

        if (!snapshot.hasData) {

            //While loading, go to loading page

            return const GenericLoading();

        }

        if (snapshot.hasError) {

            //If the app has an error, go to error page

            return const GenericLoading();

        } else {

            bool setup = snapshot.data ?? [] as bool; //Get data from Future

            if (setup == false) {

                //If setup is not complete

                return const ProfileSetupWelcome(); //Go to Setup page

            } else if (setup == true) {

                //If setup is complete

                return const Navigation(); //Go to main page

            } else {

                //If there is an error

                return const GenericLoading(); // Go to error page

            }

        }

    }

}
```

```
    }

    },

);

}
```

cart_service.dart

```
import 'package:alleat/services/localprofiles_service.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:sqflite/sqflite.dart' as sql;

class SQLiteCartItems {

    // -----
    // Cart Items Table
    // -----


    static Future<void> createTableCartItems(sql.Database database) async {

        //Create cart table (Used to store items and their customised details)

        await database.execute("""CREATE TABLE cartItems(
            cartid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
            profileid INTEGER,
            itemid INT,
```

```

customised TEXT,
quantity INT,
FOREIGN KEY (profileid) REFERENCES localprofiles(profileid)
)
""");
}

static Future<sql.Database> cartdb() async {
//If tables don't exist, create tables
return sql.openDatabase(
'alleatlocal.db',
version: 1,
onCreate: (sql.Database database, int version) async {
await SQLiteLocalProfiles.createTableProfile(database);
await createTableCartItems(database);
},
);
}

//-----
// Edit Cart
//-----

```

```
// Add to Cart

static Future<bool> addToCart(int itemid, dynamic customised, int quantity)
async {

    try {

        final db = await SQLiteCartItems.cartdb();

        final prefs = await SharedPreferences.getInstance();

        final String? profileid = prefs.getString('serverprofileid'); //Try to get
profileid of current user and convert

        if (profileid == null) {

            //If there is no current user selected

            return false;
        }

        int profileidInt = int.parse(profileid);

        await db.insert("cartItems", {

            "profileid": profileidInt,
            "itemid": itemid,
            "customised": customised,
            "quantity": quantity,
        });
    }

    return true;
} catch (e) {
```

```

        //If fails to add to cart

        return false;

    }

}

// Get a list of profiles which are in the cart

static Future<List> getProfilesInCart() async {

    final db = await SQLiteCartItems.cartdb();

    List profilesInCart = await db.rawQuery("SELECT profileid FROM cartItems");

    List singleProfilesInCart = [];

    for (int i = 0; i < profilesInCart.length; i++) {

        if (!singleProfilesInCart.contains(profilesInCart[i]["profileid"])) {

            singleProfilesInCart.add(profilesInCart[i]["profileid"]);

        }

    }

    return singleProfilesInCart;

}

// Get a list of items that are under a profile ID

static Future<List> getProfileCart(profileID) async {

```

```

final db = await SQLiteCartItems.cartdb();

List itemsInCart = await db.rawQuery("SELECT cartid, itemid, customised,
quantity FROM cartItems WHERE profileid = $profileID");

return itemsInCart;

}

//Remove item from cart

static Future<void> removeItem(cartID) async {

final db = await SQLiteCartItems.cartdb();

await db.rawDelete("DELETE FROM cartItems WHERE cartid = $cartID");

}

//Remove all items from cart

static Future<void> clearCart() async {

final db = await SQLiteCartItems.cartdb();

await db.rawDelete("DELETE FROM cartItems");

}

}

```

dataencryption.dart

```
import 'package:encrypt/encrypt.dart' as encrypt;
```

```
import 'dart:convert' as convert;

import 'package:crypto/crypto.dart' as crypto;

class DataEncryption {

    static Future<String> encrypt(plaintext) async {

        dynamic encryptPassword;

        String strkey =
            'ZC8cegCGG45d1IjIACEtrfypDXtkgJ1rA+4JABPncUE='; //Static key and iv

        String striv = 'yvhsTTh739b2yUW9NNWrcKmHTtLTZNbjbiV3F/cSRzM=';

        var iv = crypto.sha256 //Grab the substring of the key and iv
            .convert(convert.utf8.encode(striv))
            .toString()
            .substring(0, 16);

        var key = crypto.sha256
            .convert(convert.utf8.encode(strkey))
            .toString()
            .substring(0, 16);

        encrypty.IV ivObj = encrypty.IV.fromUtf8(iv);

        encrypty.Key keyObj = encrypty.Key.fromUtf8(key);

        final encrypter =
            encrypty.Encrypter(encrypty.AES(keyObj, mode: encrypty.AESMode.cbc));

        encryptPassword = encrypter.encrypt(plaintext,
```

```
        iv: ivObj); //Use the key and iv to encrypt the plaintext password

    encryptPassword = encryptPassword.base64;

    return encryptPassword;

}

}
```

localprofiles_service.dart

```
import 'dart:math';

import 'package:alleat/services/cart_service.dart';

import 'package:flutter/material.dart';

import 'package:sqflite/sqflite.dart' as sql;

class SQLiteLocalProfiles {

    // -----
    // Local Profiles Table
    // -----


    static Future<void> createTableProfile(sql.Database database) async {

        //Create localprofiles table (Used to store local logged in profiles)

        await database.execute("""CREATE TABLE localprofiles(
            id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

```

```

        profileid INT,
        firstname TEXT,
        lastname TEXT,
        email TEXT,
        password TEXT,
        selected TEXT,
        profilecolorred INT,
        profilecolorgreen INT,
        profilecolorblue INT,
        createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
    )
    """);
}

static Future<sql.Database> db() async {
    //If tables don't exist, create tables
    return sql.openDatabase(
        'alleatlocal.db',
        version: 1,
        onCreate: (sql.Database database, int version) async {
            await createTableProfile(database);
            await SQLiteCartItems.createTableCartItems(database);
        }
    );
}

```



```
final db = await SQLiteLocalProfiles.db();

db.insert("localprofiles", {
    //Insert data into localprofiles table
    "profileid": profileid,
    "firstname": firstname,
    "lastname": lastname,
    "email": email,
    "password": password,
    "profilecolorred": randomProfileColor[0],
    "profilecolorgreen": randomProfileColor[1],
    "profilecolorblue": randomProfileColor[2],
});

selectProfile(email);

}

static Future<void> selectProfile(dynamic email) async {

    final db = await SQLiteLocalProfiles.db();

    db.rawUpdate('UPDATE localprofiles SET selected = false'); //Set all profiles
selected status to false

    db.rawUpdate('UPDATE localprofiles SET selected = true WHERE email =
"$email"');

}
```



```

        return db.rawQuery('SELECT profilecolorred, profilecolorgreen,
profilecolorblue FROM localprofiles ORDER BY id ASC LIMIT 1');

    }

}

// Get all unselected profiles to be displayed

static Future<List<Map<String, Object?>>> getDisplayProfilesList() async {

    final db = await SQLiteLocalProfiles.db();

    return db.rawQuery(
        'SELECT id, profileid, firstname, lastname, profilecolorred,
profilecolorgreen, profilecolorblue FROM localprofiles ORDER BY selected DESC');

}

}

// Get profile info from ID

static Future<List<Map<String, Object?>>> getProfileFromID(id) async {

    final db = await SQLiteLocalProfiles.db();

    return db.rawQuery('SELECT profileid, firstname, lastname, email FROM
localprofiles WHERE id = $id');

}

}

//-----
// Profile Modify
//-----

```

```
// Update an profile by id

static Future<void> updateProfile(int id, String firstname, String lastname,
String email, String password) async {

    //Update profile using firstname, lastname, email and encrypted password

    final db = await SQLiteLocalProfiles.db();

    final data = {

        'firstname': firstname,

        'lastname': lastname,

        'email': email,

        'password': password,

    };

    await db.update('localprofiles', data, where: "id = ?", whereArgs: [id]);

}

// Delete profile

static Future<void> deleteProfile(int id) async {

    //Delete profile associated with the id inputted

    final db = await SQLiteLocalProfiles.db();

    try {

        //Try to delete profile
```

```
        await db.delete("localprofiles", where: "id = ?", whereArgs: [id]);  
    } catch (err) {  
        // If it fails, prints the error to the console  
        debugPrint("Something went wrong when deleting an item: $err");  
    }  
}  
}
```

queryserver.dart

```
import 'package:http/http.dart' as http;  
  
import 'dart:convert' as convert;  
  
  
class QueryServer {  
  
    static Future<Map> query(phpurl, body) async {  
  
        //Get url and data being sent  
  
        try {  
  
            //Try to send data to server  
  
            var res = await http.post(Uri.parse(phpurl), body: body);  
  
            if (res.statusCode != 200) {  
  
                //If the server rejects the data return the error code  
  
                return {"error": true, "message": "Error ${res.statusCode}"};  
            }  
        } catch (e) {  
            print(e);  
        }  
    }  
}
```

```

} else {

    // If server successfully gets data

    var data = convert.json.decode(res.body); // Get the data sent back from
the server and decode

    if (data["error"]) {

        //If the data has an error

        return {"error": true, "message": data["message"]}; //Return error 500
(server rejects data)

    } else {

        return {"error": false, "message": data};

    }

}

} catch (e) {

    //If there is an error, return error message

    return {"error": true, "message": "ERROR: $e"};

}

}

```

setselected.dart

```

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/queryserver.dart';

```

```
import 'package:shared_preferences/shared_preferences.dart';

class SetSelected {

    static Future<bool> selectProfile(
        serverid, firstname, lastname, email) async {
        dynamic favReataurantList = await QueryServer.query(
            //Get favourites from server
            'https://alleat.cpur.net/query/favouriterestaurantlist.php',
            {"profileemail": email});

        if (favReataurantList["Error"] == true) {
            //If getting favourites fails, return false
            return false;
        } else {
            try {
                //Try to change the shared preferences to the selected profile
                final prefs = await SharedPreferences.getInstance();
                await prefs.setString('serverprofileid', serverid.toString());
                await prefs.setString('firstname', firstname);
                await prefs.setString('lastname', lastname);
                await prefs.setString('email', email);
                await prefs.setString('favrestaurants',

```

```
((favReataurantList["message"])[ "restaurantids" ]).toString());  
  
SQLiteLocalProfiles.selectProfile(  
  
    email); //Select profile in the database  
  
return true;  
  
} catch (e) {  
  
    //If there is an error, return false  
  
    return false;  
  
}  
  
}  
  
}  
  
}
```

theme.dart

```
library globals;  
  
  
import 'package:flutter/material.dart';  
  
  
dynamic appthemepreference = 0;  
  
  
  
class ThemeClass {  
  
    static Color primaryLight = const Color(0xff5806FF);
```

```
static Color secondaryLight = const Color(0xffAD84FF);

static Color successLight = const Color(0xff07852A);

static Color errorLight = const Color(0xffAF0E17);

static Color textLight = const Color(0xff1A1A1A);

static Color bgLight = const Color(0xffFAFAFA);

static Color bottomAppBarLight = const Color(0xffffffff);

static Color primaryDark = const Color(0xffC1A3FF);

static Color secondaryDark = const Color(0xffAD84FF);

static Color successDark = const Color(0xff60F68A);

static Color errorDark = const Color(0xFF2555F);

static Color textDark = const Color(0xffEAEEAE);

static Color bgDark = const Color(0xff0C0C0C);

static Color bottomAppBarDark = const Color(0xff161616);

static Color mono900 = const Color(0xff0C0C0C);

static Color mono800 = const Color(0xff161616);

static Color mono700 = const Color(0xff292929);

static Color mono600 = const Color(0xff434343);

static Color mono500 = const Color(0xff676767);

static Color mono400 = const Color(0xff818181);

static Color mono300 = const Color(0xff979797);
```

```
static Color mono200 = const Color(0xffB6B6B6);

static Color mono100 = const Color(0xffD2D2D2);

static Color mono050 = const Color(0xffEAEEAE);

static Color mono000 = const Color(0xffffffff);

static Color primary900 = const Color(0xff060011);

static Color primary800 = const Color(0xff160041);

static Color primary700 = const Color(0xff2B0082);

static Color primary600 = const Color(0xff4100C4);

static Color primary500 = const Color(0xff5806FF);

static Color primary400 = const Color(0xff9866FF);

static Color primary300 = const Color(0xffAD84FF);

static Color primary200 = const Color(0ffc1A3FF);

static Color primary100 = const Color(0ffd6C2FF);

static Color primary050 = const Color(0xffEBE0FF);

static ThemeData lightTheme = ThemeData(

    //COLOUR

    primaryColor: primaryLight,

    colorScheme: ColorScheme.fromSwatch().copyWith(
```

```
secondary: secondaryLight,  
  
error: errorLight,  
  
tertiary: successLight,  
  
onBackground: textLight,  
  
onSurface: mono000,  
,  
  
// MAIN APP  
  
bottomNavigationBarTheme: BottomNavigationBarThemeData(  
  
    backgroundColor: mono000,  
  
    selectedItemColor: primaryLight,  
  
    unselectedItemColor: mono400),  
  
appBarTheme: AppBarTheme(backgroundColor: primaryLight),  
  
backgroundColor: bgLight,  
  
scaffoldBackgroundColor: bgLight,  
  
snackBarTheme: SnackBarThemeData(  
  
    contentTextStyle: TextStyle(  
  
        color: mono900,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w600,  
  
        fontSize: 14),
```

```
backgroundColor: mono100),  
  
// TEXT  
  
textTheme: TextTheme(  
  
    headline1: TextStyle(  
  
        color: textLight,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w800,  
  
        fontSize: 32),  
  
    headline2: TextStyle(  
  
        color: mono900,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w700,  
  
        fontSize: 28),  
  
    headline3: TextStyle(  
  
        color: mono800,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w600,  
  
        fontSize: 21),  
  
    headline4: TextStyle(  
  
        color: mono800,
```

```
        fontFamily: 'Satoshi',  
        fontWeight: FontWeight.w500,  
        fontSize: 21),  
  
    headline5: TextStyle(  
  
        color: mono500,  
        fontFamily: 'Satoshi',  
        fontWeight: FontWeight.w600,  
        fontSize: 20),  
  
    headline6: TextStyle(  
  
        color: mono400,  
        fontFamily: 'Satoshi',  
        fontWeight: FontWeight.w600,  
        fontSize: 16),  
  
    bodyText1: TextStyle(  
  
        color: textLight,  
        fontWeight: FontWeight.w600,  
        fontFamily: 'NotoSans',  
        fontSize: 14),  
  
    bodyText2: TextStyle(  
  
        color: textLight,  
        fontWeight: FontWeight.w400,  
        fontFamily: 'NotoSans',
```



```
        fontSize: 16))),

outlinedButtonTheme: OutlinedButtonThemeData(
    style: OutlinedButton.styleFrom(
        foregroundColor: primaryLight,
        padding: const EdgeInsets.only(
            top: 15, bottom: 15, left: 30, right: 30),
        side: BorderSide(color: primaryLight, width: 3),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(5), // <-- Radius
        ),
        textStyle: TextStyle(
            color: primaryLight,
            fontFamily: 'Satoshi',
            fontWeight: FontWeight.w600,
            fontSize: 16))),
textButtonTheme: TextButtonThemeData(
    style: TextButton.styleFrom(
        foregroundColor: primaryLight,
        padding: const EdgeInsets.only(
            top: 18, bottom: 18, left: 30, right: 30),
        textStyle: TextStyle(
            color: primaryLight,
```

```
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16))),
//FORM FIELD

inputDecorationTheme: InputDecorationTheme(
    filled: false,
    hintStyle: TextStyle(
        color: mono300,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16),
    contentPadding: const EdgeInsets.all(10),
    border: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono100)),
    focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: primaryLight)),
    disabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono050)),
    errorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorLight))),
```



```
headline4: TextStyle(  
    color: mono100,  
    fontFamily: 'Satoshi',  
    fontWeight: FontWeight.w500,  
    fontSize: 21),  
  
headline5: TextStyle(  
    color: mono300,  
    fontFamily: 'Satoshi',  
    fontWeight: FontWeight.w500,  
    fontSize: 20),  
  
headline6: TextStyle(  
    color: mono400,  
    fontFamily: 'Satoshi',  
    fontWeight: FontWeight.w600,  
    fontSize: 16),  
  
bodyText1: TextStyle(  
    color: textDark,  
    fontWeight: FontWeight.w600,  
    fontFamily: 'NotoSans',  
    fontSize: 14),  
  
bodyText2: TextStyle(  
    color: mono000,
```

```
fontWeight: FontWeight.w400,  
  
fontFamily: 'NotoSans',  
  
fontSize: 16)),  
  
//BUTTONS  
  
buttonTheme: ButtonThemeData(  
  
shape:  
  
    RoundedRectangleBorder(borderRadius: BorderRadius.circular(10.0)),  
  
padding:  
  
    const EdgeInsets.only(top: 18, bottom: 18, left: 30, right: 30)),  
  
elevatedButtonTheme: ElevatedButtonThemeData(  
  
style: ElevatedButton.styleFrom(  
  
foregroundColor: bgDark,  
  
backgroundColor: primaryDark,  
  
padding: const EdgeInsets.only(  
  
    top: 15, bottom: 15, left: 30, right: 30),  
  
shape: RoundedRectangleBorder(  
  
borderRadius: BorderRadius.circular(5), // <-- Radius  
),  
  
textStyle: TextStyle(  
  
color: mono900,
```

```
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16))),
outlinedButtonTheme: OutlinedButtonThemeData(
    style: OutlinedButton.styleFrom(
        foregroundColor: primaryDark,
        padding: const EdgeInsets.only(
            top: 15, bottom: 15, left: 30, right: 30),
        side: BorderSide(color: primaryDark, width: 3),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(5), // <-- Radius
        ),
        // minimumSize: const Size.fromHeight(50),
    ),
    textStyle: TextStyle(
        color: primaryDark,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16))),
textButtonTheme: TextButtonThemeData(
    style: TextButton.styleFrom(
        foregroundColor: primaryDark,
```

```
padding: const EdgeInsets.only(
    top: 18, bottom: 18, left: 30, right: 30),
textStyle: TextStyle(
    color: primaryDark,
    fontFamily: 'Satoshi',
    fontWeight: FontWeight.w600,
    fontSize: 16)),


// FORM FIELD


inputDecorationTheme: InputDecorationTheme(
    filled: false,
    hintStyle: TextStyle(
        color: mono400,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16),
    contentPadding: const EdgeInsets.all(10),
    border: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono700)),
    focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: primaryDark)),
```

```
disabledBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: mono800)),  
  
errorBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: errorDark)),  
  
focusedErrorBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: errorLight)),  
  
enabledBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: mono600))));  
}
```

browse_categories.dart

```
library globals;  
  
  
  
import 'package:flutter/material.dart';  
  
  
  
dynamic appthemepreference = 0;  
  
  
  
class ThemeClass {  
  
    static Color primaryLight = const Color(0xff5806FF);  
  
    static Color secondaryLight = const Color(0xffAD84FF);  
  
    static Color successLight = const Color(0xff07852A);
```

```
static Color errorLight = const Color(0xffAF0E17);

static Color textLight = const Color(0xff1A1A1A);

static Color bgLight = const Color(0xffFAFAFA);

static Color bottomAppBarLight = const Color(0xffffffff);

static Color primaryDark = const Color(0xffC1A3FF);

static Color secondaryDark = const Color(0xffAD84FF);

static Color successDark = const Color(0xff60F68A);

static Color errorDark = const Color(0xffF2555F);

static Color textDark = const Color(0xffEAEEAE);

static Color bgDark = const Color(0xff0C0C0C);

static Color bottomAppBarDark = const Color(0xff161616);

static Color mono900 = const Color(0xff0C0C0C);

static Color mono800 = const Color(0xff161616);

static Color mono700 = const Color(0xff292929);

static Color mono600 = const Color(0xff434343);

static Color mono500 = const Color(0xff676767);

static Color mono400 = const Color(0xff818181);

static Color mono300 = const Color(0xff979797);

static Color mono200 = const Color(0xffB6B6B6);

static Color mono100 = const Color(0xffD2D2D2);
```

```
static Color mono050 = const Color(0xffEAEAEA);

static Color mono000 = const Color(0xffffffff);

static Color primary900 = const Color(0xff060011);

static Color primary800 = const Color(0xff160041);

static Color primary700 = const Color(0xff2B0082);

static Color primary600 = const Color(0xff4100C4);

static Color primary500 = const Color(0xff5806FF);

static Color primary400 = const Color(0xff9866FF);

static Color primary300 = const Color(0xffAD84FF);

static Color primary200 = const Color(0ffc1A3FF);

static Color primary100 = const Color(0ffd6C2FF);

static Color primary050 = const Color(0xffEBE0FF);

static ThemeData lightTheme = ThemeData(

    //COLOUR

    primaryColor: primaryLight,

    colorScheme: ColorScheme.fromSwatch().copyWith(

        secondary: secondaryLight,

        error: errorLight,
```



```
// TEXT

textTheme: TextTheme(
    headline1: TextStyle(
        color: textLight,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w800,
        fontSize: 32),
    headline2: TextStyle(
        color: mono900,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w700,
        fontSize: 28),
    headline3: TextStyle(
        color: mono800,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 21),
    headline4: TextStyle(
        color: mono800,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w500,
```

```
fontSize: 21),  
  
headline5: TextStyle(  
  
    color: mono500,  
  
    fontFamily: 'Satoshi',  
  
    fontWeight: FontWeight.w600,  
  
    fontSize: 20),  
  
headline6: TextStyle(  
  
    color: mono400,  
  
    fontFamily: 'Satoshi',  
  
    fontWeight: FontWeight.w600,  
  
    fontSize: 16),  
  
bodyText1: TextStyle(  
  
    color: textLight,  
  
    fontWeight: FontWeight.w600,  
  
    fontFamily: 'NotoSans',  
  
    fontSize: 14),  
  
bodyText2: TextStyle(  
  
    color: textLight,  
  
    fontWeight: FontWeight.w400,  
  
    fontFamily: 'NotoSans',  
  
    fontSize: 16)),
```

```
//BUTTONS

buttonTheme: ButtonThemeData(
    shape:
        RoundedRectangleBorder(borderRadius: BorderRadius.circular(10.0)),
    padding:
        const EdgeInsets.only(top: 18, bottom: 18, left: 30, right: 30)),
elevatedButtonTheme: ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
        foregroundColor: bgLight,
        backgroundColor: primaryLight,
        padding: const EdgeInsets.only(
            top: 15, bottom: 15, left: 30, right: 30),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(5), // <-- Radius
        ),
        textStyle: TextStyle(
            color: bgLight,
            fontFamily: 'Satoshi',
            fontWeight: FontWeight.w600,
            fontSize: 16))),
outlinedButtonTheme: OutlinedButtonThemeData(
```

```
style: OutlinedButton.styleFrom(  
    foregroundColor: primaryLight,  
    padding: const EdgeInsets.only(  
        top: 15, bottom: 15, left: 30, right: 30),  
    side: BorderSide(color: primaryLight, width: 3),  
    shape: RoundedRectangleBorder(  
        borderRadius: BorderRadius.circular(5), // <-- Radius  
    ),  
    textStyle: TextStyle(  
        color: primaryLight,  
        fontFamily: 'Satoshi',  
        fontWeight: FontWeight.w600,  
        fontSize: 16))),  
  
textButtonTheme: TextButtonThemeData(  
    style: TextButton.styleFrom(  
        foregroundColor: primaryLight,  
        padding: const EdgeInsets.only(  
            top: 18, bottom: 18, left: 30, right: 30),  
        textStyle: TextStyle(  
            color: primaryLight,  
            fontFamily: 'Satoshi',  
            fontWeight: FontWeight.w600,
```

```
        fontSize: 16))),

//FORM FIELD

inputDecorationTheme: InputDecorationTheme(
    filled: false,
    hintStyle: TextStyle(
        color: mono300,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16),
    contentPadding: const EdgeInsets.all(10),
    border: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono100)),
    focusedBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: primaryLight)),
    disabledBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: mono050)),
    errorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorLight)),
    focusedErrorBorder: UnderlineInputBorder(
        borderSide: BorderSide(width: 3, color: errorDark)),
```

```
enabledBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: mono100))));  
  
static ThemeData darkTheme = ThemeData(  
  
    //COLOUR  
  
    primaryColor: primaryDark,  
    colorScheme: ColorScheme.fromSwatch().copyWith(  
        secondary: secondaryDark,  
        error: errorDark,  
        tertiary: successDark,  
        onBackground: textDark,  
        onSurface: mono700,  
    ),  
  
    //MAIN APP  
  
    bottomNavigationBarTheme: BottomNavigationBarThemeData(  
        backgroundColor: mono800, unselectedItemColor: mono600),  
        appBarTheme: AppBarTheme(backgroundColor: mono800),  
        backgroundColor: bgDark,
```

```
scaffoldBackgroundColor: bgDark,  
  
// TEXT  
  
textTheme: TextTheme(  
  
    headline1: TextStyle(  
  
        color: textDark,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w800,  
  
        fontSize: 32),  
  
    headline2: TextStyle(  
  
        color: mono050,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w700,  
  
        fontSize: 28),  
  
    headline3: TextStyle(  
  
        color: mono100,  
  
        fontFamily: 'Satoshi',  
  
        fontWeight: FontWeight.w600,  
  
        fontSize: 21),  
  
    headline4: TextStyle(  
  
        color: mono100,
```

```
fontFamily: 'Satoshi',
fontWeight: FontWeight.w500,
fontSize: 21),

headline5: TextStyle(
color: mono300,
fontFamily: 'Satoshi',
fontWeight: FontWeight.w500,
fontSize: 20),

headline6: TextStyle(
color: mono400,
fontFamily: 'Satoshi',
fontWeight: FontWeight.w600,
fontSize: 16),

bodyText1: TextStyle(
color: textDark,
fontWeight: FontWeight.w600,
fontFamily: 'NotoSans',
fontSize: 14),

bodyText2: TextStyle(
color: mono000,
fontWeight: FontWeight.w400,
fontFamily: 'NotoSans',
```

```
    fontSize: 16)),  
  
//BUTTONS  
  
buttonTheme: ButtonThemeData(  
  
    shape:  
  
        RoundedRectangleBorder(borderRadius: BorderRadius.circular(10.0)),  
  
    padding:  
  
        const EdgeInsets.only(top: 18, bottom: 18, left: 30, right: 30)),  
  
elevatedButtonTheme: ElevatedButtonThemeData(  
  
    style: ElevatedButton.styleFrom(  
  
        foregroundColor: bgDark,  
  
        backgroundColor: primaryDark,  
  
        padding: const EdgeInsets.only(  
  
            top: 15, bottom: 15, left: 30, right: 30),  
  
        shape: RoundedRectangleBorder(  
  
            borderRadius: BorderRadius.circular(5), // <-- Radius  
>),  
  
        textStyle: TextStyle(  
  
            color: mono900,  
  
            fontFamily: 'Satoshi',  
  
            fontWeight: FontWeight.w600,
```

```
        fontSize: 16))),

outlinedButtonTheme: OutlinedButtonThemeData(
    style: OutlinedButton.styleFrom(
        foregroundColor: primaryDark,
        padding: const EdgeInsets.only(
            top: 15, bottom: 15, left: 30, right: 30),
        side: BorderSide(color: primaryDark, width: 3),
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(5), // <-- Radius
        ),
        // minimumSize: const Size.fromHeight(50),
    ),

    textStyle: TextStyle(
        color: primaryDark,
        fontFamily: 'Satoshi',
        fontWeight: FontWeight.w600,
        fontSize: 16)),
)

textButtonTheme: TextButtonThemeData(
    style: TextButton.styleFrom(
        foregroundColor: primaryDark,
        padding: const EdgeInsets.only(
            top: 18, bottom: 18, left: 30, right: 30),
    ),
)
```



```
errorBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: errorDark)),  
  
focusedErrorBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: errorLight)),  
  
enabledBorder: UnderlineInputBorder(  
    borderSide: BorderSide(width: 3, color: mono600))));  
}
```

elements.dart

```
import 'package:flutter/material.dart';

class ScreenBackButton extends StatelessWidget {

    final dynamic data;

    const ScreenBackButton({Key? key, this.data})

        : super(key: key); //Get any data that needs to be sent to previous screen

    @override

    Widget build(BuildContext context) {

        return Padding(
            padding: const EdgeInsets.only(top: 50),
            child: TextButton.icon(

```

```
 onPressed: () =>

    Navigator.of(context).pop(data), //Remove current screen

    label: Text(
        "Back",
        style: Theme.of(context)
            .textTheme
            .headline6!
            .copyWith(color: Theme.of(context).primaryColor),
    ),
    icon: Icon(
        Icons.chevron_left,
        color: Theme.of(context).primaryColor,
    )));
}

}
```

profiles_buttons.dart

```
import 'package:flutter/material.dart';

class ProfileButton extends StatelessWidget {
    final dynamic icon;
```



```
        ],  
        ),  
        child: InkWell(  
            child: Row(  
                children: [  
                    Icon(icon, color: Theme.of(context).colorScheme.onBackground),  
                    const SizedBox(width: 20),  
                    Text(  
                        name,  
                        style: Theme.of(context).textTheme.headline5?.copyWith(  
                            color: Theme.of(context).colorScheme.onBackground,  
                            fontWeight: FontWeight.w700),  
                    )  
                ],  
            ),  
            onTap: () {  
                action;  
            },  
        ),  
    );  
}  
}
```

profiles_selection.dart

```
import 'package:alleat/screens/profilesetup/profilesetup_existing.dart';

import 'package:alleat/screens/profilesetup/welcomeScreen.dart';

import 'package:alleat/services/localprofiles_service.dart';

import 'package:alleat/services/setselected.dart';

import 'package:alleat/widgets/navbar.dart';

import 'package:flutter/material.dart';

class ProfileList extends StatefulWidget {

  const ProfileList({super.key});

  @override

  State<ProfileList> createState() => _ProfileListState();
}

class _ProfileListState extends State<ProfileList> {

  bool edit = false;

  Future<List> getDisplayableProfiles() async {

    return await SQLiteLocalProfiles.getDisplayProfilesList();
}
```

```
}

Future<void> selectProfile(id) async {

    var getToSelect = (await SQLiteLocalProfiles.getProfileFromID(id))[0];

    bool trySelect =
        await SetSelected.selectProfile(getToSelect['profileid'],
getToSelect['firstname'], getToSelect['lastname'], getToSelect['email']);

    if (trySelect == true) {
        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                    content: Text(
                        'Switched profiles successfully',
                    ),
                ),
            );
        });
    } else {
        setState(() {
            ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
                content: Text(
                    'Failed to switch profiles.',
                ),
            ));
        });
    }
}
```



```
Container(  
    width: 120,  
    height: 120,  
    decoration: BoxDecoration(shape: BoxShape.circle,  
color: Theme.of(context).navigationBarTheme.backgroundColor)  
,  
);  
  
{ else if (snapshot.hasData) {  
  
    // Once there is information  
  
    List? profileInfo = snapshot.data; // Store in profileInfo  
  
  
  
    if (profileInfo != null && profileInfo.isNotEmpty) {  
  
        //If it contains a profile  
  
        return Padding(  
            padding: const EdgeInsets.only(left: 30),  
            child: SizedBox(  
  
                //Set width of data to the number of profiles by  
120px  
  
                width: (profileInfo.length) * 120,  
                child: ListView.builder(  
  
                    //For each profile  
  
                    itemCount: profileInfo.length,  
                    scrollDirection: Axis.horizontal, //Make  
scrollable list
```

```
itemBuilder: (context, index) {  
  
    Map<String, dynamic> profile =  
profileInfo[index]; //Store current profile being created in profile  
  
    if (index == 0) {  
  
        return InkWell(  
  
            onTap: (() {}),  
  
            onLongPress: () {  
  
                //Remove profile from device  
  
                showDialog(  
  
                    context: context,  
  
                    builder: (BuildContext context)  
=> AlertDialog(  
  
                        backgroundColor:  
Theme.of(context).backgroundColor,  
  
                        title: Text(  
  
                            'Remove Profile',  
  
                            style:  
Theme.of(context)  
  
                                .textTheme  
  
                                .headline5  
  
                                ?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
                        ),  
  
                        content: Text(  
)
```

```
'Are you sure you want  
to remove this profile from the device',  
  
        style:  
Theme.of(context).textTheme.bodyText2,  
  
(),  
  
actions: <Widget>[  
  
    TextButton(  
  
        //Cancel button to  
close the popup  
  
        onPressed: () =>  
Navigator.pop(context, 'Cancel'),  
  
        child: const  
Text('Cancel'),  
  
(),  
  
    TextButton(  
  
        //On confirm button  
press delete profile from device and if there is no remaining profiles, go to  
welcome screen  
  
        onPressed: () async {  
  
SQLiteLocalProfiles.deleteProfile(profileInfo[index]["id"]);  
  
Navigator.pop(context);  
  
        var  
availableProfiles = await SQLiteLocalProfiles.getDisplayProfilesList();  
  
        if  
(availableProfiles.isEmpty) {
```

```

Navigator.pop(context);

Navigator.of(context).push(
  MaterialPageRoute(builder: (_) => const ProfileSetupWelcome()),
);

} else {
  //If there are
  profiles remaining get the first profile in the database and set the selected
  profile to the first profile

  var
newSelectedProfile = await SQLiteLocalProfiles.getFirstProfile();

  bool
checkSelected = await SetSelected.selectProfile(
  newSelectedProfile[0]["profileid"],

  newSelectedProfile[0]["firstname"],

  newSelectedProfile[0]["lastname"],

  newSelectedProfile[0]["email"]);

  if (checkSelected
== true) {
    setState(() {
  {}});
}
}

```

```
        },

        child: const

Text('Confirm'),

        ),

    ]));

},

child: Container(

padding: const

EdgeInsets.symmetric(horizontal: 10),

child: Column(children: [

Stack(alignment:

Alignment.center, children: [

Container(

width: 93,

height: 93,

decoration: BoxDecoration(

shape: BoxShape.circle,

border:

Border.all(color:

Theme.of(context).primaryColor, width: 3, style: BorderStyle.solid)),

),

Container(
```

```

                // Create a circle with a
purple outline and the first letter of first and last name

                width: 80,
height: 80,
decoration: BoxDecoration(
shape: BoxShape.circle,
color:
Color.fromRGBO(profile['profilecolorred'], profile['profilecolorgreen'],
profile['profilecolorblue'], 1)),
child: Align(
alignment:
Alignment.center,
child:
Text('${profile['firstname'][0]}${profile['lastname'][0]}',
style:
Theme.of(context)

.textTheme
.headline3

?.copyWith(color: Theme.of(context).backgroundColor)))))

]),

const SizedBox(height: 15),
SizedBox(
//Display profile name

```

```

width: 100,
child: Text(
"${profile['firstname']}",
textAlign:
TextAlign.center,
style:
Theme.of(context).textTheme.headline5?.copyWith(
fontWeight:
FontWeight.w600, color: Theme.of(context).textTheme.headline1?.color),
overflow:
TextOverflow.ellipsis,
)))
]));
} else if (index > 0) {
return InkWell(
onTap: () {
selectProfile(profile['id']);
}),
onLongPress: () {
//Remove profile from device
showDialog(
context: context,
builder: (BuildContext context)
=> AlertDialog(

```

```
        backgroundColor:  
Theme.of(context).backgroundColor,  
  
        title: Text(  
          'Remove Profile',  
  
        style:  
Theme.of(context)  
  
          .textTheme  
          .headline5  
  
          ?.copyWith(color:  
Theme.of(context).textTheme.headline1?.color),  
  
) ,  
  
        content: Text(  
          'Are you sure you want  
to remove this profile from the device',  
  
        style:  
Theme.of(context).textTheme.bodyText2,  
  
) ,  
  
        actions: <Widget>[  
          TextButton(  
            //Cancel button to  
close the popup  
  
            onPressed: () =>  
Navigator.pop(context, 'Cancel'),  
  
            child: const  
Text('Cancel'),  
  
) ,
```

```
TextButton(  
    //On confirm button  
    press delete profile from device and if there is no remaining profiles, go to  
    welcome screen  
  
    onPressed: () async {  
  
SQLiteLocalProfiles.deleteProfile(profileInfo[index]["id"]);  
  
Navigator.pop(context);  
  
    var  
availableProfiles = await SQLiteLocalProfiles.getDisplayProfilesList();  
  
    if  
(availableProfiles.isEmpty) {  
  
Navigator.pop(context);  
  
Navigator.pop(context);  
  
Navigator.of(context).push(  
  
MaterialPageRoute(builder: (_) => const ProfileSetupWelcome()),  
  
);  
  
} else {  
  
    //If there are  
    profiles remaining get the first profile in the database and set the selected  
    profile to the first profile  
  
    var  
newSelectedProfile = await SQLiteLocalProfiles.getFirstProfile();
```

```

    bool
checkSelected = await SetSelected.selectProfile(
  newSelectedProfile[0]["profileid"],
  newSelectedProfile[0]["firstname"],
  newSelectedProfile[0]["lastname"],
  newSelectedProfile[0]["email"]);

  if (checkSelected == true) {
    setState(() {}
  });
}

},
child: const
Text('Confirm'),
),
]);
},
child: Container(
padding: const
EdgeInsets.symmetric(horizontal: 10),
child: Column(children: [

```

```

Container(
    // Create a circle with a
purple outline and and the first letter of first and last name

    width: 88,
    height: 88,
    decoration: BoxDecoration(
        shape: BoxShape.circle,
        color:
Color.fromRGBO(profile['profilecolorred'], profile['profilecolorgreen'],
profile['profilecolorblue'], 1)),

    child: Align(
        alignment:
Alignment.center,
        child:
Text('${profile['firstname'][0]}${profile['lastname'][0]}',
style:
Theme.of(context)

        .textTheme
        .headline3
        ?.copyWith(color:
Theme.of(context).backgroundColor))),

    const SizedBox(height: 15),
    SizedBox(
        //Display profile name

```

```
        width: 100,  
  
        child: Text(  
  
          "${profile['firstname']}",  
  
          textAlign:  
          TextAlign.center,  
  
          style:  
          Theme.of(context).textTheme.headline5?.copyWith(  
  
            fontWeight:  
            FontWeight.w600, color: Theme.of(context).textTheme.headline1?.color),  
  
            overflow:  
            TextOverflow.ellipsis,  
  
          )),  
  
        ]));  
  
      } else {  
  
        return Container(  
  
          padding: const  
          EdgeInsets.symmetric(horizontal: 10),  
  
          child: Column(children: [  
  
            Container(  
  
              // Create a circle with a purple  
              outline and and the first letter of first and last name  
  
              width: 100,  
  
              height: 100,  
  
              decoration: BoxDecoration(shape:  
              BoxShape.circle, color: Theme.of(context).colorScheme.error),  
        
```

```
        child: Align(  
  
            alignment: Alignment.center,  
  
            child: Text(  
  
                "Error",  
  
                style:  
Theme.of(context).textTheme.headline5?.copyWith(color: const Color(0xffffffff)),  
  
(  
  
))  
  
]));  
  
}  
  
});  
  
} else {  
  
    return Column(  
  
        children: [  
  
            Container(width: 100, height: 100, decoration:  
BoxDecoration(shape: BoxShape.circle, color: Theme.of(context).errorColor))  
  
        ],  
  
);  
  
}  
  
} else {  
  
    return Column(  
  
        children: [  
  
            Container(width: 100, height: 100, decoration:  
BoxDecoration(shape: BoxShape.circle, color: Theme.of(context).errorColor))  
  
        ],  
  
);  
  
}
```

```
        ],
      );
    }
  )),
Column(
  children: [
    Padding(
      padding: const EdgeInsets.symmetric(horizontal: 20),
      child: Column(children: [
        ElevatedButton(
          onPressed: () {
            Navigator.push(context, MaterialPageRoute(builder: (context) => const ProfileSetupExisting()));
          },
          style: ElevatedButton.styleFrom(shape: const CircleBorder(), padding: const EdgeInsets.all(22)),
          child: Icon(
            Icons.add,
            color: Theme.of(context).backgroundColor,
            size: 40,
          ),
        ),
        const SizedBox(height: 20),
      ],
    ),
  ],
);
```

```
        ]))

    ],
)

])
]));
```

search.dart

```
import 'package:flutter/material.dart';

class SearchBar extends StatelessWidget {

  const SearchBar({super.key});

  @override
  Widget build(BuildContext context) {
    return InkWell(
      child: Container(
        width: double.infinity,
        height: 60,
        margin: const EdgeInsets.symmetric(vertical: 10, horizontal: 30),

```

```
padding: const EdgeInsets.symmetric(horizontal: 20),  
  
decoration: BoxDecoration(  
  
    borderRadius: const BorderRadius.all(Radius.circular(10)),  
  
    color: Theme.of(context).colorScheme.onSurface,  
  
    boxShadow: [  
  
        BoxShadow(  
  
            color: Theme.of(context).colorScheme.onBackground.withOpacity(0.05),  
  
            spreadRadius: 10,  
  
            blurRadius: 45,  
  
            offset: const Offset(0, 30), // changes position of shadow  
  
  
    ],  
,  
  
child: Row(children: [  
  
    Icon(  
  
        Icons.search,  
  
        color: Theme.of(context).textTheme.headline1?.color,  
  
    ),  
  
    const SizedBox(  
  
        width: 20,  
  
    ),  
  
    Text(  

```

```
'Try searching "Pizza"',
      style: Theme.of(context).textTheme.bodyText2,
    )
  ],
));
}

}
```

genericloading.dart

```
import 'package:flutter/material.dart';

class GenericLoading extends StatefulWidget {
  const GenericLoading({super.key});

  @override
  State<GenericLoading> createState() => _GenericLoadingState();
}

class _GenericLoadingState extends State<GenericLoading> {
  @override
  Widget build(BuildContext context) {
```

```
return MaterialApp(  
    title: "Loading",  
    home: Scaffold(  
        resizeToAvoidBottomInset: false,  
        body: Column(  
            mainAxisAlignment: MainAxisAlignment.center,  
            crossAxisAlignment: CrossAxisAlignment.center,  
            children: const [CircularProgressIndicator()])),  
    theme: Theme.of(context)); //Show blank page with circular loading circle  
)  
}  
}
```

navigationbar.dart

```
import 'package:alleat/screens/navigationscreens/browse.dart';  
  
import 'package:alleat/screens/navigationscreens/foryou.dart';  
  
import 'package:alleat/screens/navigationscreens/homepage.dart';  
  
import 'package:alleat/screens/navigationscreens/profiles.dart';  
  
  
class Navigation extends StatefulWidget {  
  
    const Navigation({Key? key}) : super(key: key);  
}
```

```
@override

State<Navigation> createState() => _NavigationState();

}

class _NavigationState extends State<Navigation> {

int _selectedIndex = 0; //Start at Home page

final List _screens = [
    {"screen": const HomePage()},
    {"screen": const ForYouPage()},
    {"screen": const BrowsePage()},
    {"screen": const ProfilePage()}
];

void _onItemTapped(int index) {
    //When user click button on bottom navigation bar, change to that index
    setState(() {
        _selectedIndex = index;
    });
}
```

```
@override

Widget build(BuildContext context) {

  return WillPopScope(
    onWillPop: () async => false,
    child: MaterialApp(
      title: "All Eat.",
      theme: Theme.of(context),
      home: Scaffold(
        body: _screens[_selectedIndex]["screen"],
        bottomNavigationBar: Theme(
          data: Theme.of(context).copyWith(
            canvasColor: Theme.of(context)
              .bottomNavigationBarTheme
              .backgroundColor),
        child: BottomNavigationBar(
          // If button is selected, show purple. If button is not
          // selected show greyed out text and icon
          selectedItemColor: Theme.of(context)
            .bottomNavigationBarTheme
            .selectedItemColor,
          unselectedItemColor: Theme.of(context)
            .bottomNavigationBarTheme
            .unselectedItemColor,
        
```

```
backgroundColor: Theme.of(context)

    .bottomNavigationBarTheme

    .backgroundColor,

items: <BottomNavigationBarItem>[

    //Bottom navigation bar items

    BottomNavigationBarItem(
        icon: Container(
            padding: const EdgeInsets.only(
                bottom:
                    3), //Each have padding to separate from the
text

        child: const Icon(Icons
            .home_outlined)), //Unselected icon is outlined

        label: 'Home',

        activeIcon: Container(
            padding: const EdgeInsets.only(bottom: 3),
            child: const Icon(
                Icons.home)), // Selected icon is filled

    ),
    BottomNavigationBarItem(
        //Bottom navigation bar options

        icon: Container(
            padding: const EdgeInsets.only(bottom: 3),
            child:

```

```
        child: const Icon(Icons.assistant_outlined)),  
  
        label: 'For You',  
  
        activeIcon: Container(  
  
            padding: const EdgeInsets.only(bottom: 3),  
  
            child: const Icon(Icons.assistant)),  
  
(),  
  
BottomNavigationBarItem(  
  
    icon: Container(  
  
        padding: const EdgeInsets.only(bottom: 3),  
  
        child: const Icon(Icons.manage_search_rounded)),  
  
    label: 'Browse',  
  
    activeIcon: Container(  
  
        padding: const EdgeInsets.only(bottom: 3),  
  
        child: const Icon(Icons.manage_search)),  
  
(),  
  
BottomNavigationBarItem(  
  
    icon: Container(  
  
        padding: const EdgeInsets.only(bottom: 3),  
  
        child: const Icon(Icons.person_outlined)),  
  
    label: 'Profile',  
  
    activeIcon: Container(  
  
        padding: const EdgeInsets.only(bottom: 3),
```

```

        child: const Icon(Icons.person)),
    ),
],
onTap: _onItemTapped, //On tap, change selected option
currentIndex:
    _selectedIndex, //Make current index the one last
selected (defaults to home)

),
))));

}

}

```

restaurants_list.dart

```

import 'package:alleat/screens/restaurant/restaurant_main.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';

import 'dart:math';

class RestaurantList extends StatefulWidget {
    const RestaurantList({Key? key}) : super(key: key);
}

```

```
    @override

    State<RestaurantList> createState() => _RestaurantListState();

}

class _RestaurantListState extends State<RestaurantList> {

    late bool error, sending, success, serverOffline;

    late String msg;

    Map metadataTemp = {

        "error": false,

        "sort": "distance",

        "favourite": false,

        "price": [1, 2, 3, 4],

        "maxDelivery": 4.0,

        "minOrder": 40.0,

        "latitude": 0.0,

        "longitude": 0.0

    };

}

    @override

    void initState() {

        //Default values
```

```

        error = false;

        sending = false;

        success = false;

        msg = "";

        super.initState();

    }

Future<Map> getUserMetadata() async {

    final prefs = await SharedPreferences.getInstance();

    final String? filterSortEncoded = prefs.getString('filtersort');

    final double? locationLatitude = prefs.getDouble('locationLatitude');

    final double? locationLongitude = prefs.getDouble('locationLongitude');

    final String? profileid = prefs.getString("serverprofileid");

    metadataTemp["profileid"] = profileid;

    if (filterSortEncoded != null) {

        Map filterSort = json.decode(filterSortEncoded);

        metadataTemp["sort"] = filterSort["sort"];

        metadataTemp["favourite"] = filterSort["favourite"];

        metadataTemp["price"] = filterSort["price"];

        metadataTemp["maxDelivery"] = filterSort["maxDelivery"];

        metadataTemp["minOrder"] = filterSort["minOrder"];

    }

}

```

```
    if (locationLatitude == null || locationLatitude == 0 || locationLongitude == null || locationLongitude == 0) {

        metadataTemp["error"] = true;

        return metadataTemp;

    } else {

        metadataTemp["latitude"] = locationLatitude;

        metadataTemp["longitude"] = locationLongitude;

        return metadataTemp;

    }

}

Future<List> getRestaurants() async {

    Map metadata = await getUserMetadata();

    if (metadata["error"] != true) {

        metadata.remove("error");

        switch (metadata["favourite"]) {

            case (true):

                metadata.remove("favourite");

                metadata["favourite"] = "true";

                break;

            case (false):

                metadata.remove("favourite");


```

```

        metadata["favourite"] = "false";

        break;

    }

    try {

        metadata["price"] = metadata["price"].join(",").toString();

    } catch (e) {

        metadata["price"] = "1,2,3,4";

    }

    metadata["maxDelivery"] = metadata["maxDelivery"].toString();

    metadata["profileid"] = metadata["profileid"].toString();

    metadata["minOrder"] = metadata["minOrder"].toString();

    metadata["latitude"] = metadata["latitude"].toString();

    metadata["longitude"] = metadata["longitude"].toString();

    String phpurl = "https://alleat.cpur.net/query/restaurantlist.php"; //Get
restaurant list

    try {

        var res = await http.post(Uri.parse(phpurl), body: metadata);

        if (res.statusCode == 200) {

            //If sends successfully

            var data = json.decode(res.body); //Decode to array

            if (data["error"]) {

                //If fails to perform query

```

```

List error = [
    {"error": true, "message": "Server Response: ${data["message"]}"},
    "restaurants": "[]" //Send blank list of restaurants
];

return error;

} else {

List listdata = await getDistance(data);

if (listdata[0] == false) {

List error = [
{
    "error": true,
    "message": "Failed to get Location. \nTry changing your
destination address",
    "restaurants": "[]"

} //Send blank list of restaurants

];

return error;

} else {

return [listdata[1]];

}

//If success, send the list of restaurants back

}

} else {

```

```
List error = [  
  
    {"error": true, "message": "Error ${res.statusCode}: Failed to  
connect to server.", "restaurants": "[]"}  
  
];  
  
return error;  
  
}  
  
} catch (e) {  
  
List error = [  
  
    {"error": true, "message": "An unexpected error occurred.\n Try  
reopening the app. \n\n ERROR: $e", "restaurants": "[]"}  
  
];  
  
return error;  
  
}  
  
} else {  
  
List error = [  
  
{  
  
    "error": true,  
  
    "message": "Failed to get Location. \nTry changing your destination  
address",  
  
    "restaurants": "[]"  
  
} //Send blank list of restaurants  
  
];  
  
return error;  
  
}
```

```

}

Future<List> getDistance(restaurantdata) async {

    var p = 0.017453292519943295; //Convert constant from degrees to radians

    final prefs = await SharedPreferences.getInstance();

    final double? savedLocationLat = prefs.getDouble('locationLatitude'); //lat2

    final double? savedLocationLng = prefs.getDouble('locationLongitude'); //lng2

    if (savedLocationLat != null && savedLocationLng != null) {

        for (var i = 0; i < restaurantdata["restaurants"].length; i++) {

            double latRestaurant = double.parse(restaurantdata["restaurants"][i][4]);
//lat1

            double lngRestaurant = double.parse(restaurantdata["restaurants"][i][5]);
//lng1

            double distance = 12742 *

                asin(sqrt(0.5 -

                    cos((savedLocationLat - latRestaurant) * p) / 2 +

                    cos(latRestaurant * p) * cos(savedLocationLat * p) * (1 -
cos((savedLocationLng - lngRestaurant) * p)) / 2));

            restaurantdata["restaurants"][i].add(distance);

        }

        return [true, restaurantdata];
    } else {

        return [false, restaurantdata];
    }
}

```

```
}

Future<String> favouriteRestaurant(restaurantID, action) async {

    String phpurl =
        "https://alleat.cpur.net/query/favouriterestaurant.php"; //Favourite &
unfavourite restaurant for sepcific profile using their email

    final prefs = await SharedPreferences.getInstance();

    final String? email = prefs.getString('email');

    try {

        var res = await http.post(Uri.parse(phpurl), body: {
            "action": action.toString(), //Action determines if it will favourite or
unfavourite depending on input

            "profileemail": email,
            "restaurantid": restaurantID,
        });

        if (res.statusCode == 200) {
            //On successfull send

            var data = json.decode(res.body); //Decode to array

            if (data["error"]) {
                return "failed";
            } else {
                getFavourites(); //get favourite restaurant list

                return "success";
            }
        }
    }
}
```

```

    }

} else {

    return "failed";
}

} catch (e) {

    return "failed";
}

}

Future<List> getFavourites() async {

    String phpurl = "https://alleat.cpur.net/query/favouriterestaurantlist.php";
//Get favourite restaurant ids using profile email

    final prefs = await SharedPreferences.getInstance();

    final String? email = prefs.getString('email');

    try {

        var res = await http.post(Uri.parse(phpurl), body: {
            "profileemail": email.toString(),
        });

        if (res.statusCode == 200) {

            //If successfull send data

            var data = json.decode(res.body); //Decode to array

            if (data["error"]) {

                //If error querying data

```

```
List error = [  
  
    {"error": true, "favouriterestaurants": "[]"} //Send empty favourite  
restaurant ids list  
  
];  
  
return error;  
  
} else {  
  
    //If successful query  
  
    List listdata = [data]; //Send list of restaurant ids back  
  
    return listdata;  
  
}  
  
} else {  
  
    List error = [  
  
        {"error": true, "favouriterestaurants": "[]"}  
  
    ];  
  
    return error;  
  
}  
  
} catch (e) {  
  
    List error = [  
  
        {"error": true, "favouriterestaurants": "[]"}  
  
    ];  
  
    return error;  
  
}  
  
}
```

```
@override

Widget build(BuildContext context) {
    return FutureBuilder<List>(
        //Get list of restaurants data from future (rebuild on change of data)
        future: getRestaurants(),
        builder: ((context, snapshot) {
            if (snapshot.hasData) {
                //If received data
                List restaurantsdata = snapshot.data ?? [];
                //Data stored in restaurantsdata
                if (restaurantsdata[0]["error"] == true) {
                    return Padding(
                        padding: const EdgeInsets.only(left: 20, right: 20, top: 10,
                        bottom: 10),
                        child: Container(
                            decoration: BoxDecoration(
                                borderRadius: const
                                BorderRadius.all(Radius.circular(20)),
                                color: Theme.of(context).colorScheme.onSurface,
                                boxShadow: [
                                    BoxShadow(
                                        color: Colors.black.withOpacity(0.1),
                                        spreadRadius: 2,
```

```
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes position of  
shadow  
  
    ),  
  
]),  
  
child: Column(children: [  
  
    Padding(  
  
        padding: const EdgeInsets.all(30),  
  
        child: SizedBox(  
  
            width: double.infinity,  
  
            child: Center(  
  
                child: Column(children: [  
  
                    Text(  
  
                        "${restaurantsdata[0]["message"]}",  
  
                        textAlign: TextAlign.center,  
  
                        style: Theme.of(context).textTheme.headline6,  
  
                    ),  
  
                    const SizedBox(  
  
                        height: 20,  
  
                    ),  
  
                    ElevatedButton(  
  
                        onPressed: () {  
  
                            setState(() {  

```

```

        getRestaurants();

    });

    },
    child: const Text("Retry")

]),

)
])));

} else {

try {

List restaurants = restaurantsdata[0]["restaurants"];

return FutureBuilder<List>(

//Get favourites list from future

future: getFavourites(),

builder: ((context, snapshot) {

if (!snapshot.hasData) {

//While no data received, show loading bar

return const LinearProgressIndicator(color:
Color(0xff4100C4), backgroundColor: Color(0xffEBE0FF));

} else {

//If data received

List restaurantFavourites = snapshot.data ?? [];

return ListView.builder(

```

```

        //Show number of containers depending on number of
restaurants

        physics: const NeverScrollableScrollPhysics(), //Dont
allow scrolling (Done by main page)

        scrollDirection: Axis.vertical,

        shrinkWrap: true,

        itemCount: restaurantsdata[0]["restaurants"].length,
//For each restaurant

        itemBuilder: (context, index) {

            return Center(child: LayoutBuilder(builder: (context,
constraints) {

                if (restaurants.isEmpty) {

                    //If there are no restaurants show container saying
no restaurants

                    return Padding(
padding: const EdgeInsets.only(left: 20, right:
20, top: 10, bottom: 10),

                    child: Container(
decoration: BoxDecoration(
borderRadius: const
BorderRadius.all(Radius.circular(20)),
color: const Color(0xffffffff),
boxShadow: [
BoxShadow(

```

```
        color:  
Colors.black.withOpacity(0.1),  
  
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0, 10), //  
changes position of shadow  
  
    ),  
  
]),  
  
child: Column(children: const [  
  
    Padding(  
  
        padding: EdgeInsets.all(30),  
  
        child: SizedBox(  
  
            width: double.infinity,  
  
            child: Center(child: Text("No  
restaurants found"))), //Display no restaurants text  
  
    )  
  
])));  
  
} else {  
  
    // If there is restaurant  
  
    return InkWell(  
  
        //Create clickable container  
  
        onTap: () {  
  
            List restaurantinfodata =  
restaurantsdata[0]["restaurants"][index];
```

```
        Navigator.push(  
            context,  
            MaterialPageRoute(  
                builder:  
                    (context) => //On tap, send  
                    restaurant data associated with container to main page and go to that new screen  
                    RestaurantMain(  
                        resid:  
                        restaurantinfodata[0],  
                        resname:  
                        restaurantinfodata[1],  
                        reslogo:  
                        restaurantinfodata[2],  
                        resbanner:  
                        restaurantinfodata[3],  
                        resdistance:  
                        (restaurantsdata[0]["restaurants"][index][8]).toStringAsFixed(1),  
                        resordermin:  
                        restaurantinfodata[6],  
                        resdelivery:  
                        restaurantinfodata[7]),  
                    ));  
            },  
            child: Container(  
                margin: const  
                EdgeInsets.symmetric(horizontal: 30, vertical: 20),
```

```
decoration: BoxDecoration(  
    borderRadius: const  
BorderRadius.all(Radius.circular(10)),  
    color:  
Theme.of(context).colorScheme.onSurface,  
) ,  
child: Stack(  
    children: [  
        Container(  
            //Container filled with restaurant  
banner  
            width:  
MediaQuery.of(context).size.width,  
            height: 150,  
            decoration: BoxDecoration(  
                borderRadius: const  
BorderRadius.only(  
                topLeft: Radius.circular(10),  
                topRight: Radius.circular(10),  
) ,  
                image: DecorationImage(  
                    fit: BoxFit.fitWidth,  
                    image:  
NetworkImage(restaurants[index][3].toString()),  
) ,
```

```
        ),  
  
        ),  
  
        Align(  
  
            alignment: Alignment.topCenter,  
  
            child: Container(  
  
                margin: const  
EdgeInsets.only(top: 110),  
  
                width: 70,  
  
                height: 70,  
  
                decoration: BoxDecoration(  
  
                    shape: BoxShape.circle,  
  
                    color:  
Theme.of(context).colorScheme.onSurface,  
  
                ))),  
  
        Align(  
  
            alignment: Alignment.topCenter,  
  
            child: Container(  
  
                margin: const  
EdgeInsets.only(top: 115),  
  
                width: 60,  
  
                height: 60,  
  
                decoration: BoxDecoration(  
  
                    image: DecorationImage(  
  
                        fit: BoxFit.cover,
```



```

        },

        child: Container(
          margin: const
        EdgeInsets.only(top: 10, right: 10),
          width: 45,
          height: 45,
          decoration: BoxDecoration(
            shape: BoxShape.circle,
            color:
          Theme.of(context).colorScheme.onSurface.withOpacity(0.8),
          ),
          child:
        restaurantFavourites[0]["restaurantids"].contains(restaurants[index][0].toString())
        ) ? Icon(Icons.favorite,
          size: 30, // ? =
        favourited
          color:
        Theme.of(context).colorScheme.secondary)
          : Icon(
            // : = unfavourited
          Icons.favorite_border_outlined,
          size: 30,
        )
      )
    )
  )
}

```

```
        color:  
Theme.of(context).colorScheme.onBackground)))),  
  
        Align(  
  
            alignment: Alignment.bottomLeft,  
  
            child: Padding(  
  
                padding: const  
EdgeInsets.only(top: 190, left: 20, right: 20),  
  
                child: Column(  
  
                    children: [  
  
                    Row(  
  
                        mainAxisAlignment:  
MainAxisAlignment.spaceBetween,  
  
                        children: [  
  
                        Flexible(  
  
                            child: Container(  
  
                                padding: const  
EdgeInsets.only(right: 13.0),  
  
                                child: Text(  
  
restaurants[index][1],  
  
                                overflow:  
TextOverflow.ellipsis,  
  
                                style:  
Theme.of(context).textTheme.headline4,  
  
                                ))),
```

```
Row(  
  children: [  
    Icon(  
      Icons.star,  
      size: 20,  
      color:  
        Theme.of(context).primaryColor,  
    ),  
    const SizedBox(  
      width: 5,  
    ),  
    Text(  
      "N/A",  
      style:  
        Theme.of(context).textTheme.bodyText1,  
    )  
  ],  
),  
],  
(  
  const SizedBox(height: 10),  
  Row(  
    children: [  

```

```

Text("${(restaurantsdata[0]["restaurants"][index][8]).toStringAsFixed(1)} km
away",
      style:
Theme.of(context).textTheme.bodyText1!.copyWith(fontSize: 12)),


const SizedBox(
width: 5,
),


Text(
".",
style:
Theme.of(context).textTheme.bodyText1!.copyWith(fontSize: 12),
),


const SizedBox(
width: 5,
),


Text(


(double.parseDouble(restaurantsdata[0]["restaurants"][index][7]) == 0)
? "Free
Delivery"
:
"${(restaurantsdata[0]["restaurants"][index][7])} Delivery",
style:
Theme.of(context).textTheme.bodyText1!.copyWith(fontSize: 12)),
],

```

```
        ),  
  
        const SizedBox(height: 20),  
  
    ],  
  
    ))))  
  
],  
  
))));  
  
}  
  
});  
  
});  
  
} catch (e) {  
  
    return Padding(  
  
        padding: const EdgeInsets.only(left: 20, right: 20, top: 10,  
bottom: 10),  
  
        child: Container(  
  
            decoration: BoxDecoration(  
  
                borderRadius: const  
BorderRadius.all(Radius.circular(20)),  
  
                color: Theme.of(context).colorScheme.onSurface,  
  
                boxShadow: [  
  
                    BoxShadow(  

```

```
        color: Colors.black.withOpacity(0.1),  
  
        spreadRadius: 2,  
  
        blurRadius: 10,  
  
        offset: const Offset(0, 10), // changes position of  
shadow  
,  
],  
  
child: Column(children: [  
  
    Padding(  
  
        padding: const EdgeInsets.all(30),  
  
        child: SizedBox(  
  
            width: double.infinity,  
  
            child: Center(  
  
                child: Column(children: [  
  
                    const Text(  
  
                        "An unexpected error occurred. \nPlease try  
again",  
  
                    textAlign: TextAlign.center,  
  
                ),  
  
                const SizedBox(  
  
                    height: 20,  
  
                ),  
  
                ElevatedButton(  
)
```

```
        onPressed: () {
            setState(() {
                getRestaurants();
            });
        },
        child: const Text("Retry"))
    ])));
)
]));


}
}

} else {
    return const LinearProgressIndicator(color: Color(0xff4100C4),
backgroundColor: Color.fromARGB(0, 235, 224, 255));
}
},
);
}
}
```

topbar.dart

```
import 'package:alleat/screens/checkout/cart.dart';
```

```
import 'package:alleat/screens/locationselection.dart';

import 'package:flutter/material.dart';

import 'package:shared_preferences/shared_preferences.dart';



class MainAppBar extends StatelessWidget implements PreferredSizeWidget {

    final double height;

    const MainAppBar({Key? key, required this.height}) : super(key: key);





    Future<String> getSavedLocation() async {

        final prefs = await SharedPreferences.getInstance();

        final List<String>? savedLocationText =

            prefs.getStringList('locationPlacemark');

        if (savedLocationText != null) {

            return savedLocationText[1];

        } else {

            return ("No Location Set");

        }

    }







    @override

    Widget build(BuildContext context) {

        return Container(
```

```
//Container of height 120px

height: 120,

color: Theme.of(context).bottomNavigationBarTheme.backgroundColor,

child: SafeArea(
    child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
            InkWell(
                onTap: () {
                    Navigator.push(
                        context,
                        MaterialPageRoute(
                            builder: (context) => const SelectLocation()));
                },
            ),
            Padding(
                padding: const EdgeInsets.all(10),
                child: Icon(
                    //Location icon
                    Icons.location_on_outlined,
                    color: Theme.of(context).textTheme.headline1?.color,
                ),
            ),
        ],
    ),
),
```

```
        ))),  
  
    Column(mainAxisAlignment: MainAxisAlignment.center, children: [  
  
        //Column that will display the destination the food will be  
delivered to  
  
        Text(  
  
            "Location",  
  
            style: Theme.of(context)  
  
                .textTheme  
  
                .headline6  
  
                ?.copyWith(fontWeight: FontWeight.w500),  
  
(  
        ),  
  
        const SizedBox(  
  
            height: 2,  
  
(  
        ),  
  
        FutureBuilder<String>(  
  
            future: getSavedLocation(),  
  
            builder: (context, snapshot) {  
  
                if (!snapshot.hasData) {  
  
                    return const Text("Location loading");  
  
                }  
  
                if (snapshot.hasData) {  
  
                    var location = snapshot.data ?? [];  
  
                    return Text(location.toString());  
                }  
            },  
        ),  
    ]  
);
```

```
        } else {

            return const Text("No Location Set");

        }

    })

]),

InkWell(child: Padding(padding: const EdgeInsets.all(10),child:

Icon(

    //Cart icon

    Icons.shopping_cart_outlined,

    color: Theme.of(context).textTheme.headline1?.color,

)),,

onTap: () {

    Navigator.push(
        //go to cart page

        context,
        MaterialPageRoute(
            builder: (context) => const Cart()));
    }

}),
]),

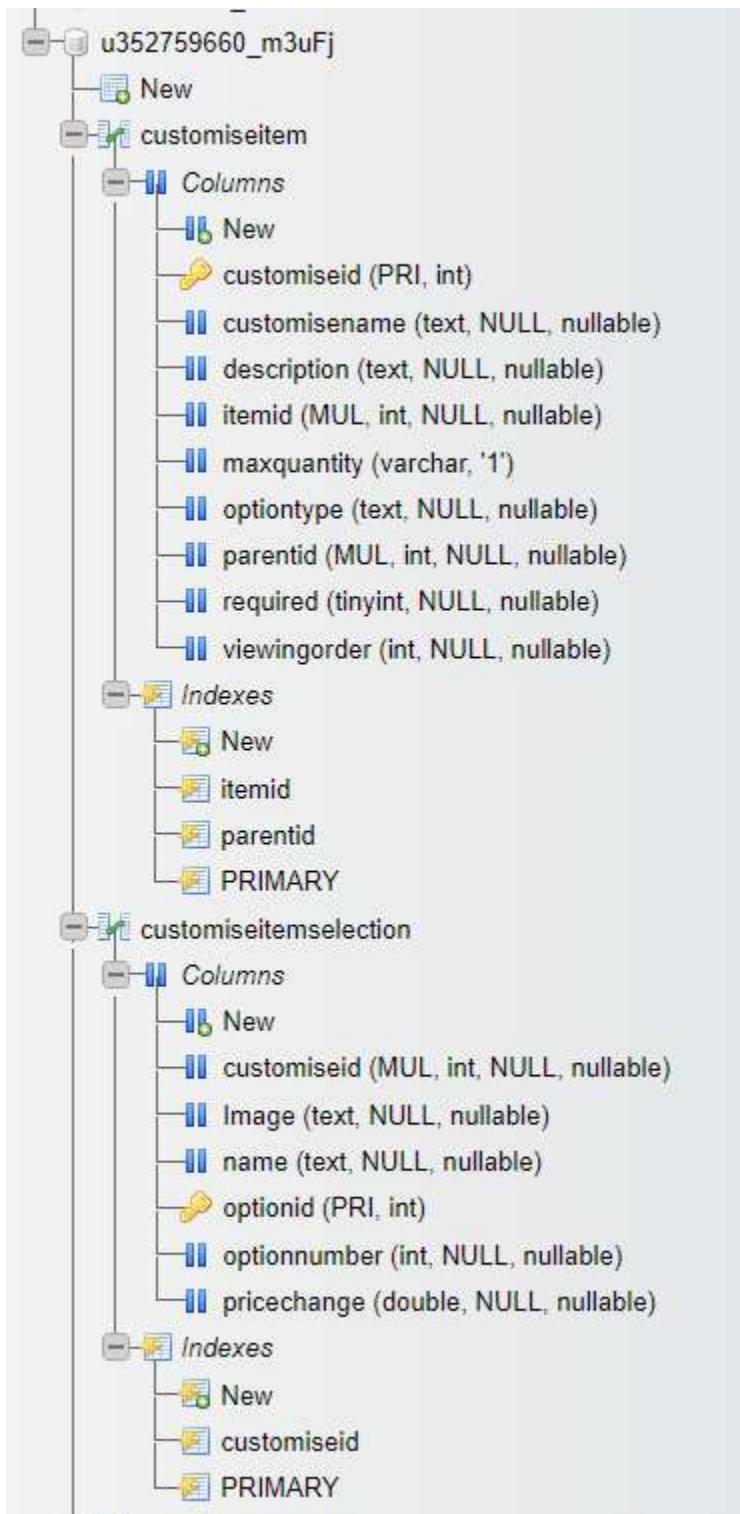
));
}

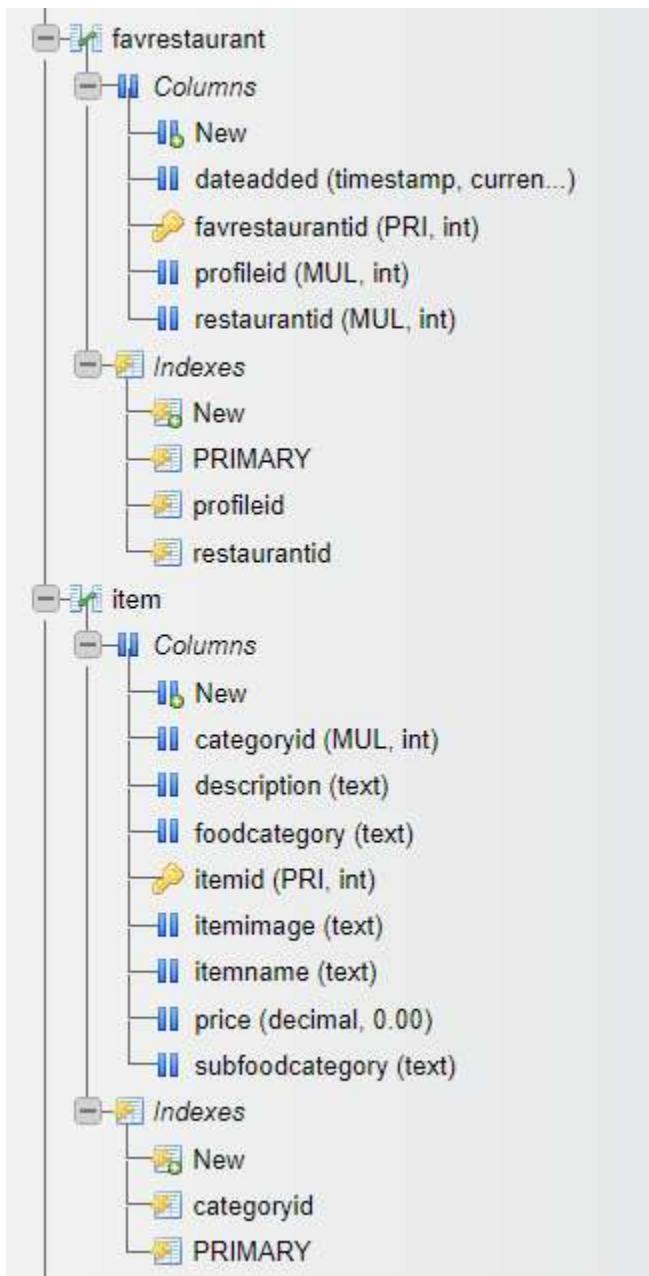
}
```

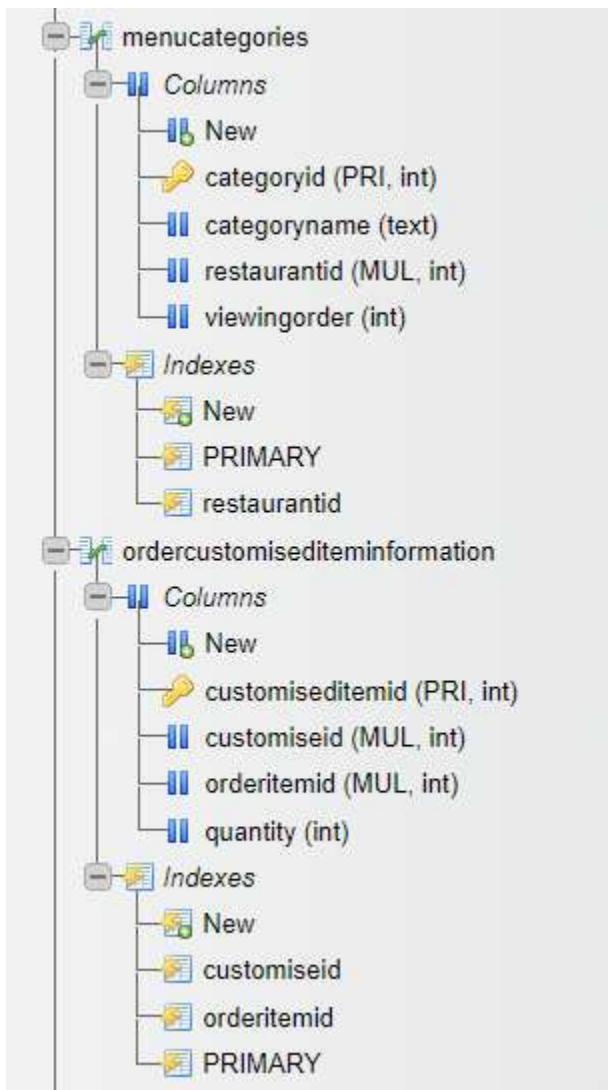
```
@override  
Size get preferredSize => Size.fromHeight(height);  
}
```

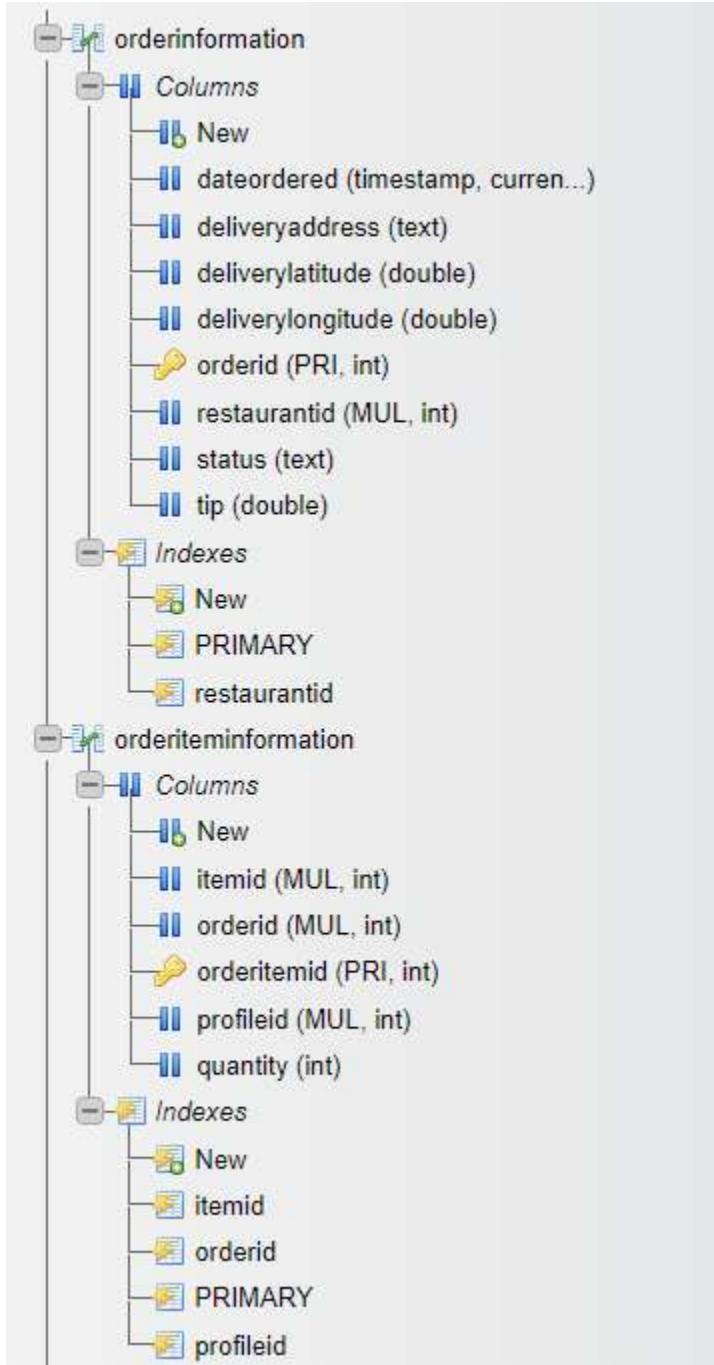
Server Database

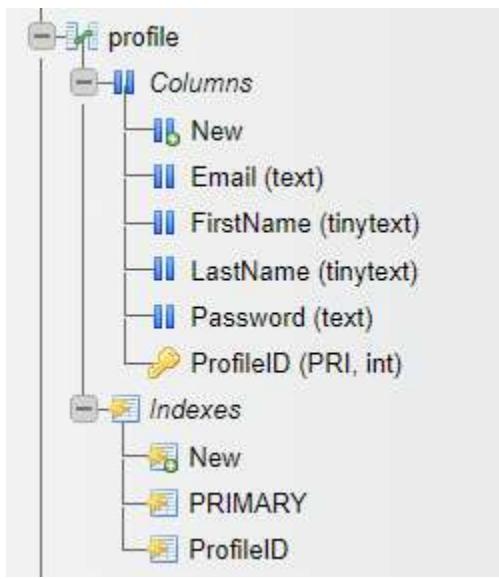
Structure



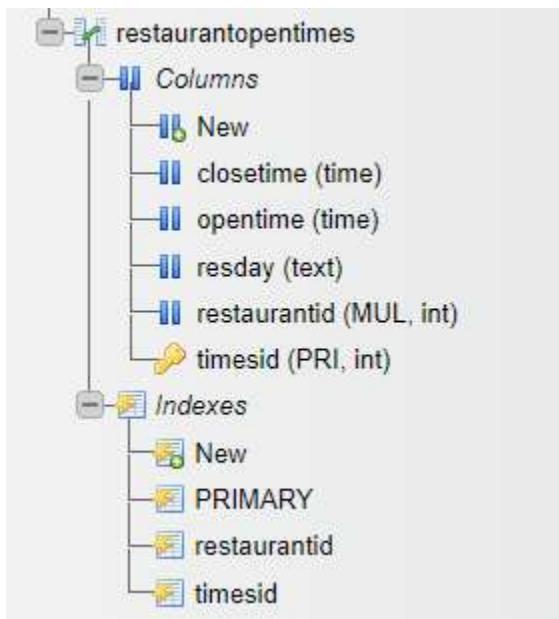












Server File Directory

Structure

Assets > Images > restaurant_banner				
Name	Size	Last modified		
 dojasianfusionbanner.jpg	183.35 KB	5 months ago	-rw-rw-r--	x
 papajohnsbanner.jpg	238.94 KB	5 months ago	-rw-rw-r--	x
 pizzaexpressbanner.jpg	272.39 KB	5 months ago	-rw-rw-r--	x
 riversidefishandchipsbanner.jpeg	164.12 KB	5 months ago	-rw-rw-r--	x
 starbucksbanner.jpg	248.01 KB	5 months ago	-rw-rw-r--	x
 thefirstbuuddhabanner.jpg	309 KB	5 months ago	-rw-rw-r--	x
 themeatingroombanner.jpg	302.55 KB	5 months ago	-rw-rw-r--	x
 zerosushibanner.jpg	188.37 KB	5 months ago	-rw-rw-r--	x

Assets > Images > restaurant_item				
Name	Size	Last modified		
 7ufree.webp	11.32 KB	5 months ago	-rw-rw-r--	x
 allthemerts.webp	60.34 KB	5 months ago	-rw-rw-r--	x
 american.webp	63.52 KB	5 months ago	-rw-rw-r--	x
 americanhot.webp	33.65 KB	5 months ago	-rw-rw-r--	x
 bbqchickenandbaconpapadias.webp	20.61 KB	5 months ago	-rw-rw-r--	x
 bbqchickenclassic.jpg	57.0 KB	5 months ago	-rw-rw-r--	x
 bbqclip.webp	8.00 KB	5 months ago	-rw-rw-r--	x
 bbqmeatfeast.webp	34.64 KB	5 months ago	-rw-rw-r--	x
 bolognese.webp	37.05 KB	5 months ago	-rw-rw-r--	x
 calzonenduja.webp	53.82 KB	5 months ago	-rw-rw-r--	x
 calzoneverdure.webp	48.51 KB	5 months ago	-rw-rw-r--	x
 cheeseardtomato.jpg	50.45 KB	5 months ago	-rw-rw-r--	x
 cheesegarlicsticks.webp	39.76 KB	5 months ago	-rw-rw-r--	x
 chickenpoppers.webp	21.5 KB	5 months ago	-rw-rw-r--	x
 chocolatefudgebrownie.webp	19.33 KB	5 months ago	-rw-rw-r--	x
 cookiedough.webp	19.19 KB	5 months ago	-rw-rw-r--	x
 cookiesorcookiедoughndairy.webp	28.53 KB	5 months ago	-rw-rw-r--	x
 familysharer.webp	31.22 KB	5 months ago	-rw-rw-r--	x
 garlicandherbdip.webp	7.43 KB	5 months ago	-rw-rw-r--	x
 garlickid.webp	8.57 KB	5 months ago	-rw-rw-r--	x

Name ↑	Size	Last modified	
dojiaasianfusionlogo.png	41.3 KB	5 months ago	-rw-r--r--X
papajohnslogo.png	160.55 KB	5 months ago	-rw-r--r--X
pizzaexpresslogo.png	376.2 KB	5 months ago	-rw-r--r--X
riversidefishandchipslogo.jpg	49.94 KB	9 months ago	-rw-r--r--X
starbuckslogo.png	552.05 KB	5 months ago	-rw-r--r--X
thefatburgerlogo.png	38.42 KB	5 months ago	-rw-r--r--X
themeatinggroomlogo.png	253.55 KB	5 months ago	-rw-r--r--X
zerosushilogo.png	22.01 KB	5 months ago	-rw-r--r--X

Name ↑	Size	Last modified	
cartiteminfo.php	3.24 KB	12 days ago	-rwxr--r--X
favouriterestaurant.php	2.38 KB	5 months ago	-rwxr--r--X
favouriterestaurantdata.php	2.49 KB	5 months ago	-rwxr--r--X
favouriterestaurantlist.php	1.81 KB	4 months ago	-rwxr--r--X
itemcustomise.php	1.7 KB	a month ago	-rwxr--r--X
login.php	2.97 KB	2 days ago	-rwxr--r--X
orders.php	6.08 KB	2 days ago	-rwxr--r--X
register.php	4.96 KB	8 hours ago	-rwxr--r--X
restaurantlist.php	3.38 KB	2 months ago	-rwxr--r--X
restaurantmenucategories.php	1.25 KB	5 months ago	-rwxr--r--X
restaurantmenuitems.php	1.36 KB	5 months ago	-rwxr--r--X

Files

cartiteminfo.php

```
<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host

$connection = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

if ($connection) {
    echo "Connected successfully";
} else {
    die("Connection failed: " . mysqli_connect_error());
}

$return["error"] = false;

$return["message"] = "";


```

```

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["type"]) && ($_POST["term"]); //validate if the fields have
been sent from the client

if($val){

    $type = $_POST["type"];

    $term = $_POST["term"];

    if ($type == "item"){ //If the query type is item

        $term = intval($term);

        $sqlitem = "SELECT item.itemid, item.itemname, item.price,
item.itemimage, restaurant.restaurantid, restaurant.restaurantname,
restaurant.delivery, restaurant.ordermin, restaurant.address,
restaurant.latitude, restaurant.longitude FROM item AS item INNER JOIN
menucategories AS menucategories ON item.categoryid = menucategories.categoryid
INNER JOIN restaurant AS restaurant ON menucategories.restaurantid =
restaurant.restaurantid WHERE item.itemid = $term"; //Select item information and
restaurant information by linking using the item to the category id and
restaurant id from the category

        if($result = mysqli_query($link, $sqlitem)){ //If it successfully queries
the database

            $iteminfo = []; //array to be sent back to the client

            while($row = mysqli_fetch_assoc($result)) {

                array_push($iteminfo, $row["itemid"], $row["itemname"],
$row["price"], $row["itemimage"], $row["restaurantid"], $row["restaurantname"],
$row["delivery"], $row["ordermin"], $row["address"], $row["latitude"],
$row["longitude"]); //Add the item information to array to be sent back

            }
        }
    }
}

```

```

$return["message"] = $iteminfo;

} else{

    $return["error"] = true;

    $return["message"] = "Failed item query";

}

}elseif ($type == "title"){ //If the query type is customise title

    $term = json_decode($term); //decode the array to be a list of
customisetitle ids

    $sqltitle = "SELECT customiseid, customisename, optiontype FROM
customiseitem WHERE customiseid IN (".implode(', ', $term).")"; //select
customise title information including id, title and type (add, subtract or
select). Query for only the customise titles under $term

if($result = mysqli_query($link, $sqltitle)){ //If it successfully
queries database

    $titleinfo = []; //Store information about each title to be sent back to
the client

    while($row = mysqli_fetch_assoc($result)) { //For each title returned

        array_push($titleinfo, [$row["customiseid"],
$row["customisename"], $row["optiontype"]]); //Add title information to the array

    }

    $return["message"] = $titleinfo;

} else {

    $return["error"] = true;
}

```

```

$return["message"] = "Failed to query customised item";

}

} elseif ($type == "option"){ //If the query type is customise options

    $term = json_decode($term); //Decode to be an array including the
customise option ids to find information for

    $sqloption = "SELECT optionid, name, pricechange FROM
customiseitemselection WHERE optionid IN (".implode(', ', $term).")"; //Select
option id, option name and price change for each customise id in $term

if($result = mysqli_query($link, $sqloption)){ //If the database
successfully queries

    $optioninfo = []; //Store option information to be sent back to client

    while($row = mysqli_fetch_assoc($result)) { //For each result

        array_push($optioninfo, [$row["optionid"], $row["name"],
$row["pricechange"]]);

    } //Add information to $optioninfo as a 2D array

    $return["message"] = $optioninfo;

} else {

    $return["error"] = true;

    $return["message"] = "Failed to query customised item";

}

} else {

    $return["error"] = true;

```

```

$return["message"] = "Query type unknown";

}

}

else{$return["error"] = true;

        $return["message"] = "data not specified";}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>

```

favouriterestaurant.php

```

<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host

```

```

$return["error"] = false;

$return["message"] = "";

$return["success"] = false;

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["action"]) && ($_POST["profileemail"]) &&
isset($_POST["restaurantid"]);

if($val){

    $email = $_POST["profileemail"]; //grabing the data from headers

    $restaurantid = $_POST["restaurantid"];

    $action = $_POST["action"];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";

    if($result = mysqli_query($link, $sql)) {

        if(mysqli_num_rows($result) == 1){

            while($row = mysqli_fetch_array($result)){

                $ProfileID = $row['ProfileID'];

            }

            // Free result set

            mysqli_free_result($result);

            if ($action == "unfavourite"){


```

```

    $sqlunfavourite = "DELETE FROM favrestaurant WHERE profileid =
$ProfileID AND restaurantid = $restaurantid";

    if ($link->query($sqlunfavourite) === TRUE) {

        $return["success"] = true;

    } else{

        $return["error"] = true;

    }

}

else if ($action == "favourite"){

    $sqlfavourite = "INSERT INTO favrestaurant (profileid,
restaurantid) VALUES ($ProfileID, $restaurantid)";

    if ($link->query($sqlfavourite) == TRUE) {

        $return["success"] = true;

    } else{

        $return["error"] = true;

    }

}

else{

    $return["error"] = true;

$return["message"] = "action unspecified";

```

```

        }

    }

else{

    $return["error"] = true;

    $return["message"] = "duplicate found";

}

}

else{

    $return["error"] = true;

    $return["message"] = "failed";

}

}

else{$return["error"] = true;

    $return["message"] = "data not specified";}

mysql_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

```

```
?>
```

favouriterestaurantdata.php

```
<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host


$return["error"] = false;

$return["message"] = "";

$return["restaurants"] = array();

$return["temp"] = array();

$restaurantids = array();


$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileemail"]);

if($val){

    $query = "SELECT * FROM users WHERE email = '$val'";

    $result = mysqli_query($link, $query);

    if(mysqli_num_rows($result) > 0){
```

```

$email = $_POST["profileemail"];

$sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";

if($result = mysqli_query($link, $sql)){
    if(mysqli_num_rows($result) == 1){

        while($row = mysqli_fetch_array($result)){
            $ProfileID = $row['ProfileID'];
        }

        // Free result set
        mysqli_free_result($result);
    }
}

else{
    $return["error"] = true;
    $return["message"] = "duplicate found";
}

$sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
profileid = $ProfileID";

if($result2 = mysqli_query($link, $sqlfavrestaurant)){
    while($row2 = mysqli_fetch_assoc($result2)) {
        array_push($restaurantids, $row2["restaurantid"]);
    }
}

mysqli_free_result($result2);

```

```

}

foreach ($restaurantids as &$value) {

    $sqlrestaurantinfo = "SELECT res.restaurantid, res.restaurantname,
res.restaurantlogo, res.restaurantbanner FROM restaurant res WHERE
res.restaurantid = $value";

    $return["temp"] = $sqlrestaurantinfo;

    if($result3 = mysqli_query($link, $sqlrestaurantinfo)){
        while($row3 = mysqli_fetch_assoc($result3)) {
            array_push($return["restaurants"], [$row3["restaurantid"],
$row3["restaurantname"], $row3["restaurantlogo"], $row3["restaurantbanner"]]);
        }
        mysqli_free_result($result3);
    }
}

else{
    $return["error"] = true;
    $return["message"] = "failed";
}
}

}

```

```
else{

    $return["error"] = true;

    $return["message"] = "data not specified 2";

}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>
```

favouriterestaurantlist.php

```
<?php

$db = "u352759660_m3uFj"; //database name
```

```

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host


$return["error"] = false;

$return["message"] = "";

$return["restaurantids"] = array();

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["profileemail"]);

if($val){

    $email = $_POST["profileemail"];

    $sql = "SELECT ProfileID FROM profile WHERE Email = '$email' LIMIT 1";

    if($result = mysqli_query($link, $sql)) {

        if(mysqli_num_rows($result) == 1){

            while($row = mysqli_fetch_array($result)) {

                $ProfileID = $row['ProfileID'];

            }

            // Free result set
    }
}

```

```
mysqli_free_result($result);

}

else{

$return["error"] = true;

$return["message"] = "duplicate found";

}

}

else{

$return["error"] = true;

$return["message"] = "failed";

}

$sqlfavrestaurant = "SELECT restaurantid FROM favrestaurant WHERE
profileid = $ProfileID";

if($result2 = mysqli_query($link, $sqlfavrestaurant)){
while($row2 = mysqli_fetch_assoc($result2)) {

array_push($return["restaurantids"], $row2["restaurantid"]);

}

mysqli_free_result($result2);
```

```

        }

    }

else{

    $return["error"] = true;

    $return["message"] = "data not specified";

}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>

```

itemcustomise.php

```

<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

```

```
$dbpassword = "FO7&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host

$return["error"] = false;

$return["message"] = "";

$return["customiseitem"] = [];

$return["test"] = [];

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["itemid"]);

if($val){

    $itemid = $_POST["itemid"];

    $sqlcustomisetitles = "SELECT * FROM customiseitem WHERE itemid = $itemid
ORDER BY viewingorder ASC";

    if($result = mysqli_query($link, $sqlcustomisetitles)){

        while($row = mysqli_fetch_assoc($result)) {

            array_push($return["customiseitem"], [$row["customiseid"],
$row["customisename"], $row["description"], $row["optiontype"], $row["required"],
$row["parentid"], $row["maxquantity"], []]);
        }
    }
}
```

```

    }

    mysqli_free_result($result);

}

for ($i = 0; $i < count($return["customiseitem"]); $i++){

    $sqlcustomiseoptions = "SELECT * FROM customiseitemselection WHERE
customiseid = {$return['customiseitem'][$i][0]}";

    if($result2 = mysqli_query($link, $sqlcustomiseoptions)) {

        while($row2 = mysqli_fetch_assoc($result2)) {

            array_push($return["customiseitem"][$i][7], [$row2["optionid"],
$row2["name"], $row2["image"], $row2["pricechange"]]);

        }
    }
}

else{

    $return["error"] = true;

    $return["message"] = "Unknown itemid";

}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

```

```
// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>
```

login.php

```
<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host


$return["error"] = false;

$return["message"] = "";

$return["exists"] = false;

$return["profile"] = "";


$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

//connecting to database server


$val = isset($_POST["email"]) && isset($_POST["password"]);
```

```

if($val){

    //checking if there is POST data

    $email = mysqli_real_escape_string($link, $_POST["email"]);

    $password = mysqli_real_escape_string($link, $_POST["password"]);

    $secret_iv = '5fd0b31f582beb1f';

    $secret_key = 'e16a3f356b2366c1';

    $cipher = "AES-128-CBC";

    $decryptedString = openssl_decrypt ($password, $cipher, $secret_key, 0,
$secret_iv);

    $sql = "SELECT ProfileID, FirstName, LastName, Email, Password FROM
profile WHERE Email = '$email'";

    if($result = mysqli_query($link, $sql)) {

        if(mysqli_num_rows($result) == 1){

            while($row = mysqli_fetch_array($result)){

                $ProfileIDR = $row['ProfileID'];

                $FirstNameR = $row['FirstName'];

                $LastNameR = $row['LastName'];

                $EmailR = $row['Email'];

                $PasswordR = $row['Password'];

            }

            // Free result set
        }
    }
}

```

```

        mysqli_free_result($result);

        $ispasswordsame = password_verify($decryptedString,
>PasswordR);

        if ($ispasswordsame == true){

            $PasswordREncrypted = openssl_encrypt($PasswordR,
$cipher, $secret_key, 0, $secret_iv);

            $return["exists"] = true;

            $return["profile"] = array($ProfileIDR, $FirstNameR,
$LastNameR, $EmailR, $PasswordREncrypted);

        }

        else{

            $return["error"] = false;

            $return["message"] = "Email or password incorrect";

            $return["exists"] = false;

            $return["profile"] = null;

        }

    } else{

        $return["error"] = false;

        $return["message"] = "No matching profiles";

        $return["exists"] = false;

        $return["profile"] = null;

    }

} else{

    $return["error"] = true;

```

```

        $return["message"] = "Failed profile search";

    }

}else{

$return["error"] = true;

$return["message"] = 'Send all parameters.';

}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>

```

orders.php

```

<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

```

```

$dbhost = "localhost"; //database host

$return["error"] = false;

$return["message"] = "";

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["type"]);

if($val){

    if ($_POST["type"] == "add"){ //If the order query type is add

        $addval = isset($_POST["tip"]) && isset($_POST["restaurantid"]) &&
        isset($_POST["address"]) && isset($_POST["latitude"]) &&
        isset($_POST["longitude"]) && isset($_POST["iteminfo"]) &&
        isset($_POST["customiseinfo"]); //Check for if it contains the correct variables

        if($addval){ //If it contains the correct information

            $tip = round((floatval(mysqli_real_escape_string($link,
$_POST["tip"]))), 2); //Set the tip to 2dp

            $restaurantid = intval(mysqli_real_escape_string($link,
$_POST["restaurantid"])); //Set restaurant id to an integer

            $address = mysqli_real_escape_string($link, $_POST["address"]);

            $latitude = floatval(mysqli_real_escape_string($link,
$_POST["latitude"]));

            $longitude = floatval(mysqli_real_escape_string($link,
$_POST["longitude"]));

```

```

$iteminfo = json_decode(mysqli_real_escape_string($link,
$_POST["iteminfo"]), true); //Decode the json string into a php array

$customiseinfo = json_decode(mysqli_real_escape_string($link,
$_POST["customiseinfo"]), true); //Decode the json string into a php array

$sqlorderinfo = "INSERT INTO orderinformation SET restaurantid =
'$restaurantid', tip = '$tip', deliveryaddress = '$address', deliverylatitude =
'$latitude', deliverylongitude = '$longitude', status = 'confirming'"; //Create
the SQL insert query to add the order information to the orderinformation table

if($resorderinfo = mysqli_query($link, $sqlorderinfo)){ //If the
insert query was successful get the orderid (primary key) using the most recent
order id that fits under the same latitude and longitude inputted

    $sqlorderid = "SELECT orderid FROM orderinformation WHERE
deliverylatitude = '$latitude' AND deliverylongitude = '$longitude' ORDER BY
orderid DESC LIMIT 1";

    if($orderidinfo = mysqli_query($link, $sqlorderid)){

        $orderidrecord = mysqli_fetch_assoc($orderidinfo); //Get the
first result and save the orderid

        $orderid = intval($orderidrecord["orderid"]);

        for ($i = 0; $i < count($_POST["iteminfo"]); $i++){ //For
each item in the list of items

            $profileid = intval($iteminfo[$i][0]); //Set profile id
to an integer

            $itemid = intval($iteminfo[$i][1]); //Set item id to be
an integer

            $itemquantity = intval($iteminfo[$i][2]); //Set item
quantity to be an integer

```

```

$sqlorderiteminfo = "INSERT INTO orderiteminformation
SET orderid = '$orderid', profileid = '$profileid', itemid = '$itemid', quantity
= '$itemquantity'"; //Create an SQL insert query to add the item to the
orderiteminformation info using orderid, profileid and itemid as foreign keys

if($orderiteminfo = mysqli_query($link,
$sqlorderiteminfo)){ //If is successfully inserted the item information into the
table for the order

    $sqlorderitemid = "SELECT orderitemid FROM
orderiteminformation WHERE profileid = '$profileid' AND itemid = '$itemid' ORDER
BY orderitemid DESC LIMIT 1"; //Get the orderitemid (primary key) from
orderiteminformation

    if($orderitemidinfo = mysqli_query($link,
$sqlorderitemid)){ //If the orderitemid is queried successfully

        $orderitemidrecord =
mysqli_fetch_assoc($orderitemidinfo); //Get the first record

        $orderitemid =
intval($orderitemidrecord["orderitemid"]); // Store it under orderitemid as an
integer

        for ($j = 0; $j < count($customiseinfo); $j++){
//For each item customise option

            if ($customiseinfo[$j][0] == $i){ //If the
profile is equal to the index of the item, insert it into the table using the sql
insert query

                $customiseid =
intval($customiseinfo[$j][1]);

                $quantity =
intval($customiseinfo[$j][2]);

```

```

$sqlorderitemoptioninfo = "INSERT INTO
ordercustomisediteminformation SET orderitemid = '$orderitemid', customiseid =
'$customiseid', quantity = '$quantity'";

if($sqlorderitemoptioninfo =
mysqli_query($link, $sqlorderitemoptioninfo)){
    $return["message"] = "success";
} else{
    $return["error"] = true;
    $return["message"] = "failed to save
customised item";
}

}

}

}

else {
    $return["error"] = true;
    $return["message"] = "failed to verify item
info";
}

}

}

else{
    $return["error"] = true;
    $return["message"] = "failed to save item info";
}

```

```

        }

    }else{

        $return[ "error" ] = true;

        $return[ "message" ] = "failed to verify order info";

    }

}else{

    $return[ "error" ] = true;

    $return[ "message" ] = "failed to save order";

}

}

} else{ //If the data of adding an order doesnt have the correct data
return error

    $return[ "error" ] = true;

    $return[ "message" ] = "data not specified";}

}

} else{ //If the order query type is not the specified type, return error

    $return[ "error" ] = true;

    $return[ "message" ] = "order query type unknown";

}

}

```

```
else{$return["error"] = true;

        $return["message"] = "data not specified";}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>
```

register.php

```
<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host

$return["error"] = false;

$return["message"] = "";

$return["hash"] = "";

$return["profile"] = "";
```

```

$return["exist"] = false;

$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);
//connecting to database server

$val = isset($_POST["firstname"]) && isset($_POST["lastname"])
      && isset($_POST["email"]) && isset($_POST["password"]);

if($val){
    //checking if there is POST data

    $firstname = $_POST["firstname"]; //grabing the data from headers
    $lastname = $_POST["lastname"];
    $email = $_POST["email"];
    $password = $_POST["password"];

    $secret_iv = '5fd0b31f582beb1f';
    $secret_key = 'e16a3f356b2366c1';
    $cipher = "AES-128-CBC";

    $decryptedString = openssl_decrypt ($password, $cipher, $secret_key, 0,
$secret_iv);

    $hashpassword = password_hash($decryptedString, PASSWORD_DEFAULT);

    $base64hash = base64_encode($hashpassword);
}

```

```

$encryptedhashedpassword = openssl_encrypt ($base64hash, $cipher,
$secret_key, 0, $secret_iv);

$return["hash"] = $encryptedhashedpassword; //Encrypt hashed password to
be sent back

if($return["error"] == false){

    $firstname = mysqli_real_escape_string($link, $firstname);

    $lastname = mysqli_real_escape_string($link, $lastname);

    $email = mysqli_real_escape_string($link, $email);

    $hashpassword = mysqli_real_escape_string($link, $hashpassword);

    //escape inverted comma query conflict from string

    $sqlverify = "SELECT email FROM profile WHERE Email = '$email'";

    $res = mysqli_query($link, $sqlverify);

    if (mysqli_num_rows($result) == 0) {

        $sql = "INSERT INTO profile SET

                FirstName = '$firstname',

                LastName = '$lastname',

                Email = '$email',

                Password = '$hashpassword'";

        $res = mysqli_query($link, $sql);

    }

}

$sqlget = "SELECT ProfileID, FirstName, LastName, Email, Password
FROM profile WHERE Email = '$email'";

```

```

if($result = mysqli_query($link, $sqlget)){
    if(mysqli_num_rows($result) == 1){
        while($row = mysqli_fetch_array($result)){
            $ProfileIDR = $row['ProfileID'];
            $FirstNameR = $row['FirstName'];
            $LastNameR = $row['LastName'];
            $EmailR = $row['Email'];
            $PasswordR = $row['Password'];
        }
        // Free result set
        mysqli_free_result($result);
        $ispasswordsame = password_verify($decryptedString,
$PasswordR);

        if ($ispasswordsame == true){

            $PasswordREncrypted = openssl_encrypt($PasswordR,
$cipher, $secret_key, 0, $secret_iv);

            $return["exists"] = true;

            $return["profile"] = array($ProfileIDR, $FirstNameR,
$LastNameR, $EmailR, $PasswordREncrypted);

        }
        else{
            $return["error"] = true;

            $return["message"] = "Email or password incorrect";

            $return["exists"] = false;
        }
    }
}

```

```

        $return["profile"] = null;

    }

} else if (mysqli_num_rows($result) > 1){

    $return["error"] = true;

    $return["message"] = "Multiple profiles found under profile.
Email unavailable";

    $return["exists"] = false;

    $return["profile"] = null;

} else{

    $return["error"] = true;

    $return["message"] = "Profile search failed on server. Try
another email./";

    $return["exists"] = false;

    $return["profile"] = null;

}

} else{

    $return["error"] = true;

    $return["message"] = "Failed profile search";

}

if($res){

    //write success

```

```
        }else{

            $return["error"] = true;

            $return["message"] = "Database error";

        }

    }

    else{

        $return["exist"] = true;

    }

}

}

}

else{

    $return["error"] = true;

    $return["message"] = 'Send all parameters.';

}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>
```

restaurantmenucategories.php

```
<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host


$return["error"] = false;

$return["message"] = "";

$return["restaurantcategories"] = array();


$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["restaurantid"]);

if($val){

$resid = $_POST["restaurantid"];



$sql = "SELECT categoryname, categoryid FROM menucategories WHERE
restaurantid = '$resid' ORDER BY viewingorder ASC";

if($result = mysqli_query($link, $sql)){


```

```

        while($row = mysqli_fetch_assoc($result)) {

            array_push($return["restaurantcategories"], [$row["categoryname"],
$row["categoryid"]]);

        }

        mysqli_free_result($result);

    }

    else{

        $return["error"] = true;

        $return["message"] = "restaurant not sepcified";

    }

}

else{

    $return["error"] = true;

    $return["message"] = "data not specified";

}

mysqli_close($link); //close mysqli

header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

```

```
?>
```

restaurantmenuitems.php

```
<?php

$db = "u352759660_m3uFj"; //database name

$dbuser = "u352759660_GDfIr"; //database username

$dbpassword = "F07&Ruvr;0G"; //database password

$dbhost = "localhost"; //database host


$return["error"] = false;

$return["message"] = "";

$return["menuitems"] = array();


$link = mysqli_connect($dbhost, $dbuser, $dbpassword, $db);

$val = isset($_POST["categoryid"]);

if($val){

    $rescategory = $_POST["categoryid"];
```

```

$sql = "SELECT itemid, foodcategory, subfoodcategory, itemname, description,
price, itemimage FROM item WHERE categoryid = '$rescategory' ORDER BY itemid
ASC";

if($result = mysqli_query($link, $sql)){
    while($row = mysqli_fetch_assoc($result)) {
        array_push($return["menuitems"], [$row["itemid"],
$row["foodcategory"], $row["subfoodcategory"], $row["itemname"],
$row["description"], $row["price"], $row["itemimage"]]);
    }
    mysqli_free_result($result);
}

else{
    $return["error"] = true;
    $return["message"] = "category not sepcified";
}

}

else{
    $return["error"] = true;
    $return["message"] = "data not specified";
}

mysqli_close($link); //close mysqli

```

```
header('Content-Type: application/json');

// tell browser that its a json data

echo json_encode($return);

//converting array to JSON string

?>
```