**Design Consideration**

**(including discussion of usability, accessibility and design**

**guidelines)**

**Implementation:**

**Brief description and reflection of your code implementation,**

**level of complexity, how the key features are implemented, and any additional**

**tools/APIs used to develop the app.**

**Critique**

**: What worked and what didn’t work and how could the program be**

**Improved.**

**Description of the app:**

The Menu app allows users to run their own takeaway/restaurant from their own kitchens. The app has two types of users; Chefs, and eaters. Chefs sell food made in their kitchen to eaters. Food can be either sold as takeaway or eaters can eat in chefs’ kitchens.

Considoring that the app is designed for the android platform and android is a google product, choosing the google design styles for android made sense, since the design style is tailored for the products in which the app runs.

In terms of colours, the google design styles suggest using:

* A primary colour with dark and light variants
* A secondary colour, also with dark and light variants and which compliments the primary
* And sparing use of surface, background and error colours

The primary colour used for this app is pink, with the secondary colour as blue.

The weak toned colours grey and light grey were chosen as surface and background colours to not distract the user from more important UI elements.

These colours were chosen with colour blindness in mind.

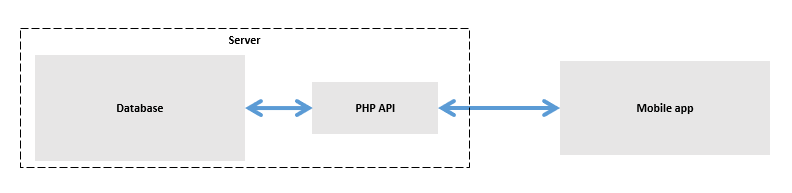
Thanks to Android’s configuration settings allowing users to scale text larger if needed, little consideration was made as to the size of the apps text.

Therefore, the text of the app has been designed to be legible for users without any text size vision impairments.

The fact that text may be scaled has however been accounted for when it comes to the positioning of text in order to maintain a clear appearance.

The app requires a database in order to store and make information available to all users. The chosen platform for this was an SQL server. In order to connect the SQL server with instances of the mobile app a PHP API was created as seen in the diagram.

This API makes the architecture modular, and therefore easy to change. This also makes breaching out to other deployment apps easy, such as IOS as only the front-end needs coding.



The SQL database has these main tables: User, Meal and Order.

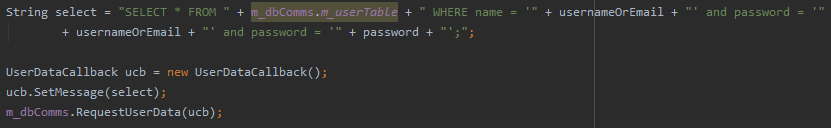
The android java code had classes representing these tables.

To make use of this API within the android app, the DatabaseCommunicator class was created.

The database communicator class had four **public function** operations. **RequestUserData, requestOrderData, requestMealData all for data retrieval and GenericUplaod for uploading data.**

These functions have matching callback classes instances as parameters. They are of the classes UsersCallback, OrdersCallback, MealsCallback all of which inherit from BaseCallback.

When performing a database action, a function of an instance of the database communicator class would be called and passed a callback function.



Once the operation had completed this call back function would be executed. Having call back functions removes the issue of a timely networking operation affecting the main thread, meaning no GUI operations freeze.

Loader page

When the app is booting up a loader page is displayed.

Login page

* On the login page the user can enter in their **credentials**, either a **username/password or email/password combination**.
* In the future, hooking up the login feature with existing social media accounts such as Gmail or Facebook may prove useful
* Nearly all operations involving an internet connection make use of a progress bar in order to notify the user of networking.
* If the incorrect details are entered as demonstrated here, a prompt notifying the user will appear.

Sign up page

* The sign up page allows new users to sign up to the app. Users are required to enter their details.
* At the bottom there are 3 buttons,
  + Address, where we enter in our address
  + Be a chef, this opens up another activity whereby the user can sign up to be a chef on the app. The empty textbox acts as a placeholder for the disclaimer tailored for the governmental requirements of the local area. Should the app be rolled out.

Home page:

* Once logged in we are brought to the home page.
* Here we see maps displaying the user’s location.
* At the top left-hand side of the screen we see three buttons. At the bottom is the location button which resets the display of the user’s location as such.
* At the top is the navigation bar button, which brings up the navigation bar. The navigation bar is used to get around to various locations of the app, these will be explored later in the demonstration. Each option has a custom icon next to it. At the top of the navigation bar we can see our user account.
* By panning around the map the user can see nearby chefs.
* The middle of the three left hand buttons can be used to refresh the locations of chefs.
* Clicking on a chef shows their name and food speciality.
* Clicking on this then brings up a list of all the meals a chef has. Each meal has a name, rating, and weather or not it is on sale, clicking on one of these displays the meal. We will check out this functionality in a second via an alternative route.
* Pressing the search icon on the home page brings up the search page. Here a list of all chefs is displayed, along with an image, their rating and food specialty.
* You can search for a chef using the search bar as such.
* Selecting a chef brings up the page we have seen previously displaying all the meals of a chef.
* Clicking on the meal takes us to the meals page.
* Here we see the name, price, rating, ingredients of the meal. Clicking here (the I) displays the meals allergen information.
* Below this is the address of the chef.
* To order a meal, simply enter the number of meals you would like to order and specify ‘eat in’ or ‘takeaway’. Then press the order button. A prompt requesting if you would like to continue shopping appears. Let’s continue shopping and add one more meal.
* Now let’s checkout our meals.

Basket page

* This brings us to the basket page, where the items ordered are listed. Each item shows a picture, name and price of the meal as well as the options to edit the amount of meals to order, or remove the meal from the basket. The price displayed is the price of an individual meal. Once the items have been checkout however this price is talleyed up.
* The basket page can be accessed from the navigation bar as such.
* Pressing checkout at the bottom of the page navigates the user to the ‘my orders’ page, which is also accessible from the adds the basket items to the users orders.
* Here the user can see a list of their current active orders, which display the tallied up price, the number of portions and the name of the meal orderd. Clicking on an order brings us to the meetup and chat page.
* Here we see the name of the chef were meeting along with their address. A chat is available to discuss delivery times and progress depending on who’s meeting who.
* Here’s an example chat. And response.
* Once the meal is finished, the user can press the done button.
* A rating bar comes up for the user to rate their experience.
* After pressing done, a final prompt to finalize the order pops up.
* Pressing yes finalizes the order, and brings us back to the home page.

Home page 🡪 My meals

* From here, chefs can use the navigation bar to navigate to the my meals page.
* This page displays all a chefs meals. Selecting the new button opens a page to add a new meal. A chef simply adds their information into the page. Here we can select mealtimes, and the option to set a picture for the meal is available. This would be added to the database once the add button is pressed.
* Selecting a meal pops up a page to edit the meal.
* Lets change the time when this meal is available. And also lets change the allergen information of the meal. Pressing the update button sends us back to the ‘my meals’ page as well as updating the menu app’s database with updated meals information. As soon as the update button is pressed, the meals properties are changed for all viewers of the meals.

My meals 🡪 settings

* Using the navigation bar, we can navigate to the user’s settings page. Here we see the user’s settings that we made previously. The address and be a chef options are also there. These can be updated with the save button.

Nav bar 🡪

* Finally, we have the options section of the navigation bar. Here you can report an issue.
* Pressing ‘ok’ navigates the user to their preferred emailing app, and adds the menu app support contact email as the recipient. Unfortunately, email is not set up on this emulator.
* And finally, we have the sign out button. Which as expected, signs out the user.