

MY COOKERY MASTER

A PROJECT REPORT

Submitted by

Shubham Mahajan(19BCS4275)

Utkarsh Chauhan(19BCS4270)

Anand Svarup Bhatia(19BCS4257)

Eish Jindal(19BCS4251)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

WITH SPECIALIZATION

IN

MOBILE COMPUTING



CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

NOVEMBER 2022

MY COOKERY MASTER

A PROJECT REPORT

Submitted by

Shubham Mahajan(19BCS4275)

Utkarsh Chauhan(19BCS4270)

Anand Svarup Bhatia(19BCS4257)

Eish Jindal(19BCS4251)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

WITH SPECIALIZATION

IN

MOBILE COMPUTING



CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

NOVEMBER 2022



BONAFIDE CERTIFICATE

Certified that this project report "**My Cookery Master**" is the bonafide work of "**Shubham Mahajan, Utkarsh Chauhan, Anand Svarup Bhatia, Eish Jindal**" who carried out the project work under my/our supervision.

Mr. AMAN KAUSHIK
HEAD OF THE DEPARTMENT
APEX INSTITUTE OF TECHNOLOGY
CHANDIGARH UNIVERSITY

MR. CHANDRA BHAN SINGH
SUPERVISOR
ASSISTANT PROFESSOR
APEX INSTITUTE OF TECHNOLOGY
CHANDIGARH UNIVERSITY

Submitted for the project viva-voice examination held on November 2022.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

For this final year project, we would like to thank you for all the advice and guidance from my supervisor **Mr. Chandra Bhan Singh**, parents, and friends. Thank you so much for offering the assistance to my food recipe social application development. I really appreciate that because this project could not be completed without these relevant advice and suggestions for improvement. When I encountered problems, the assistance that I've received is invaluable and it makes me getting better in project development and problem solving as well. I am grateful from the bottom of my heart.

Abstract

In order to achieve long-distance communication, social platform has played a very important role in this modern era because it can connect people all over the world. For ordinary social networks like Instagram and Facebook, the information published on these channels usually has a high diversity. If the social network doesn't have a specific theme like food, it may cause problem for users that want to look for particular relevant content such as food recipes. In social network like Facebook, users face difficulty when searching for a recipe that only serves one people and have a less than twenty minutes cook time because Facebook is not specific for recipes sharing. Besides that, ordinary social network has made it more difficult for user to make friends that have same interest as well. Therefore, food social network is necessary to be proposed to solve such problem. The final deliverable of this project is a food-recipes social platform that can gather users with the same interest and share food-related information and recipes. By using the template provided in this food recipe social network, users can categorize the recipe, state the serve size and cook time before publishing to the community. Besides that, this social network also allow user to publish posts to reflect the food taste, service quality and sanitary condition of visited restaurant so other user can get basic understanding of that restaurant before visit there. Several food social networks have been reviewed in order to study the working logic of similar social network. Basic functionalities such as sign up, login, register and logout are implemented as well to make the application more systematic. This project is developed as an android native application using java and firebase as a backend for authentication and storing all data. Moreover, agile method is implemented during development, which is prototype approach, the continuous improvements are implemented based on the initial project prototype until delivered a final product. The agile development approach can reduce cost and time required to develop a mobile application project. Although this project is completed, but improvements can still be done in the future.

List of Figures

Figure 1: System Flow Chart	4.
Figure 2 - 7: Yummly	8-9.
Figure 8-13: Food Tribe	10-11.
Figure 14-19: I'm Hungry	12-13.
Figure 20: Gant Chart	14.
Figure 21: Waterfall Methodology	15.
Figure 22: Software Requirement	16.
Figure 23: Hardware Requirement	16.
Figure 24: Framework	17.
Figure 25: Context Diagram	18.
Figure 26: Activity Diagram	19.
Figure 27: Class Diagram for Servers	19.
Figure 28: Data Flow Diagram	20.
Figure 29: ER Diagram	20.
Figure 30: Android Lifecycle	22.
Figure 31: Service Lifecycle	23.
Figure 32: Creating New Android Application	25.
Figure 33: Configure Project	25.
Figure 34: Android Device Chooser	27.
Figure 35: Android Virtual Device Manager	28.
Figure 36: Android Virtual Device Edit	16.
Figure 37–153: Code for Developing My Cookery Master	29-87.

Figure 154-155: Recipe Page (My Cookery Master)	89.
Figure 156: Explore Page (My Cookery Master)	89.
Figure 157: Bookmark Page (My Cookery Master)	89.
Figure 158: Profile Page (My Cookery Master)	90.
Figure 159: Grocery Page (My Cookery Master)	90.

Table of Content

Acknowledgement	i
Abstract	ii
List of Figures	iii-iv
1. CHAPTER 1: INTRODUCTION	1-6
1.1 Background Information	1-2
1.2 Problem Statement	2
1.3 Project Objective	3
1.4 Project Scope	3-4
1.5 Proposed Approach	4-5
1.6 Highlights of What have been achieved	5-6
1.7 Summary	6
2. CHAPTER 2: LITERATURE SURVEY	7-13
2.1 Overview	7
2.2 Yummly	7-9
2.3 FoodTribe	9-11
2.4 I'm Hungry	11-13
3. CHAPTER 3: DESIGN FLOW/PROCESS	14-87
3.1 Planning & Timeline	14
3.2 Methodology	15-16
3.3 System Requirement	16
3.4 Framework	17
3.5 Process Model	18-19
3.6 Data Model	19-20
3.7 Android Development Basics	21-29
3.8 Code for Developing My Cookery Master	29-87
4. CHAPTER 4: RESULT ANALYSIS AND VALIDATION	88-90
4.1 Steeple Analysis	88
4.2 Output for My Cookery Master	89-90
5. CHAPTER 5: CONCLUSION & FUTURE WORKS	91
5.1 Conclusion	91
5.2 Future Works	91
REFERENCES	92-93

CHAPTER 1: INTRODUCTION

1.1. Background Information: In the past two decades, the interaction between humanity across a long distance has always been a concern. As human beings are social animals, people need the communication to maintain healthy mind and strengthen the relationship with another individual. With such a demand, social network was born for connecting people all over the world. For example, Facebook, Instagram and Twitter are the most well known type of social network in current market. Not only provide a platform for connecting people all around the world, but social network is also a great platform for information and knowledge sharing as well. For some community, social network may just design for a specific theme to focus on a specific knowledge and information sharing. Food-related information could be a social network theme because food lover is a large community that willing to share food-related knowledge and information such as food recipe. With food social network, food lovers who have the same interest can get in touch with each other easily. According to investigation, social network applications are the highest used mobile application over the world, and it has been growing at exponential rates. It is a long story for social network development history. Some basic features of a social network such as build profile, build conversation, upload content was defined in this long journey. The first created social media site was Six Degrees in 1997, this site allows user to create and set up the profile page, add other users into friend list and send messages via network. From 1997 to 2001, Six Degrees had reached a peak of about 1 million users. The major features of Six Degrees made it became a foundation of social application nowadays. The profile mechanism can greatly represent a user's background and information within the profile page is totally customizable. Moreover, the message features can help users to establish a connection with friends. The occurrence of Six Degrees had become an important milestone in social application development. The first blogging sites, LiveJournal was proposed in year 1999. It is a platform that allow users to write blog in form format to keep updating daily lives. The blog format introduced by LiveJournal has become a social media sensation that is still popular nowadays where the Facebook status initially was designed base on blog format. After the popularization of blog site, social media channel began to increase in popularity. In early 2000s, sites like Friendster and LinkedIn are launched and started to gain prominence. Friendster is a dating site that allows users to create personal profile, update daily status to reveal user's mood. For LinkedIn, it is a social media site that linked up with business as it is widely used in job finding and employee recruitment. Users of LinkedIn can post resume and send private message to other users for looking job opportunities. In February 2004, the Facebook was launched by Mark Zuckerberg, along with his Harvard roommates. Initially, Facebook is only available for Harvard students but after months and years, the membership of Facebook is distributed worldwide. In 2012, Facebook announced that the number of users had reached a milestone – 1 billion. After 16 years, Facebook is still a leading giant in social network field and keep affecting the world. During these 16 years, the feature of Facebook is

getting better and complete eventually and it has become a sense of universality for having a Facebook account.

1.2. Problem Statement: In ordinary social network, the information published by user are usually not uniform. Different users can publish different contents to reflect daily status or interests. For food lover community, if the social network doesn't have a specific theme like food, it may cause problem when user try to look for particular relevant content such as food recipes and food-related posts. For example, users of general social network may face difficulty when search for a food recipe that only serves one people and have a less than twenty minutes cook time because the social network is not specific for recipes sharing, many features that can improve recipe sharing experience is not supported such as recipe template for easier recipe creation and recipe filter for easier search. Besides that, social networks without specific theme have made it more difficult for user to make friends that have same interest. Therefore, it is very necessary to build a food social network with specific features for food lover community to solve the problems above.

1) Features that improve recipe sharing experience are not supported by ordinary social networks. For users who like to cook, recipe sharing is a very useful and interesting feature because through the recipe publish by others, user is able to learn how to cook cuisine more quickly and effectively. Although popular social networks nowadays are very mature, but unfortunately some problems may still occur when it comes to recipe sharing. That is because these social networks do not provide specific features that support recipe sharing such as recipe template and recipe posts filter. Therefore, it may affect food lover experience when sharing recipe or searching relevant recipes. Generally, a recipe is consisting of basic information such as recipe title, ingredient list and step by step tutorial. In ordinary social network, recipe post usually is written in a status format. Therefore, user may be confused when viewing the recipe because the recipe content layout that manually arranged by the author may not be intuitive. Moreover, for aged users, extra efforts are needed when arrange recipe content manually. Besides that, searching favourite recipe posts on ordinary social network can be inconvenient because it is lack of recipe filter feature. In this case, user cannot find any effective way when searching a recipe base on category, serve size and cook time.

2) Sharing among people with the same interests is difficult in social network without specific theme.

Food lover is an enormous community. There is always a group of people who are willing to share food information and knowledge with each other. The reason of why ordinary social network is not suitable for such community is because the information published on these social networks have a high diversity and not uniform. Different information such as international news, entertainment and sport can be found when using these social networks. In this case, difficulty may occur when user try to share information with others that have same interest.

1.3. Project Objective: The main objective of this project is to develop a food social network for food lover community to support food-related information sharing such as food recipes and food posts become more effective and easily. According to the problem statements, two objectives are stated as below:

1) Implement features that can improve food recipes sharing experience.

When users publish a recipe in social network like Facebook or Instagram, the content of the recipe must be arranged manually which is not convenient and might confuse the reader when viewing the recipe. Hence, a prebuild template that support recipe sharing should be propose in this food social network. This recipe template should have a visual friendly and reasonable layout that allow user to fill in all recipe data such as recipe title, recipe description, serve size, estimate cook time, category, ingredient list and step by step tutorial. Besides that, this food social network should also support recipe filter so that all users are able to search interested recipes more quickly and effectively. User can filter food recipes base on category, serve size, cook time, popularity and timestamp. These rich filter options can enhance user experience when sharing or retrieving food recipes.

2) Create a social network with food theme and implement features that support social functionality to enhance usability and sharing effectiveness.

In general, a social network should support features such as register, login, reset password, user profile, chat, like, comment, and etc. Therefore, this project is going to implement these features as well. In addition, this food social network should also support user to write and publish food post for sharing daily meal status. Bookmark feature should also be implemented to let users save favourite recipes and food posts. After bookmark operation is done, the bookmarked items are saved into the user profile thus user can easily retrieve it.

1.4. Project Scope: A food social network known as “My Cookery Master” are delivered. It is a social application that design for food lovers to share food-related information and knowledge such as food recipes. The main motivation to develop My Cookery Master is because ordinary social network without theme does not support features that improve recipe sharing experience such as recipe template and recipe filter. Instead of just support recipe sharing, user can also write and publish food post to reflect daily meal status and interest. Besides that. “My Cookery Master” supports some social app general features such as authentication, profile management, chat system, bookmark feature, and etc. First, user can register an account by using personal email and password. If the user accidentally forgot the password, user can choose the forgot password option in login interface and follow the instruction given to reset the password. With account authentication, the user identity is verified before login to My Cookery Master and the security of user’s data can be ensured. Besides that, users can customize their profile to reflect personal characteristic. Users are able to upload profile picture and modify personal information such as username, profession, bio and email. User can search another user by username and

view other user's profile. All created and bookmarked recipes and posts are displayed in user's profile. My Cookery Master also supports chatting feature among the users. There is a channel for user to chat with others to establish social connection. Users can write post to reflect daily meal status as well. User can attach the food photo from phone gallery and write text to describe the post. For post about dining experience in a restaurant, other user who read the post can have a basic understanding of that restaurant before visit there. For food recipe sharing feature, user can create recipes via the prebuild template to improve the content layout. The recipe content such as image, title, description, serve size, cook time, category, ingredient list and step by step tutorial are filled by user according to the template layout. User can then publish the recipe to the community, other user can leave like and comment on that recipe. The popularity of a recipe is depending on the likes number, if the recipe has the most likes, it is going to appear at the top when other user search food recipe base on popularity. By default, all recipes are displayed according to the timestamp, which is the latest on top. In addition, recipe filter provided can enable user to filter recipes base on category, serve size and cook time. Therefore, user can get the targeted recipe more quickly and easily.

1.5. Proposed Approach:

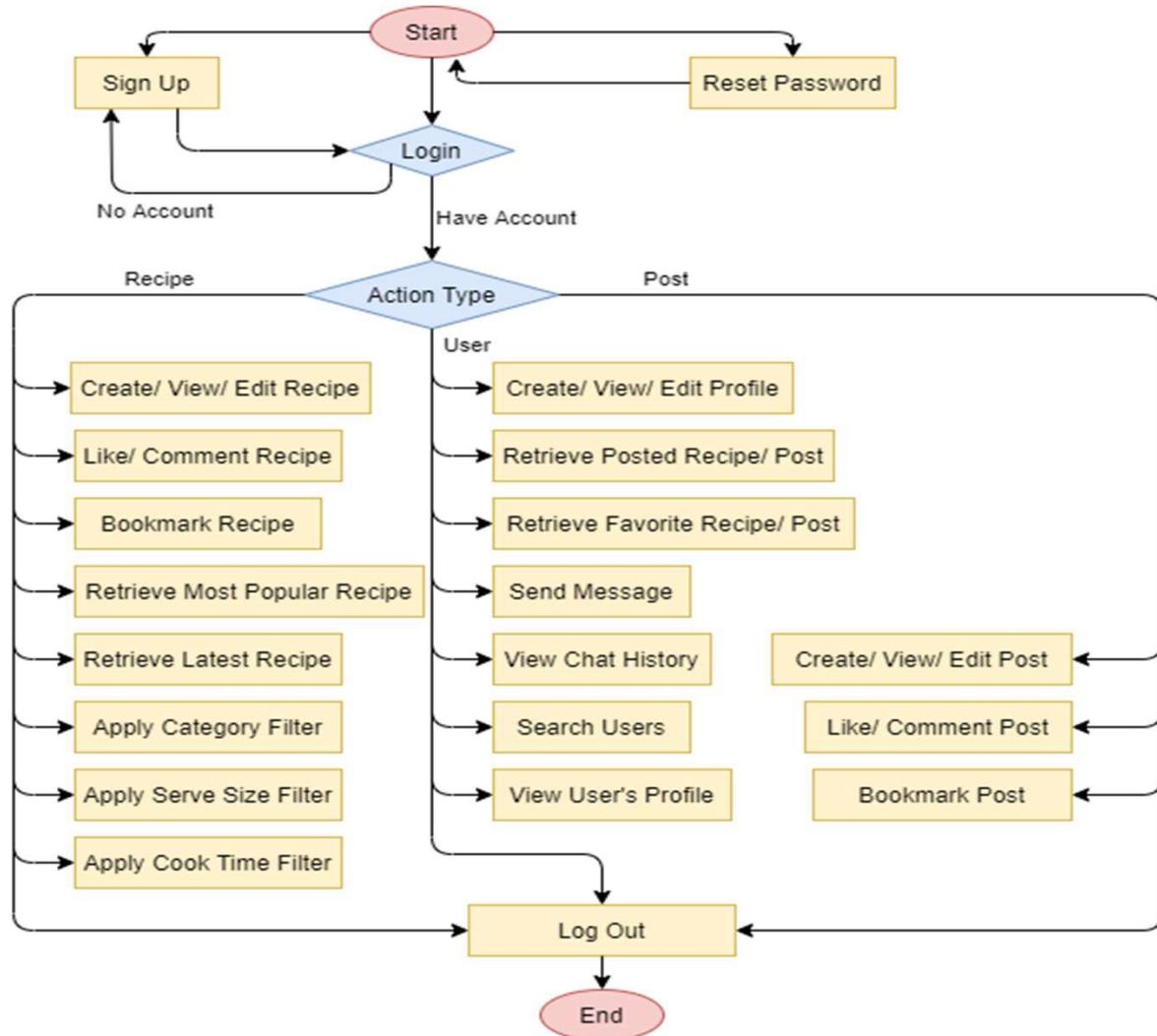


Fig. 1: System Flow Chart

Firstly, user must register an account to have access to the social network. After profile setup is done, user is directed to the profile page. In this page, user can edit profile or access to posted recipes, bookmarked recipes, posted posts and bookmarked posts. User can go to the recipe page through the recipe tab on the bottom navigation menu bar. User can create and publish, modify or delete own recipes. User can also like, comment or bookmark interest recipes as well. User can click into each recipe to view the full details of that recipe. In addition, User can create own food post and publish in the post page. Similar with recipe, user is able to like comment, bookmark recipes as well. Modify and delete operation are only allow when user is the author of the post. Besides that, user can search other users by name, user can also send message to other user and view the chat history. Moreover, user can apply different filters when searching recipes such as category filter, serve size filter and cook time filter. Instead of filter, user can also sort recipes by popularity or latest on top. Lastly, user can sign out the account or reset password.

1.6. Highlight of What Have been Achieved: My Cookery Master food social network is developed for food lover community that helps in recipe sharing and food-related information sharing. A lot of features have been implemented to achieve the project objectives. First of all, a complete user account mechanism is implemented to help in user identity authentication. Authentication and profile management is very essential in a social network as it is the way of how individuals are represented on the platform. Besides that, the recipe sharing feature has been successfully implemented as well. When user developed some unique recipes and want to share it out to benefit more people, user can make use of the prebuild recipe template to fill in every needed recipe data such as image, title, description, serve size, cook time, category, ingredients list, and step by step directions. With this template, user can save a lot of time as there is no need to manually arranged the recipe content. The number of ingredients field and directions field are totally managed by user as well. As there is no certain number of ingredient and direction for a recipe, user can click on add button to insert a new ingredient or direction field or click on delete button to remove ingredient and direction field, which make things easier to manage and modify. After the recipe is published, edit and delete operation will only be enabled for author by using identity validation, thus no user can make change or delete other user's recipe. In addition, user can apply several filters such as category, serve size and cook time base on requirement. User can also sort the result by popularity or latest display on top. With help of filters, it can enhance user experience in recipe searching and save a lot of time for user to find the interest recipe. Instead of publish recipe, user can also write and publish food post to reflect daily meal status or share food-related information. For both recipes and food posts, user can perform operations such as like, comment or bookmark. Like numbers can reflect the popularity while comment feature can enable user to communicate or discuss with others. Bookmark feature helps user to save interested recipes into profile for easier retrieval in future. In addition, user can search other users by username in users' page. The search bar will detect the input text and search the user database for result generation.

User can then click on image and enter to the user's profile page. The message feature has been implemented as well. When viewing another user's profile, user can click on send message button to enter the chat interface. After message is sent, sender can check the message status of whether the message has been seen by receiver. These implemented features have completed this food social network system as the main goal of this food social network is to help food lover connect with others and share knowledge or information. The recipe sharing features such as recipe template and filter are also enhanced the recipe sharing experience as well.

1.7. Chapter Summary: In Chapter 1, the background information section has introduced the history of social sites development. An overview has been done from the very first recognized social site – Six Degrees, until today most popular social network – Facebook. The problem domain and motivation of this project is introduced with two problem statements. After that the project scope and 2 project objectives are discussed as well. The project scope described that the deliverable at the end of the project. After that, the proposed approach has been shown and achievements of this project have been highlighted and described as well.

CHAPTER 2: LITERATURE SURVEY

2.1. Overview: In this chapter, three food-related applications are reviewed which are Yummly, Food Tribe, and I'm Hungry. Yummly and I'm Hungry are more emphasized on recipe feature while Food Tribe is more emphasized on sharing daily food-related post. For each of them, an account is registered to test the major features to point out the strengths and weaknesses.

2.2. Yummly: Yummly is a very popular recipe mobile app that provides recipe recommendation base on individual's taste. Users are not allowed to publish or share personal recipe in this app. It acts like a recipe library on hand as many recipes can be found in this app. It also supports create shopping list and one-hour grocery delivery service within country. This app is available on both iOS and android device. In 2014, Yummly had 15 million active users in the US and has launched international websites in the UK, Germany, and The Netherlands. User can create account on Yummly easily through Google or Facebook login option. After registration is done, user is directed to the home page. Yummly app will display some trending recipes that may suit user's taste in this page. When user click on the interest recipe, the complete recipe content will load in a single page. Information such as image, title, nutrition, servings, preparation time, ingredient list and step by step tutorial are shown here. After user has tried out the recipe, user can write review message and these reviews will be shown at the bottom of recipe page. It also supports shopping feature where user can add all ingredients into shopping list and buy from local retailer, but this service is only available in some countries. To get step by step direction of some premium recipes, user can click on "Get Direction" button at the bottom of recipe page. Then, user is directed to a website that contains full tutorial guide. Users are not allowed to create and publish personal recipe on Yummly. There are many recipe categories available on Yummly app such as video guided, trending now, kid friendly, diets, etc. User can search interest recipes according to the category provided. When user click into interested category, Yummly will display most relevant recipes recorded in recipe library. These recipes are managed and saved by Yummly, so every recipe has been reviewed critically to ensure the recipe quality. In profile page, the recipes that liked before by user are stored in several default recipe category tabs. User can also create a new collection by giving a collection name and description.

2.2.1. Strengths of Yummly

1. Rich recipes library, up to 2 million recipes.
2. Up to 12 categories of recipes is sorted.
3. User can save recipe into collections.
4. Few steps to register an account.
5. The quality of recipe is high as all recipes is qualified before listed.
6. Does not support search recipes by serve size and preparation time.

2.2.2 Weaknesses of Yummly

1. User cannot publish personal recipe on this platform which means sharing of personal recipe is not allowed.
2. Profile only consist of name and profile image, which is not enough to reflect personal characteristic.
3. Does not support multilingual.
4. The ingredient shopping feature only supported in some countries.

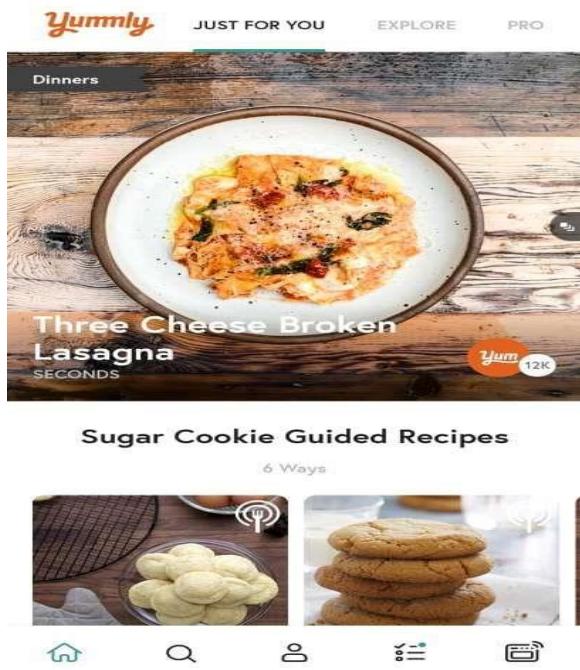


Fig. 2: Home Page

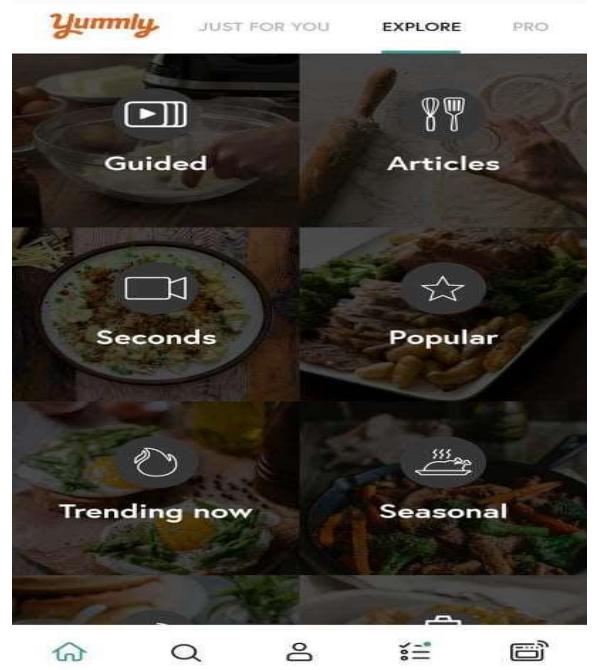


Fig. 3: Explore Page

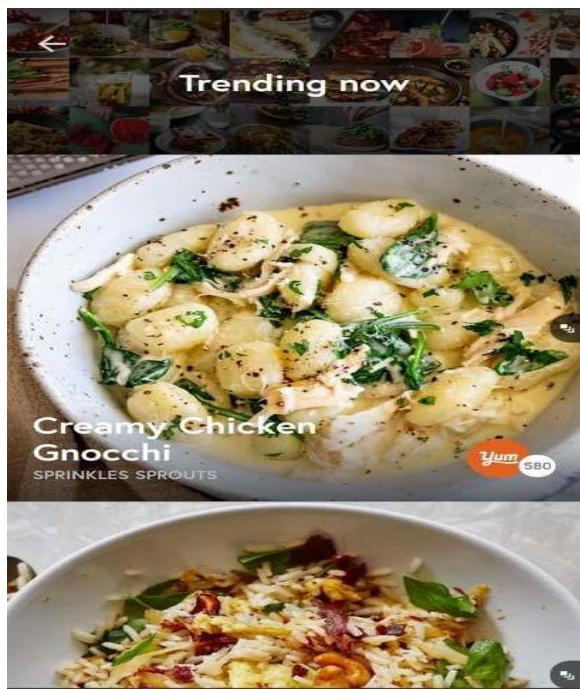


Fig. 4: Categorized Page

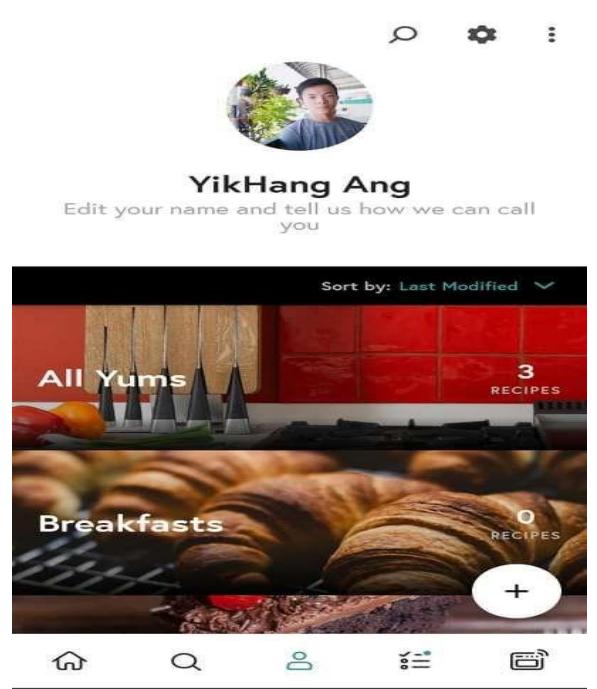


Fig. 5: Profile Page

The screenshot shows a man in a red shirt standing in a kitchen. At the top, there are navigation icons: a left arrow, a double arrow, a right arrow, and a 'Yum' logo. Below the video, the word 'seconds' is displayed. The main content area is titled 'Ingredients' with a note '2/10 ITEMS IN PANTRY'. It includes icons for shopping and buying, and indicates '4 SERVINGS'. A button 'Add All to Shopping List' is present. The ingredient list is as follows:

- + 4 garlic cloves
- + 1/2 box lasagna noodles
- + 480 grams crushed tomatoes (canned)
- + 1 handful basil
- + 227 grams mascarpone cheese
- + 227 grams mozzarella cheese

Fig. 6: Ingredient List

The screenshot shows the same man in the kitchen. The title 'Directions 15 Minutes' is at the top. Below it are three steps:

- STEP 1**: Turn on your broiler. Bring a large pot of salted water to a boil for your lasagna noo...
- STEP 2**: Add the tomatoes to the pan of garlic and season with a pinch of salt. Reduce the sa...
- STEP 3**: Grab your mascarpone and mozzarella cheeses. If you've had a tough day, take a...

A button 'View 1 more' is visible. At the bottom, there are nutritional facts: 440 CALORIES, 14G SODIUM, and 10G FAT.

Fig. 7: Direction Details

2.3. Food Tribe: Food Tribe is a social media that design for food lovers. It is downloadable in google play store and it is totally free for use. The major feature of this application is allowed user share food-related status and information. Users are prompted to select the interest tribes when a new registration is done. There are many tribes available for user to follow such as Home Cooking, Sweet Treats, Meat, Vegan, Street Food etc. The tribes are divided into 49 categories and at least one of them should be choose by users. After followed the tribe, the tribe-related post published by others are shown on user's timeline. Moreover, tribes followed by user is going to display in user profile page hence users can know the interested tribes of each other. User can also modify the interest tribes whenever needed. When found some interesting post, user can follow the publisher to receive the new post on time. Every user has a follower list and following list and it is totally transparent to others. After followed a user, the chat feature is enabled, and chat data is not public in order to ensure user privacy. Like general social applications, user is allowed to like and comment to any post publish by others. User can also choose whether publish the post on profile or tribes. User should at least attach a photo or video in every post and the caption can be write at template's bottom section. The overall layout of post in Food Tribe is quite simple. A thumbnail photo is displayed at the top of the post and following by the post title. The post content is arranged under the title as well but due to the text alignment, the content displayed is not neat as expected. Moreover, there is another attractive feature in Food Tribe which is the quizzes feature. User can create quiz base on a particular tribe, the questions can be set by publisher and answer type can also be define as checkbox, radio button, short answer when needed. At the bottom of each quiz, the participants can comment to share opinion against the quiz question. The quiz feature in Food Tribe can help to spread knowledges and enhance the interactivity level among the users.

2.3.1. Strengths of Food Tribe:

1. User can publish food-related post and categorize it into corresponding tribe easily.
2. Up to 49 tribes that can be subscribed by users.
3. Few steps to set up an account. Design flow/Process
4. The overall design of user interface is neat and easy to view.
5. Quiz can be conduct by users to share food knowledges and ideas.
6. Completed social features such as like, comment and chat are available.

2.3.2. Weaknesses of Food Tribe:

1. Does not support recipe sharing.
2. Does not support multilingual.
3. The random advertisements that showed in application affects the user's experience.
4. Not friendly for new user as it is lack of tutorial and documentation, new user may not know how to perform some feature such as create new quiz.
5. Every post needs to attach at least one photo or video, it may cause problem when there is no suitable photo for use.
6. Text alignment of post is not neat.

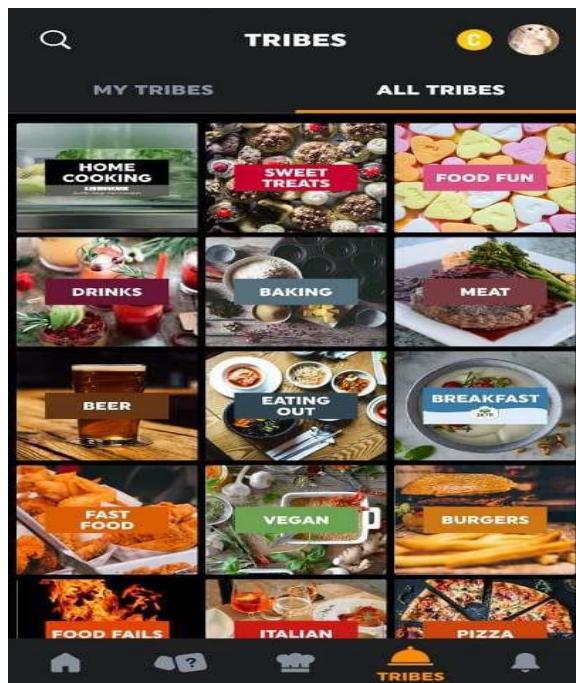


Fig. 8: Tribes Page

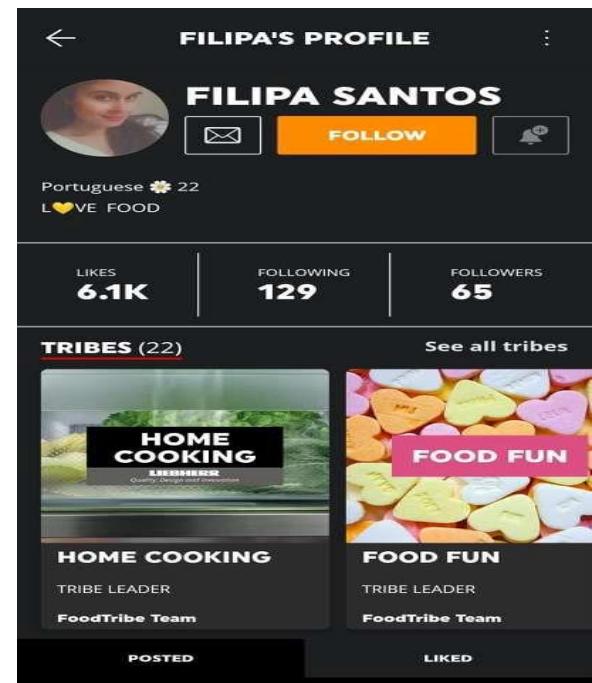


Fig. 9: User Page

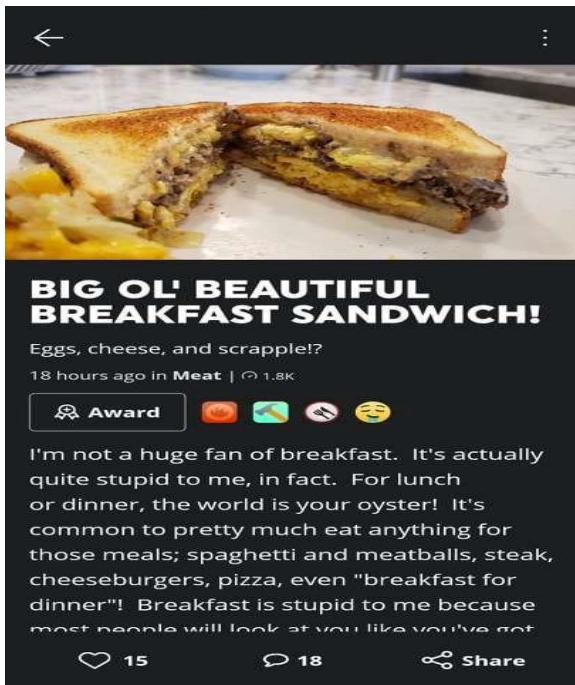


Fig. 10: Food Post

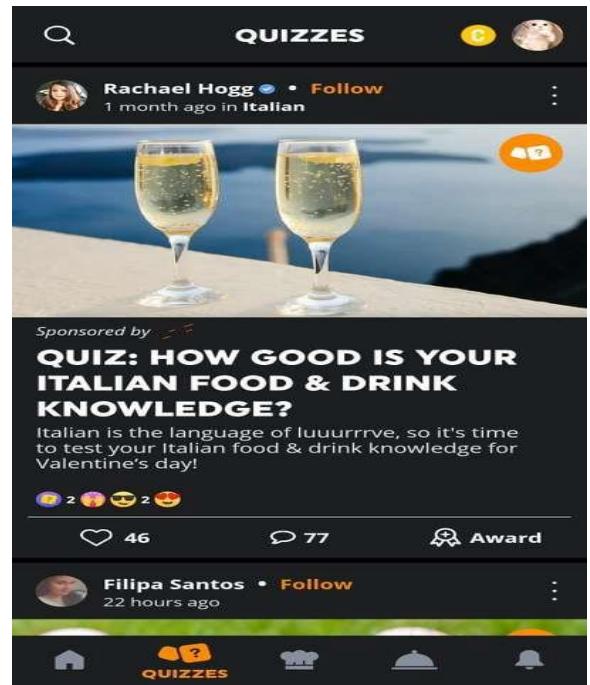


Fig. 11: Quick Page

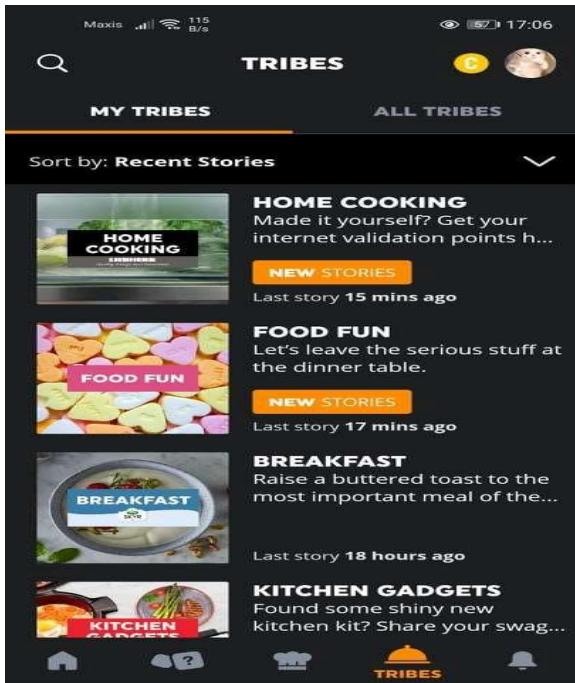


Fig. 12: My Tribe (FoodTribe)



Fig. 13: Notification

2.4. I'm Hungry: I'm Hungry is an android app that helps people in discovering food recipes. User can quickly sign up an account by using google account sign up option. After registered, user is directed to the home page that displayed random food recipes. In this application, users are not allowed to create and share personal recipes. Every recipe displayed in this application is basically from website source. For each recipe, the website domain name is shown under the food recipe title, means that the food recipe is getting from this website. When user click into interested recipe, user is directed to a recipe page that contains all recipe data such as recipe title, website source, category, preparation time, and ingredient list.

At the bottom of the recipe page, user can click on view full recipe button and user is directed to the website that contain step by step direction tutorial. The application also provided search recipe feature. User can search recipes base on keyword or directly select the food category provided. In addition, this recipe has supported meal plan feature. User can attach recipe to a particular day to have a more appropriate diet plan. This meal plan feature can help user organize the diet habit more easily as it can remind user of what should eat in the coming days. Moreover, this application is also vegan-friendly. User can choose to disable all meat recipes and allow only vegetarian recipes display in home screen by adjust the setting in profile page.

2.4.1. Strengths of I'm Hungry

1. Vegan-friendly, can disable all meat recipes from being displayed.
2. Number of recipes is large as most of the recipes are from internet source.
3. Search recipe by category can help user filter recipes more quickly.
4. Support meal plan feature.

2.4.2. Weaknesses of I'm Hungry

1. Lack of social functionality such as user profile, chat and search user are not available.
2. User cannot share personal recipe on this platform.
3. Does not support multilingual
4. Rough application user interface design
5. Does not provide search recipes by serve size and preparation time.

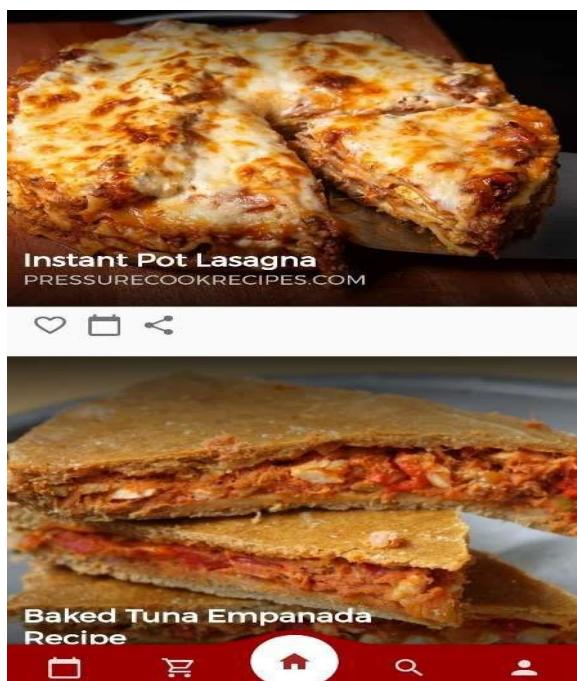


Fig. 14: Home Page

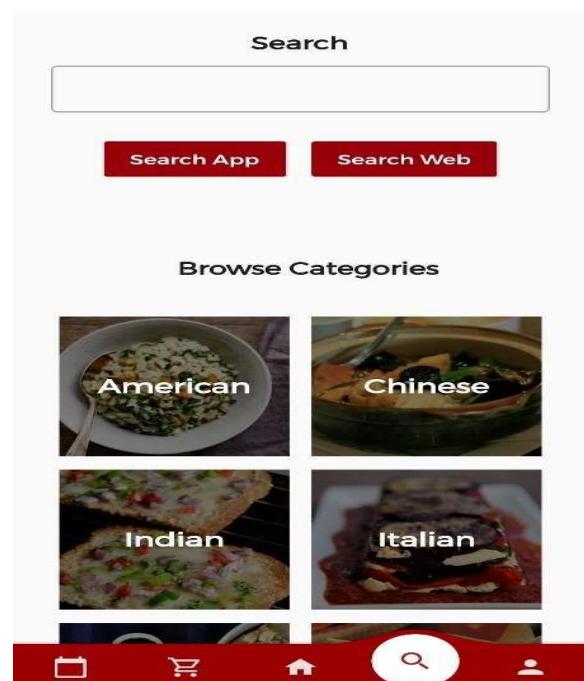


Fig. 15: Search Page

Your Weekly Meal Plan
Week of 2021-03-28

Sunday
+ Add to Sunday

Monday
+ Add to Monday

Tuesday
+ Add to Tuesday

Wednesday
+ Add to Wednesday

Thursday

Fig. 16: Meal Plan



Instant Pot Lasagna

pressurecookrecipes.com

American-Italian Main

50:00

INGREDIENTS

- 8 (160g - 200g) oven-ready lasagna
- 2 cups (150g - 180g) Mozzarella chee
- ¼ cup (30g - 40g) Parmesan chee
- 2.5 - 3 cups (600g - 750g) meat sa

Fig. 17: Recipe Page

Instant Pot Lasagna
pressurecookrecipes.com

American-Italian Main

50:00

INGREDIENTS

- 8 (160g - 200g) oven-ready lasagna
- 2 cups (150g - 180g) Mozzarella chee
- ¼ cup (30g - 40g) Parmesan chee
- 2.5 - 3 cups (600g - 750g) meat sa
- Salt + pepper to taste
- ¾ cup (200g) Ricotta cheese
- 1 teaspoon (1g) Italian seasoning
- 1 large egg
- ½ teaspoon kosher salt

VIEW FULL RECIPE

Fig. 18: Ingredient List

AMY + JACKY
PRESSURE COOK RECIPES.COM

Instructions

- 1 Create Ricotta Cheese Mixture:** In a mixing bowl, beat one large egg, then add in ¼ cup (200g) Ricotta cheese. Season with 1 tsp (1g) Italian seasoning, ground black pepper, and ½ tsp kosher salt.
- 2 Assemble Instant Pot Lasagna:** Line a 7-inch springform pan with cut-out parchment paper (optional). Break the uncooked oven-ready lasagna noodles into smaller pieces, then layer them at bottom of the pan in a single layer. Layer ⅓ portion of the meat sauce on the lasagna noodles. Layer Ricotta cheese

Fig. 19: Direction Details

CHAPTER 3: DESIGN FLOW/PROCESS

3.1. Planning & Timeline:

Gantt Chart

Week \ Task	W 1	W 2	W 3	W 4	W 5	W 6	W 7	W 8	W 9	W 10	W 11	W 12	W 13	W 14
Introduction														
Literature Review														
Proposal Progress Presentation & Evaluation														
Discussion & Correction														
Proposed Solution Methodology														
Proof of Concept														
Drafting Report of the Proposal														

Fig 20. Gant Chart for Project Processes and Completion

1. **27th September 2022:** Getting started.
2. **1st October 2022:** Basic Widgets modelling.
3. **3rd October 2022:** Designing Splash Screen and certain pages.
4. **5th October 2022:** Applying Scroll View.
5. **8th October 2022:** Making Interactive Widgets for users.
6. **10th October 2022:** Routing and Navigating the app with Navigator 2.0.
7. **12th October 2022:** Applying Deep links and web URL's.
8. **14th October 2022:** Using Shared Preferences to Store data.
9. **18th October 2022:** Serialization with JSON.
10. **20th October 2022:** Providing a network.
11. **25th October 2022:** Using Chopper Library in app.
12. **28th October 2022:** Building a better state management architecture.
13. **1st November 2022:** Saving data with SQLite.
14. **3rd November 2022:** Adding platform specific app assets.
15. **5th November 2022:** Adding Fire store functionality.
16. **7th November 2022:** Adding chat functionality in the app.

3.2. Methodology: To develop “My Cookery Master” apps, I am going to use Waterfall Model methodology. This methodology consists of different stages which are requirement analysis, system design, implementation, testing, deployment, and maintenance.

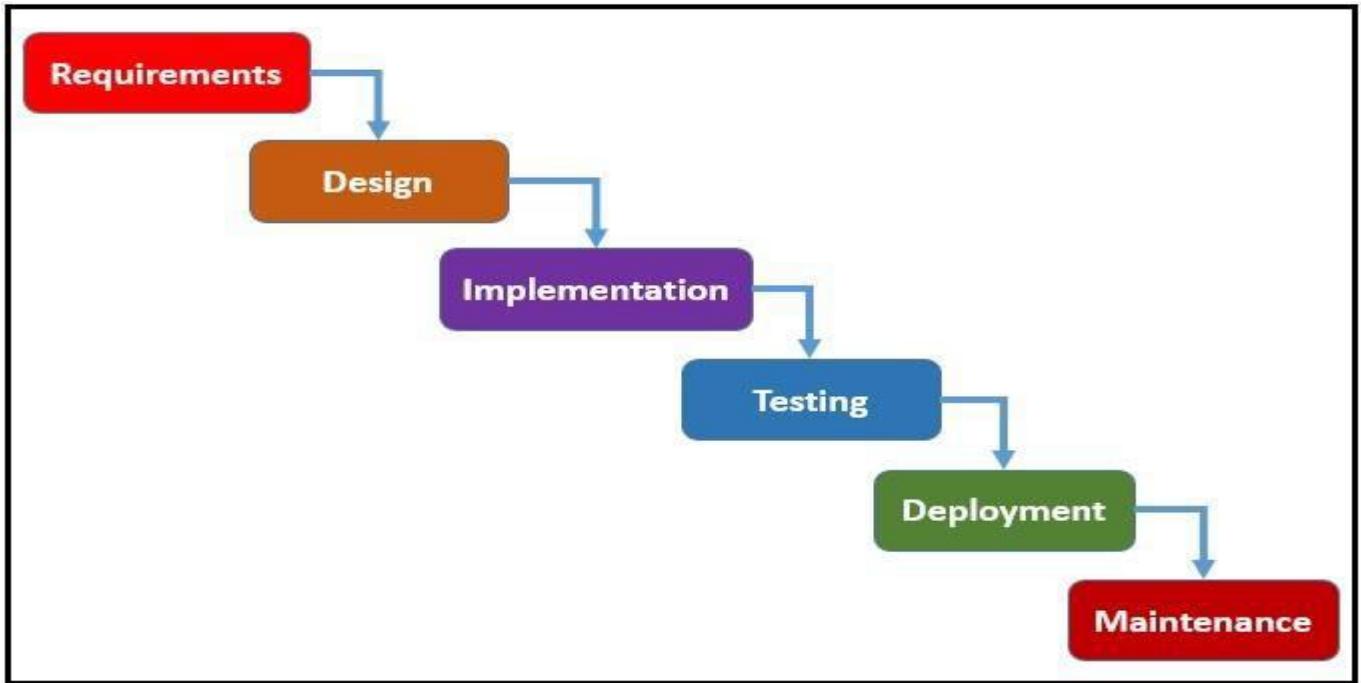


Fig 21. Waterfall Model Methodology

3.2.1. Requirement Analysis: During this early stage, the possible requirements of the application are analysed thoroughly and documented so that it can be refer during the future development. The outcome result about requirements that has been documented shows that what the application should do, but not how it should do it. All the requirements for the application were gathered by brainstorming and analysing existing applications. For this project, the solution is brand new and needs to be created as a set of ideas that people can agree to. So, brainstorming is simply one of the best techniques to get as many ideas as possible. Brainstorming also is the most effective for solving problems, generate a list of ideas or focus on the serious issue. After all the ideas are generated, prioritize one idea that think the best for this solution. The resulting agreement of best ideas is used for the early requirements.

3.2.2. Design: For this stage, the idea from brainstorming and result from analysing existing is studied to design a better interface to satisfy the user. The design for interface should cover some of basic features in My Cookery Master app. The system is proposed to design an app-based recipe guidance that being able for user to search and sharing recipe.

3.2.3. Implementation: The implementation stage is begin once the design is approved. This project is develop using PHP, MySQL, Android Studio and JSON. JSON is a syntax for storing and exchanging data. Mostly the coding language used is PHP because PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server.

3.2.4. Testing: At this stage, the apps will be tested. If there is any error occur or detected, it must be solved at this phase and if there any changes made, it need to be re-implement back to design phase to make sure that the flow of the apps is not affected.

3.2.5. Deployment: After the apps is free from error and bugs, the apps will then be deployed for market. The testing that takes place will approve the validity of the system for marketing. By then, it is mandatory to monitor the apps time to time to make sure that if any changes had to be done, it is noticed.

3.2.6. Maintenance: For the last stage, which is maintenance, the apps will be updates as necessary to keep up to date with its environment. It focus on change associated with error correction.

3.3. System Requirement: Below are the software and hardware requirement needed to develop and run the application smoothly.

Software Requirement:

No	Software	Description
1	MySQL	System database application
2	Android	Operating system to run the application
3	Android Studio	To develop the application
4	Xampp	Connection with the database
5	Notepad++	For the coding development

Fig. 22: Software Requirement

Hardware Requirement:

No	Hardware	Description
1	Notebook	Intel® Core™ i5-6300HQ CPU @ 2.30GHZ
2	Mobile Phone	Xiaomi Redmi Note 4
3	Printer	Print the report
4	Flash Drive	Used to back up the project

Fig. 23: Hardware Requirement

3.4. Framework: For the framework of the My Cookery Master apps, the user can publish, share, search recipe, comment, and rate recipe at the apps. After that, the apps can show the recipe, comment, and rate to the users. The moderator role at the apps is to report any suspicious behaviour user to admin while they also can view recipe, comment, and users' activities. For the administrator, the admin can manage recipe, manage comment and rating of the users. Lastly, for the database, it will store recipe, comment, rating of the recipe and users' details and also it will send recipe, comment and rating to the apps so that the users can view it.



Fig. 24: Framework of the My Cookery Master App

3.5. Process Model: The My Cookery Master Apps consists of 3 entities which are Administrator, Moderator and User. As usual, these entities need to login into the application before able to access the interface. Admin can view users' activities, manage the published recipe, comment, and rating of the users. For user, they can search, publish, share, comment, and rate recipe. As for moderator, they can report any misbehaviour user to the user to maintain the environment in the apps.

Context Diagram (CD)

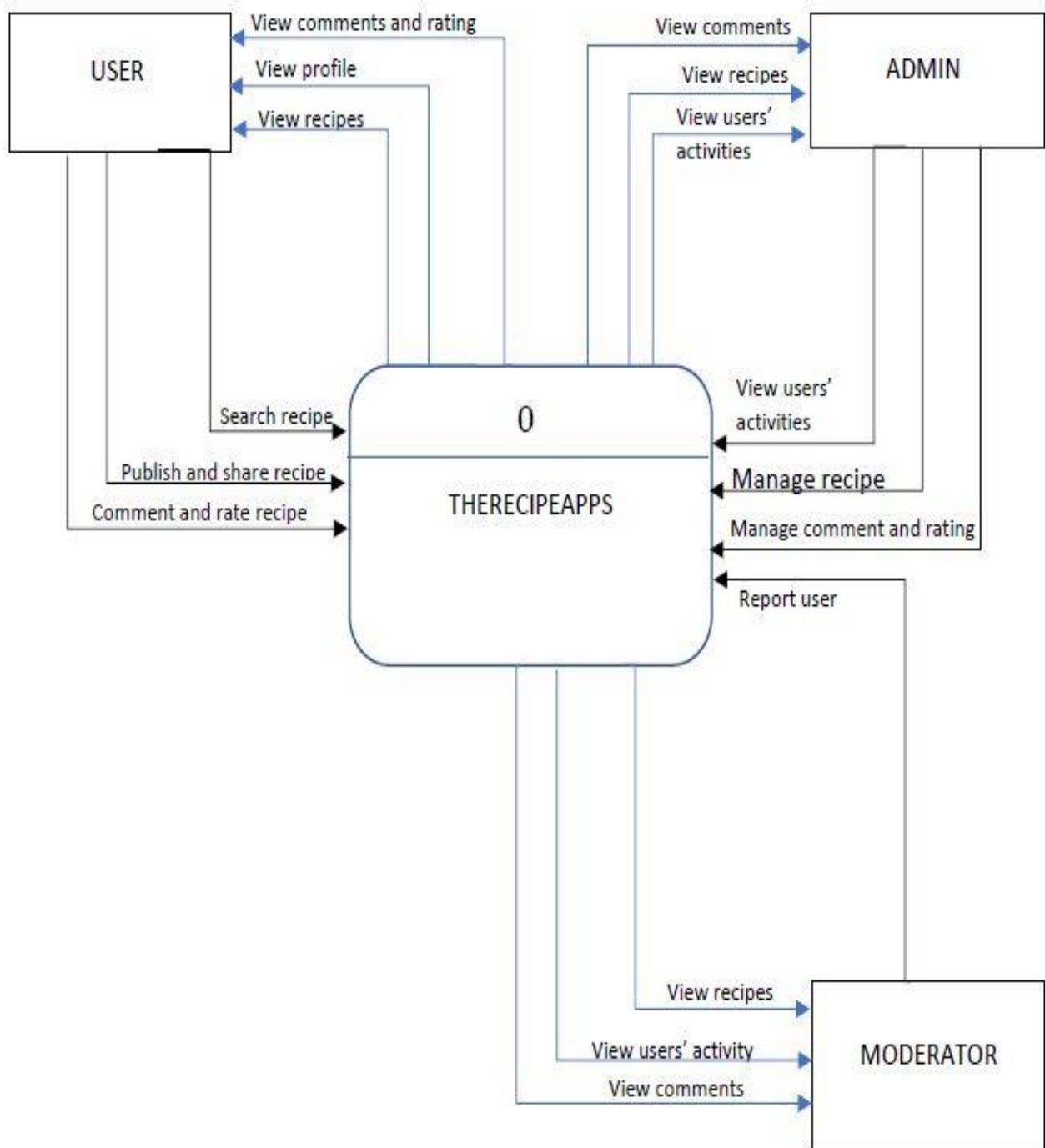


Fig. 25: Context Diagram for My Cookery Master

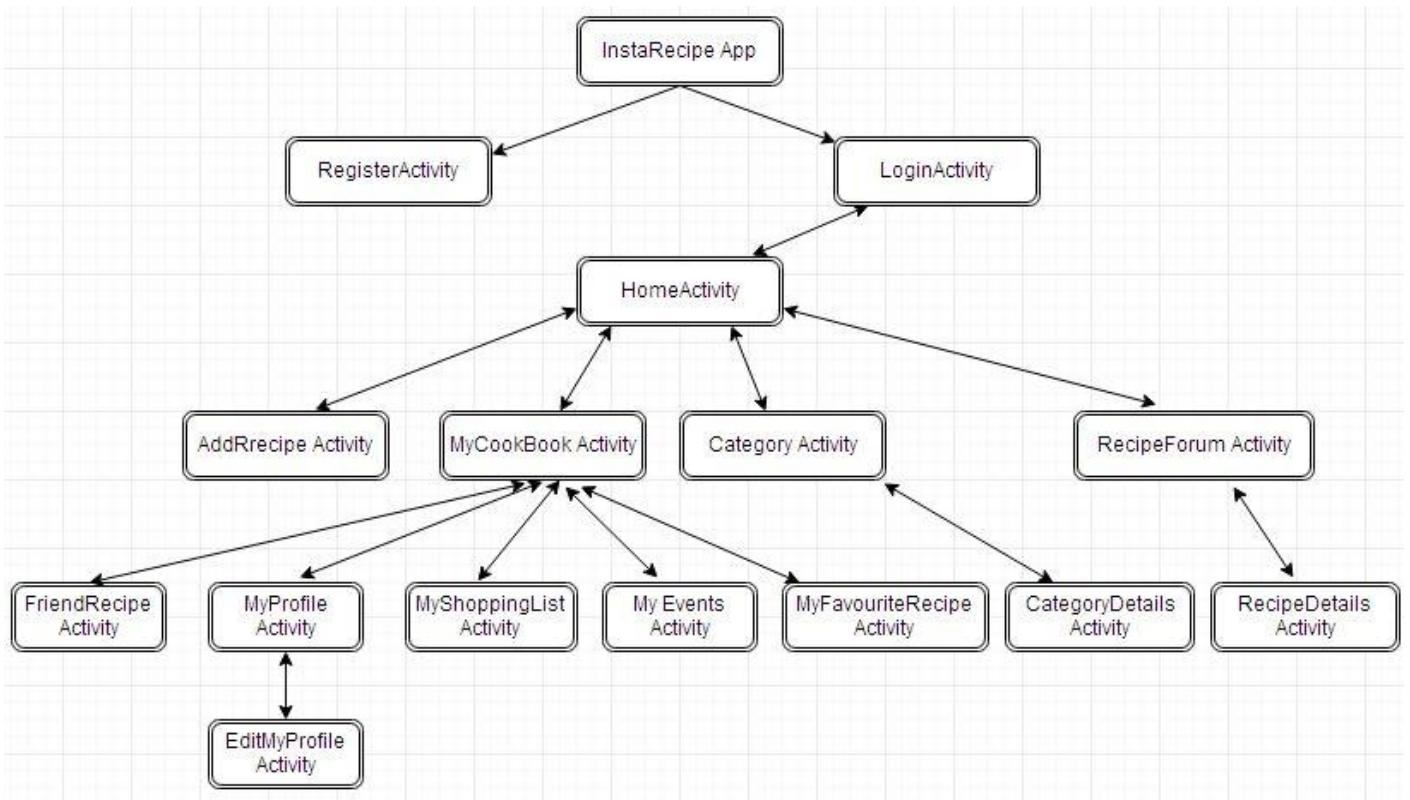


Fig. 26: Activity Diagram

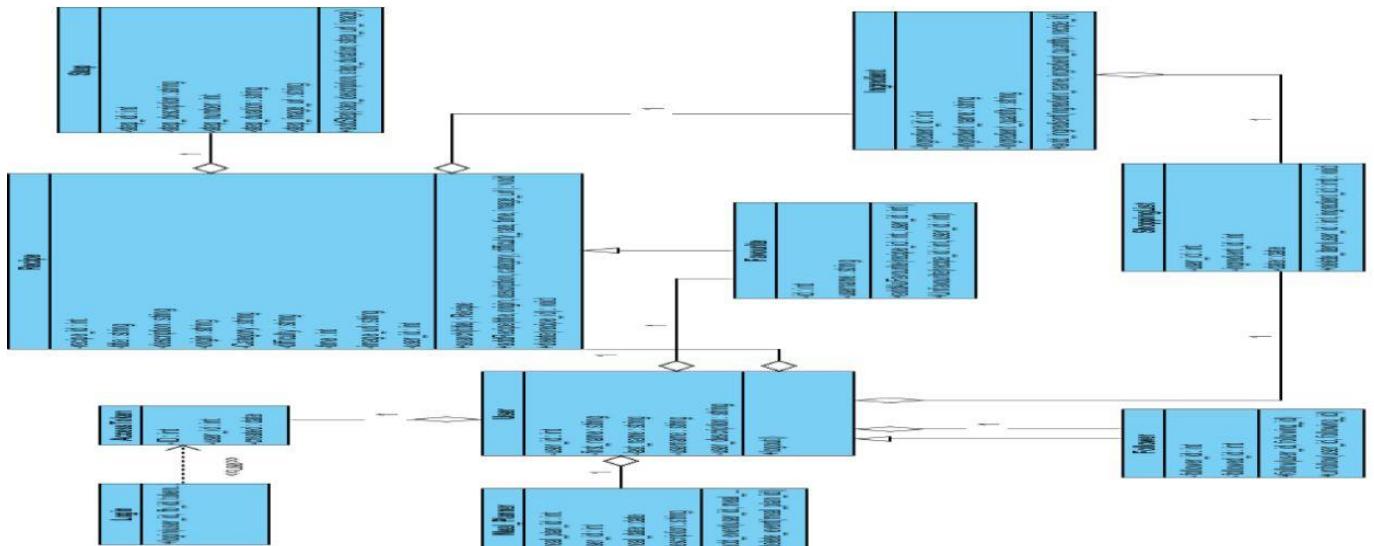


Fig. 27: Class Diagram for Server

3.6. Data Model:

3.6.1. Data Flow Diagram (DFD): A Data Flow Diagram (DFD) is a graphical representation of a flow of data through the application. In figure above, there are five processes involved. Processes in this system are sign up or login, publish and share recipe, search recipe, comment and rate recipe and report. There are four data stores involved which is login, recipe, comment and rate and report details. We can know that the first process is sign up or login followed by publish and search recipe, search the recipe, comment and rate recipe and lastly report the user.

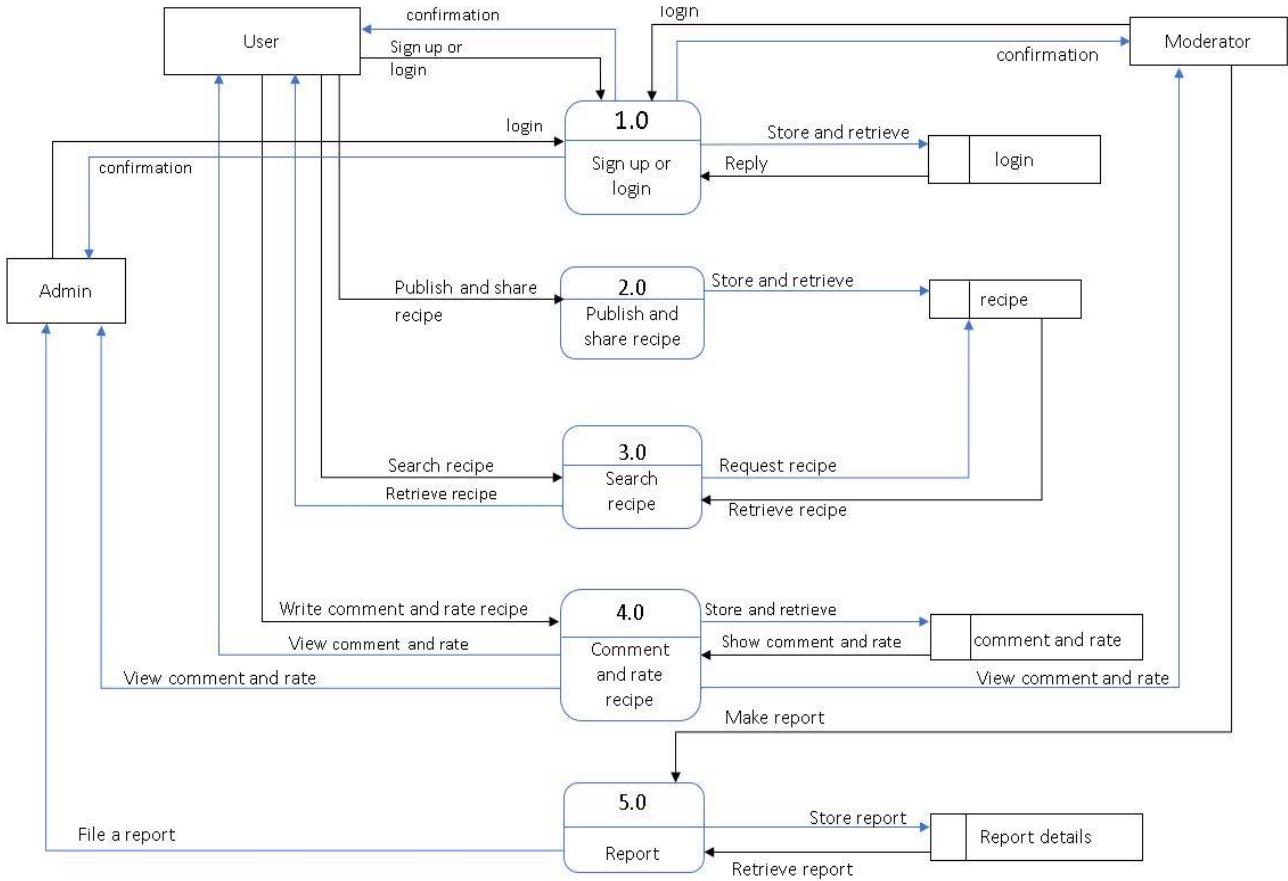


Fig. 28: DFD for My Cookery Master Application

3.6.2. Entity Relationship Diagram (ERD): Figure shows the entity relational diagram of the system. This diagram shows the database and their entity that involve in the process for the system. All the updated data in the data store is self-updated by user. In the diagram, we can see several lookup tables that is to be inferred by the app's system. There are three database which is comment, recipe, and report. For the login, the user need password and username to proceed to the main page.

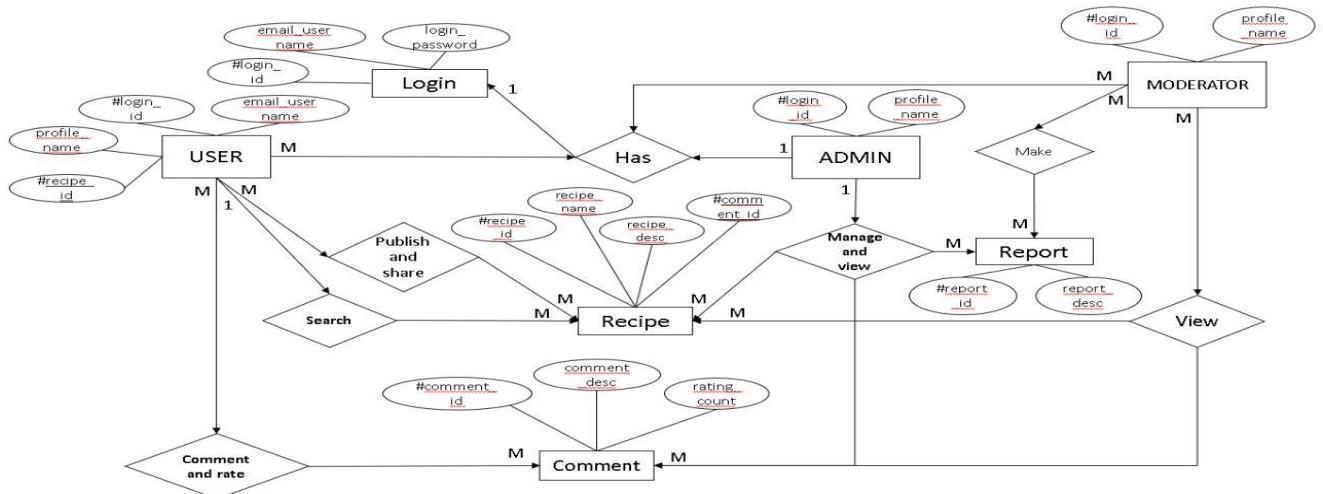


Fig. 29: ER Diagram for My Cookery Master Application

3.7. Android Development Basics: Android is a Linux-based operating system designed primarily for touch screen mobile devices such as smartphones and tablet computers. To make things easier for developers, Google had made Android as open source and releases the code under the Apache License. Android has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in a customized version of the Java programming language [1].

Quick Recipes application is written in java programming language using SQLite database. The code of Quick Recipes App is compiled using Android SDK tools into an android package, an archive file with an .apk suffix. This .apk file is considered as one application and is used to install the application in all android devices like tablets and phones.

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact [1][2]. There are four different types of application components: activities, services, broadcast receivers and content providers.

3.7.1. Four Types of Application Components: Android provides four application components: activities, services, content providers and broadcast receivers. Below section describes each component in detail.

3.7.1.1. Activities: Activity represents a single screen with a user interface. For example, in this Quick Recipes application there are many activities such as search by title and ingredient activity, category activity, cookbook create recipe activity and so on. One activity in the application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity in order to perform different actions. Whenever a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, it is pushed onto the back stack and takes user focus. This back stack follows LIFO (last in first out) stack mechanism, so, when the user is done with the current activity and presses the back button, it is popped from the stack (and destroyed) and the previous activity resumes.

Activity must be declared in the manifest file to be accessible by the system. Intent filters can also be specified in the element to declare how other application components may activate it.

The activity automatically includes an intent filter that declares the activity responds to the "main" action and should be placed in the "launcher" category.



Fig. 30: Activity Lifecycle

The callback methods for activity are as follows which are shown in Figure 10. **onCreate()**: This is called when the activity is first created. This is always followed by **onStart()**.

onStart(): This is called just before the activity becomes visible to the user. This is followed by **onResume()** if the activity comes to the foreground, or **onStop()** if it becomes hidden.

OnResume(): This is called just before the activity starts interacting with the user. This is always followed by **onPause()**. **onPause()**: This is called when the system is about to start resuming another activity. And this is followed either by **onResume()** if the activity returns back to the front, or by **onStop()** if it becomes invisible to the user.

onStop(): This is called when the activity is no longer visible to the user. This is followed either by **onRestart()** if the activity is coming back to interact with the user, or by **onDestroy()** if this activity is going away .

onRestart(): This is called after the activity has been stopped, just prior to it being started again. This is always followed by onStart(). **OnDestroy()**: This is called before the activity is destroyed. This is followed by nothing[1].

3.7.1.2. Services: A service is a component which would be running in background without direct interaction with the user. As the service has no user interface it is not bound to the lifecycle of an activity. Services are used mostly when there is a time consuming and long task, like loading an Image, or a File, or download something for the Internet and asynchronous tasks in general. Service has two forms: Started and Bound. Below section describes in detail:

1. **Started:** When an activity starts start Service (), then a service is "started". A service can run in the background indefinitely once started, even if the component that started it is destroyed. For example, it might download a image over the network. When the download is completed, the service should stop itself.
2. **Bound:** When application component calls bind Service (), a service is "bound". A bound service provides a client-server interface that allows components to interact with the service, send requests and get results. Figure 11 shows lifecycle of services in android.

The callback methods are:

onStartCommand(): This method is called when an activity requests for a service to be started by calling startService().

onBind(): This method is called when an activity wants to bind with the service by calling bindService().

onCreate(): This method is called when the service is first created to perform setup of the service before onStartCommand or onBind is called.

onDestroy(): This method is called when service is no longer required and is to be destroyed[1].

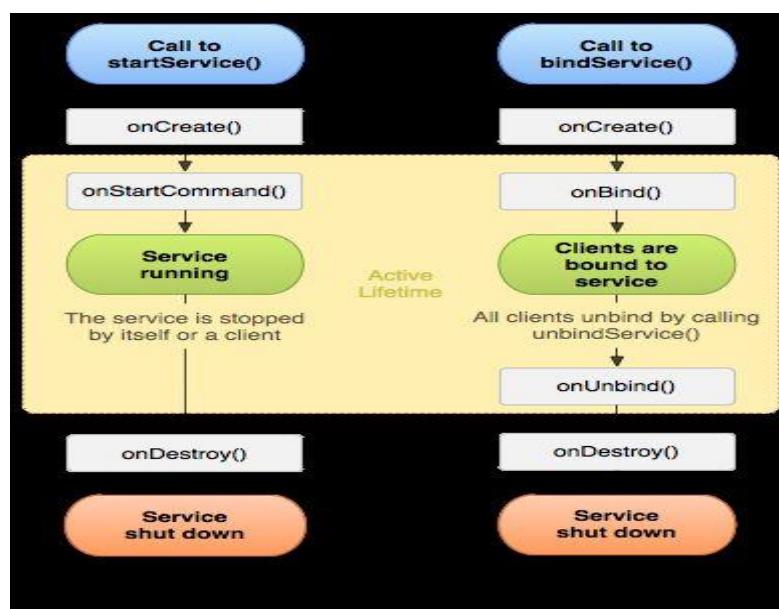


Fig. 31: Service Lifecycle

3.7.1.3. Content Providers: Content providers are used to provide access to other applications to a shared set of application data i.e content providers provide a mechanism through which the data stored by one Android application can be made accessible to other applications. The data is usually stored in file system or SQLite database which the application can access and through content provider, other applications can query or modify the date depending on the settings of the content provider.

A content provider is implemented as a subclass of `ContentProvider` and an application accesses the data from a content provider with a `ContentResolver` client object. In order to insert data into provider, `ContentResolver.insert()` method is used. This method inserts a new row into the provider and returns a content URI for that row. In order to update a row or delete a row `ContentResolver.update()` and `ContentResolver.delete()` methods can be used respectively[1].

3.7.1.4. Broadcast Receivers: A broadcast receiver allows registering for system or application events. When any registered event occurs, receivers for an event will be notified by the Android runtime. System broadcast include screen turning off, the battery is low, or a picture was captured. Applications broadcast would include letting other applications know that some data has been downloaded to the device and is available for them to use. A receiver can be registered via `AndroidManifest.xml`.

The implementing class for a receiver extends the `BroadcastReceiver` class. If the event for which the broadcast receiver has registered happens the `onReceive()` method of the receiver is called by the Android system[5].

3.2. Setup Android Environment: The first step in starting a project would be to download the Android ADT bundle. The bundle provides all features needed to develop the app and also includes a version of the Eclipse IDE with built in ADT (Android Developer Tools) . Perform the following steps after downloading:

1. Unpack the ZIP file (named `adt-bundle-.zip`) and save it to an appropriate location, such as a "Development" directory in our home directory.
2. Then open the `adt-bundle-/eclipse/` directory and launch eclipse.
3. In Eclipse, click on the SDK manager and download the latest SDK tools and platforms.

3.7.3. Creating sample Android Application: The android project consists of all the code required for the android application. The SDK tools which we downloaded provide basic functionality for us to start coding our application [6].

1. First need to click 'New' in the toolbar.
2. When the window appear appears, we need to open the Android folder, select Android Application Project, and then click on Next.

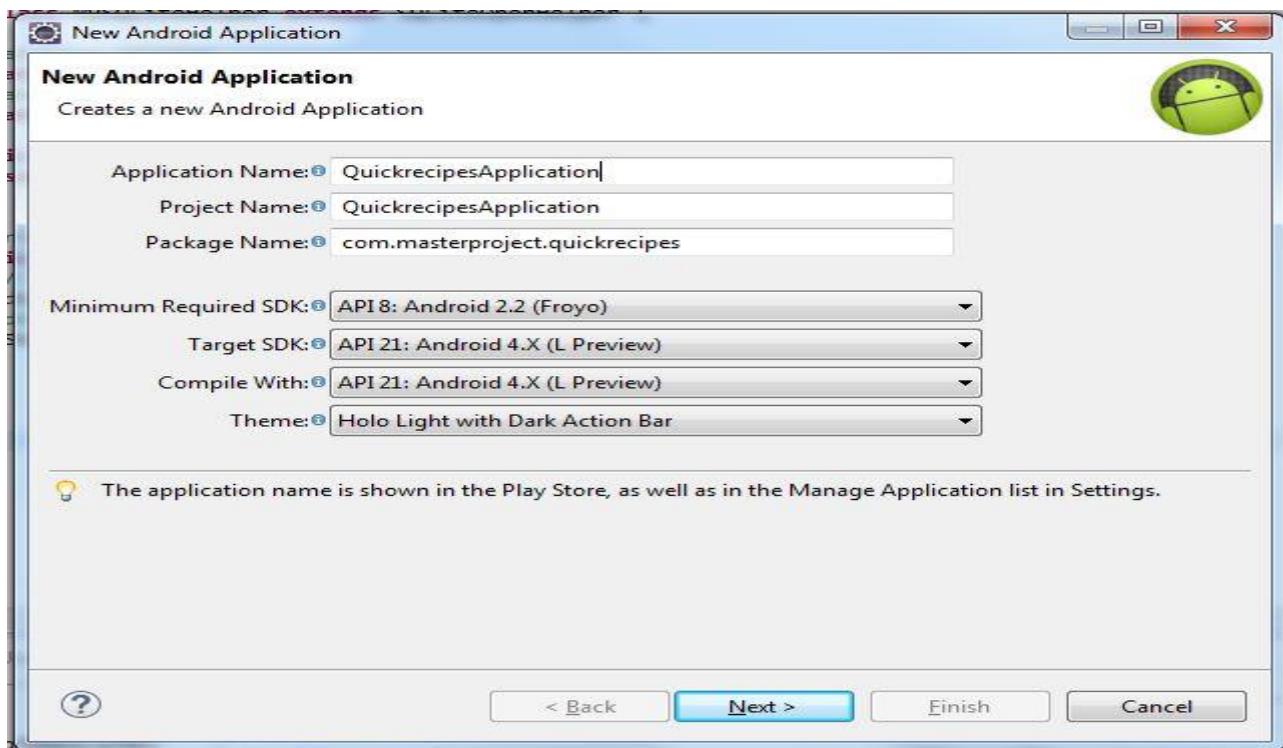


Fig. 32: Creating New Android Application

In the android application box, we need to fill appropriate values like the application name is the app name that appears to users which is "QuickrecipesApplication" for this sample project. The project name is the name of our project directory and the name visible in Eclipse. The Package Name is the package namespace for our app which is "com.masterproject.quickrecipes" for this sample project. The lowest version that this app supports is the minimum required SDK. If we want to support multiple devices, this needs to be set to lowest version available such that the application can perform its basic operations. The target SDK is the highest version of Android for this application. The platform version with which we compile this sample app is specified in the Compile With. The theme specifies the Android UI style to apply for your app.

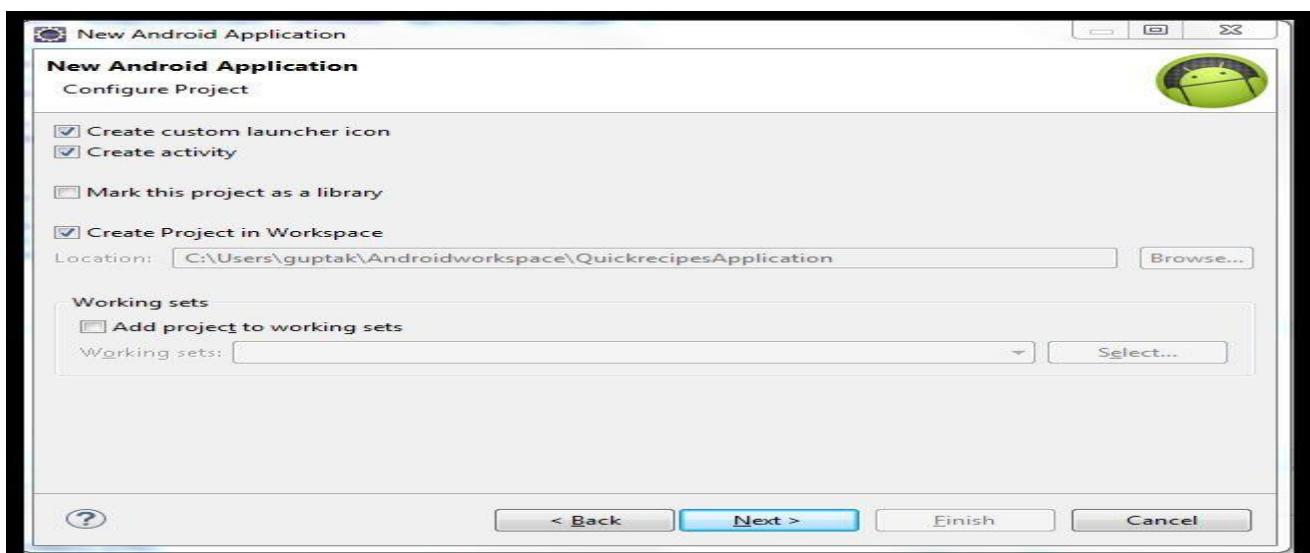


Fig. 33: Configure Project

This screen is to configure the project and is left at default selections On clicking next, the next screen will create a launcher icon for this app. On clicking Next, for this sample project we select Blank Activity and click Next. The other details are left at default and click Finish.

The app by default stores hello world code and we need to just run it. The code looks as follows:

```
package com.masterproject.quickrecipes;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

The Manifest file will provides the characteristics of the app. The code looks like this:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="21" />
<TextView
```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
    </RelativeLayout>

```

The src directory contains all source files required for our application.

The res directory contains sub directories like layout, drawable-hdpi, menu, values etc.

The manifest file which describes the fundamental characteristics of the application and defines each of its components [7].

The app is run by clicking on the QuickrecipesApplication package-> RunAs -> Android application. On running this, the main activity class starts and loads a layout file that says "Hello World."

3.7.4. Running the Application: The application can be run either on device or on emulator.

3.7.4.1. On Device: There are two ways the application can be run on device:

- The android device needs to be connected to the laptop with USB cable. On connecting the device, option of downloading USB driver would appear in case the driver is not already installed. Then enable USB debugging on the device.

This can be found under Settings->Developer options

- The second way is to run the app from Eclipse:

- On Eclipse, click on Run from the toolbar.
- In the Run as window that appears, need to select Android Application and click OK.

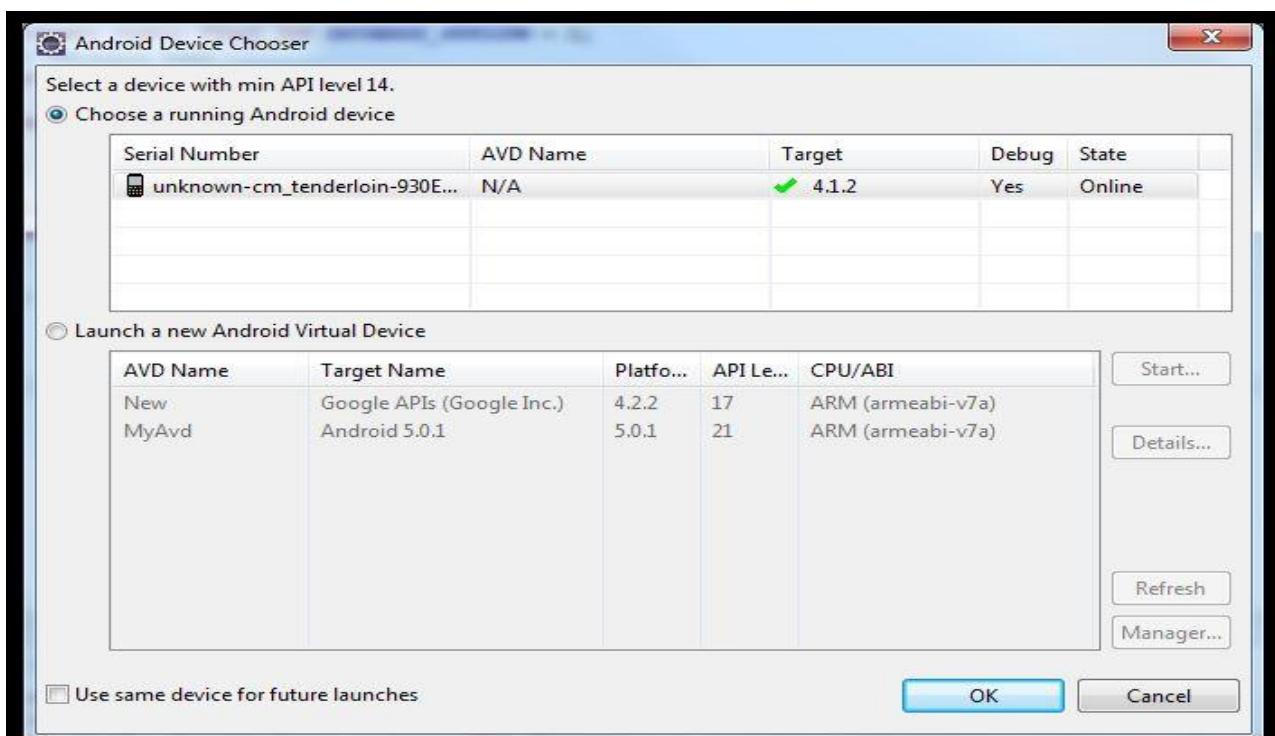


Fig. 34: Android Device Chooser

3.7.4.2. On Emulator: When the application needs to be run on Eclipse, then first step is to set the AVD i.e Android Virtual Device. The AVD is a device configuration for the Android emulator that allows to model different devices.

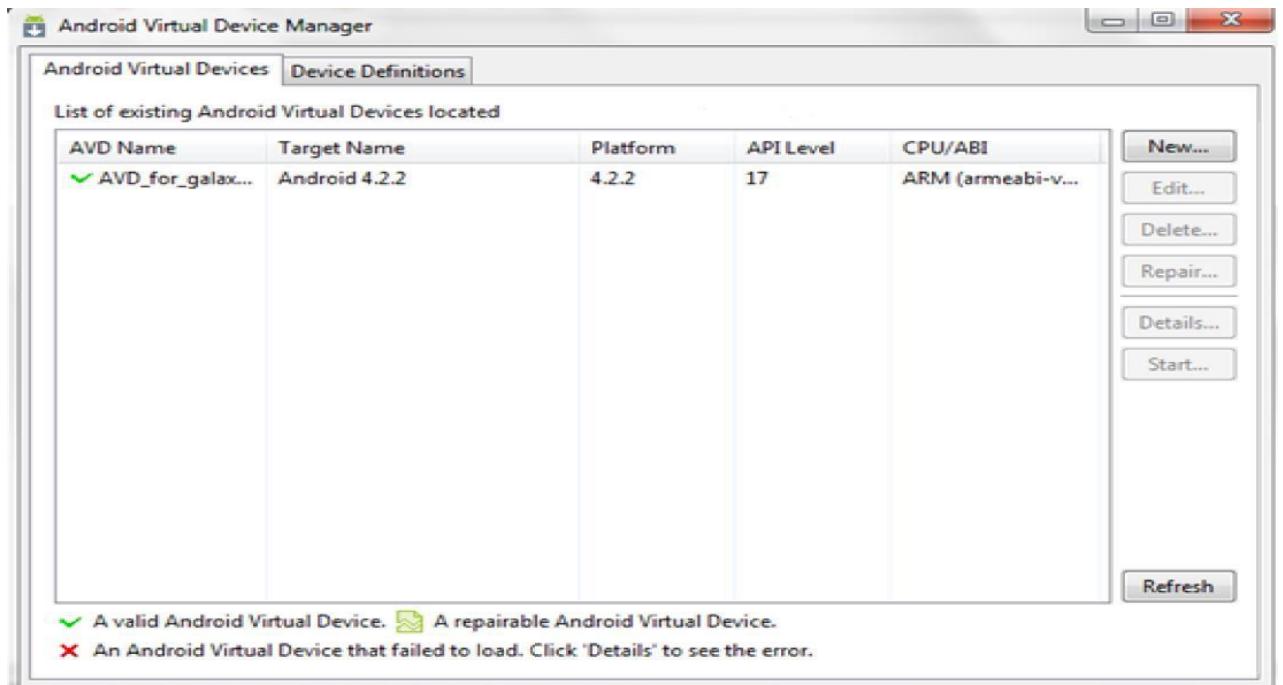


Fig. 35: Android Virtual Device Manager

In order to create AVD, first need to launch the Android Virtual Device Manager by clicking on the toolbar. The next step is to click on ‘New’ in the Android Virtual Device Manager panel. Next step is to fill in the details for the AVD like name, a platform target, an SD card size.

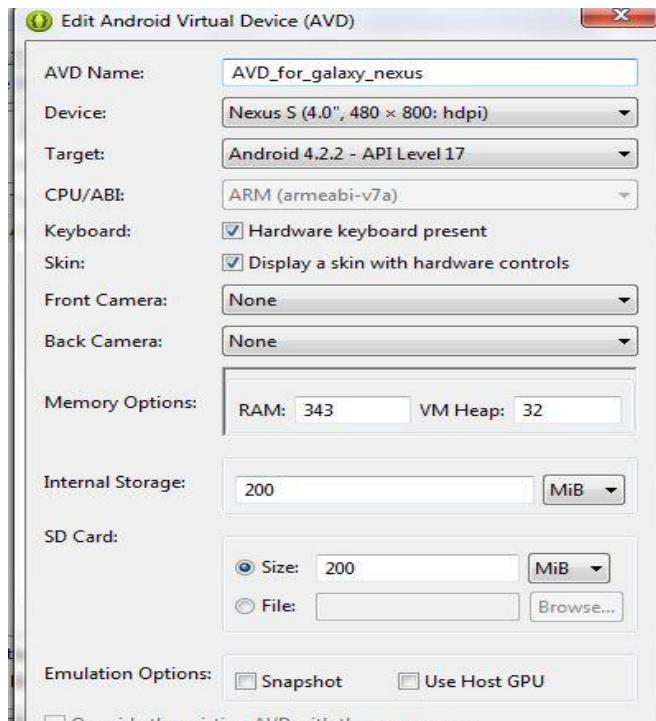


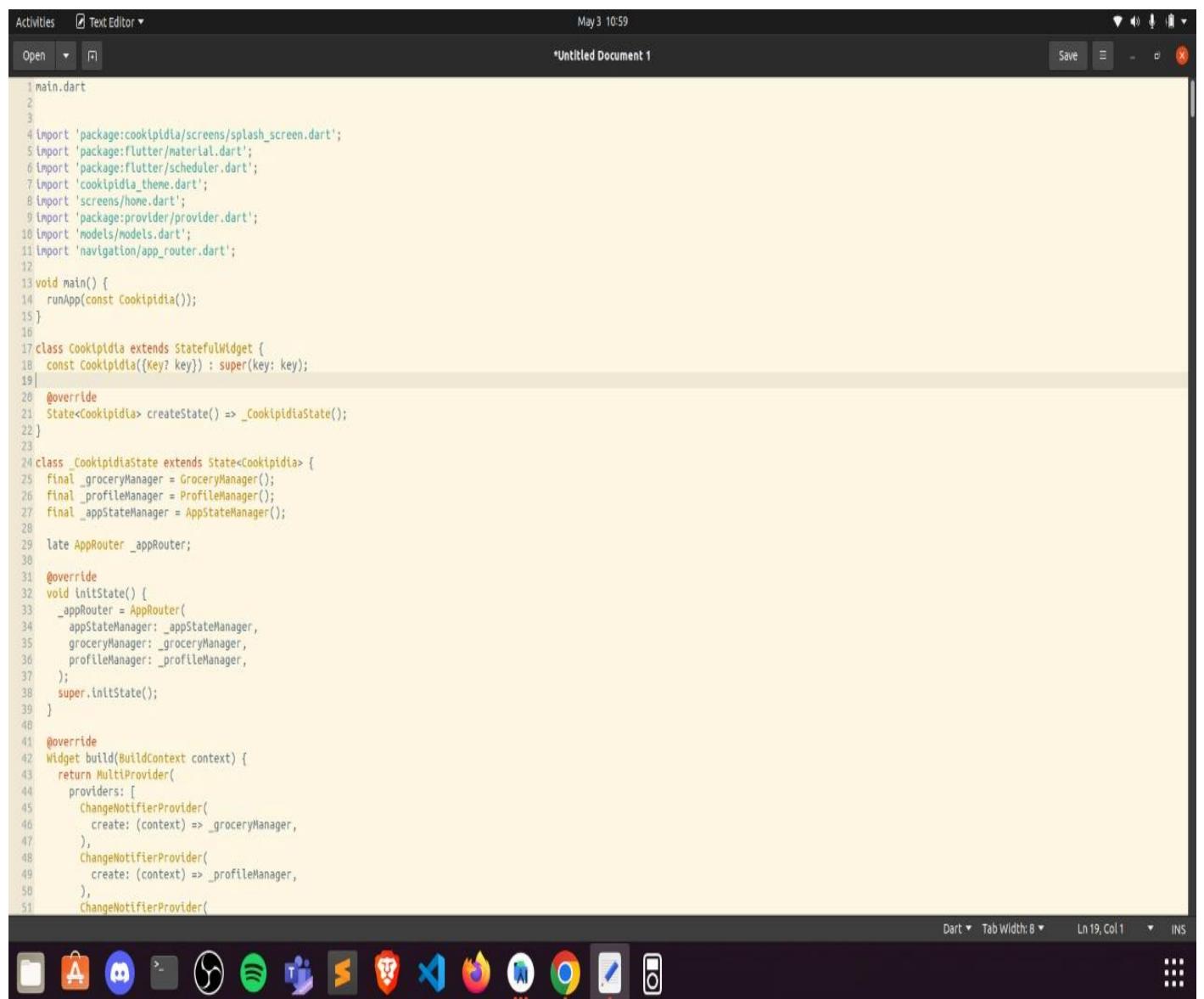
Fig. 36: Android Virtual Device Edit.

Click Okay. The new AVD will appear in the Android Virtual Device Manager screen. Select the new AVD and click Start. Wait for the emulator to boot up and then unlock the emulator screen.

In order to run the app from Eclipse, click Run from the toolbar. And then select Android Application and click OK. Eclipse will install the app on the AVD and starts it.

3.8. Code for Developing the My Cookery Master

The following code was used in developing My Cookery Master. The code snippets are written in dart using flutter SDK and kotlin NDK with database being google firebase and few other data connectors and API's were used which are as follows:



A screenshot of a Text Editor window titled "Untitled Document 1". The window shows Dart code for the "main.dart" file. The code defines the main function and the Cookipidia class, which extends StatefulWidget. It initializes state managers for grocery, profile, and app states, and sets up a multi-provider widget for the grocery and profile managers. The code uses imports for various packages like cookipidia, flutter, provider, and navigation.

```
1 main.dart
2
3
4 import 'package:cookipidia/screens/splash_screen.dart';
5 import 'package:flutter/material.dart';
6 import 'package:flutter/scheduler.dart';
7 import 'cookipidia_theme.dart';
8 import 'screens/home.dart';
9 import 'package:provider/provider.dart';
10 import 'models/models.dart';
11 import 'navigation/app_router.dart';
12
13 void main() {
14   runApp(const Cookipidia());
15 }
16
17 class Cookipidia extends StatelessWidget {
18   const Cookipidia({Key? key}) : super(key: key);
19
20   @override
21   State<Cookipidia> createState() => _CookipidiaState();
22 }
23
24 class _CookipidiaState extends State<Cookipidia> {
25   final _groceryManager = GroceryManager();
26   final _profileManager = ProfileManager();
27   final _appStateManager = AppStateManager();
28
29   late AppRouter _appRouter;
30
31   @override
32   void initState() {
33     _appRouter = AppRouter(
34       appBarManager: _appStateManager,
35       groceryManager: _groceryManager,
36       profileManager: _profileManager,
37     );
38     super.initState();
39   }
40
41   @override
42   Widget build(BuildContext context) {
43     return MultiProvider(
44       providers: [
45         ChangeNotifierProvider(
46           create: (context) => _groceryManager,
47         ),
48         ChangeNotifierProvider(
49           create: (context) => _profileManager,
50         ),
51         ChangeNotifierProvider(
```

Fig. 37: Code for Home Screen for My Cookery Master

A screenshot of a Linux desktop environment. At the top, there's a header bar with 'Activities' and a 'Text Editor' tab. The main area shows an 'Untitled Document 1' containing Dart code for a theme class. The code defines two static TextTheme objects: 'lightTextTheme' and 'darkTextTheme'. Both themes use Google Fonts' Open Sans font. The 'lightTextTheme' has a bodyText1 font size of 14.0, headline1 font size of 32.0, and headline2 font size of 21.0. The 'darkTextTheme' has a bodyText1 font size of 14.0, headline1 font size of 32.0, and headline2 font size of 21.0. The code also includes imports for 'package:google_fonts/google_fonts.dart' and 'package:cookipidia/cockipidia_theme.dart'. Below the code editor is a dock with various application icons. The status bar at the bottom right shows 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'.

```
88 import 'package:google_fonts/google_fonts.dart';
89
90 class CookipidiaTheme {
91   // 1
92   static TextTheme lightTextTheme = TextTheme(
93     bodyText1: GoogleFonts.openSans(
94       fontSize: 14.0,
95       fontWeight: FontWeight.w700,
96       color: Colors.black,
97     ),
98     headline1: GoogleFonts.openSans(
99       fontSize: 32.0,
100      fontWeight: FontWeight.bold,
101      color: Colors.black,
102    ),
103     headline2: GoogleFonts.openSans(
104       fontSize: 21.0,
105       fontWeight: FontWeight.w700,
106       color: Colors.black,
107     ),
108     headline3: GoogleFonts.openSans(
109       fontSize: 16.0,
110       fontWeight: FontWeight.w600,
111       color: Colors.black,
112     ),
113   );
114   static TextTheme darkTextTheme = TextTheme(
115     bodyText1: GoogleFonts.openSans(
116       fontSize: 14.0,
117       fontWeight: FontWeight.w700,
118       color: Colors.white,
119     ),
120     headline1: GoogleFonts.openSans(
121       fontSize: 32.0,
122       fontWeight: FontWeight.bold,
123       color: Colors.white,
124     ),
125     headline2: GoogleFonts.openSans(
126       fontSize: 21.0,
127       fontWeight: FontWeight.w700,
128       color: Colors.white,
129     ),
130     headline3: GoogleFonts.openSans(
131       fontSize: 16.0,
132       fontWeight: FontWeight.w600,
133       color: Colors.white,
134     ),
135   );
136 }
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350 card1.dart
351 import 'package:cookipidia/cockipidia_theme.dart';
352 import 'package:cookipidia/main.dart';
353 import 'package:flutter/material.dart';
354 import 'package:cookipidia/cockipidia_theme.dart';
355
356 import './models/explore_recipe.dart';
357
358 class Card1 extends StatelessWidget {
359   final ExploreRecipe recipe;
360
361   const Card1({
362     Key key,
363     required this.recipe,
364   }) : super(key: key);
365
366   // 2
367   @override
368   Widget build(BuildContext context) {
369   // 3
370     return Center(
371       child: Container(
372         child: Stack(
373           children: [
374             Text(
375               recipe.subtitle,
376               style: CookipidiaTheme.darkTextTheme.bodyText1,
377             ),
378             Positioned(
379               child: Text(
380                 recipe.title,
381                 style: CookipidiaTheme.darkTextTheme.headline2,
382               ),
383               top: 20,
384             ),
385             Positioned(
386               child: Text(
387                 recipe.message,
388                 style: CookipidiaTheme.darkTextTheme.bodyText1,
389               ),
390               bottom: 30,
391               right: 0,
392             )
393           ],
394         ),
395       ),
396     );
397   }
398 }
```

Fig. 38

A screenshot of a Linux desktop environment. At the top, there's a header bar with 'Activities' and a 'Text Editor' tab. The main area shows an 'Untitled Document 1' containing Dart code for a 'Card1' widget. The code imports 'package:cookipidia/cockipidia_theme.dart', 'package:cookipidia/main.dart', and 'package:flutter/material.dart'. It defines a 'Card1' class that extends ' StatelessWidget'. The 'build' method returns a 'Center' widget containing a 'Container' with a 'Stack' of children. The stack contains three text elements: a subtitle at the top, a title in the middle positioned 20 units from the top, and a message at the bottom positioned 30 units from the bottom. The code also includes imports for 'package:google_fonts/google_fonts.dart' and 'package:cookipidia/cockipidia_theme.dart'. Below the code editor is a dock with various application icons. The status bar at the bottom right shows 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'.

```
342   ],
343   ],
344   ],
345   );
346 }
347
348
349
350 card1.dart
351 import 'package:cookipidia/cockipidia_theme.dart';
352 import 'package:cookipidia/main.dart';
353 import 'package:flutter/material.dart';
354 import 'package:cookipidia/cockipidia_theme.dart';
355
356 import './models/explore_recipe.dart';
357
358 class Card1 extends StatelessWidget {
359   final ExploreRecipe recipe;
360
361   const Card1({
362     Key key,
363     required this.recipe,
364   }) : super(key: key);
365
366   // 2
367   @override
368   Widget build(BuildContext context) {
369   // 3
370     return Center(
371       child: Container(
372         child: Stack(
373           children: [
374             Text(
375               recipe.subtitle,
376               style: CookipidiaTheme.darkTextTheme.bodyText1,
377             ),
378             Positioned(
379               child: Text(
380                 recipe.title,
381                 style: CookipidiaTheme.darkTextTheme.headline2,
382               ),
383               top: 20,
384             ),
385             Positioned(
386               child: Text(
387                 recipe.message,
388                 style: CookipidiaTheme.darkTextTheme.bodyText1,
389               ),
390               bottom: 30,
391               right: 0,
392             )
393           ],
394         ),
395       ),
396     );
397   }
398 }
```

Fig. 39

Activities Text Editor ▾

Open ▾

May 3 11:01

*Untitled Document 1

Save

```
429 card2.dart
430 import 'package:flutter/material.dart';
431 import '../models/explore_recipe.dart';
432 import 'author_card.dart';
433 import '../cookipedia_theme.dart';
434
435 class Card2 extends StatelessWidget {
436   final ExploreRecipe recipe;
437
438   const Card2({
439     Key? key,
440     required this.recipe,
441   }) : super(key: key);
442
443   @override
444   Widget build(BuildContext context) {
445     return Center(
446       child: Container(
447         constraints: const BoxConstraints.expand(
448           width: 350,
449           height: 450,
450         ),
451         decoration: BoxDecoration(
452           image: DecorationImage(
453             image: AssetImage(recipe.backgroundImage),
454             fit: BoxFit.cover,
455           ),
456           borderRadius: const BorderRadius.all(Radius.circular(10.0)),
457         ),
458         child: Column(
459           children: [
460             AuthorCard(
461               authorName: recipe.authorName,
462               title: recipe.role,
463               imageProvider: AssetImage(recipe.profileImage),
464             ),
465             Expanded(
466               child: Stack(
467                 children: [
468                   Positioned(
469                     bottom: 16,
470                     right: 16,
471                     child: Text(
472                       recipe.title,
473                       style: CookipediaTheme.lightTextTheme.headline1,
474                     ),
475                   ),
476                   Positioned(
477                     bottom: 70,
478                     left: 16,
479                     child: RotatedBox(
480
481
482
483
484
485
486
487
488 card3.dart
489 import 'package:flutter/material.dart';
490 import '../cookipedia_theme.dart';
491 import '../models/explore_recipe.dart';
492
493 class Card3 extends StatelessWidget {
494   final ExploreRecipe recipe;
495
496   const Card3({
497     Key? key,
498     required this.recipe,
499   }) : super(key: key);
500
501   List<Widget> createTagChips() {
502     final chips = <Widget>[];
503     recipe.tags.take(6).forEach((element) {
504       final chip = Chip(
505         label: Text(
506           element,
507           style: CookipediaTheme.darkTextTheme.bodyText1,
508         ),
509         backgroundColor: Colors.black.withOpacity(0.7),
510       );
511       chips.add(chip);
512     });
513
514     return chips;
515   }
516
517   @override
518   Widget build(BuildContext context) {
519     return Center(
520       child: Container(
521         constraints: const BoxConstraints.expand(
522           width: 350,
523           height: 450,
524         ),
525       ),
526     );
527   }
528 }
```

Dart Tab Width: 8 Ln 19, Col 1 INS

Fig. 40

Activities Text Editor ▾

Open ▾

May 3 11:01

*Untitled Document 1

Save

```
473
474   ),
475   ),
476   ],
477   ],
478   ],
479   ],
480   ],
481   ],
482   ],
483   ],
484   );
485 }
486
487 card3.dart
488 import 'package:flutter/material.dart';
489 import '../cookipedia_theme.dart';
490 import '../models/explore_recipe.dart';
491
492 class Card3 extends StatelessWidget {
493   final ExploreRecipe recipe;
494
495   const Card3({
496     Key? key,
497     required this.recipe,
498   }) : super(key: key);
499
500   List<Widget> createTagChips() {
501     final chips = <Widget>[];
502     recipe.tags.take(6).forEach((element) {
503       final chip = Chip(
504         label: Text(
505           element,
506           style: CookipediaTheme.darkTextTheme.bodyText1,
507         ),
508         backgroundColor: Colors.black.withOpacity(0.7),
509       );
510       chips.add(chip);
511     });
512
513     return chips;
514   }
515
516   @override
517   Widget build(BuildContext context) {
518     return Center(
519       child: Container(
520         constraints: const BoxConstraints.expand(
521           width: 350,
522           height: 450,
523         ),
524       ),
525     );
526   }
527 }
```

Dart Tab Width: 8 Ln 19, Col 1 INS

Fig. 41

A screenshot of a code editor window titled "Text Editor" showing Dart code for a Flutter application. The code defines a class `circle_image.dart` which extends `StatelessWidget`. It contains a large nested `Container` block with various properties like `decoration`, `padding`, and `children` containing `Icon` and `Text` widgets. The code is heavily commented with numbers (e.g., 525, 526, 527, etc.) preceding each line. The status bar at the bottom shows tabs for "Dart", "Tab Width: 8", "Ln 19, Col 1", and "INS".

```
May 3 11:01 *Untitled Document 1 Save □ ×
525     image: DecorationImage(
526       image: AssetImage(recipe.backgroundImage),
527     ),
528   ),
529   borderRadius: const BorderRadius.all(
530     Radius.circular(10.0),
531   ),
532 },
533 ),
534 child: Stack(
535   children: [
536     Container(
537       decoration: BoxDecoration(
538         color: Colors.black.withOpacity(0.6),
539         borderRadius: const BorderRadius.all(Radius.circular(10.0)),
540       ),
541     ),
542     Container(
543       padding: const EdgeInsets.all(16),
544       child: Column(
545         crossAxisAlignment: CrossAxisAlignment.start,
546         children: [
547           const Icon(
548             Icons.book,
549             color: Colors.white,
550             size: 40,
551           ),
552           const SizedBox(height: 8),
553           Text(recipe.title,
554             style: CookipediaTheme.darkTextTheme.headline2),
555           const SizedBox(height: 30),
556         ],
557       ),
558     ),
559   ],
560   Center(
561     child: Wrap(
562       alignment: WrapAlignment.start,
563       spacing: 12,
564       runSpacing: 12,
565       children: createTagChips(),
566     ),
567   ),
568 );
569 }
570 }
571
572 circle_image.dart
573 import 'package:flutter/material.dart';
574 class CircleImage extends StatelessWidget {
575   const CircleImage()
576 }
```

Fig. 42

A screenshot of a code editor window titled "Text Editor" showing Dart code for a Flutter application. The code defines a class `FriendPostListView` which extends `StatelessWidget`. It includes imports for `ImageProvider`, `CircleAvatar`, and various components from the project. The build method returns a `CircleAvatar` with a specific radius and background color. The status bar at the bottom shows tabs for "Dart", "Tab Width: 8", "Ln 19, Col 1", and "INS".

```
May 3 11:01 *Untitled Document 1 Save □ ×
577   final ImageProvider? imageProvider;
578   final double imageRadius;
579 } : super(key: key);
580
581   @override
582   final double imageRadius;
583   final ImageProvider? imageProvider;
584
585   @override
586   Widget build(BuildContext context) {
587     // ...
588     return CircleAvatar(
589       backgroundColor: Colors.white,
590       radius: imageRadius,
591       // ...
592       child: CircleAvatar(
593         radius: imageRadius - 5,
594         backgroundImage: imageProvider,
595       ),
596     );
597   }
598
599 components.dart
600 auth_card.dart';
601 export 'card1.dart';
602 export 'card2.dart';
603 export 'card3.dart';
604 export 'circle_image.dart';
605 export 'today_recipe_list_view.dart';
606 export 'friend_post_tile.dart';
607 export 'friend_post_list_view.dart';
608 export 'recipe_thumbnail.dart';
609 export 'recipes_grid_view.dart';
610 export 'today_recipe_list_view.dart';
611
612 friend_post_list_view.dart
613 import 'package:flutter/material.dart';
614 import '../models/models.dart';
615 import 'components.dart';
616
617 class FriendPostListView extends StatelessWidget {
618   // ...
619   final List<Post> friendPosts;
620
621   const FriendPostListView({
622     Key? key,
623     required this.friendPosts,
624   }) : super(key: key);
625
626   @override
627   Widget build(BuildContext context) {
628     // ...
629   }
630 }
```

Fig. 43

Activities Text Editor ▾

May 3 11:01

*Untitled Document 1

Save

Open

```
002 jj) : super(key: key);
003
004 @override
005 Widget build(BuildContext context) {
006 // ...
007 return Row(
008   mainAxisAlignment: MainAxisAlignment.start,
009   mainAxisSize: MainAxisSize.start,
010   children: [
011 // 2
012   CircleImage(
013     imageProvider: AssetImage(post.profileImageUrl),
014     imageRadius: 28,
015   ),
016 // 3
017   const SizedBox(width: 16),
018 // 4
019   Expanded(
020     child: Column(
021       crossAxisAlignment: CrossAxisAlignment.start,
022       children: [
023 // 5
024         Text(post.comment),
025 // 6
026         Text(
027           '${post.timestamp} mins ago',
028           style: const TextStyle(fontWeight: FontWeight.w700),
029         ),
030       ],
031     ),
032   ),
033 ],
034 );
035 }
036 }
037 }
038 );
039 }
040 }
041 }
042 }
043 }
044 }
045 }
046 }
047 }
048 }
049 }
050 }
051 }
052 }
053 }
054 }
055 }
056 }
057 }
058 }
059 }
060 }
061 }
062 }
063 }
064 }
065 }
066 }
067 }
068 }
069 }
070 }
071 }
072 }
073 }
074 }
075 }
076 }
077 }
078 }
079 }
080 }
081 }
082 }
083 }
084 }
085 }
086 }
087 }
088 }
089 }
090 }
091 }
092 }
093 }
094 }
095 }
096 }
097 }
098 }
099 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
719 }
720 }
721 Import 'package:flutter/material.dart';
722 Import 'package:google_fonts/google_fonts.dart';
723 Import 'package:intl/intl.dart';
724 Import 'package:cookiida/models/grocery_item.dart';
725
726 class GroceryTile extends StatelessWidget {
727 // ...
728   final GroceryItem item;
729
730 // ...
731   final Function(bool)? onComplete;
732 }
```

Dart ▾ Tab Width: 8 ▾ Ln 19, Col 1 ▾ INS



Fig. 44

Activities Text Editor ▾ May 3 11:01 Untitled Document 1 Save

Open

```
735 // 4
737 GroceryTile({
738   Key? key,
739   required this.item,
740   this.onComplete,
741 }) : textDecoration = item.isComplete ? TextDecoration.lineThrough : TextDecoration.none,
742   super(key: key);
743
744 @override
745 Widget build(BuildContext context) {
746   // TODO: Z1: Change this Widget
747   return SizedBox(
748     height: 100.0,
749   );
750
751   child: Row(
752     mainAxisAlignment: MainAxisAlignment.spaceBetween,
753     children: [
754       // 2
755       Container(
756         width: 5.0,
757         color: item.color,
758       ),
759     ],
760   );
761 }
762 // 4
763 Column(
764   mainAxisAlignment: MainAxisAlignment.center,
765   crossAxisAlignment: CrossAxisAlignment.start,
766   children: [
767     // 5
768     Text(
769       item.name,
770       style: GoogleFonts.lato(
771         decoration: textDecoration,
772         fontSize: 21.0,
773         fontWeight: FontWeight.bold),
774     ),
775     const SizedBox(height: 4.0),
776     buildDate(),
777     const SizedBox(height: 4.0),
778     buildImportance(),
779   ],
780 ),
781
782 Row(
783   children: [
784     // 7
785     Text(
```

Dart Tab Width: 8 Ln 19, Col 1 INS

Fig. 45

Activities Text Editor May 3 11:01 *Untitled Document 1 Save

```
786     style: GoogleFonts.lato(
787       decoration: textDecoration,
788       fontSize: 21.0,
789     ),
790   ),
791 },
792 // B
793 buildCheckbox(),
794 ],
795 ],
796 ],
797 ),
798 );
799 }
800
801 Widget buildImportance() {
802   if (item.importance == Importance.low) {
803     return Text('Low', style: GoogleFonts.lato(decoration: textDecoration));
804   } else if (item.importance == Importance.medium) {
805     return Text('Medium',
806       style: GoogleFonts.lato(
807         fontWeight: FontWeight.w800, decoration: textDecoration));
808   } else if (item.importance == Importance.high) {
809     return Text(
810       'High',
811       style: GoogleFonts.lato(
812         color: Colors.red,
813         fontWeight: FontWeight.w900,
814         decoration: textDecoration,
815       ),
816     );
817   } else {
818     throw Exception('This importance type does not exist');
819   }
820 }
821
822 Widget buildDate() {
823   final dateFormatter = DateFormat('EEEE dd h:mm a');
824   final dateString = dateFormatter.format(item.date);
825   return Text(
826     dateString,
827     style: TextStyle(decoration: textDecoration),
828   );
829 }
830
831 Widget buildCheckbox() {
832   return Checkbox(
833     // 1
834     value: item.isComplete,
835     // 2
836     onChanged: onComplete,
837   );
838 }
```

Save

Open

Save

Activities Text Editor May 3 11:01 *Untitled Document 1

Tab Width: 8 ▾ Ln 19, Col 1 INS

Fig. 46

Activities Text Editor Save May 3 11:01 *Untitled Document 1

```
761
762 // 4
763
764 Column(
765   mainAxisAlignment: MainAxisAlignment.center,
766   crossAxisAlignment: CrossAxisAlignment.start,
767   children: [
768     Text(
769       item.name,
770       style: GoogleFonts.lato(
771         decoration: textDecoration,
772         fontSize: 21.0,
773         fontWeight: FontWeight.bold),
774     ),
775     const SizedBox(height: 4.0),
776     buildDate(),
777     const SizedBox(height: 4.0),
778     buildImportance(),
779   ],
780 ),
781
782 Row(
783   children: [
784 // 7
785   Text(
786     item.quantity.toString(),
787     style: GoogleFonts.lato(
788       decoration: textDecoration,
789       fontSize: 21.0,
790     ),
791   ),
792 // 8
793   buildCheckbox(),
794 ],
795 ),
796 ],
797 ),
798 );
799 }
800
801 Widget buildImportance() {
802   if (item.importance == Importance.low) {
803     return Text('Low', style: GoogleFonts.lato(decoration: textDecoration));
804   } else if (item.importance == Importance.medium) {
805     return Text('Medium',
806       style: GoogleFonts.lato(
807         fontWeight: FontWeight.w800, decoration: textDecoration));
808   } else if (item.importance == Importance.high) {
809     return Text(
810       'High',
811       style: GoogleFonts.lato(
```

Fig. 47

```
822     ),
823     decoration: textDecoration,
824   );
825 }
826 } else {
827   throw Exception('This importance type does not exist');
828 }
829 }
830
831 Widget buildDate() {
832   final dateFormatter = DateFormat('MMMM dd h:mm a');
833   final dateString = dateFormatter.format(item.date);
834   return Text(
835     dateString,
836     style: TextStyle(decoration: textDecoration),
837   );
838 }
839
840 Widget buildCheckbox() {
841   return Checkbox(
842     value: item.isComplete,
843     onChanged: onComplete,
844   );
845 }
846
847 recipe_thumbnail.dart
848 import 'package:flutter/material.dart';
849 import '../models/models.dart';
850
851 class RecipeThumbnail extends StatelessWidget {
852   // 1
853   final SimpleRecipe recipe;
854
855   const RecipeThumbnail({
856     Key? key,
857     required this.recipe,
858   }) : super(key: key);
859
860   @override
861   Widget build(BuildContext context) {
862     // 2
863     return Container(
864       padding: const EdgeInsets.all(8),
865     // 3
866     child: Column(
867       crossAxisAlignment: CrossAxisAlignment.start,
868       children: [
869         // 4
870         Expanded(
871           // 5
872             child: ClipRRect(
873               // 6
874               child: Image.asset(
875                 '${recipe.dishImage}',
876                 fit: BoxFit.cover,
877               ),
878               // 7
879               borderRadius: BorderRadius.circular(12),
880             ),
881         ),
882         const SizedBox(height: 10),
883         Text(
884           recipe.title,
885           maxLines: 1,
886           style: Theme.of(context).textTheme.bodyText1,
887         ),
888         Text(
889           recipe.duration,
890           style: Theme.of(context).textTheme.bodyText1,
891         ),
892       ],
893     );
894   }
895 }
```

Fig. 48

```
840
841
842 recipe_thumbnail.dart
843 import 'package:flutter/material.dart';
844 import '../models/models.dart';
845
846 class RecipeThumbnail extends StatelessWidget {
847   // 1
848   final SimpleRecipe recipe;
849
850   const RecipeThumbnail({
851     Key? key,
852     required this.recipe,
853   }) : super(key: key);
854
855   @override
856   Widget build(BuildContext context) {
857     // 2
858     return Container(
859       padding: const EdgeInsets.all(8),
860     // 3
861     child: Column(
862       crossAxisAlignment: CrossAxisAlignment.start,
863       children: [
864         // 4
865         Expanded(
866           // 5
867             child: ClipRRect(
868               // 6
869               child: Image.asset(
870                 '${recipe.dishImage}',
871                 fit: BoxFit.cover,
872               ),
873               // 7
874               borderRadius: BorderRadius.circular(12),
875             ),
876         ),
877         const SizedBox(height: 10),
878         Text(
879           recipe.title,
880           maxLines: 1,
881           style: Theme.of(context).textTheme.bodyText1,
882         ),
883         Text(
884           recipe.duration,
885           style: Theme.of(context).textTheme.bodyText1,
886         ),
887       ],
888     );
889   }
890 }
```

Fig. 49

```
866 // 5
867     child: ClipRect(
868       child: Image.asset(
869         '${recipe.dishImage}',
870         fit: BoxFit.cover,
871       ),
872       borderRadius: BorderRadius.circular(12),
873     ),
874   ),
875   const SizedBox(height: 10),
876 // 7
877   Text(
878     recipe.title,
879     maxLines: 1,
880     style: Theme.of(context).textTheme.bodyText1,
881   ),
882   Text(
883     recipe.duration,
884     style: Theme.of(context).textTheme.bodyText1,
885   ),
886 ],
887 ),
888 );
889 )
890 }
891
892 recipes_grid_view.dart
893 import 'package:flutter/material.dart';
894 import '../components/components.dart';
895 import '../models/models.dart';
896
897 class RecipesGridView extends StatelessWidget {
898 // 1
899   final List<SimpleRecipe> recipes;
900
901   const RecipesGridView({
902     Key key,
903     required this.recipes,
904   }) : super(key: key);
905
906   @override
907   Widget build(BuildContext context) {
908 // 2
909     return Padding(
910       padding: const EdgeInsets.only(
911         left: 16,
912         right: 16,
913         top: 16,
914       ),
915 // 3
916     child: GridView.builder(

```

Fig. 50

```
898 // 1
899   final List<SimpleRecipe> recipes;
900
901   const RecipesGridView({
902     Key key,
903     required this.recipes,
904   }) : super(key: key);
905
906   @override
907   Widget build(BuildContext context) {
908 // 2
909     return Padding(
910       padding: const EdgeInsets.only(
911         left: 16,
912         right: 16,
913         top: 16,
914       ),
915 // 3
916     child: GridView.builder(
917 // 4
918       itemCount: recipes.length,
919 // 5
920       gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
921         crossAxisCount: 2,
922       ),
923       itemBuilder: (context, index) {
924 // 6
925         final simpleRecipe = recipes[index];
926         return RecipeThumbnail(recipe: simpleRecipe);
927       },
928     );
929   );
930 }
931
932
933 today_recipe_list_view.dart
934 import 'package:cookipidia/cookipidia_theme.dart';
935 import 'package:flutter/material.dart';
936
937
938 // 1
939 import '../components/components.dart';
940 import '../models/models.dart';
941
942 class TodayRecipeListView extends StatelessWidget {

```

Fig. 51

```
Activities Text Editor
Open Untitled Document 1
May 3 11:01
919 // 5
920     gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
921         crossAxisCount: 2,
922     ),
923     itemBuilder: (context, index) {
924 // 6
925         final simpleRecipe = recipes[index];
926         return RecipeThumbnail(recipe: simpleRecipe);
927     },
928 );
929 ],
930 ]
931 }
932
933
934 today_recipe_list_view.dart
935 import 'package:cookipedia/cookipedia_theme.dart';
936 import 'package:flutter/material.dart';
937
938 // 1
939 import '../components/components.dart';
940 import '../models/models.dart';
941
942 class TodayRecipeListView extends StatelessWidget {
943 // 2
944     final List<ExploreRecipe> recipes;
945
946     const TodayRecipeListView({
947         Key? key,
948         required this.recipes,
949     }) : super(key: key);
950
951     @override
952     Widget build(BuildContext context) {
953 // 3
954         return Padding(
955             padding: const EdgeInsets.only(
956                 left: 16,
957                 right: 16,
958                 top: 16,
959             ),
960 // 4
961         child: Column(
962             crossAxisAlignment: CrossAxisAlignment.start,
963             children: [
964 // 5
965             Text(
966                 'Recipes of the Day',
967                 style: Theme.of(context).textTheme.headline2,
968             ),
969         ],
970     );
971 // 6
972     const SizedBox(height: 16),
973     Container(
974         height: 400,
975         // 1
976         color: Colors.transparent,
977     );
978 // 3
979     child: ListView.separated(
980         scrollDirection: Axis.horizontal,
981         itemCount: recipes.length,
982     );
983 // 5
984 // 6
985         itemBuilder: (context, index) {
986             final recipe = recipes[index];
987             return buildCard(recipe);
988         },
989     );
990 // 7
991     separatorBuilder: (context, index) {
992         return const SizedBox(width: 16);
993     },
994 );
995 }

```

Fig. 52

```
Activities Text Editor
Open Untitled Document 1
May 3 11:01
945
946     const TodayRecipeListView({
947         Key? key,
948         required this.recipes,
949     }) : super(key: key);
950
951     @override
952     Widget build(BuildContext context) {
953 // 3
954         return Padding(
955             padding: const EdgeInsets.only(
956                 left: 16,
957                 right: 16,
958                 top: 16,
959             ),
960 // 4
961         child: Column(
962             crossAxisAlignment: CrossAxisAlignment.start,
963             children: [
964 // 5
965             Text(
966                 'Recipes of the Day',
967                 style: Theme.of(context).textTheme.headline2,
968             ),
969         ],
970     );
971 // 6
972     const SizedBox(height: 16),
973     Container(
974         height: 400,
975         // 1
976         color: Colors.transparent,
977     );
978 // 3
979     child: ListView.separated(
980         scrollDirection: Axis.horizontal,
981         itemCount: recipes.length,
982     );
983 // 5
984 // 6
985         itemBuilder: (context, index) {
986             final recipe = recipes[index];
987             return buildCard(recipe);
988         },
989     );
990 // 7
991     separatorBuilder: (context, index) {
992         return const SizedBox(width: 16);
993     },
994 );
995 }
```

Fig. 53

A screenshot of a code editor window titled "Text Editor". The code is written in Dart. It includes a Container widget with a height of 400 pixels, a separator builder for a horizontal scrollable list, and a buildCard function that returns different card types based on recipe card type. It also includes an AppCache class with methods for invalidating and caching user data using SharedPreferences.

```
971 // 7
972 Container(
973   height: 400,
974   // 1
975   color: Colors.transparent,
976 // 2
977   child: ListView.separated(
978 // 3
979     scrollDirection: Axis.horizontal,
980     itemCount: recipes.length,
981     itemBuilder: (context, index) {
982 // 4
983       final recipe = recipes[index];
984       return buildCard(recipe);
985     },
986   ),
987   separatorBuilder: (context, index) {
988 // 5
989     return const SizedBox(width: 16);
990   },
991 },
992 ),
993 ),
994 ],
995 ),
996 ),
997 );
998 }
999 Widget buildCard(ExploreRecipe recipe) {
1000   if (recipe.cardType == RecipeCardType.card1) {
1001     return Card1(recipe: recipe);
1002   } else if (recipe.cardType == RecipeCardType.card2) {
1003     return Card2(recipe: recipe);
1004   } else if (recipe.cardType == RecipeCardType.card3) {
1005     return Card3(recipe: recipe);
1006   } else {
1007     throw Exception('This card doesn\'t exist yet');
1008   }
1009 }
1010 ]
1011
1012
1013 app_cache.dart
1014 import 'package:shared_preferences/shared_preferences.dart';
1015
1016 class AppCache {
1017   static const kUser = 'user';
1018   static const kOnboarding = 'onboarding';
1019
1020 Future<void> invalidate() async {
1021   final prefs = await SharedPreferences.getInstance();
1022 }
```

Fig. 54

A screenshot of a code editor window titled "Text Editor". The code is identical to Fig. 54, showing Dart code for a UI component and a shared preferences class.

```
971 // 7
972 Container(
973   height: 400,
974   // 1
975   color: Colors.transparent,
976 // 2
977   child: ListView.separated(
978 // 3
979     scrollDirection: Axis.horizontal,
980     itemCount: recipes.length,
981     itemBuilder: (context, index) {
982 // 4
983       final recipe = recipes[index];
984       return buildCard(recipe);
985     },
986   ),
987   separatorBuilder: (context, index) {
988 // 5
989     return const SizedBox(width: 16);
990   },
991 },
992 ),
993 ),
994 ],
995 ),
996 ),
997 );
998 }
999 Widget buildCard(ExploreRecipe recipe) {
1000   if (recipe.cardType == RecipeCardType.card1) {
1001     return Card1(recipe: recipe);
1002   } else if (recipe.cardType == RecipeCardType.card2) {
1003     return Card2(recipe: recipe);
1004   } else if (recipe.cardType == RecipeCardType.card3) {
1005     return Card3(recipe: recipe);
1006   } else {
1007     throw Exception('This card doesn\'t exist yet');
1008   }
1009 }
1010 ]
1011
1012
1013 app_cache.dart
1014 import 'package:shared_preferences/shared_preferences.dart';
1015
1016 class AppCache {
1017   static const kUser = 'user';
1018   static const kOnboarding = 'onboarding';
1019
1020 Future<void> invalidate() async {
1021   final prefs = await SharedPreferences.getInstance();
1022   await prefs.setBool(kUser, false);
1023   await prefs.setBool(kOnboarding, false);
1024 }
1025
1026 Future<void> cacheUser() async {
1027   final prefs = await SharedPreferences.getInstance();
1028   await prefs.setBool(kUser, true);
1029 }
1030
1031 Future<void> completeOnboarding() async {
1032   final prefs = await SharedPreferences.getInstance();
1033   await prefs.setBool(kOnboarding, true);
1034 }
1035
1036 Future<bool> isUserLoggedIn() async {
1037   final prefs = await SharedPreferences.getInstance();
1038   return prefs.getBool(kUser) ?? false;
1039 }
1040
1041 Future<bool> didCompleteOnboarding() async {
1042   final prefs = await SharedPreferences.getInstance();
1043   return prefs.getBool(kOnboarding) ?? false;
1044 }
1045 }
```

Fig. 55

A screenshot of a Linux desktop environment. At the top, there's a dark header bar with the title "Activities Text Editor" and the date "May 3 11:01". Below the header is an "Untitled Document 1" tab. On the right side of the header, there are icons for "Save", "New", and "Close". The main area is a code editor displaying a Dart file. The code defines a class `CookipidiaTab` and an `AppStateManager` class that extends `ChangeNotifier`. It includes methods for logging in, completing onboarding, and navigating tabs. The code uses `async` and `await` for asynchronous operations like database access. The bottom of the screen shows a dock with various application icons, including a terminal, a file manager, and several browser and productivity apps. A status bar at the bottom right indicates "Dart Tab Width: 8 Ln 19, Col 1 INS".

```
1050 import 'package:flutter/material.dart';
1051
1052 import 'app_cache.dart';
1053
1054 class CookipidiaTab {
1055   static const int explore = 0;
1056   static const int recipes = 1;
1057   static const int toBuy = 2;
1058 }
1059
1060 class AppStateManager extends ChangeNotifier {
1061   bool _initialized = false;
1062   bool _loggedin = false;
1063   bool _onboardingComplete = false;
1064   int _selectedTab = CookipidiaTab.explore;
1065   final _appCache = AppCache();
1066
1067   bool get isInitialized => _initialized;
1068   bool get isloggedin => _loggedin;
1069   bool get isOnboardingComplete => _onboardingComplete;
1070   int get selectedTab => _selectedTab;
1071
1072   void initializeApp() async {
1073     _loggedin = await _appCache.lsUserLoggedIn();
1074     _onboardingComplete = await _appCache.didCompleteOnboarding();
1075
1076     Timer(
1077       const Duration(milliseconds: 2000),
1078       () {
1079         _initialized = true;
1080         notifyListeners();
1081       },
1082     );
1083   }
1084 }
1085
1086 void login(String username, String password) async {
1087   _loggedin = true;
1088   await _appCache.cacheUser();
1089   notifyListeners();
1090 }
1091
1092 void completeOnboarding() async {
1093   _onboardingComplete = true;
1094   await _appCache.completeOnboarding();
1095   notifyListeners();
1096 }
1097
1098 void gotoTab(index) {
1099   _selectedTab = index;
1100   notifyListeners();
1101 }
```

Fig. 56

A screenshot of a Linux desktop environment, similar to Fig. 56. The title bar says "Activities Text Editor" and "May 3 11:01". The main area shows a continuation of the code from Fig. 56, specifically the `AppStateManager` class. It includes methods for logging out, navigating to recipes, and handling grocery item details. The code uses `async` and `await` for database operations. The bottom of the screen shows a dock with various application icons and a status bar indicating "Dart Tab Width: 8 Ln 19, Col 1 INS".

```
1070
1071   Timer(
1072     const Duration(milliseconds: 2000),
1073     () {
1074       _initialized = true;
1075       notifyListeners();
1076     },
1077   );
1078
1079
1080 void login(String username, String password) async {
1081   _loggedin = true;
1082   await _appCache.cacheUser();
1083   notifyListeners();
1084 }
1085
1086 void completeOnboarding() async {
1087   _onboardingComplete = true;
1088   await _appCache.completeOnboarding();
1089   notifyListeners();
1090 }
1091
1092 void gotoTab(index) {
1093   _selectedTab = index;
1094   notifyListeners();
1095 }
1096
1097 void gotoRecipes() {
1098   _selectedTab = CookipidiaTab.recipes;
1099   notifyListeners();
1100 }
1101
1102 void logout() async {
1103   _initialized = false;
1104   _selectedTab = 0;
1105   await _appCache.invalidate();
1106
1107   initializeApp();
1108   notifyListeners();
1109 }
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119 cookipidia_pages.dart
1120 class CookipidiaPages {
1121   static String splashPath = '/splash';
1122   static String loginPath = '/login';
1123   static String onboardPath = '/onboarding';
1124   static String home = '/';
1125   static String groceryItemDetails = '/item';
1126   static String profilePath = '/profile';
1127 }
```

Fig. 57

```
Activities Text Editor
Open Untitled Document 1
May 3 11:01
1103 void goToRecipes() {
1104     _selectedTab = CookipediaTab.recipes;
1105     notifyListeners();
1106 }
1107
1108 void logout() async {
1109     _initialized = false;
1110     _selectedTab = 0;
1111     await _appCache.invalidate();
1112     initializeApp();
1113     notifyListeners();
1114 }
1115 }
1116
1117
1118 cookipedia_pages.dart
1119 class CookipediaPages {
1120     static String splashPath = '/splash';
1121     static String loginPath = '/login';
1122     static String onboardPath = '/onboarding';
1123     static String home = '/';
1124     static String groceryItemDetails = '/item';
1125     static String profilePath = '/profile';
1126     static String raywenderlich = '/raywenderlich';
1127     static String accountPath = '/account';
1128 }
1129
1130
1131
1132 explore_data.dart
1133 import 'models.dart';
1134
1135 class ExploreData {
1136     final List<ExploreRecipe> todayRecipes;
1137     final List<Post> friendPosts;
1138
1139     ExploreData(
1140         this.todayRecipes,
1141         this.friendPosts,
1142     );
1143 }
1144
1145
1146 explore_recipe.dart
1147 part 'ingredient.dart';
1148 part 'instruction.dart';
1149
1150
1151 class RecipeCardType {
1152     static const card1 = 'card1';
1153     static const card2 = 'card2';
1154 }
```

Dart Tab Width: 8 Ln 19, Col 1 INS

Fig. 58

```
Activities Text Editor
Open Untitled Document 1
May 3 11:01
Save
May 3 11:01
1129 ]
1130
1131
1132 explore_data.dart
1133 import 'models.dart';
1134
1135 class ExploreData {
1136     final List<ExploreRecipe> todayRecipes;
1137     final List<Post> friendPosts;
1138
1139     ExploreData(
1140         this.todayRecipes,
1141         this.friendPosts,
1142     );
1143 }
1144
1145
1146 explore_recipe.dart
1147 part 'ingredient.dart';
1148 part 'instruction.dart';
1149
1150
1151 class RecipeCardType {
1152     static const card1 = 'card1';
1153     static const card2 = 'card2';
1154     static const card3 = 'card3';
1155 }
1156
1157 class ExploreRecipe {
1158     String id;
1159     String cardType;
1160     String title;
1161     String subtitle;
1162     String backgroundImage;
1163     String backgroundImageSource;
1164     String message;
1165     String authorName;
1166     String authorRole;
1167     String profileImage;
1168     int durationInMinutes;
1169     String dietType;
1170     int calories;
1171     List<String> tags;
1172     String description;
1173     String source;
1174     List<Ingredients> ingredients;
1175     List<Instruction> instructions;
1176
1177     ExploreRecipe({
1178         required this.id,
1179         required this.cardType,
1180     });
1181 }
```

Dart Tab Width: 8 Ln 19, Col 1 INS

Fig. 59

A screenshot of a code editor window titled "Untitled Document 1". The code is written in Dart and defines a class named `ExploreRecipe`. The class has several properties: `id`, `cardType`, `title`, `subtitle`, `backgroundImage`, `backgroundImageSource`, `message`, `authorName`, `role`, `profileImage`, `durationInMinutes`, `dietType`, `calories`, `tags`, `description`, `source`, `Ingredients`, and `Instructions`. The constructor `ExploreRecipe({})` initializes these properties to empty strings or zero values. A factory method `factory ExploreRecipe.fromJson(Map<String, dynamic> json)` takes a JSON map and creates a new `ExploreRecipe` object by setting its properties from the map. The code editor interface includes tabs for "Activities" and "Text Editor", a toolbar with icons for file operations, and a status bar at the bottom.

```
1156
1157 class ExploreRecipe {
1158   String id;
1159   String cardType;
1160   String title;
1161   String subtitle;
1162   String backgroundImage;
1163   String backgroundImageSource;
1164   String message;
1165   String authorName;
1166   String role;
1167   String profileImage;
1168   int durationInMinutes;
1169   String dietType;
1170   int calories;
1171   List<String> tags;
1172   String description;
1173   String source;
1174   List<Ingredients> ingredients;
1175   List<Instruction> instructions;
1176
1177   ExploreRecipe({
1178     required this.id,
1179     required this.cardType,
1180     required this.title,
1181     required this.subtitle,
1182     this.backgroundImage = '',
1183     this.backgroundImageSource = '',
1184     this.message = '',
1185     this.authorName = '',
1186     this.role = '',
1187     this.profileImage = '',
1188     this.durationInMinutes = 0,
1189     this.dietType = '',
1190     this.calories = 0,
1191     this.tags = const [],
1192     this.description = '',
1193     this.source = '',
1194     this.ingredients = const [],
1195     this.instructions = const [],
1196   });
1197
1198   factory ExploreRecipe.fromJson(Map<String, dynamic> json) {
1199     final ingredients = <Ingredients>[];
1200     final instructions = <Instruction>[];
1201
1202     if (json['ingredients'] != null) {
1203       json['ingredients'].forEach((v) {
1204         ingredients.add(Ingredients.fromJson(v));
1205       });
1206
1207     if (json['instructions'] != null) {
1208       json['instructions'].forEach((v) {
1209         instructions.add(Instruction.fromJson(v));
1210       });
1211     }
1212   }
1213
1214   return ExploreRecipe(
1215     id: json['id'] ?? '',
1216     cardType: json['cardType'] ?? '',
1217     title: json['title'] ?? '',
1218     subtitle: json['subtitle'] ?? '',
1219     backgroundImage: json['backgroundImage'] ?? '',
1220     backgroundImageSource: json['backgroundImageSource'] ?? '',
1221     message: json['message'] ?? '',
1222     authorName: json['authorName'] ?? '',
1223     role: json['role'] ?? '',
1224     profileImage: json['profileImage'] ?? '',
1225     durationInMinutes: json['durationInMinutes'] ?? 0,
1226     dietType: json['dietType'] ?? '',
1227     calories: json['calories'] ?? 0,
1228     tags: json['tags'].cast<String>() ?? [],
1229     description: json['description'] ?? '',
1230     source: json['source'] ?? '',
1231     ingredients: ingredients,
1232   );
1233 }
```

Fig. 60

A screenshot of a code editor window titled "Untitled Document 1". The code is identical to the one in Fig. 60, defining the `ExploreRecipe` class with its properties and factory method. The code editor interface is similar, with tabs for "Activities" and "Text Editor", a toolbar with icons, and a status bar at the bottom.

```
1156
1157 class ExploreRecipe {
1158   String id;
1159   String cardType;
1160   String title;
1161   String subtitle;
1162   String backgroundImage;
1163   String backgroundImageSource;
1164   String message;
1165   String authorName;
1166   String role;
1167   String profileImage;
1168   int durationInMinutes;
1169   String dietType;
1170   int calories;
1171   List<String> tags;
1172   String description;
1173   String source;
1174   List<Ingredients> ingredients;
1175   List<Instruction> instructions;
1176
1177   ExploreRecipe({
1178     required this.id,
1179     required this.cardType,
1180     required this.title,
1181     required this.subtitle,
1182     this.backgroundImage = '',
1183     this.backgroundImageSource = '',
1184     this.message = '',
1185     this.authorName = '',
1186     this.role = '',
1187     this.profileImage = '',
1188     this.durationInMinutes = 0,
1189     this.dietType = '',
1190     this.calories = 0,
1191     this.tags = const [],
1192     this.description = '',
1193     this.source = '',
1194     this.ingredients = const [],
1195     this.instructions = const [],
1196   });
1197
1198   factory ExploreRecipe.fromJson(Map<String, dynamic> json) {
1199     final ingredients = <Ingredients>[];
1200     final instructions = <Instruction>[];
1201
1202     if (json['ingredients'] != null) {
1203       json['ingredients'].forEach((v) {
1204         ingredients.add(Ingredients.fromJson(v));
1205       });
1206
1207     if (json['instructions'] != null) {
1208       json['instructions'].forEach((v) {
1209         instructions.add(Instruction.fromJson(v));
1210       });
1211     }
1212   }
1213
1214   return ExploreRecipe(
1215     id: json['id'] ?? '',
1216     cardType: json['cardType'] ?? '',
1217     title: json['title'] ?? '',
1218     subtitle: json['subtitle'] ?? '',
1219     backgroundImage: json['backgroundImage'] ?? '',
1220     backgroundImageSource: json['backgroundImageSource'] ?? '',
1221     message: json['message'] ?? '',
1222     authorName: json['authorName'] ?? '',
1223     role: json['role'] ?? '',
1224     profileImage: json['profileImage'] ?? '',
1225     durationInMinutes: json['durationInMinutes'] ?? 0,
1226     dietType: json['dietType'] ?? '',
1227     calories: json['calories'] ?? 0,
1228     tags: json['tags'].cast<String>() ?? [],
1229     description: json['description'] ?? '',
1230     source: json['source'] ?? '',
1231     ingredients: ingredients,
1232   );
1233 }
```

Fig. 61

A screenshot of a Linux desktop environment. At the top, there's a header bar with 'Activities' and 'Text Editor'. The main area is a code editor titled 'Untitled Document 1' with the file path 'grocery_item.dart'. The code is a Dart class definition for 'GroceryItem'. It includes a constructor 'GroceryItem' with required parameters: id, name, importance, color, quantity, and date. It also includes a copyWith method for creating new instances. The code editor has syntax highlighting for Dart. Below the code editor is a dock with various application icons. The status bar at the bottom shows 'May 3 11:01', 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'.

```
1208     if (json['instructions'] != null) {
1209       json['instructions'].forEach((v) {
1210         instructions.add(Instruction.fromJson(v));
1211       });
1212     }
1213   }
1214   return ExploreRecipe(
1215     id: json['id'] ?? '',
1216     cardType: json['cardtype'] ?? '',
1217     title: json['title'] ?? '',
1218     subtitle: json['subtitle'] ?? '',
1219     backgroundImage: json['backgroundImage'] ?? '',
1220     backgroundImageSource: json['backgroundImageSource'] ?? '',
1221     message: json['message'] ?? '',
1222     authorName: json['authorName'] ?? '',
1223     role: json['role'] ?? '',
1224     profileImage: json['profileImage'] ?? '',
1225     durationInMinutes: json['durationInMinutes'] ?? 0,
1226     dietType: json['dietype'] ?? '',
1227     calories: json['calories'] ?? 0,
1228     tags: json['tags'].cast<String>() ?? [],
1229     description: json['description'] ?? '',
1230     source: json['source'] ?? '',
1231     ingredients: ingredients,
1232     instructions: instructions,
1233   );
1234 }
1235 }
1236
1237
1238 grocery_item.dart
1239 import 'package:flutter/painting.dart';
1240
1241 // 1
1242 enum Importance {
1243   low,
1244   medium,
1245   high,
1246 }
1247
1248 class GroceryItem {
1249 // 2
1250   final String id;
1251
1252 // 3
1253   final String name;
1254   final Importance importance;
1255   final Color color;
1256   final int quantity;
1257   final DateTime date;
1258   final bool isComplete;
1259 }
```

Fig. 62

A screenshot of a Linux desktop environment, similar to Fig. 62. The code editor window is open with the same file 'grocery_item.dart'. The code has been modified to include a constructor 'GroceryItem' with required parameters: id, name, importance, color, quantity, and date. It also includes a copyWith method for creating new instances. The code editor has syntax highlighting for Dart. Below the code editor is a dock with various application icons. The status bar at the bottom shows 'May 3 11:01', 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'.

```
1234 }
1235 ]
1236
1237
1238 grocery_item.dart
1239 import 'package:flutter/painting.dart';
1240
1241 // 1
1242 enum Importance {
1243   low,
1244   medium,
1245   high,
1246 }
1247
1248 class GroceryItem {
1249 // 2
1250   final String id;
1251
1252 // 3
1253   final String name;
1254   final Importance importance;
1255   final Color color;
1256   final int quantity;
1257   final DateTime date;
1258   final bool isComplete;
1259
1260   GroceryItem({
1261     required this.id,
1262     required this.name,
1263     required this.importance,
1264     required this.color,
1265     required this.quantity,
1266     required this.date,
1267     this.isComplete = false,
1268   });
1269
1270   GroceryItem copyWith({
1271     String? id,
1272     String? name,
1273     Importance? importance,
1274     Color? color,
1275     int? quantity,
1276     DateTime? date,
1277     bool? isComplete,
1278   }) {
1279     return GroceryItem(
1280       id: id ?? this.id,
1281       name: name ?? this.name,
1282       importance: importance ?? this.importance,
1283       color: color ?? this.color,
1284       quantity: quantity ?? this.quantity,
1285     );
1286   }
1287 }
```

Fig. 63

Activities Text Editor

May 3 11:01 *Untitled Document 1 Save

```
1260 GroceryItem({  
1261   required this.id,  
1262   required this.name,  
1263   required this.importance,  
1264   required this.color,  
1265   required this.quantity,  
1266   required this.date,  
1267   this.isComplete = false,  
1268 });  
1269  
1270 GroceryItem copyWith({  
1271   String? id,  
1272   String? name,  
1273   Importance? importance,  
1274   Color? color,  
1275   int? quantity,  
1276   DateTime? date,  
1277   bool? isComplete,  
1278 }) {  
1279   return GroceryItem(  
1280     id: id ?? this.id,  
1281     name: name ?? this.name,  
1282     importance: importance ?? this.importance,  
1283     color: color ?? this.color,  
1284     quantity: quantity ?? this.quantity,  
1285     date: date ?? this.date,  
1286     isComplete: isComplete ?? this.isComplete,  
1287   );  
1288 }  
1289  
1290 grocery_manager.dart  
1291  
1292 import 'package:flutter/material.dart';  
1293  
1294 import 'grocery_item.dart';  
1295  
1296 class GroceryManager extends ChangeNotifier {  
1297   final _groceryItems = <GroceryItem>[];  
1298   int _selectedIndex = -1;  
1299   bool _createNewItem = false;  
1300  
1301 List<GroceryItem> get groceryItems => List.unmodifiable(_groceryItems);  
1302 int get selectedIndex => _selectedIndex;  
1303 GroceryItem? get selectedGroceryitem =>  
1304   _selectedIndex != -1 ? _groceryItems[_selectedIndex] : null;  
1305 bool get isCreatingNewItem => _createNewItem;  
1306  
1307 void createNewItem() {  
1308   _createNewItem = true;  
1309   notifyListeners();  
1310 }
```

Save Tab Width: 8 Ln 19, Col 1 INS

Fig. 64

Activities Text Editor May 3 11:01 *Untitled Document 1 Save

```
1286     });
1287   );
1288 }
1289 }
1290
1291 grocery_manager.dart
1292 import 'package:flutter/material.dart';
1293
1294 import 'grocery_item.dart';
1295
1296 class GroceryManager extends ChangeNotifier {
1297   final _groceryItems = <GroceryItem>[];
1298   int _selectedIndex = -1;
1299   bool _createNewItem = false;
1300
1301   List<GroceryItem> get groceryItems => List.unmodifiable(_groceryItems);
1302   int get selectedIndex => _selectedIndex;
1303   GroceryItem? get selectedGroceryItem =>
1304     _selectedIndex != -1 ? _groceryItems[_selectedIndex] : null;
1305   bool get isCreatingNewItem => _createNewItem;
1306
1307   void createNewItem() {
1308     _createNewItem = true;
1309     notifyListeners();
1310   }
1311
1312   void deleteItem(int index) {
1313     _groceryItems.removeAt(index);
1314     notifyListeners();
1315   }
1316
1317   void groceryItemTapped(int index) {
1318     _selectedIndex = index;
1319     _createNewItem = false;
1320     notifyListeners();
1321   }
1322
1323   void addNewItem(GroceryItem item) {
1324     _groceryItems.add(item);
1325     _createNewItem = false;
1326     notifyListeners();
1327   }
1328
1329   void updateItem(GroceryItem item, int index) {
1330     _groceryItems[index] = item;
1331     _selectedIndex = -1;
1332     _createNewItem = false;
1333     notifyListeners();
1334   }
1335 }
1336 void onCompleteItem(int index, bool channel, f
```

Fig. 65

```
Activities Text Editor
Open Untitled Document 1 Save
May 3 11:01
1313 void deleteItem(int index) {
1314     _groceryItems.removeAt(index);
1315     notifyListeners();
1316 }
1317
1318 void groceryItemTapped(int index) {
1319     _selectedIndex = index;
1320     _createNewItem = false;
1321     notifyListeners();
1322 }
1323
1324 void addItem(GroceryItem item) {
1325     _groceryItems.add(item);
1326     _createNewItem = false;
1327     notifyListeners();
1328 }
1329
1330 void updateItem(GroceryItem item, int index) {
1331     _groceryItems[index] = item;
1332     _selectedIndex = -1;
1333     _createNewItem = false;
1334     notifyListeners();
1335 }
1336
1337 void completeItem(int index, bool change) {
1338     final item = _groceryItems[index];
1339     _groceryItems[index] = item.copyWith(isComplete: change);
1340     notifyListeners();
1341 }
1342
1343
1344 ingredient.dart
1345 part of 'explore_recipe.dart';
1346
1347 class Ingredients {
1348     String imageUrl;
1349     String title;
1350     String source;
1351
1352     Ingredients({
1353         required this.imageUrl,
1354         required this.title,
1355         required this.source,
1356     });
1357
1358     factory Ingredients.fromJson(Map<String, dynamic> json) {
1359         return Ingredients(
1360             imageUrl: json['imageUrl'] ?? '',
1361             title: json['title'] ?? '',
1362             source: json['source'] ?? '',
1363         );
1364     }
1365 }
1366
1367 instruction.dart
1368 part of 'explore_recipe.dart';
1369
1370 class Instruction {
1371     String imageUrl;
1372     String description;
1373     int durationInMinutes;
1374
1375     Instruction({
1376         required this.imageUrl,
1377         required this.description,
1378         required this.durationInMinutes,
1379     });
1380
1381     factory Instruction.fromJson(Map<String, dynamic> json) {
1382         return Instruction(
1383             imageUrl: json['imageUrl'] ?? '',
1384             description: json['description'] ?? '',
1385             durationInMinutes: json['durationInMinutes'] ?? '',
1386         );
1387     }
1388 }
```

Fig. 66

```
Activities Text Editor
Open Untitled Document 1 Save
May 3 11:01
1339     _groceryItems[index] = item.copyWith(isComplete: change);
1340     notifyListeners();
1341 }
1342
1343
1344 ingredient.dart
1345 part of 'explore_recipe.dart';
1346
1347 class Ingredients {
1348     String imageUrl;
1349     String title;
1350     String source;
1351
1352     Ingredients({
1353         required this.imageUrl,
1354         required this.title,
1355         required this.source,
1356     });
1357
1358     factory Ingredients.fromJson(Map<String, dynamic> json) {
1359         return Ingredients(
1360             imageUrl: json['imageUrl'] ?? '',
1361             title: json['title'] ?? '',
1362             source: json['source'] ?? '',
1363         );
1364     }
1365 }
1366
1367 instruction.dart
1368 part of 'explore_recipe.dart';
1369
1370 class Instruction {
1371     String imageUrl;
1372     String description;
1373     int durationInMinutes;
1374
1375     Instruction({
1376         required this.imageUrl,
1377         required this.description,
1378         required this.durationInMinutes,
1379     });
1380
1381     factory Instruction.fromJson(Map<String, dynamic> json) {
1382         return Instruction(
1383             imageUrl: json['imageUrl'] ?? '',
1384             description: json['description'] ?? '',
1385             durationInMinutes: json['durationInMinutes'] ?? '',
1386         );
1387     }
1388 }
```

Fig. 67

Activities Text Editor ▾

May 3 11:01

*Untitled Document 1

Save    

```
342
343     ],
344   ),
345 );
346 }
347 }
348 }
349
350 card1.dart
351 import 'package:cookipidia/cookipidia_theme.dart';
352 import 'package:cookipidia/main.dart';
353 import 'package:flutter/material.dart';
354 import 'package:cookipidia/cookipidia_theme.dart';
355
356 import '../models/explore_recipe.dart';
357
358 class Card1 extends StatelessWidget {
359   final ExploreRecipe recipe;
360
361   const Card1({
362     Key? key,
363     required this.recipe,
364   }) : super(key: key);
365
366   // 2
367   @override
368   Widget build(BuildContext context) {
369     // 3
370     return Center(
371       child: Container(
372         child: Stack(
373           children: [
374             Text(
375               recipe.subtitle,
376               style: CookipidiaTheme.darkTextTheme.bodyText1,
377             ),
378             Positioned(
379               child: Text(
380                 recipe.title,
381                 style: CookipidiaTheme.darkTextTheme.headline2,
382               ),
383               top: 20,
384             ),
385             Positioned(
386               child: Text(
387                 recipe.message,
388                 style: CookipidiaTheme.darkTextTheme.bodyText1,
389               ),
390               bottom: 30,
391               right: 0,
392             )
393           ],
394         ),
395       ),
396     );
397   }
398 }
```

Dart  Tab Width: 8  Ln 19, Col 1  INS 

Fig. 68

Activities Text Editor ▾ May 3 11:01 Untitled Document 1 Save

Open

```
420
421 card2.dart
422 import 'package:flutter/material.dart';
423 import './models/explore_recipe.dart';
424 import 'author_card.dart';
425 import './cookipidia_theme.dart';
426
427 class Card2 extends StatelessWidget {
428   final ExploreRecipe recipe;
429
430   const Card2({
431     Key? key,
432     required this.recipe,
433   }) : super(key: key);
434   @override
435   Widget build(BuildContext context) {
436     return Center(
437       child: Container(
438         constraints: const BoxConstraints.expand(
439           width: 350,
440           height: 450,
441         ),
442         decoration: BoxDecoration(
443           image: DecorationImage(
444             image: AssetImage(recipe.backgroundImage),
445             fit: BoxFit.cover,
446           ),
447           borderRadius: const BorderRadius.all(Radius.circular(10.0)),
448         ),
449         child: Column(
450           children: [
451             AuthorCard(
452               authorName: recipe.authorName,
453               title: recipe.role,
454               imageProvider: AssetImage(recipe.profileImage),
455             ),
456             Expanded(
457               child: Stack(
458                 children: [
459                   Positioned(
460                     bottom: 16,
461                     right: 16,
462                     child: Text(
463                       recipe.title,
464                       style: CookipidiaTheme.lightTextTheme.headline1,
465                     ),
466                   ),
467                   Positioned(
468                     bottom: 70,
469                     left: 16,
470                     child: RotatedBox(
471
```

Dart Tab Width:8 Ln 19, Col 1 INS

File Edit View Insert Run Project Tools Help

Flutter Dart DevTools VS Code GitHub

Fig. 69

A screenshot of a Linux desktop environment showing a text editor window titled "Untitled Document 1". The code is written in Dart and defines a class named Card3. The code includes imports for flutter/material.dart, cookipedia_theme.dart, and models/explore_recipe.dart. It also uses CookipediaTheme and ExploreRecipe. The class Card3 extends StatelessWidget and contains methods for creating tag chips and building the widget's UI. The UI includes a Container with BoxDecoration, a Column with CrossAxisAlignment.start, and a Wrap alignment. The code is annotated with line numbers from 473 to 513.

```
473     recipe.subtitle,
474     style: CookipediaTheme.lightTextTheme.headline1,
475   ),
476   ],
477   ],
478   ],
479   ],
480   ],
481   ],
482   ],
483   );
484 }
485 }
486 }
487
488 card3.dart
489 import 'package:flutter/material.dart';
490 import '../cookipedia_theme.dart';
491 import '../models/explore_recipe.dart';
492
493 class Card3 extends StatelessWidget {
494   final ExploreRecipe recipe;
495
496   const Card3({
497     Key? key,
498     required this.recipe,
499   }) : super(key: key);
500
501   List<Widget> createTagChips() {
502     final chips = <Widget>[];
503     recipe.tags.take(6).forEach((element) {
504       final chip = Chip(
505         label: Text(
506           element,
507           style: CookipediaTheme.darkTextTheme.bodyText1,
508         ),
509         backgroundColor: Colors.black.withOpacity(0.7),
510       );
511       chips.add(chip);
512     });
513
514     return chips;
515   }
516   @override
517   Widget build(BuildContext context) {
518     return Center(
519       child: Container(
520         constraints: const BoxConstraints.expand(
521           width: 350,
522           height: 450,
523         ),
524         decoration: DecorationImage(
525           image: AssetImage(recipe.backgroundImage),
526           fit: BoxFit.cover,
527         ),
528         borderRadius: const BorderRadius.all(
529           Radius.circular(10.0),
530         ),
531       ),
532       child: Stack(
533         children: [
534           Container(
535             decoration: BoxDecoration(
536               color: Colors.black.withOpacity(0.6),
537               borderRadius: const BorderRadius.all(Radius.circular(10.0)),
538             ),
539           ),
540           Container(
541             padding: const EdgeInsets.all(16),
542             child: Column(
543               crossAxisAlignment: CrossAxisAlignment.start,
544               children: [
545                 const Icon(
546                   Icons.book,
547                   color: Colors.white,
548                   size: 40,
549                 ),
550                 const SizedBox(height: 8),
551                 Text(recipe.title,
552                   style: CookipediaTheme.darkTextTheme.headline2),
553                 const SizedBox(height: 30),
554               ],
555             ),
556           ),
557           Center(
558             child: Wrap(
559               alignment: WrapAlignment.start,
560               spacing: 12,
561               runSpacing: 12,
562               children: createTagChips(),
563             ),
564           ),
565         ],
566       ),
567     );
568   }
569 }
570
571 circle_image.dart
572 import 'package:flutter/material.dart';
573 class CircleImage extends StatelessWidget {
574   const CircleImage({
```

Fig. 70

A screenshot of a Linux desktop environment showing a text editor window titled "Untitled Document 1". The code is written in Dart and defines a class named CircleImage. It imports flutter/material.dart. The class extends StatelessWidget and contains a single constructor. The constructor takes a parameter named image and initializes a CircleAvatar with that image. The code is annotated with line numbers from 525 to 578.

```
525   image: DecorationImage(
526     image: AssetImage(recipe.backgroundImage),
527     fit: BoxFit.cover,
528   ),
529   borderRadius: const BorderRadius.all(
530     Radius.circular(10.0),
531   ),
532   ),
533   child: Stack(
534     children: [
535       Container(
536         decoration: BoxDecoration(
537           color: Colors.black.withOpacity(0.6),
538           borderRadius: const BorderRadius.all(Radius.circular(10.0)),
539         ),
540       ),
541       Container(
542         padding: const EdgeInsets.all(16),
543         child: Column(
544           crossAxisAlignment: CrossAxisAlignment.start,
545           children: [
546             const Icon(
547               Icons.book,
548               color: Colors.white,
549               size: 40,
550             ),
551             const SizedBox(height: 8),
552             Text(recipe.title,
553               style: CookipediaTheme.darkTextTheme.headline2),
554             const SizedBox(height: 30),
555           ],
556         ),
557       ),
558       Center(
559         child: Wrap(
560           alignment: WrapAlignment.start,
561           spacing: 12,
562           runSpacing: 12,
563           children: createTagChips(),
564         ),
565       ),
566     ],
567   );
568 }
569
570 circle_image.dart
571 import 'package:flutter/material.dart';
572 class CircleImage extends StatelessWidget {
573   const CircleImage({
```

Fig. 71

A screenshot of a Flutter IDE interface. The top bar shows 'Activities' and 'Text Editor'. The status bar indicates 'May 3 11:01' and 'Untitled Document 1'. The main area contains Dart code for the 'FriendPostListView' class. The code includes imports for 'author_card.dart', 'card1.dart', 'card2.dart', 'card3.dart', 'circle_image.dart', 'today_recipe_list_view.dart', 'friend_post_tile.dart', 'friend_post_list_view.dart', 'models.dart', and 'components.dart'. It defines a class 'FriendPostListView' that extends 'StatelessWidget'. The build method returns a 'CircleAvatar' with a white background and a radius of 20, containing another 'CircleAvatar' with a radius of 5 and a background image from 'imageProvider'. The bottom of the screen shows a toolbar with various icons for file operations and navigation.

```
578     this.imageRadius = 20,
579   ) : super(key: key);
580 
581   final double imageRadius;
582   final ImageProvider? imageProvider;
583 
584   @override
585   Widget build(BuildContext context) {
586     // ...
587     return CircleAvatar(
588       backgroundColor: Colors.white,
589       radius: imageRadius,
590     );
591     child: CircleAvatar(
592       radius: imageRadius - 5,
593       backgroundImage: imageProvider,
594     ),
595   );
596 }
597 }
598 
599 components.dart
600 export 'author_card.dart';
601 export 'card1.dart';
602 export 'card2.dart';
603 export 'card3.dart';
604 export 'circle_image.dart';
605 export 'today_recipe_list_view.dart';
606 export 'friend_post_tile.dart';
607 export 'friend_post_list_view.dart';
608 export 'recipe_thumbnail.dart';
609 export 'recipes_grid_view.dart';
610 export 'today_recipe_list_view.dart';
611 
612 friend_post_list_view.dart
613 import 'package:flutter/material.dart';
614 import '../models/models.dart';
615 import 'components.dart';
616 
617 class FriendPostListView extends StatelessWidget {
618   // ...
619   final List<Post> friendPosts;
620 
621   const FriendPostListView({
622     Key? key,
623     required this.friendPosts,
624   }) : super(key: key);
625 
626   @override
627   Widget build(BuildContext context) {
628     // ...
629     padding: const EdgeInsets.only(
630       left: 16,
631       right: 16,
632       top: 0,
633     ),
634     ),
635   );
636     child: Column(
637       crossAxisAlignment: CrossAxisAlignment.start,
638       children: [
639       // ...
640       Text('Social Chefs ! ', style: Theme.of(context).textTheme.headline1),
641       // ...
642       const SizedBox(height: 16),
643       // TODO: Add PostListTile here
644       ListView.separated(
645         // ...
646         primary: false,
647         // ...
648         physics: const NeverScrollableScrollPhysics(),
649       );
650       shrinkWrap: true,
651       scrollDirection: Axis.vertical,
652       itemCount: friendPosts.length,
653       itemBuilder: (context, index) {
654       );
655       final post = friendPosts[index];
656       return FriendPostTitle(post: post);
657     ],
658     separatorBuilder: (context, index) {
659       );
660       return const SizedBox(height: 16);
661     },
662   ),
663   const SizedBox(height: 16),
664   ],
665   ),
666   );
667 );
668 ]
669 }
670 friend_post_tile.dart
671 import 'package:flutter/material.dart';
672 import '../components/components.dart';
673 import '../models/models.dart';
674 
675 class FriendPostTitle extends StatelessWidget {
676   final Post post;
677 
678   const FriendPostTitle({
679     Key? key,
680   }) : super(key: key);
```

Fig. 72

A screenshot of a Flutter IDE interface. The top bar shows 'Activities' and 'Text Editor'. The status bar indicates 'May 3 11:01' and 'Untitled Document 1'. The main area contains Dart code for the 'FriendPostTitle' class. The code includes imports for 'post.dart', 'material.dart', 'components.dart', and 'models.dart'. It defines a class 'FriendPostTitle' that extends ' StatelessWidget'. The build method takes a 'Post' parameter and returns a 'Text' widget with the title 'Social Chefs ! '. The bottom of the screen shows a toolbar with various icons for file operations and navigation.

```
630   padding: const EdgeInsets.only(
631     left: 16,
632     right: 16,
633     top: 0,
634   ),
635   );
636   child: Column(
637     crossAxisAlignment: CrossAxisAlignment.start,
638     children: [
639     // ...
640     Text('Social Chefs ! ', style: Theme.of(context).textTheme.headline1),
641     // ...
642     const SizedBox(height: 16),
643     // TODO: Add PostListTile here
644     ListView.separated(
645       // ...
646       primary: false,
647       // ...
648       physics: const NeverScrollableScrollPhysics(),
649     );
650     shrinkWrap: true,
651     scrollDirection: Axis.vertical,
652     itemCount: friendPosts.length,
653     itemBuilder: (context, index) {
654     );
655     final post = friendPosts[index];
656     return FriendPostTitle(post: post);
657   ],
658   separatorBuilder: (context, index) {
659     );
660     return const SizedBox(height: 16);
661   },
662   ),
663   const SizedBox(height: 16),
664   ],
665   ),
666   );
667 );
668 ]
669 }
670 post.dart
671 import 'package:flutter/material.dart';
672 import '../components/components.dart';
673 import '../models/models.dart';
674 
675 class FriendPostTitle extends StatelessWidget {
676   final Post post;
```

Fig. 73

Activities Text Editor

May 3 11:01 *Untitled Document 1 Save

```
002     j) : super(key: key);
003
004     @override
005     Widget build(BuildContext context) {
006     // ...
007     return Row(
008       mainAxisAlignment: MainAxisAlignment.start,
009       mainAxisSize: MainAxisSize.start,
010       children: [
011         // 2
012         CircleImage(
013           imageProvider: AssetImage(post.profileImageUrl),
014           imageRadius: 20,
015         ),
016         // 3
017         const SizedBox(width: 10),
018         // 4
019         Expanded(
020           child: Column(
021             mainAxisAlignment: MainAxisAlignment.start,
022             children: [
023               // 5
024               Text(post.comment),
025               // 6
026               Text(
027                 '${post.timestamp} mins ago',
028                 style: const TextStyle(fontWeight: FontWeight.w700),
029               ),
030             ],
031           ),
032         ),
033       ],
034     );
035   }
036 }
037 }
038 }
039 }
040 }
041 }
042 }
043 }
044 }
045 }
046 }
047 }
048 }
049 }
050 }
051 }
052 }
053 }
054 }
055 }
056 }
057 }
058 }
059 }
060 }
061 }
062 }
063 }
064 }
065 }
066 }
067 }
068 }
069 }
070 }
071 }
072 }
073 }
074 }
075 }
076 }
077 }
078 }
079 }
080 }
081 }
082 }
083 }
084 }
085 }
086 }
087 }
088 }
089 }
090 }
091 }
092 }
093 }
094 }
095 }
096 }
097 }
098 }
099 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
```

Fig. 74

Fig. 75

Activities Text Editor May 3 11:01 *Untitled Document 1 Save

```
786     style: GoogleFonts.lato(
787       decoration: textDecoration,
788       fontSize: 21.0,
789     ),
790   ),
791 },
792 // B
793 buildCheckbox(),
794 ],
795 ],
796 ],
797 ),
798 );
799 }
800
801 Widget buildImportance() {
802   if (item.importance == Importance.low) {
803     return Text('Low', style: GoogleFonts.lato(decoration: textDecoration));
804   } else if (item.importance == Importance.medium) {
805     return Text('Medium',
806       style: GoogleFonts.lato(
807         fontWeight: FontWeight.w800, decoration: textDecoration));
808   } else if (item.importance == Importance.high) {
809     return Text(
810       'High',
811       style: GoogleFonts.lato(
812         color: Colors.red,
813         fontWeight: FontWeight.w900,
814         decoration: textDecoration,
815       ),
816     );
817   } else {
818     throw Exception('This importance type does not exist');
819   }
820 }
821
822 Widget buildDate() {
823   final dateFormatter = DateFormat('EEEE dd h:mm a');
824   final dateString = dateFormatter.format(item.date);
825   return Text(
826     dateString,
827     style: TextStyle(decoration: textDecoration),
828   );
829 }
830
831 Widget buildCheckbox() {
832   return Checkbox(
833     // 1
834     value: item.isComplete,
835     // 2
836     onChanged: onComplete,
837   );
838 }
```

Save

Open

Save

Activities Text Editor May 3 11:01 *Untitled Document 1

Tab Width: 8 ▾ Ln 19, Col 1 INS

Fig. 76

Activities Text Editor Save May 3 11:01 *Untitled Document 1

```
761
762 // 4
763
764 Column(
765   mainAxisAlignment: MainAxisAlignment.center,
766   crossAxisAlignment: CrossAxisAlignment.start,
767   children: [
768     Text(
769       item.name,
770       style: GoogleFonts.lato(
771         decoration: textDecoration,
772         fontSize: 21.0,
773         fontWeight: FontWeight.bold),
774     ),
775     const SizedBox(height: 4.0),
776     buildDate(),
777     const SizedBox(height: 4.0),
778     buildImportance(),
779   ],
780 ),
781
782 Row(
783   children: [
784 // 7
785   Text(
786     item.quantity.toString(),
787     style: GoogleFonts.lato(
788       decoration: textDecoration,
789       fontSize: 21.0,
790     ),
791   ),
792 // 8
793   buildCheckbox(),
794 ],
795 ),
796 ],
797 ),
798 );
799 }
800
801 Widget buildImportance() {
802   if (item.importance == Importance.low) {
803     return Text('Low', style: GoogleFonts.lato(decoration: textDecoration));
804   } else if (item.importance == Importance.medium) {
805     return Text('Medium',
806       style: GoogleFonts.lato(
807         fontWeight: FontWeight.w800, decoration: textDecoration));
808   } else if (item.importance == Importance.high) {
809     return Text(
810       'High',
811       style: GoogleFonts.lato(
```

Fig. 77

A screenshot of a Linux desktop environment. At the top, there's a header bar with the title "Activities" and "Text Editor". The main area shows an untitled document with Dart code. The code defines a class `RecipeThumbnail` that extends `StatelessWidget`. It imports `package:flutter/material.dart` and `../models/models.dart`. The class has a constructor `const RecipeThumbnail({Key? key, required this.recipe})` and an overridden `build` method. The `build` method returns a `Container` with padding and a `Column` child. The `Column` has a `crossAxisAlignment: CrossAxisAlignment.start` and a list of children. The first child is an `Expanded` widget containing a `ClipRRect` with a `BoxFit.cover` fit, a `borderRadius: BorderRadius.circular(12)`, and a `Image.asset` child with the path `\${recipe.dishImage}`. The second child is a `const SizedBox(height: 10)`. The third child is a `Text` widget with `recipe.title` as its content, with a `maxLines: 1` constraint and a `style: Theme.of(context).textTheme.bodyText1` applied. The fourth child is another `Text` widget with `recipe.duration` as its content, with a `style: Theme.of(context).textTheme.bodyText1` applied. The fifth child is a blank line. The bottom right of the window shows status information: "Dart", "Tab Width: 8", "Ln 19, Col 1", and "INS". Below the code editor is a dock with various application icons.

```
May 3 11:01
*Untitled Document 1
Save
Open ▾
813     decoration: textDecoration,
814   ),
815   ),
816 }
817 } else {
818   throw Exception('This importance type does not exist');
819 }
820 }
821
822 Widget buildDate() {
823   final dateFormatter = DateFormat('MMMM dd h:mm a');
824   final dateString = dateFormatter.format(item.date);
825   return Text(
826     dateString,
827     style: TextStyle(decoration: textDecoration),
828   );
829 }
830
831 Widget buildCheckbox() {
832   return Checkbox(
833 // 1
834     value: item.isComplete,
835 // 2
836     onChanged: onComplete,
837   );
838 }
839 }
840
841 recipe_thumbnail.dart
842 import 'package:flutter/material.dart';
843 import '../models/models.dart';
844
845 class RecipeThumbnail extends StatelessWidget {
846 // 1
847   final SimpleRecipe recipe;
848
849   const RecipeThumbnail({
850     Key? key,
851     required this.recipe,
852   }) : super(key: key);
853
854   @override
855   Widget build(BuildContext context) {
856 // 2
857   return Container(
858     padding: const EdgeInsets.all(0),
859 // 3
860     child: Column(
861       crossAxisAlignment: CrossAxisAlignment.start,
862       children: [
863
864 // 4
865       Expanded(
866 // 5
867         child: ClipRRect(
868           child: Image.asset(
869             '${recipe.dishImage}',
870             fit: BoxFit.cover,
871           ),
872           borderRadius: BorderRadius.circular(12),
873         ),
874       ),
875       const SizedBox(height: 10),
876 // 7
877       Text(
878         recipe.title,
879         maxLines: 1,
880         style: Theme.of(context).textTheme.bodyText1,
881       ),
882       Text(
883         recipe.duration,
884         style: Theme.of(context).textTheme.bodyText1,
885       ),
886     ],
887   );
888 }
889 }
```

Fig. 78

A screenshot of a Linux desktop environment, similar to Fig. 78. It shows an untitled document with Dart code. The code defines a class `RecipeThumbnail` that extends `StatelessWidget`. It imports `package:flutter/material.dart` and `../models/models.dart`. The class has a constructor `const RecipeThumbnail({Key? key, required this.recipe})` and an overridden `build` method. The `build` method returns a `Container` with padding and a `Column` child. The `Column` has a `crossAxisAlignment: CrossAxisAlignment.start` and a list of children. The first child is an `Expanded` widget containing a `ClipRRect` with a `BoxFit.cover` fit, a `borderRadius: BorderRadius.circular(12)` and a `Image.asset` child with the path `\${recipe.dishImage}`. The second child is a `const SizedBox(height: 10)`. The third child is a `Text` widget with `recipe.title` as its content, with a `maxLines: 1` constraint and a `style: Theme.of(context).textTheme.bodyText1` applied. The fourth child is another `Text` widget with `recipe.duration` as its content, with a `style: Theme.of(context).textTheme.bodyText1` applied. The fifth child is a blank line. The bottom right of the window shows status information: "Dart", "Tab Width: 8", "Ln 19, Col 1", and "INS". Below the code editor is a dock with various application icons.

```
May 3 11:01
*Untitled Document 1
Save
Open ▾
840
841
842 recipe_thumbnail.dart
843 import 'package:flutter/material.dart';
844 import '../models/models.dart';
845
846 class RecipeThumbnail extends StatelessWidget {
847 // 1
848   final SimpleRecipe recipe;
849
850   const RecipeThumbnail({
851     Key? key,
852     required this.recipe,
853   }) : super(key: key);
854
855   @override
856   Widget build(BuildContext context) {
857 // 2
858   return Container(
859     padding: const EdgeInsets.all(8),
860 // 3
861     child: Column(
862       crossAxisAlignment: CrossAxisAlignment.start,
863       children: [
864 // 4
865       Expanded(
866 // 5
867         child: ClipRRect(
868           child: Image.asset(
869             '${recipe.dishImage}',
870             fit: BoxFit.cover,
871           ),
872           borderRadius: BorderRadius.circular(12),
873         ),
874       ),
875       const SizedBox(height: 10),
876 // 7
877       Text(
878         recipe.title,
879         maxLines: 1,
880         style: Theme.of(context).textTheme.bodyText1,
881       ),
882       Text(
883         recipe.duration,
884         style: Theme.of(context).textTheme.bodyText1,
885       ),
886     ],
887   );
888 }
889 }
```

Fig. 79

A screenshot of a Linux desktop environment. In the foreground, a code editor window titled "Untitled Document 1" is open, displaying Dart code for a Flutter application. The code defines a class `RecipesGridView` that extends `StatelessWidget`. It imports various packages and defines a `GridView.builder` to build a grid of recipe cards. The code is annotated with line numbers from 866 to 900. The desktop background is light beige, and the taskbar at the bottom shows icons for various applications like a file manager, terminal, and browser.

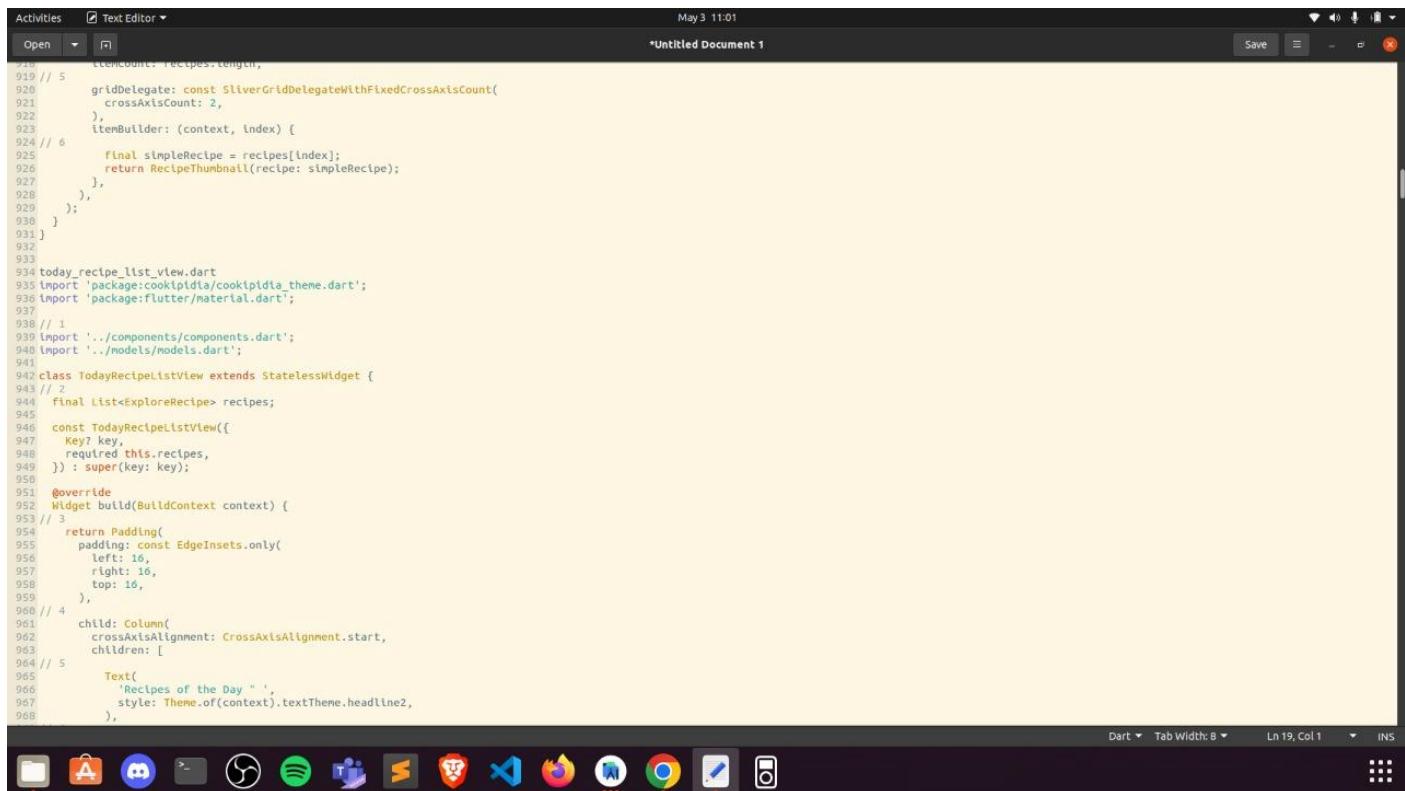
```
866 // 5
867     child: ClipRect(
868       child: Image.asset(
869         '${recipe.dishImage}',
870         fit: BoxFit.cover,
871       ),
872       borderRadius: BorderRadius.circular(12),
873     ),
874   ),
875   const SizedBox(height: 10),
876 // 7
877   Text(
878     recipe.title,
879     maxLines: 1,
880     style: Theme.of(context).textTheme.bodyText1,
881   ),
882   Text(
883     recipe.duration,
884     style: Theme.of(context).textTheme.bodyText1,
885   )
886   ],
887 ),
888 );
889 }
890 }
891
892 recipes_grid_view.dart
893 import 'package:flutter/material.dart';
894 import '../components/components.dart';
895 import '../models/models.dart';
896
897 class RecipesGridView extends StatelessWidget {
898 // 1
899   final List<SimpleRecipe> recipes;
900
901   const RecipesGridView({
902     Key key,
903     required this.recipes,
904   }) : super(key: key);
905
906   @override
907   Widget build(BuildContext context) {
908 // 2
909     return Padding(
910       padding: const EdgeInsets.only(
911         left: 16,
912         right: 16,
913         top: 16,
914       ),
915 // 3
916       child: GridView.builder(
917 // 4
918         itemCount: recipes.length,
919 // 5
920         gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
921           crossAxisCount: 2,
922         ),
923         itemBuilder: (context, index) {
924 // 6
925           final simpleRecip = recipes[index];
926           return RecipeThumbnail(recipe: simpleRecip);
927         },
928       );
929     );
930   }
931 }
932
933 today_recipe_list_view.dart
934 import 'package:cookipedia/cookipedia_theme.dart';
935 import 'package:flutter/material.dart';
936
937
938 // 1
939 import '../components/components.dart';
940 import '../models/models.dart';
941
942 class TodayRecipeListView extends StatelessWidget {
```

Fig. 80

A screenshot of a Linux desktop environment, similar to Fig. 80. In the foreground, a code editor window titled "Untitled Document 1" is open, displaying Dart code for a Flutter application. The code defines a class `TodayRecipeListView` that extends `StatelessWidget`. It imports various packages and defines a `GridView.builder` to build a grid of recipe cards. The code is annotated with line numbers from 893 to 942. The desktop background is light beige, and the taskbar at the bottom shows icons for various applications like a file manager, terminal, and browser.

```
893 recipes_grid_view.dart
894 import 'package:flutter/material.dart';
895 import '../components/components.dart';
896 import '../models/models.dart';
897
898 class RecipesGridView extends StatelessWidget {
899 // 1
900   final List<SimpleRecipe> recipes;
901
902   const RecipesGridView({
903     Key key,
904     required this.recipes,
905   }) : super(key: key);
906
907   @override
908   Widget build(BuildContext context) {
909 // 2
910     return Padding(
911       padding: const EdgeInsets.only(
912         left: 16,
913         right: 16,
914         top: 16,
915       ),
916 // 3
917       child: GridView.builder(
918 // 4
919         itemCount: recipes.length,
920 // 5
921         gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
922           crossAxisCount: 2,
923         ),
924         itemBuilder: (context, index) {
925 // 6
926           final simpleRecip = recipes[index];
927           return RecipeThumbnail(recipe: simpleRecip);
928         },
929       );
930     );
931   }
932
933 today_recipe_list_view.dart
934 import 'package:cookipedia/cookipedia_theme.dart';
935 import 'package:flutter/material.dart';
936
937
938 // 1
939 import '../components/components.dart';
940 import '../models/models.dart';
941
942 class TodayRecipeListView extends StatelessWidget {
```

Fig. 81

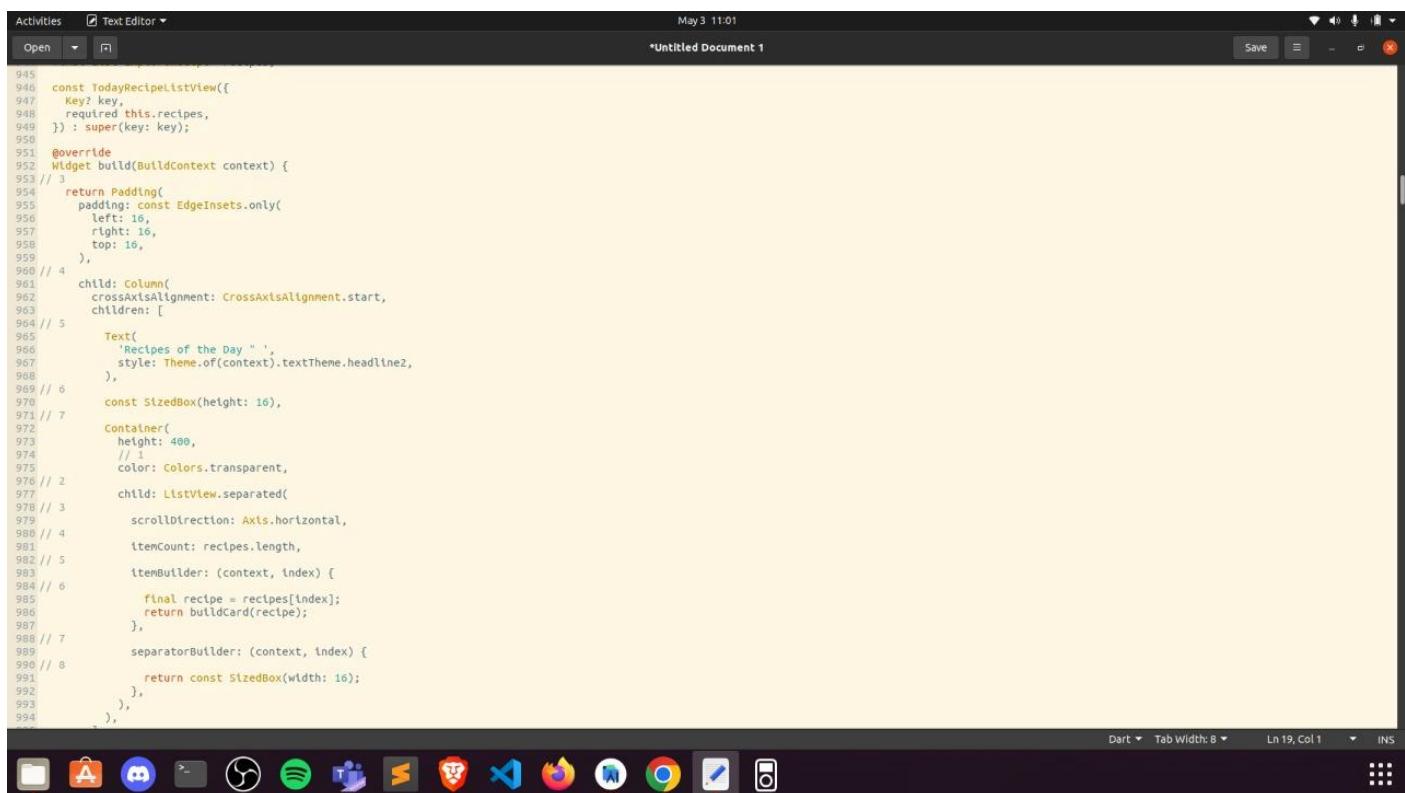


```
May 3 11:01 *Untitled Document 1 Save Activities Text Editor Open □
```

```
919     itemCount: recipes.length,
920   // 5
921   gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
922     crossAxisCount: 2,
923   ),
924   itemBuilder: (context, index) {
925     // 6
926     final simpleRecipe = recipes[index];
927     return RecipeThumbnail(recipe: simpleRecipe);
928   },
929 },
930 );
931 }
932
933 today_recipe_list_view.dart
934 import 'package:cookipedia/cookipedia_theme.dart';
935 import 'package:flutter/material.dart';
936
937 // 1
938 import '../components/components.dart';
939 import '../models/models.dart';
940
941 class TodayRecipeListView extends StatelessWidget {
942 // 2
943   final List<ExploreRecipe> recipes;
944
945   const TodayRecipeListView({
946     Key? key,
947     required this.recipes,
948   }) : super(key: key);
949
950   @override
951   Widget build(BuildContext context) {
952 // 3
953     return Padding(
954       padding: const EdgeInsets.only(
955         left: 16,
956         right: 16,
957         top: 16,
958       ),
959     );
960 // 4
961     child: Column(
962       mainAxisAlignment: MainAxisAlignment.start,
963       children: [
964 // 5
965       Text(
966         'Recipes of the Day',
967         style: Theme.of(context).textTheme.headline2,
968       ),
969     ],
970   );
971 // 6
972   const SizedBox(height: 16),
973   Container(
974     height: 400,
975     // 1
976     color: Colors.transparent,
977   );
978 // 3
979   child: ListView.separated(
980     scrollDirection: Axis.horizontal,
981     itemCount: recipes.length,
982 // 5
983     itemBuilder: (context, index) {
984 // 6
985     final recipe = recipes[index];
986     return buildCard(recipe);
987   },
988 // 7
989   separatorBuilder: (context, index) {
990 // 8
991     return const SizedBox(width: 16);
992   },
993 ),
994 );
```

```
Dart Tab Width: 8 Ln 19, Col 1 INS
```

Fig. 82



```
May 3 11:01 *Untitled Document 1 Save Activities Text Editor Open □
```

```
945 const TodayRecipeListView({
946   Key? key,
947   required this.recipes,
948 }) : super(key: key);
949
950 @override
951 Widget build(BuildContext context) {
952 // 3
953   return Padding(
954     padding: const EdgeInsets.only(
955       left: 16,
956       right: 16,
957       top: 16,
958     ),
959   );
960 // 4
961   child: Column(
962     mainAxisAlignment: MainAxisAlignment.start,
963     children: [
964 // 5
965       Text(
966         'Recipes of the Day',
967         style: Theme.of(context).textTheme.headline2,
968       ),
969     ],
970   );
971 // 6
972   const SizedBox(height: 16),
973   Container(
974     height: 400,
975     // 1
976     color: Colors.transparent,
977   );
978 // 3
979   child: ListView.separated(
980     scrollDirection: Axis.horizontal,
981     itemCount: recipes.length,
982 // 5
983     itemBuilder: (context, index) {
984 // 6
985     final recipe = recipes[index];
986     return buildCard(recipe);
987   },
988 // 7
989   separatorBuilder: (context, index) {
990 // 8
991     return const SizedBox(width: 16);
992   },
993 ),
994 );
```

```
Dart Tab Width: 8 Ln 19, Col 1 INS
```

Fig. 83

A screenshot of a Linux desktop environment. At the top, there's a standard window title bar with 'Activities' and 'Text Editor'. The main area is a code editor titled '*Untitled Document 1' with the date 'May 3 11:01' at the top right. The code is written in Dart and defines a class `AppCache` with methods for building cards and managing shared preferences. The code editor has a dark theme with syntax highlighting. Below the code editor is a dock with various application icons. The bottom right corner shows the terminal status bar with 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'.

```
971 // 7
972     Container(
973         height: 400,
974         // 1
975         color: Colors.transparent,
976     // 2
977         child: ListView.separated(
978             scrollDirection: Axis.horizontal,
979             itemCount: recipes.length,
980             // 4
981             itemBuilder: (context, index) {
982             // 5
983                 final recipe = recipes[index];
984                 return buildCard(recipe);
985             },
986             // 6
987             separatorBuilder: (context, index) {
988                 return const SizedBox(width: 16);
989             },
990             // 7
991             ),
992         ),
993     ),
994     ],
995     ],
996     ),
997 );
998 }
999 Widget buildCard(ExploreRecipe recipe) {
1000     if (recipe.cardType == RecipeCardType.card1) {
1001         return Card1(recipe: recipe);
1002     } else if (recipe.cardType == RecipeCardType.card2) {
1003         return Card2(recipe: recipe);
1004     } else if (recipe.cardType == RecipeCardType.card3) {
1005         return Card3(recipe: recipe);
1006     } else {
1007         throw Exception('This card doesn\'t exist yet');
1008     }
1009 }
1010 ]
1011
1012 app_cache.dart
1013 import 'package:shared_preferences/shared_preferences.dart';
1014
1015 class AppCache {
1016     static const kUser = 'user';
1017     static const kOnboarding = 'onboarding';
1018
1019     Future<void> invalidate() async {
1020         final prefs = await SharedPreferences.getInstance();
1021     }
1022
1023     Future<void> cacheUser() async {
1024         final prefs = await SharedPreferences.getInstance();
1025         await prefs.setBool(kUser, true);
1026     }
1027
1028     Future<void> completeOnboarding() async {
1029         final prefs = await SharedPreferences.getInstance();
1030         await prefs.setBool(kOnboarding, true);
1031     }
1032
1033     Future<bool> isUserLoggedIn() async {
1034         final prefs = await SharedPreferences.getInstance();
1035         return prefs.getBool(kUser) ?? false;
1036     }
1037
1038     Future<bool> didCompleteOnboarding() async {
1039         final prefs = await SharedPreferences.getInstance();
1040         return prefs.getBool(kOnboarding) ?? false;
1041     }
1042 }
1043
1044 }
```

Fig. 84

A screenshot of a Linux desktop environment, similar to Fig. 84. It shows a code editor with the same Dart code for `AppCache`. The code includes methods for invalidating preferences, caching user data, completing onboarding, checking if a user is logged in, and determining if onboarding is completed. The interface is identical to Fig. 84, with a dark-themed code editor, a dock at the bottom, and a terminal status bar at the bottom right.

```
971 // 7
972     Container(
973         height: 400,
974         // 1
975         color: Colors.transparent,
976     // 2
977         child: ListView.separated(
978             scrollDirection: Axis.horizontal,
979             itemCount: recipes.length,
980             // 4
981             itemBuilder: (context, index) {
982             // 5
983                 final recipe = recipes[index];
984                 return buildCard(recipe);
985             },
986             // 6
987             separatorBuilder: (context, index) {
988                 return const SizedBox(width: 16);
989             },
990             // 7
991             ),
992         ),
993     ),
994     ],
995     ],
996     ),
997 );
998 }
999 Widget buildCard(ExploreRecipe recipe) {
1000     if (recipe.cardType == RecipeCardType.card1) {
1001         return Card1(recipe: recipe);
1002     } else if (recipe.cardType == RecipeCardType.card2) {
1003         return Card2(recipe: recipe);
1004     } else if (recipe.cardType == RecipeCardType.card3) {
1005         return Card3(recipe: recipe);
1006     } else {
1007         throw Exception('This card doesn\'t exist yet');
1008     }
1009 }
1010 ]
1011
1012 app_cache.dart
1013 import 'package:shared_preferences/shared_preferences.dart';
1014
1015 class AppCache {
1016     static const kUser = 'user';
1017     static const kOnboarding = 'onboarding';
1018
1019     Future<void> invalidate() async {
1020         final prefs = await SharedPreferences.getInstance();
1021         await prefs.setBool(kUser, false);
1022         await prefs.setBool(kOnboarding, false);
1023     }
1024
1025
1026     Future<void> cacheUser() async {
1027         final prefs = await SharedPreferences.getInstance();
1028         await prefs.setBool(kUser, true);
1029     }
1030
1031     Future<void> completeOnboarding() async {
1032         final prefs = await SharedPreferences.getInstance();
1033         await prefs.setBool(kOnboarding, true);
1034     }
1035
1036     Future<bool> isUserLoggedIn() async {
1037         final prefs = await SharedPreferences.getInstance();
1038         return prefs.getBool(kUser) ?? false;
1039     }
1040
1041     Future<bool> didCompleteOnboarding() async {
1042         final prefs = await SharedPreferences.getInstance();
1043         return prefs.getBool(kOnboarding) ?? false;
1044     }
1045 }
1046
1047 }
```

Fig. 85

A screenshot of a Linux desktop environment. At the top, there's a standard window title bar with 'Activities' and 'Text Editor'. The main area shows a code editor with Dart code for a Flutter application. The code includes imports for 'package:flutter/material.dart' and 'app_cache.dart'. It defines a class `CookipidiaTab` with static constants for explore, recipes, and toBuy counts. A `StateManager` class extends `ChangeNotifier`, initializing variables like `_initialized`, `_loggedin`, and `_onboardingComplete`. It has methods for logging in, completing onboarding, switching tabs, and navigating to recipes. A `Timer` is used to set `_initialized` to true after a 2-second delay. The code editor interface includes tabs for 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'. Below the code editor is a dock with various application icons.

```
May 3 11:01
*Untitled Document 1
1050
1051 import 'package:flutter/material.dart';
1052
1053 import 'app_cache.dart';
1054
1055 class CookipidiaTab {
1056   static const int explore = 0;
1057   static const int recipes = 1;
1058   static const int toBuy = 2;
1059 }
1060
1061 class AppStateManager extends ChangeNotifier {
1062   bool _initialized = false;
1063   bool _loggedin = false;
1064   bool _onboardingComplete = false;
1065   int _selectedTab = CookipidiaTab.explore;
1066   final _appCache = AppCache();
1067
1068   bool get isInitialized => _initialized;
1069   bool get isloggedin => _loggedin;
1070   bool get isOnboardingComplete => _onboardingComplete;
1071   int get selectedTab => _selectedTab;
1072
1073 void initializeApp() async {
1074   _loggedin = await _appCache.isUserLoggedIn();
1075   _onboardingComplete = await _appCache.didCompleteOnboarding();
1076
1077   Timer(
1078     const Duration(milliseconds: 2000),
1079     () {
1080       _initialized = true;
1081       notifyListeners();
1082     },
1083   );
1084 }
1085
1086 void login(String username, String password) async {
1087   _loggedin = true;
1088   await _appCache.cacheUser();
1089   notifyListeners();
1090 }
1091
1092 void completeOnboarding() async {
1093   _onboardingComplete = true;
1094   await _appCache.completeOnboarding();
1095   notifyListeners();
1096 }
1097
1098 void goToTab(index) {
1099   _selectedTab = index;
1100   notifyListeners();
1101 }
1102
1103 void goToRecipes() {
1104   _selectedTab = CookipidiaTab.recipes;
1105   notifyListeners();
1106 }
1107
1108 void logout() async {
1109   _initialized = false;
1110   _selectedTab = 0;
1111   await _appCache.invalidate();
1112
1113   initializeApp();
1114   notifyListeners();
1115 }
1116 }
1117
1118 cookipidia_pages.dart
1119 class CookipidiaPages {
1120   static String splashPath = '/splash';
1121   static String loginPath = '/login';
1122   static String onboardingPath = '/onboarding';
1123   static String home = '/';
1124   static String groceryItemDetails = '/item';
1125   static String profilePath = '/profile';
1126 }
```

Fig. 86

A screenshot of a Linux desktop environment, similar to Fig. 86, showing a code editor with Dart code for a Flutter application. The code is identical to Fig. 86 but includes additional methods: `logout()` which invalidates the cache and sets `_initialized` to false, and `goToRecipes()` which sets the selected tab to `CookipidiaTab.recipes`. The code editor interface includes tabs for 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'. Below the code editor is a dock with various application icons.

```
May 3 11:01
*Untitled Document 1
1070
1071 Timer(
1072   const Duration(milliseconds: 2000),
1073   () {
1074     _initialized = true;
1075     notifyListeners();
1076   },
1077 );
1078
1079 void login(String username, String password) async {
1080   _loggedin = true;
1081   await _appCache.cacheUser();
1082   notifyListeners();
1083 }
1084
1085 void completeOnboarding() async {
1086   _onboardingComplete = true;
1087   await _appCache.completeOnboarding();
1088   notifyListeners();
1089 }
1090
1091 void goToTab(index) {
1092   _selectedTab = index;
1093   notifyListeners();
1094 }
1095
1096 void goToRecipes() {
1097   _selectedTab = CookipidiaTab.recipes;
1098   notifyListeners();
1099 }
1100
1101 void logout() {
1102   _initialized = false;
1103   _selectedTab = 0;
1104   await _appCache.invalidate();
1105
1106   initializeApp();
1107   notifyListeners();
1108 }
1109
1110 cookipidia_pages.dart
1111 class CookipidiaPages {
1112   static String splashPath = '/splash';
1113   static String loginPath = '/login';
1114   static String onboardingPath = '/onboarding';
1115   static String home = '/';
1116   static String groceryItemDetails = '/item';
1117   static String profilePath = '/profile';
1118 }
```

Fig. 87

```
Activities Text Editor
Open Untitled Document 1
May 3 11:01
1103 void goToRecipes() {
1104     _selectedTab = CookipediaTab.recipes;
1105     notifyListeners();
1106 }
1107
1108 void logout() async {
1109     _initialized = false;
1110     _selectedTab = 0;
1111     await _appCache.invalidate();
1112     initializeApp();
1113     notifyListeners();
1114 }
1115 }
1116
1117
1118 cookipedia_pages.dart
1119 class CookipediaPages {
1120     static String splashPath = '/splash';
1121     static String loginPath = '/login';
1122     static String onboardPath = '/onboarding';
1123     static String home = '/';
1124     static String groceryItemDetails = '/item';
1125     static String profilePath = '/profile';
1126     static String raywenderlich = '/raywenderlich';
1127     static String accountPath = '/account';
1128 }
1129
1130
1131
1132 explore_data.dart
1133 import 'models.dart';
1134
1135 class ExploreData {
1136     final List<ExploreRecipe> todayRecipes;
1137     final List<Post> friendPosts;
1138 }
1139
1140 ExploreData(
1141     this.todayRecipes,
1142     this.friendPosts,
1143 );
1144
1145
1146 explore_recipe.dart
1147 part 'ingredient.dart';
1148 part 'instruction.dart';
1149
1150
1151 class RecipeCardType {
1152     static const card1 = 'card1';
1153     static const card2 = 'card2';
1154 }
```

Dart Tab Width: 8 Ln 19, Col 1 INS

Fig. 88

```
Activities Text Editor
Open Untitled Document 1
May 3 11:01
Save
May 3 11:01
*Untitled Document 1
1129 ]
1130
1131
1132 explore_data.dart
1133 import 'models.dart';
1134
1135 class ExploreData {
1136     final List<ExploreRecipe> todayRecipes;
1137     final List<Post> friendPosts;
1138 }
1139
1140 ExploreData(
1141     this.todayRecipes,
1142     this.friendPosts,
1143 );
1144
1145
1146 explore_recipe.dart
1147 part 'ingredient.dart';
1148 part 'instruction.dart';
1149
1150
1151 class RecipeCardType {
1152     static const card1 = 'card1';
1153     static const card2 = 'card2';
1154     static const card3 = 'card3';
1155 }
1156
1157 class ExploreRecipe {
1158     String id;
1159     String cardType;
1160     String title;
1161     String subtitle;
1162     String backgroundImage;
1163     String backgroundImageSource;
1164     String message;
1165     String authorName;
1166     String authorRole;
1167     String profileImage;
1168     int durationInMinutes;
1169     String dietType;
1170     int calories;
1171     List<String> tags;
1172     String description;
1173     String source;
1174     List<Ingredients> ingredients;
1175     List<Instruction> instructions;
1176
1177     ExploreRecipe({
1178         required this.id,
1179         required this.cardType,
1180     });
1181 }
```

Dart Tab Width: 8 Ln 19, Col 1 INS

Fig. 89

```
1155 J
1156
1157 class ExploreRecipe {
1158   String id;
1159   String cardType;
1160   String title;
1161   String subtitle;
1162   String backgroundImage;
1163   String backgroundImageSource;
1164   String message;
1165   String authorName;
1166   String role;
1167   String profileImage;
1168   int durationInMinutes;
1169   String dietType;
1170   int calories;
1171   List<String> tags;
1172   String description;
1173   String source;
1174   List<Ingredients> ingredients;
1175   List<Instruction> instructions;
1176
1177   ExploreRecipe({
1178     required this.id,
1179     required this.cardType,
1180     required this.title,
1181     this.subtitle = '',
1182     this.backgroundImage = '',
1183     this.backgroundImageSource = '',
1184     this.message = '',
1185     this.authorName = '',
1186     this.role = '',
1187     this.profileImage = '',
1188     this.durationInMinutes = 0,
1189     this.dietType = '',
1190     this.calories = 0,
1191     this.tags = const [],
1192     this.description = '',
1193     this.source = '',
1194     this.ingredients = const [],
1195     this.instructions = const [],
1196   });
1197
1198   factory ExploreRecipe.fromJson(Map<String, dynamic> json) {
1199     final ingredients = <Ingredients>[];
1200     final instructions = <Instruction>[];
1201
1202     if (json['ingredients'] != null) {
1203       json['ingredients'].forEach((v) {
1204         ingredients.add(Ingredients.fromJson(v));
1205       });
1206
1207     }
1208
1209     if (json['instructions'] != null) {
1210       json['instructions'].forEach((v) {
1211         instructions.add(Instruction.fromJson(v));
1212       });
1213
1214     return ExploreRecipe(
1215       id: json['id'] ?? '',
1216       cardType: json['cardType'] ?? '',
1217       title: json['title'] ?? '',
1218       subtitle: json['subtitle'] ?? '',
1219       backgroundImage: json['backgroundImage'] ?? '',
1220       backgroundImageSource: json['backgroundImageSource'] ?? '',
1221       message: json['message'] ?? '',
1222       authorName: json['authorName'] ?? '',
1223       role: json['role'] ?? '',
1224       profileImage: json['profileImage'] ?? '',
1225       durationInMinutes: json['durationInMinutes'] ?? 0,
1226       dietType: json['dietType'] ?? '',
1227       calories: json['calories'] ?? 0,
1228       tags: json['tags'].cast<String>() ?? [],
1229       description: json['description'] ?? '',
1230       source: json['source'] ?? '',
1231       ingredients: ingredients,
1232     );
1233   }
1234 }
```

Fig. 90

```
1182     this.subtitle = '',
1183     this.backgroundImage = '',
1184     this.message = '',
1185     this.authorName = '',
1186     this.role = '',
1187     this.profileImage = '',
1188     this.durationInMinutes = 0,
1189     this.dietType = '',
1190     this.calories = 0,
1191     this.tags = const [],
1192     this.description = '',
1193     this.source = '',
1194     this.ingredients = const [],
1195     this.instructions = const [],
1196   );
1197
1198   factory ExploreRecipe.fromJson(Map<String, dynamic> json) {
1199     final ingredients = <Ingredients>[];
1200     final instructions = <Instruction>[];
1201
1202     if (json['ingredients'] != null) {
1203       json['ingredients'].forEach((v) {
1204         ingredients.add(Ingredients.fromJson(v));
1205       });
1206
1207     }
1208
1209     if (json['instructions'] != null) {
1210       json['instructions'].forEach((v) {
1211         instructions.add(Instruction.fromJson(v));
1212       });
1213
1214     return ExploreRecipe(
1215       id: json['id'] ?? '',
1216       cardType: json['cardType'] ?? '',
1217       title: json['title'] ?? '',
1218       subtitle: json['subtitle'] ?? '',
1219       backgroundImage: json['backgroundImage'] ?? '',
1220       backgroundImageSource: json['backgroundImageSource'] ?? '',
1221       message: json['message'] ?? '',
1222       authorName: json['authorName'] ?? '',
1223       role: json['role'] ?? '',
1224       profileImage: json['profileImage'] ?? '',
1225       durationInMinutes: json['durationInMinutes'] ?? 0,
1226       dietType: json['dietType'] ?? '',
1227       calories: json['calories'] ?? 0,
1228       tags: json['tags'].cast<String>() ?? [],
1229       description: json['description'] ?? '',
1230       source: json['source'] ?? '',
1231       ingredients: ingredients,
1232     );
1233   }
1234 }
```

Fig. 91

A screenshot of a Linux desktop environment. At the top, there's a header bar with 'Activities' and 'Text Editor'. The main area shows a code editor for a Dart file named 'grocery_item.dart'. The code defines a class 'GroceryItem' with properties like 'id', 'name', 'importance', 'color', 'quantity', 'date', and 'isComplete'. It also includes methods for copying the item with updated values. The code editor has syntax highlighting for Dart. Below the code editor is a dock with various application icons. The status bar at the bottom shows 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'.

```
1288     if (json['instructions'] != null) {
1289       json['instructions'].forEach((v) {
1290         instructions.add(instruction.fromJson(v));
1291       });
1292     }
1293   }
1294   return ExploreRecipe(
1295     id: json['id'] ?? '',
1296     cardType: json['cardType'] ?? '',
1297     title: json['title'] ?? '',
1298     subtitle: json['subtitle'] ?? '',
1299     backgroundImage: json['backgroundImage'] ?? '',
1300     backgroundImageSource: json['backgroundImageSource'] ?? '',
1301     message: json['message'] ?? '',
1302     authorName: json['authorName'] ?? '',
1303     role: json['role'] ?? '',
1304     profileImage: json['profileImage'] ?? '',
1305     durationInMinutes: json['durationInMinutes'] ?? 0,
1306     dietType: json['dietType'] ?? '',
1307     calories: json['calories'] ?? 0,
1308     tags: json['tags'].cast<String>() ?? [],
1309     description: json['description'] ?? '',
1310     source: json['source'] ?? '',
1311     ingredients: ingredients,
1312     instructions: instructions,
1313   );
1314 }
1315 }
1316
1317 grocery_item.dart
1318 import 'package:flutter/painting.dart';
1319
1320 // 1
1321 enum Importance {
1322   low,
1323   medium,
1324   high,
1325 }
1326
1327 class GroceryItem {
1328   // 2
1329   final String id;
1330
1331   // 3
1332   final String name;
1333   final Importance importance;
1334   final Color color;
1335   final int quantity;
1336   final DateTime date;
1337   final bool isComplete;
1338 }
```

Fig. 92

A screenshot of a Linux desktop environment. At the top, there's a header bar with 'Activities' and 'Text Editor'. The main area shows a code editor for a Dart file named 'grocery_item.dart'. The code defines a class 'GroceryItem' with properties like 'id', 'name', 'importance', 'color', 'quantity', 'date', and 'isComplete'. It also includes methods for copying the item with updated values. The code editor has syntax highlighting for Dart. Below the code editor is a dock with various application icons. The status bar at the bottom shows 'Dart', 'Tab Width: 8', 'Ln 19, Col 1', and 'INS'.

```
1234 }
1235 }
1236
1237 grocery_item.dart
1238 import 'package:flutter/painting.dart';
1239
1240 // 1
1241 enum Importance {
1242   low,
1243   medium,
1244   high,
1245 }
1246
1247 class GroceryItem {
1248   // 2
1249   final String id;
1250
1251   // 3
1252   final String name;
1253   final Importance importance;
1254   final Color color;
1255   final int quantity;
1256   final DateTime date;
1257   final bool isComplete;
1258
1259   GroceryItem({
1260     required this.id,
1261     required this.name,
1262     required this.importance,
1263     required this.color,
1264     required this.quantity,
1265     required this.date,
1266     this.isComplete = false,
1267   });
1268
1269   GroceryItem copyWith({
1270     String? id,
1271     String? name,
1272     Importance? importance,
1273     Color? color,
1274     int? quantity,
1275     DateTime? date,
1276     bool? isComplete,
1277   }) {
1278     return GroceryItem(
1279       id: id ?? this.id,
1280       name: name ?? this.name,
1281       importance: importance ?? this.importance,
1282       color: color ?? this.color,
1283       quantity: quantity ?? this.quantity,
1284     );
1285   }
1286 }
```

Fig. 93

Activities Text Editor

May 3 11:01 *Untitled Document 1 Save

```
1260 GroceryItem({  
1261   required this.id,  
1262   required this.name,  
1263   required this.importance,  
1264   required this.color,  
1265   required this.quantity,  
1266   required this.date,  
1267   this.isComplete = false,  
1268 });  
1269  
1270 GroceryItem copyWith({  
1271   String? id,  
1272   String? name,  
1273   Importance? importance,  
1274   Color? color,  
1275   int? quantity,  
1276   DateTime? date,  
1277   bool? isComplete,  
1278 }) {  
1279   return GroceryItem(  
1280     id: id ?? this.id,  
1281     name: name ?? this.name,  
1282     importance: importance ?? this.importance,  
1283     color: color ?? this.color,  
1284     quantity: quantity ?? this.quantity,  
1285     date: date ?? this.date,  
1286     isComplete: isComplete ?? this.isComplete,  
1287   );  
1288 }  
1289  
1290 grocery_manager.dart  
1291  
1292 import 'package:flutter/material.dart';  
1293  
1294 import 'grocery_item.dart';  
1295  
1296 class GroceryManager extends ChangeNotifier {  
1297   final _groceryItems = <GroceryItem>[];  
1298   int _selectedIndex = -1;  
1299   bool _createNewItem = false;  
1300  
1301 List<GroceryItem> get groceryItems => List.unmodifiable(_groceryItems);  
1302 int get selectedIndex => _selectedIndex;  
1303 GroceryItem? get selectedGroceryitem =>  
1304   _selectedIndex != -1 ? _groceryItems[_selectedIndex] : null;  
1305 bool get isCreatingNewItem => _createNewItem;  
1306  
1307 void createNewItem() {  
1308   _createNewItem = true;  
1309   notifyListeners();  
1310 }
```

Save Tab Width: 8 Ln 19, Col 1 INS

Fig. 94

Activities Text Editor May 3 11:01 *Untitled Document 1 Save

Open

```
1286     _onComplete, _onIncomplete ?? _onIncomplete);
1287   );
1288 }
1289 }
1290 }
1291 }
1292 grocery_manager.dart
1293 import 'package:flutter/material.dart';
1294
1295 import 'grocery_item.dart';
1296
1297 class GroceryManager extends ChangeNotifier {
1298   final _groceryItems = <GroceryItem>[];
1299   int _selectedIndex = -1;
1300   bool _createNewItem = false;
1301
1302   List<GroceryItem> get groceryItems => List.unmodifiable(_groceryItems);
1303   int get selectedIndex => _selectedIndex;
1304   GroceryItem? get selectedGroceryItem =>
1305     selectedIndex != -1 ? _groceryItems[_selectedIndex] : null;
1306   bool get isCreatingNewItem => _createNewItem;
1307
1308   void createNewItem() {
1309     _createNewItem = true;
1310     notifyListeners();
1311   }
1312
1313   void deleteItem(int index) {
1314     _groceryItems.removeAt(index);
1315     notifyListeners();
1316   }
1317
1318   void groceryItemTapped(int index) {
1319     _selectedIndex = index;
1320     _createNewItem = false;
1321     notifyListeners();
1322   }
1323
1324   void addNewItem(GroceryItem item) {
1325     _groceryItems.add(item);
1326     _createNewItem = false;
1327     notifyListeners();
1328   }
1329
1330   void updateItem(GroceryItem item, int index) {
1331     _groceryItems[index] = item;
1332     _selectedIndex = -1;
1333     _createNewItem = false;
1334     notifyListeners();
1335   }
1336
1337 void onCompleteItem(int index, bool channel, f
```

Save Tab Width: 8 ▾ Ln 19, Col 1 INS

Activity icons: Home, App Drawer, Notifications, Recent Apps, Spotify, Terminal, Settings, Browser, File Manager, Camera.

Fig. 95

```
Activities Text Editor
Open Untitled Document 1 Save
May 3 11:01
1313 void deleteItem(int index) {
1314     _groceryItems.removeAt(index);
1315     notifyListeners();
1316 }
1317
1318 void groceryItemTapped(int index) {
1319     _selectedIndex = index;
1320     _createNewItem = false;
1321     notifyListeners();
1322 }
1323
1324 void addItem(GroceryItem item) {
1325     _groceryItems.add(item);
1326     _createNewItem = false;
1327     notifyListeners();
1328 }
1329
1330 void updateItem(GroceryItem item, int index) {
1331     _groceryItems[index] = item;
1332     _selectedIndex = -1;
1333     _createNewItem = false;
1334     notifyListeners();
1335 }
1336
1337 void completeItem(int index, bool change) {
1338     final item = _groceryItems[index];
1339     _groceryItems[index] = item.copyWith(isComplete: change);
1340     notifyListeners();
1341 }
1342
1343
1344 ingredient.dart
1345 part of 'explore_recipe.dart';
1346
1347 class Ingredients {
1348     String imageUrl;
1349     String title;
1350     String source;
1351
1352     Ingredients({
1353         required this.imageUrl,
1354         required this.title,
1355         required this.source,
1356     });
1357
1358     factory Ingredients.fromJson(Map<String, dynamic> json) {
1359         return Ingredients(
1360             imageUrl: json['imageUrl'] ?? '',
1361             title: json['title'] ?? '',
1362             source: json['source'] ?? '',
1363         );
1364     }
1365 }
1366
1367 instruction.dart
1368 part of 'explore_recipe.dart';
1369
1370 class Instruction {
1371     String imageUrl;
1372     String description;
1373     int durationInMinutes;
1374
1375     Instruction({
1376         required this.imageUrl,
1377         required this.description,
1378         required this.durationInMinutes,
1379     });
1380
1381     factory Instruction.fromJson(Map<String, dynamic> json) {
1382         return Instruction(
1383             imageUrl: json['imageUrl'] ?? '',
1384             description: json['description'] ?? '',
1385             durationInMinutes: json['durationInMinutes'] ?? '',
1386         );
1387     }
1388 }
```

Fig. 96

```
Activities Text Editor
Open Untitled Document 1 Save
May 3 11:01
1339     _groceryItems[index] = item.copyWith(isComplete: change);
1340     notifyListeners();
1341 }
1342
1343
1344 ingredient.dart
1345 part of 'explore_recipe.dart';
1346
1347 class Ingredients {
1348     String imageUrl;
1349     String title;
1350     String source;
1351
1352     Ingredients({
1353         required this.imageUrl,
1354         required this.title,
1355         required this.source,
1356     });
1357
1358     factory Ingredients.fromJson(Map<String, dynamic> json) {
1359         return Ingredients(
1360             imageUrl: json['imageUrl'] ?? '',
1361             title: json['title'] ?? '',
1362             source: json['source'] ?? '',
1363         );
1364     }
1365 }
1366
1367 instruction.dart
1368 part of 'explore_recipe.dart';
1369
1370 class Instruction {
1371     String imageUrl;
1372     String description;
1373     int durationInMinutes;
1374
1375     Instruction({
1376         required this.imageUrl,
1377         required this.description,
1378         required this.durationInMinutes,
1379     });
1380
1381     factory Instruction.fromJson(Map<String, dynamic> json) {
1382         return Instruction(
1383             imageUrl: json['imageUrl'] ?? '',
1384             description: json['description'] ?? '',
1385             durationInMinutes: json['durationInMinutes'] ?? '',
1386         );
1387     }
1388 }
```

Fig. 97

```

import 'dart:convert';
import 'package:flutter/services.dart';
import '../models/models.dart';

// Mock recipe service that grabs sample json data to mock recipe request/response
class MockCookipidiaService {
  // Batch request that gets both today recipes and friend's feed
  Future<ExploreData> getExploreData() async {
    final todayRecipes = await _getTodayRecipes();
    final friendPosts = await _getFriendFeed();

    return ExploreData(todayRecipes, friendPosts);
  }

  // Get sample explore recipes json to display in ui
  Future<List<ExploreRecipe>> _getTodayRecipes() async {
    // Simulate api request wait time
    await Future.delayed(const Duration(milliseconds: 1000));
    // Load json from file system
    final jsonString =
        await _loadAsset('assets/sample_data/sample_explore_recipes.json');
    // Decode to json
    final Map<String, dynamic> json = jsonDecode(jsonString);

    // Go through each recipe and convert json to ExploreRecipe object.
    if (json['recipes'] != null) {
      final recipes = <ExploreRecipe>[];
      json['recipes'].forEach((v) {
        recipes.add(ExploreRecipe.fromJson(v));
      });
      return recipes;
    } else {
      return [];
    }
  }
}

```

Fig. 98

```

import 'package:cookipidia/cookipidia_theme.dart';
import '../models/explore_recipe.dart';

class Card1 extends StatelessWidget {
  final ExploreRecipe recipe;

  const Card1({
    Key key,
    required this.recipe,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        child: Stack(
          children: [
            Text(
              recipe.subtitle,
              style: CookipidiaTheme.darkTextTheme.bodyText1,
            ),
            Text(
              recipe.title,
              style: CookipidiaTheme.darkTextTheme.headline2,
            ),
            Positioned(
              top: 20,
              child: Text(
                recipe.message,
                style: CookipidiaTheme.darkTextTheme.bodyText1,
              ),
            ),
            Positioned(
              right: 0,
              child: Text(
                recipe.message,
                style: CookipidiaTheme.darkTextTheme.bodyText1,
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```

Fig. 99

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** Activities, Android Studio, May 3 11:08, cookipidia - card2.dart [cookipidia]
- Menu Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, Git, Window, Help
- Project Tree:** cookipidia (AndroidStudioProjects/cookipidia), showing components like api, components, lib, and models.
- Code Editor:** The file card2.dart is open, displaying Dart code for a StatelessWidget named Card2. The code includes imports for flutter/material.dart, models/explore_recipe.dart, author_card.dart, and cookipidia_theme.dart. It defines a Card2 class with a build method that creates a Center child, which contains a Container with constraints and a BoxDecoration. The BoxDecoration includes a DecorationImage with an AssetImage of recipe.backgroundImage, BoxFit.cover, and a BorderRadius.circular(10.0). The Container also contains a Column with an AuthorCard and an Expanded child, which is a Stack with children including a Container for the author's name and title.
- Toolbars and Status Bar:** Includes Git, TODO, Problems, Terminal, Dart Analysis, Logcat, Profiler, App Inspection, Event Log, Layout Inspector, and a status bar showing Failed to start monitoring 32000bcaba01950d (50 minutes ago), 21:35, LF, UTF-8, 2 spaces, and master.

Fig. 100

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** Activities, Android Studio, May 3 11:08, cookipidia - card3.dart [cookipidia].
- File Menu:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, Git, Window, Help.
- Toolbar:** SM A750F (mobile), main.dart, samsung SM-A750F, Git, Analyzing... (highlighted).
- Project Structure:** cookipidia (~/AndroidStudioProjects/cookipidia). The file card3.dart is selected. The code editor shows the following snippet:

```
    ), // Container
    Container(
      padding: const EdgeInsets.all(16),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          const Icon(
            Icons.book,
            color: Colors.white,
            size: 40,
          ), // Icon
          const SizedBox(height: 8),
          Text(recipe.title,
            style: cookipidiaTheme.darkTextTheme.headline2, // Text
            const SizedBox(height: 30),
            ),
        ],
      ), // Column
    ), // Container
    Center(
      child: Wrap(
        alignment: WrapAlignment.start,
        spacing: 12,
        runSpacing: 12,
        children: createTagChips(), // Wrap
      ), // Center
    ),
  ),
), // Stack
), // Container
); // Center
}
}
```

- Side Panels:** Project, Build, Resource Manager, Structure, Favorites, Build Variants, Git, TODO, Problems, Terminal, Dart Analysis, Logcat, Profiler, App Inspection.
- Bottom Bar:** Failed to start monitoring 32000bcaba01950d (50 minutes ago), Event Log, Layout Inspector.

Fig. 101

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `grocery_list_screen.dart` open. The code defines a `GroceryListScreen` class that extends `StatelessWidget`. It imports various packages and models. The `build` method creates a `ListView.separated` with a padding of 16.0. Each item in the list has a `Dismissible` widget with a red background, white text, and a delete icon. When dismissed, it calls `manager.deleteItem(index)` and shows a snack bar. The code editor has syntax highlighting and code completion suggestions.

```
import 'package:flutter/material.dart';
import '../components/grocery_tile.dart';
import '../models/models.dart';

class GroceryListScreen extends StatelessWidget {
  final GroceryManager manager;

  const GroceryListScreen({
    Key? key,
    required this.manager,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final groceryItems = manager.groceryItems;
    return Padding(
      padding: const EdgeInsets.all(16.0),
      child: ListView.separated(
        itemCount: groceryItems.length,
        itemBuilder: (context, index) {
          final item = groceryItems[index];
          return Dismissible(
            key: Key(item.id),
            direction: DismissDirection.endToStart,
            background: Container(
              color: Colors.red,
              alignment: Alignment.centerRight,
              child: const Icon(
                Icons.delete_forever,
                color: Colors.white,
                size: 50.0,
              ), // Icon
            ), // Container
            onDismissed: (direction) {
              manager.deleteItem(index);
              ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                  content: Text('Item removed'),
                ),
              );
            },
          );
        },
      );
    }
}
```

Fig. 102

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `onboarding_screen.dart` open. The code defines a `OnboardingScreen` class that extends `StatefulWidget`. It imports `cookipidia_models`, `smooth_page_indicator`, and `provider` packages. The `build` method returns a `MaterialPage` with a `MaterialPage` page. The `createState` method returns an `_OnboardingScreenState` object. The `_OnboardingScreenState` class extends `State<OnboardingScreen>` and contains a `PageController` and a `rwColor` color. The `build` method of `_OnboardingScreenState` creates a `Scaffold` with an `AppBar` and a `Text` title. The code editor has syntax highlighting and code completion suggestions.

```
import 'package:cookipidia/models/cookipidia_pages.dart';
import 'package:flutter/material.dart';
import 'package:smooth_page_indicator/smooth_page_indicator.dart';
import 'package:provider/provider.dart';
import './models/models.dart';

class OnboardingScreen extends StatefulWidget {
  static MaterialPage page() {
    return MaterialPage(
      name: CookipidiaPages.onboardingPath,
      key: ValueKey(CookipidiaPages.onboardingPath),
      child: const OnboardingScreen(),
    );
  }

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

  @override
  _OnboardingScreenState createState() => _OnboardingScreenState();
}

class _OnboardingScreenState extends State<OnboardingScreen> {
  final controller = PageController();
  final Color rwColor = const Color.fromRGBO(64, 143, 77, 1);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.transparent,
        elevation: 0.0,
        title: const Text('Getting Started'),
        leading: GestureDetector(
          child: const Icon(
            Icons.chevron_left,
            size: 35,
          ), // Icon
        ),
      ),
    );
  }
}
```

Fig. 103

Activities Android Studio ▾ May 3 11:08

File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help

cookipidia lib screens recipes_screen.dart

Project ▾

- grocery_manager.dart
- ingredient.dart
- instruction.dart
- models.dart
- post.dart
- profile_manager.dart
- simple_recipe.dart
- tab_manager.dart
- user.dart
- navigation
- app_router.dart
- screens
- empty_grocery_screen.dart
- explore_screen.dart
- grocery_item_screen.dart
- grocery_list_screen.dart
- grocery_screen.dart
- home.dart
- login_screen.dart
- onboarding_screen.dart
- profile_screen.dart
- recipes_screen.dart
- screens.dart
- splash_screen.dart
- webview_screen.dart
- cookipidia_theme.dart
- generated_plugin_registrant.dart
- main.dart

Resource Manager ▾

Favorites ▾

Build Variants

Git ▾

TODO Problems Terminal Dart Analysis Logcat Profiler App Inspection

Flutter Outline ▾

Flutter Performance ▾

Device Manager ▾

Flutter Inspector ▾

Emulator ▾

Device File Explorer ▾

Event Log Layout Inspector

23:31 LF UTF-8 2 spaces master

Failed to start monitoring 32006bcaba01950d (50 minutes ago)

Fig. 104

Activities Android Studio ▾ May 3 11:08

File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help

cookipidia lib cookipidia_theme.dart

Project ▾

- grocery_manager.dart
- ingredient.dart
- instruction.dart
- models.dart
- post.dart
- profile_manager.dart
- simple_recipe.dart
- tab_manager.dart
- user.dart
- navigation
- app_router.dart
- screens
- empty_grocery_screen.dart
- explore_screen.dart
- grocery_item_screen.dart
- grocery_list_screen.dart
- grocery_screen.dart
- home.dart
- login_screen.dart
- onboarding_screen.dart
- profile_screen.dart
- recipes_screen.dart
- screens.dart
- splash_screen.dart
- webview_screen.dart
- cookipidia_theme.dart
- generated_plugin_registrant.dart
- main.dart

Resource Manager ▾

Favorites ▾

Build Variants

Git ▾

TODO Problems Terminal Dart Analysis Logcat Profiler App Inspection

Flutter Outline ▾

Flutter Performance ▾

Device Manager ▾

Flutter Inspector ▾

Emulator ▾

Device File Explorer ▾

Event Log Layout Inspector

25:27 LF UTF-8 2 spaces master

Failed to start monitoring 32006bcaba01950d (50 minutes ago)

Fig. 105

```
import '...';

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

class Cookipidia extends StatelessWidget {
  const Cookipidia({Key? key}) : super(key: key);

  @override
  State<Cookipidia> createState() => _CookipidiaState();
}

class _CookipidiaState extends State<Cookipidia> {
  final _groceryManager = GroceryManager();
  final _profileManager = ProfileManager();
  final _appStateManager = AppStateManager();

  late AppRouter _appRouter;

  @override
  void initState() {
    _appRouter = AppRouter(
      appStateManager: _appStateManager,
      groceryManager: _groceryManager,
      profileManager: _profileManager,
    );
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(
          create: (context) => _groceryManager,
        ), // ChangeNotifierProvider
      ],
    );
  }
}
```

Fig. 106

```
import 'package:cloud_firestore/cloud_firestore.dart';

class Message {
  final String text;
  final DateTime date;
  final String? email;
  DocumentReference? reference;

  Message({
    required this.text,
    required this.date,
    this.email,
    this.reference,
  });

  factory Message.fromJson(Map<dynamic, dynamic> json) => Message(
    text: json['text'] as String,
    date: DateTime.parse(json['date'] as String),
    email: json['email'] as String?,
    Map<String, dynamic> toJson() => <String, dynamic>{
      'date': date.toString(),
      'text': text,
      'email': email,
    },
  );

  factory Message.fromSnapshot(DocumentSnapshot snapshot) {
    final message = Message.fromJson(snapshot.data() as Map<String, dynamic>);
    message.reference = snapshot.reference;
    return message;
  }
}
```

Fig. 107

```

import 'package:cloud_firestore/cloud_firestore.dart';
import 'message.dart';

class MessageDao {
    final CollectionReference collection =
        FirebaseFirestore.instance.collection('messages');

    void saveMessage(Message message) {
        collection.add(message.toJson());
    }

    Stream<QuerySnapshot> getMessageStream() {
        return collection.snapshots();
    }
}

```

Fig. 108

```

import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter/widgets.dart';
import 'package:provider/provider.dart';
import '../data/user_dao.dart';

class Login extends StatefulWidget {
    const Login({Key? key}) : super(key: key);
    @override
    _LoginState createState() => _LoginState();
}

class _LoginState extends State<Login> {
    // 1
    final _emailController = TextEditingController();
    // 2
    final _passwordController = TextEditingController();
    // 3
    final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
    @override
    void dispose() {
        // 4
        _emailController.dispose();
        _passwordController.dispose();
        super.dispose();
    }
    @override
    Widget build(BuildContext context) {
        // 1
        final userDao = Provider.of<UserDao>(context, listen: false);
        return Scaffold(
            // 2
            appBar: AppBar(
                title: const Text('Cookipedia Chat'),
            ), // AppBar
            body: Padding(

```

Fig. 109

Fig. 110

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** Activities, Android Studio, May 3 11:09, raychat - message_widget.dart [raychat]
- Menu Bar:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, Git, Window, Help
- Toolbar:** SM A750F (mobile), main.dart, Samsung SM-A750F, Git: ✓, Layout Inspector
- Project Structure:** raychat (AndroidStudioProjects/raychat), lib, ui, message_widget.dart
- Code Editor:** The file message_widget.dart is open. The code defines a StatelessWidget named MessageWidget that takes a message, date, email, and key. It uses a Column with a Container and a Padding widget.
- Side Panels:** Flutter Outline, Flutter Performance, Device Manager, Flutter Inspector, Emulator, Device File Explorer
- Bottom Navigation:** Git, TODO, Problems, Terminal, Dart Analysis, Logcat, Profiler, App Inspection, Layout Inspector
- Status Bar:** Failed to start monitoring 32000bcaba01959d (50 minutes ago), 01:15, LF, UTF-8, 2 spaces, master

Fig. 111

The screenshot shows the Android Studio interface with the project 'raychat' open. The main window displays the Dart code for the `main.dart` file. The code initializes the Flutter engine, Firebase, and defines a `MaterialApp` widget that checks if the user is logged in to determine which screen to show: `MessageList` or `Login`. The code uses `ChangeNotifierProvider` and `MultiProvider` to manage state providers.

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

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const App());
}

class App extends StatelessWidget {
  const App({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider<UserDao>(
          lazy: false,
          create: () => UserDao(),
        ), // ChangeNotifierProvider
        Provider<MessageDao>(
          lazy: false,
          create: () => MessageDao(),
        ), // Provider
      ],
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'CookipediaChat',
        theme: ThemeData(primaryColor: const Color(0xFF30814A)),
        // ...
        home: Consumer<UserDao>(
          builder: (context, userDao, child) {
            if (userDao.isLoggedIn()) {
              return const MessageList();
            } else {
              return const Login();
            }
          },
        ),
      ),
    );
  }
}
```

Fig. 112

The screenshot shows the Visual Studio Code interface with the project 'COOKIPEDIA_TEST_1' open. The main window displays the `pubspec.yaml` file. The file specifies dependencies for `moor_flutter`, `flutter_test`, and `moor_generator`. It also includes sections for `flutter_lints`, `build_runner`, `json_serializable`, and `chopper_generator`. The `flutter` section ensures the Material Icons font is included. The `assets` section lists two JSON files: `recipes1.json` and `recipes2.json`.

```
moor_flutter: ^4.1.0
dev_dependencies:
  flutter_test:
    sdk: flutter

# The "flutter_lints" package below contains a set of recommended lints to
# encourage good coding practices. The lint set provided by the package is
# activated in the "analysis_options.yaml" file located at the root of your
# package. See that file for information about deactivating specific lint
# rules and activating additional ones.
flutter_lints: ^1.0.0
build_runner: ^2.1.8
json_serializable: ^6.1.5
chopper_generator: ^4.0.5
moor_generator: ^4.6.0+1

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
flutter:

# The following line ensures that the Material Icons font is
# included with your application, so that you can use the icons in
# the material Icons class.
uses-material-design: true

assets:
  - assets/recipes1.json
  - assets/recipes2.json
```

Fig. 113

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Editor:** colors.dart - cookipidia_test_1 - Visual Studio Code
- Code Content:**

```
1 import 'dart:ui';
2
3 const Color green = Color(0xff158442);
4 const Color lightGrey = Color(0xfffff6367);
5 const Color shim = Color(0x7f000000);
```
- Explorer:** Shows the project structure for COOKIPIDIA_TEST_1, including lib, ui, test, web, flutter-plugins, flutter-plugins-dependencies, .gitignore, and .metadata.
- Terminal:** bash
- Bottom Taskbar:** Includes icons for various applications like Spotify, Firefox, and VS Code.

Fig. 114

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Editor:** main_screen.dart - cookipidia_test_1 - Visual Studio Code
- Code Content:**

```
1 import 'package:flutter/services.dart';
2 import 'package:flutter_svg/flutter_svg.dart';
3 import 'colors.dart';
4
5 import 'myrecipes/my_recipes_list.dart';
6 import 'recipes/recipe_list.dart';
7 import 'shopping/shopping_list.dart';
8 import 'package:flutter/material.dart';
9 import 'package:shared_preferences/shared_preferences.dart';
10
11 class MainScreen extends StatefulWidget {
12   const MainScreen({Key? key}) : super(key: key);
13
14   @override
15   _MainScreenState createState() => _MainScreenState();
16 }
17
18 class _MainScreenState extends State<MainScreen> {
19   int _selectedIndex = 0;
20   List<Widget> pageList = [];
21
22   static const String prefSelectedIndexKey = 'selectedIndex';
23
24   @override
25   void initState() {
26     super.initState();
27     pageList.add(const RecipeList());
28     pageList.add(const MyRecipesList());
29     pageList.add(const ShoppingList());
30     getCurrentIndex();
31   }
32 }
```
- Explorer:** Shows the project structure for COOKIPIDIA_TEST_1, including lib, ui, test, web, flutter-plugins, flutter-plugins-dependencies, .gitignore, and .metadata.
- Terminal:** bash
- Bottom Taskbar:** Includes icons for various applications like Spotify, Firefox, and VS Code.

Fig. 115

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for COOKIPIDIA_TEST_1, including files like ingredient.dart, models.dart, recipe.dart, moor, sqlite, memory_repository.dart, repository.dart, mock_service, network, ui, myrecipes, recipes, shopping, widgets, colors.dart, main_screen.dart, and recipe_card.dart.
- Editor:** The main.dart and recipe_card.dart files are open. recipe_card.dart contains Dart code for a recipe card UI.
- Terminal:** Shows the command "utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$".
- Bottom Bar:** Includes icons for various tools and a terminal.

Fig. 116

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for COOKIPIDIA_TEST_1, including files like ingredient.dart, models.dart, recipe.dart, moor, sqlite, memory_repository.dart, repository.dart, mock_service, network, ui, myrecipes, recipes, shopping, and shopping_list.dart.
- Editor:** The main.dart and shopping_list.dart files are open. shopping_list.dart contains Dart code for a shopping list screen.
- Terminal:** Shows the command "utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$".
- Bottom Bar:** Includes icons for various tools and a terminal.

Fig. 117

Activities Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

COOKIPIDIA_TEST_1

- ingredient.dart
- models.dart
- recipe.dart
- moor
- moor_db.dart
- moor_db.g.dart
- moor_repository.dart
- sqlite
- memory_repository.dart
- repository.dart
- mock_service
- network
- ui
- myrecipes
- recipes
- shopping
- shopping_list.dart
- widgets
- colors.dart
- main_screen.dart
- recipe_card.dart
- generated_plugin_registrant.dart
- main.dart

test

web

flutter-plugins

.flutter-plugins-dependencies

.gitignore

OUTLINE

TIMELINE

DEPENDENCIES

Ln 1, Col 1 Spaces: 2 UTF-8 LF Dart Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64)

May 3 23:52 shopping_list.dart - cookipidia_test_1 - Visual Studio Code

```
main.dart      shopping_list.dart ×
lib > ui > shopping > shopping_list.dart > ...
22   return StreamBuilder(
23     stream: repository.watchAllIngredients(),
24     builder: (context, snapshot) {
25       if (snapshot.connectionState == ConnectionState.active) {
26         final ingredients = snapshot.data as List<Ingredient>?;
27         if (ingredients == null) {
28           return Container();
29         }
30       }
31       return ListView.builder(
32         itemCount: ingredients.length,
33         itemBuilder: (BuildContext context, int index) {
34           return CheckboxListTile(
35             value: checkBoxValues.containsKey(index) &&
36               checkBoxValues[index]!,
37             title: Text(ingredients[index].name ?? ''),
38             onChanged: (newValue) {
39               if (newValue != null) {
40                 setState(() {
41                   checkBoxValues[index] = newValue;
42                 });
43               }
44             },
45           ); // CheckboxListTile
46         } // ListView.builder
47       } else {
48         return Container();
49       }
50     ); // StreamBuilder
51   }
52 }
```

PROBLEMS 38 OUTPUT DEBUG CONSOLE TERMINAL

utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$

Fig. 118

Activities Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

COOKIPIDIA_TEST_1

- ingredient.dart
- models.dart
- recipe.dart
- moor
- moor_db.dart
- moor_db.g.dart
- moor_repository.dart
- sqlite
- memory_repository.dart
- repository.dart
- mock_service
- network
- ui
- myrecipes
- recipes
- shopping
- shopping_list.dart
- widgets
- colors.dart
- main_screen.dart
- recipe_card.dart
- generated_plugin_registrant.dart
- main.dart

test

web

flutter-plugins

.flutter-plugins-dependencies

OUTLINE

TIMELINE

DEPENDENCIES

Ln 15, Col 5 Spaces: 2 UTF-8 LF Dart Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64)

May 3 23:52 mock_service.dart - cookipidia_test_1 - Visual Studio Code

```
main.dart      mock_service.dart ×
lib > mock_service > mock_service.dart > MockService
1 import 'dart:convert';
2 import 'dart:math';
3 import 'package:http/http.dart' as http;
4 import '../network/service_interface.dart';
5 import 'package:chopper/chopper.dart';
6 import 'package:flutter/services.dart' show rootBundle;
7 import '../network/model_response.dart';
8 import '../network/recipe_model.dart';
9
10 class MockService implements ServiceInterface {
11   // 1
12   late APIRecipeQuery _currentRecipes1;
13   late APIRecipeQuery _currentRecipes2;
14
15   // 2
16   Random nextRecipe = Random();
17
18   void create() {
19     loadRecipes();
20   }
21
22   void loadRecipes() async {
23     // 4
24     var jsonString = await rootBundle.loadString('assets/recipes1.json');
25     // 5
26     _currentRecipes1 = APIRecipeQuery.fromJson(jsonDecode(jsonString));
27     jsonString = await rootBundle.loadString('assets/recipes2.json');
28     _currentRecipes2 = APIRecipeQuery.fromJson(jsonDecode(jsonString));
29   }
30
31   @override
32   Future<Response<Result<APIRecipeQuery>>> queryRecipes()
```

PROBLEMS 38 OUTPUT DEBUG CONSOLE TERMINAL

utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$

Fig. 119

Activities Visual Studio Code ▾ May 3 23:52

File Edit Selection View Go Run Terminal Help

EXPLORER

COOKIPIDIA_TEST_1

- ingredient.dart
- models.dart
- recipe.dart
- moor
- moor_db.dart
- moor_db.g.dart
- moor_repository.dart
- sqlite
- memory_repository.dart
- repository.dart
- mock_service
- mock_service.dart
- network
- model_converter.dart
- model_response.dart
- recipe_model.dart
- recipe_model.g.dart
- recipe_service.chopper.dart
- recipe_service.dart
- service_interface.dart
- ui
- myrecipes
- recipes
- shopping
- shopping_list.dart
- widgets
- colors.dart
- main_screen.dart

main.dart model_converter.dart

```
1 import 'dart:convert';
2 import 'package:chopper/chopper.dart';
3
4 import 'model_response.dart';
5 import 'recipe_model.dart';
6
7 class ModelConverter implements Converter {
8   @override
9   Request convertRequest(Request request) {
10     final req = applyHeader(
11       request,
12       contentTypeKey,
13       jsonHeaders,
14       override: false,
15     );
16
17     return encodeJson(req);
18   }
19
20   Request encodeJson(Request request) {
21     final contentType = request.headers[contentTypeKey];
22     if (contentType != null && contentType.contains(jsonHeaders)) {
23       return request.copyWith(body: json.encode(request.body));
24     }
25     return request;
26   }
27
28   Response<BodyType> decodeJson<BodyType, InnerType>(Response response) {
29     final contentType = response.headers[contentTypeKey];
30     var body = response.body;
31     if (contentType != null && contentType.contains(jsonHeaders)) {
32       body = utf8.decode(response.bodyBytes);
33     }
34     return response;
35   }
36 }
```

PROBLEMS 38 OUTPUT DEBUG CONSOLE TERMINAL

utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$

Ln 55, Col 1 Spaces: 2 UTF-8 LF Dart Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64)

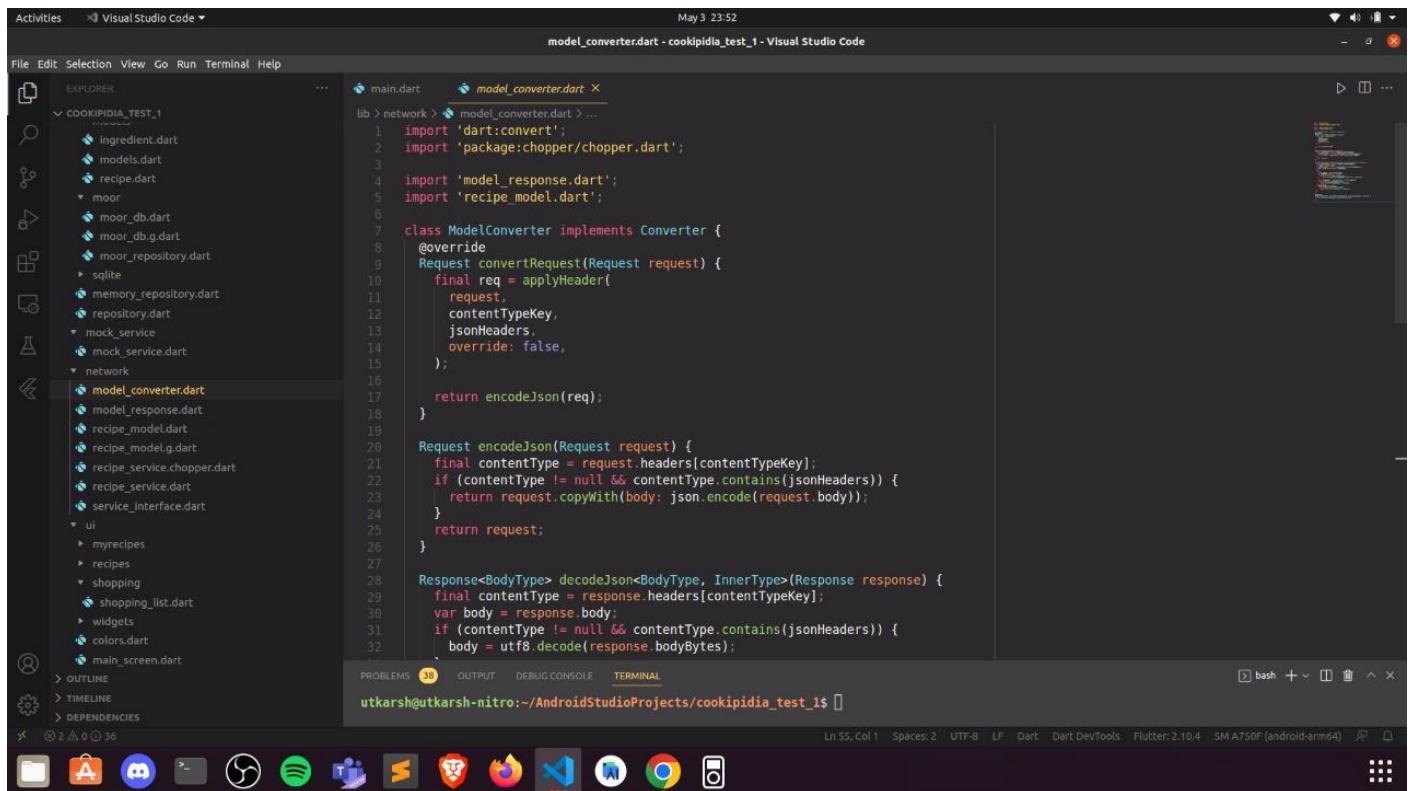


Fig. 120

Activities Visual Studio Code ▾ May 3 23:52

File Edit Selection View Go Run Terminal Help

EXPLORER

COOKIPIDIA_TEST_1

- ingredient.dart
- models.dart
- recipe.dart
- moor
- moor_db.dart
- moor_db.g.dart
- moor_repository.dart
- sqlite
- memory_repository.dart
- repository.dart
- mock_service
- mock_service.dart
- network
- model_converter.dart
- model_response.dart
- recipe_model.dart
- recipe_model.g.dart
- recipe_service.chopper.dart
- recipe_service.dart
- service_interface.dart
- ui
- myrecipes
- recipes
- shopping
- shopping_list.dart
- widgets
- colors.dart
- main_screen.dart

main.dart model_response.dart

```
1 // 1
2 abstract class Result<T> {
3 }
4 // 2
5 class Success<T> extends Result<T> {
6   final T value;
7   Success(this.value);
8 }
9 // 3
10 class Error<T> extends Result<T> {
11   final Exception exception;
12
13   Error(this.exception);
14 }
```

PROBLEMS 38 OUTPUT DEBUG CONSOLE TERMINAL

utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$

Ln 1, Col 1 Spaces: 2 UTF-8 LF Dart Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64)

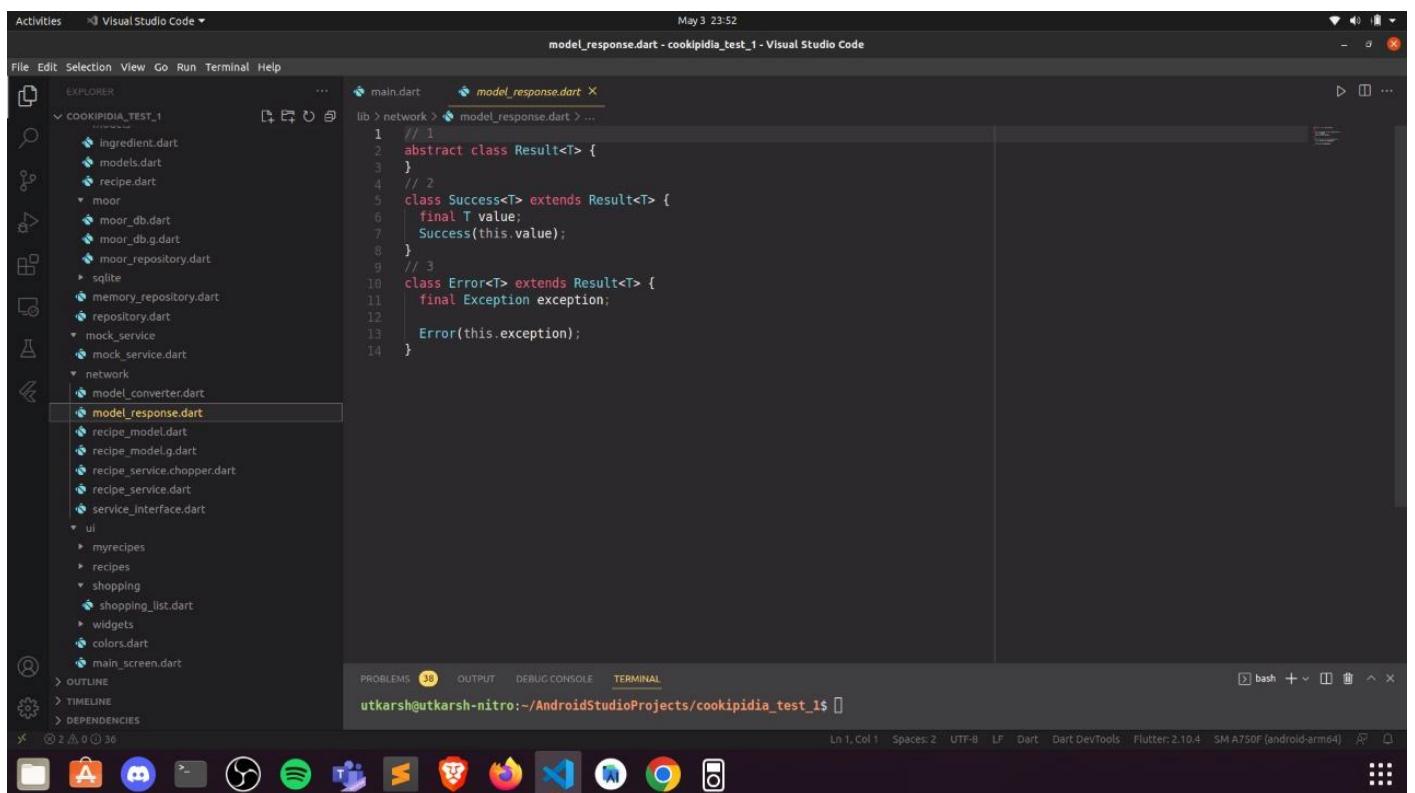


Fig. 121

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for 'COOKIPIDIA_TEST_1'. The 'recipe_model.dart' file is selected.
- Code Editor:** Displays the content of 'recipe_model.dart'. The code defines a class 'APIRecipeQuery' with methods for serialization and deserialization of API responses.
- Terminal:** Shows the command 'utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$'.
- Bottom Bar:** Includes icons for various applications like Spotify, Firefox, and Google Chrome.

Fig. 122

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for 'COOKIPIDIA_TEST_1'. The 'recipe_service.chopper.dart' file is selected.
- Code Editor:** Displays the content of 'recipe_service.chopper.dart'. It contains generated code for a 'ChopperGenerator' and a 'RecipeService' class.
- Terminal:** Shows the command 'utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$'.
- Bottom Bar:** Includes icons for various applications like Spotify, Firefox, and Google Chrome.

Fig. 123

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for 'COOKIPIDIA_TEST_1'. The 'lib' folder contains 'network', 'model_converter.dart', 'model_response.dart', 'recipe_model.dart', 'recipe_model.g.dart', 'recipe_service.chopper.dart', 'recipe_service.dart', and 'service_Interface.dart'. The 'ui' folder contains 'myrecipes', 'recipes', and 'shopping'.
- Editor:** The 'recipe_service.dart' file is open, showing Dart code for creating a RecipeService instance using ChopperClient and HttpLoggingInterceptor.
- Terminal:** The terminal shows the command 'utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$'.
- Bottom:** A dock with various icons including Finder, App Store, Mail, Spotify, Settings, Safari, Firefox, VS Code, and Google Chrome.

Fig. 124

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for 'COOKIPIDIA_TEST_1'. The 'lib' folder contains 'network', 'model_converter.dart', 'model_response.dart', 'recipe_model.dart', 'recipe_model.g.dart', 'recipe_service.chopper.dart', 'recipe_service.dart', and 'service_Interface.dart'. The 'ui' folder contains 'myrecipes', 'recipes', and 'shopping'.
- Editor:** The 'service_Interface.dart' file is open, showing Dart code for an abstract ServiceInterface class that queries recipes.
- Terminal:** The terminal shows the command 'utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$'.
- Bottom:** A dock with various icons including Finder, App Store, Mail, Spotify, Settings, Safari, Firefox, VS Code, and Google Chrome.

Fig. 125

```
lib > ui > myrecipes > my_recipes_list.dart > _MyRecipesListState > _buildRecipeList
35 // 4
36 stream: repository.watchAllRecipes(),
37 // 5
38 builder: (context, AsyncSnapshot<List<Recipe>> snapshot) {
39 // 6
40 if (snapshot.connectionState == ConnectionState.active) {
41 final recipes = snapshot.data ?? [];
42 return ListView.builder(
43 itemCount: recipes.length,
44 itemBuilder: (BuildContext context, int index) {
45 final recipe = recipes[index];
46 return SizedBox(
47 height: 100,
48 child: Slidable(
49 actionPane: const SlidableDrawerActionPane(),
50 actionExtentRatio: 0.25,
51 child: Card(
52 elevation: 1.0,
53 shape: RoundedRectangleBorder(
54 borderRadius: BorderRadius.circular(10.0),
55 ), // RoundedRectangleBorder
56 color: Colors.white,
57 child: Align(
58 alignment: Alignment.center,
59 child: Padding(
60 padding: const EdgeInsets.all(8.0),
61 child: ListTile(
62 leading: CachedNetworkImage(
63 imageUrl: recipe.image ?? '',
64 height: 120,
65 width: 60,
66 fit: BoxFit.cover,
67 
```

utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$

Fig. 126

```
flutter:
  sdk: flutter

# The following adds the Cupertino Icons font to your application.
# Use with the CupertinoIcons class for iOS style icons.
cupertino_icons: ^1.0.2
google_fonts: ^2.3.1
provider: ^6.0.2
flutter_colorpicker: ^1.0.3
intl: ^0.17.0
uuid: ^3.0.6
smooth_page_indicator: ^1.0.0+2
webview_flutter: ^3.0.1
url_launcher: ^6.0.20
shared_preferences: ^2.0.13
image_picker: ^0.8.5
path_provider: ^2.0.9
firebase_core: ^1.16.0
cloud_firestore: ^3.1.14

dev_dependencies:
  flutter_test:
    sdk: flutter

# The "flutter_lints" package below contains a set of recommended lints to
# encourage good coding practices. The lint set provided by the package is
# activated in the analysis_options.yaml file located at the root of your
# package. See that file for information about deactivating specific lint
# rules and activating additional ones.
flutter_lints: ^1.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter.
```

Fig. 127

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** COOKIPIDIA folder containing .dart_tool, .idea, android, .gradle, app, src, build.gradle, google-services.json, gradle, gitignore, build.gradle, cookipidia_android.iml, gradle.properties, gradlew, gradlew.bat, local.properties, settings.gradle, assets, build, ios, lib, api, components, data, message.dart, models, navigation, screens.
- Editor:** build.gradle (selected tab) showing code for setting up the Android build configuration.
- Bottom Status Bar:** Ln 27, Col 47, Spaces: 4, UTF-8, LF, Groovy, Dart DevTools, SM A750F (android-arm64)
- Bottom Taskbar:** Icons for various applications including Finder, App Store, Mail, Safari, Spotify, Slack, VS Code, Firefox, Chrome, and Camera.

```
1 android > app > build.gradle
2     def localProperties = new Properties()
3     def localPropertiesFile = rootProject.file('local.properties')
4     if (localPropertiesFile.exists()) {
5         localPropertiesFile.withReader('UTF-8') { reader ->
6             localProperties.load(reader)
7         }
8     }
9
10    def flutterRoot = localProperties.getProperty('flutter.sdk')
11    if (flutterRoot == null) {
12        throw new GradleException("Flutter SDK not found. Define location with flutter.sdk in the local.properties file.")
13    }
14
15    def flutterVersionCode = localProperties.getProperty('flutter.versionCode')
16    if (flutterVersionCode == null) {
17        flutterVersionCode = '1'
18    }
19
20    def flutterVersionName = localProperties.getProperty('flutter.versionName')
21    if (flutterVersionName == null) {
22        flutterVersionName = '1.0'
23    }
24
25    apply plugin: 'com.android.application'
26    apply plugin: 'kotlin-android'
27    apply from: '$flutterRoot/packages/flutter_tools/gradle/flutter.gradle'
28    apply plugin: 'com.google.gms.google-services'
29
30    android {
31        compileSdkVersion flutter.compileSdkVersion
32
33        compileOptions {
34            sourceCompatibility JavaVersion.VERSION_1_8
35            targetCompatibility JavaVersion.VERSION_1_8
36        }
37    }
38
39
```

Fig. 128

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer:** COOKIPIDIA folder containing .dart_tool, .idea, android, .gradle, app, src, build.gradle, google-services.json, gradle, gitignore, build.gradle, cookipidia_android.iml, gradle.properties, gradlew, gradlew.bat, local.properties, settings.gradle, assets, build, ios, lib, api, components, data, message.dart, models, navigation, screens.
- Editor:** cookipidia_android.iml (selected tab) showing XML code for the module.
- Bottom Status Bar:** Ln 1, Col 1, Spaces: 2, UTF-8, LF, XML, Dart DevTools, SM A750F (android-arm64)
- Bottom Taskbar:** Icons for various applications including Finder, App Store, Mail, Safari, Spotify, Slack, VS Code, Firefox, Chrome, and Camera.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <module type="JAVA_MODULE" version="4">
3     <component name="FacetManager">
4         <facet type="android" name="Android">
5             <configuration>
6                 <option name="ALLOW_USER_CONFIGURATION" value="false" />
7                 <option name="GEN_FOLDER_RELATIVE_PATH_APT" value="/gen" />
8                 <option name="GEN_FOLDER_RELATIVE_PATH_AIDL" value="/gen" />
9                 <option name="MANIFEST_FILE_RELATIVE_PATH" value="/app/src/main/AndroidManifest.xml" />
10                <option name="RES_FOLDER_RELATIVE_PATH" value="/app/src/main/res" />
11                <option name="ASSETS_FOLDER_RELATIVE_PATH" value="/app/src/main/assets" />
12                <option name="LIBS_FOLDER_RELATIVE_PATH" value="/app/src/main/libs" />
13                <option name="PROGUARD_LOGS_FOLDER_RELATIVE_PATH" value="/app/src/main/proguard_logs" />
14            </configuration>
15        </facet>
16    </component>
17    <component name="NewModuleRootManager" inherit-compiler-output="true">
18        <exclude-output />
19        <content url="file://$MODULE_DIR$">
20            <sourceFolder url="file://$MODULE_DIR$/app/src/main/java" isTestSource="false" />
21            <sourceFolder url="file://$MODULE_DIR$/app/src/main/kotlin" isTestSource="false" />
22            <sourceFolder url="file://$MODULE_DIR$/gen" isTestSource="false" generated="true" />
23        </content>
24        <orderEntry type="jdk" jdkName="Android API 29 Platform" jdkType="Android SDK" />
25        <orderEntry type="sourceFolder" forTests="false" />
26        <orderEntry type="library" name="Flutter for Android" level="project" />
27        <orderEntry type="library" name="KotlinJavaRuntime" level="project" />
28    </component>
29 </module>
```

Fig. 129

Activities Visual Studio Code ▾ May 3 23:53

File Edit Selection View Go Run Terminal Help

EXPLORER COOKIPIDIA

- .dart_tool
- .idea
- ▼ android

 - .gradle
 - app
 - src
 - ▀ build.gradle
 - ▀ google-services.json
 - ▀ gradle
 - ▀ .gitignore
 - ▀ build.gradle

- cookipidia_android.iml
- ▀ gradle.properties
- ▀ gradlew
- ▀ gradlew.bat
- ▀ local.properties
- ▀ settings.gradle

login_screen.dart message.dart M pubspec.yaml M settings.gradle

```
1 include ':app'
2
3 def localPropertiesFile = new File(rootProject.projectDir, "local.properties")
4 def properties = new Properties()
5
6 assert localPropertiesFile.exists()
7 localPropertiesFile.withReader("UTF-8") { reader -> properties.load(reader) }
8
9 def flutterSdkPath = properties.getProperty("flutter.sdk")
10 assert flutterSdkPath != null, "flutter.sdk not set in local.properties"
11 apply from: "$flutterSdkPath/packages/flutter_tools/gradle/app_plugin_loader.gradle"
```

assets build ios lib api components data message.dart models navigation screens

OUTLINE TIMELINE DEPENDENCIES

Ln 1, Col 1 Spaces: 4 UTF-8 LF Groovy Dart DevTools SM A750F (android-arm64)

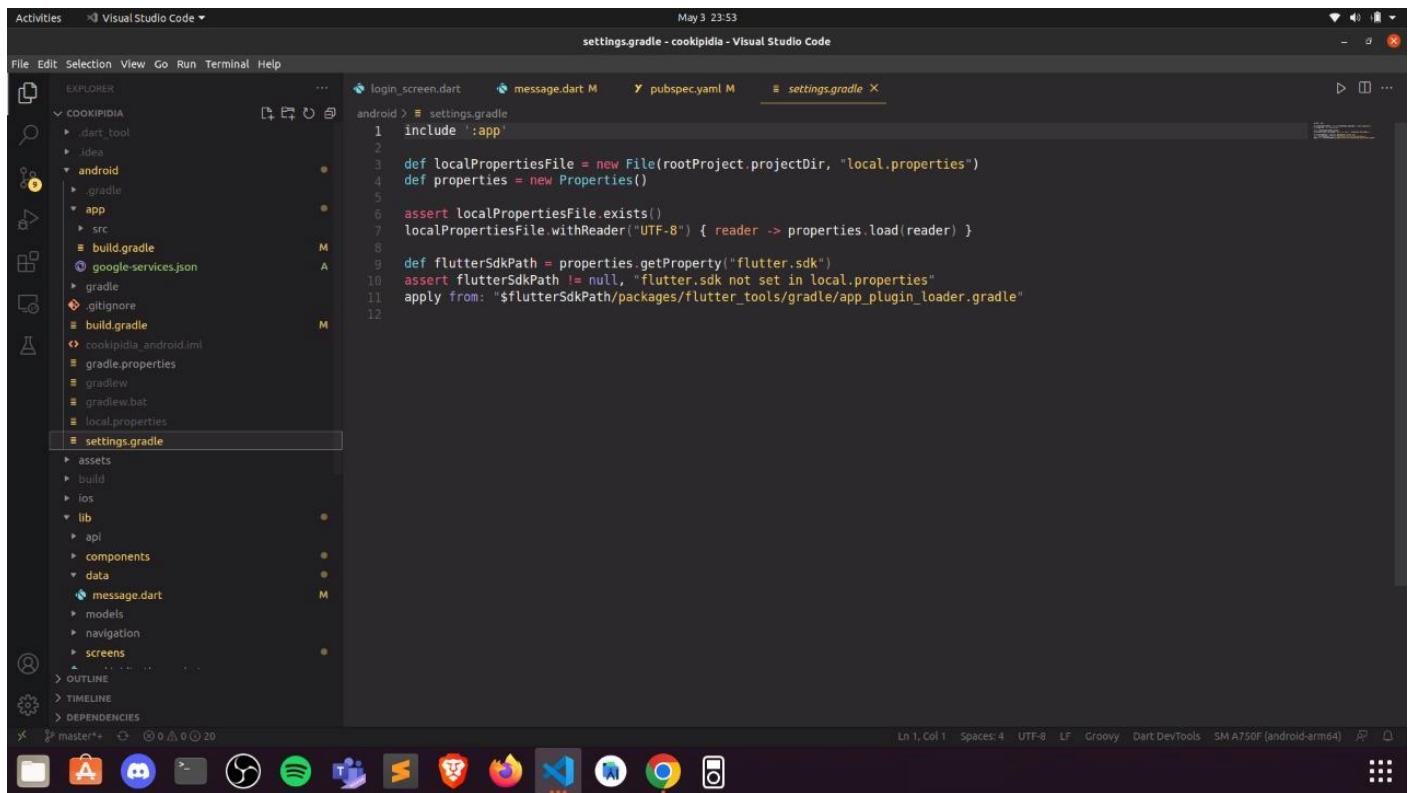


Fig. 130

Activities Visual Studio Code ▾ May 3 23:53

File Edit Selection View Go Run Terminal Help

EXPLORER COOKIPIDIA

- .dart_tool
- .idea
- ▼ android

 - .gradle
 - app
 - src
 - ▀ build.gradle
 - ▀ google-services.json
 - ▀ gradle
 - ▀ .gitignore
 - ▀ build.gradle

- cookipidia_android.iml
- ▀ gradle.properties
- ▀ gradlew
- ▀ gradlew.bat
- ▀ local.properties
- ▀ settings.gradle

login_screen.dart message.dart M pubspec.yaml M gradlew.bat

```
19 @rem Find Java home
20 if defined JAVA_HOME goto findJavaFromJavaHome
21
22 set JAVA_EXE=java.exe
23 %JAVA_EXE% -version >NUL 2>&1
24 if "%ERRORLEVEL%" == "0" goto init
25
26 echo.
27 echo ERROR: JAVA_HOME is not set and no 'java' command could be found in your PATH.
28 echo.
29 echo Please set the JAVA_HOME variable in your environment to match the
30 echo location of your Java installation.
31
32 goto fail
33
34 :findJavaFromJavaHome
35 set JAVA_HOME=%JAVA_HOME:"=%
36 set JAVA_EXE=%JAVA_HOME%/bin/java.exe
37
38 if exist "%JAVA_EXE%" goto init
39
40 echo.
41 echo ERROR: JAVA_HOME is set to an invalid directory: %JAVA_HOME%
42 echo.
43 echo Please set the JAVA_HOME variable in your environment to match the
44 echo location of your Java installation.
45
46 goto fail
47
48 :init
49 @rem Get command-line arguments, handling Windows variants
50
51 if not "%OS%" == "Windows_NT" goto win9xME_args
52 if "%@eval[2+2]" == "4" goto 4NT_args
53
54 :win9xME_args
55 @rem Slurp the command line arguments.
```

assets build ios lib api components data message.dart models navigation screens

OUTLINE TIMELINE DEPENDENCIES

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Batch Dart DevTools SM A750F (android-arm64)

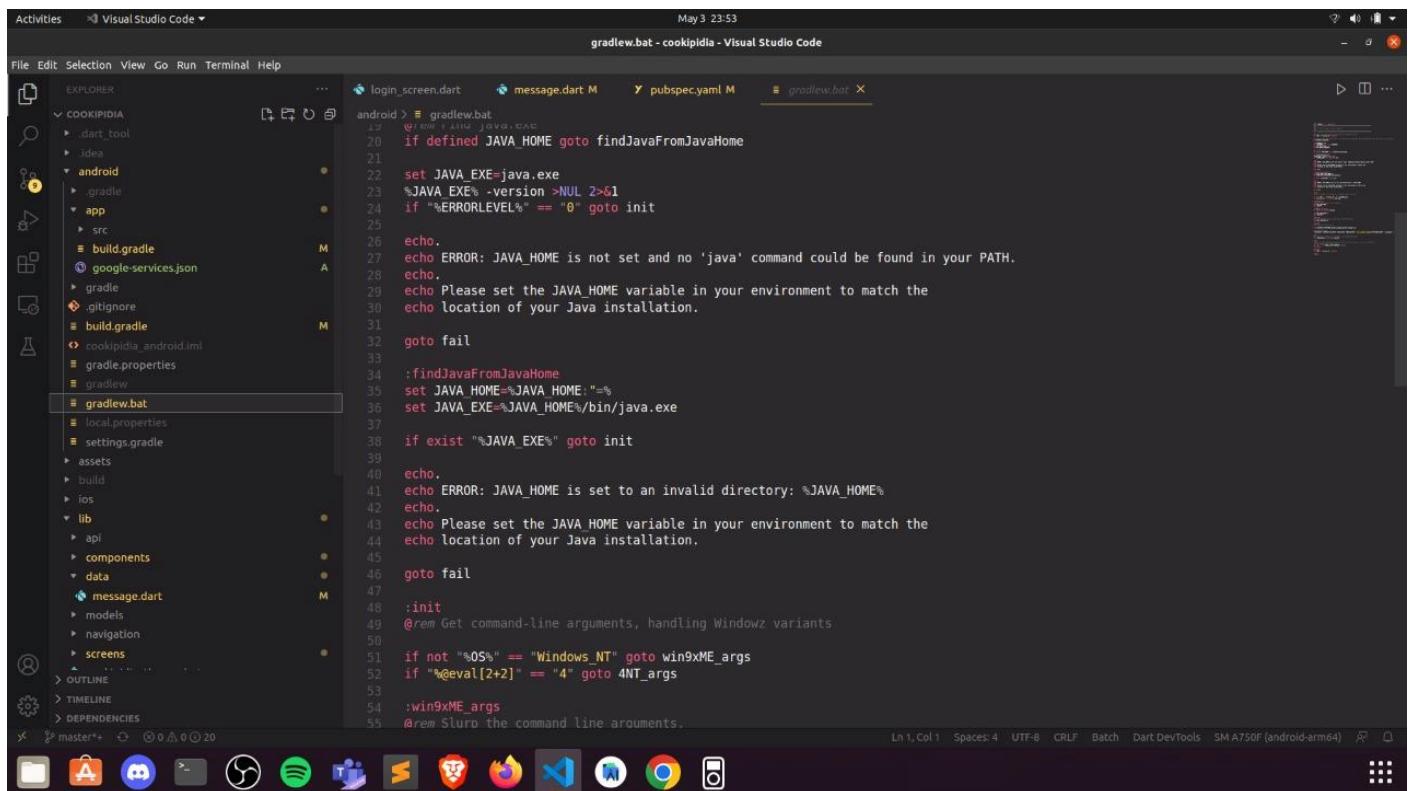


Fig. 131

Activities Visual Studio Code ▾ May 3 23:53

File Edit Selection View Go Run Terminal Help

EXPLORER COOKIPIDIA lib api mock_cookipidia_service.dart

login_screen.dart message.dart M pubspec.yaml M mock_cookipidia_service.dart

```
lib > api > mock_cookipidia_service.dart > ...
1 import 'dart:convert';
2
3 import 'package:flutter/services.dart';
4
5 import '../models/models.dart';
6
7 // Mock recipe service that grabs sample json data to mock recipe request/response
8 class MockCookipidiaService {
9   // Batch request that gets both today recipes and friend's feed
10  Future<ExploreData> getExploreData() async {
11    final todayRecipes = await _getTodayRecipes();
12    final friendPosts = await _getFriendFeed();
13
14    return ExploreData(todayRecipes, friendPosts);
15  }
16
17 // Get sample explore recipes json to display in ui
18 Future<List<ExploreRecipe>> _getTodayRecipes() async {
19   // Simulate api request wait time
20   await Future.delayed(const Duration(milliseconds: 1000));
21   // Load json from file system
22   final jsonString =
23     await _loadAsset('assets/sample_data/sample_explore_recipes.json');
24   // Decode to json
25   final Map<String, dynamic> json = jsonDecode(jsonString);
26
27   // Go through each recipe and convert json to ExploreRecipe object,
28   if (json['recipes'] != null) {
29     final recipes = <ExploreRecipe>[];
30     json['recipes'].forEach((v) {
31       recipes.add(ExploreRecipe.fromJson(v));
32     });
33     return recipes;
34   } else {
35     return [];
36   }
37 }
```

Ln 1, Col 1 Spaces:2 UTF-8 LF Dart Dart DevTools Flutter:2.10.4 SM A750F (android-arm64)

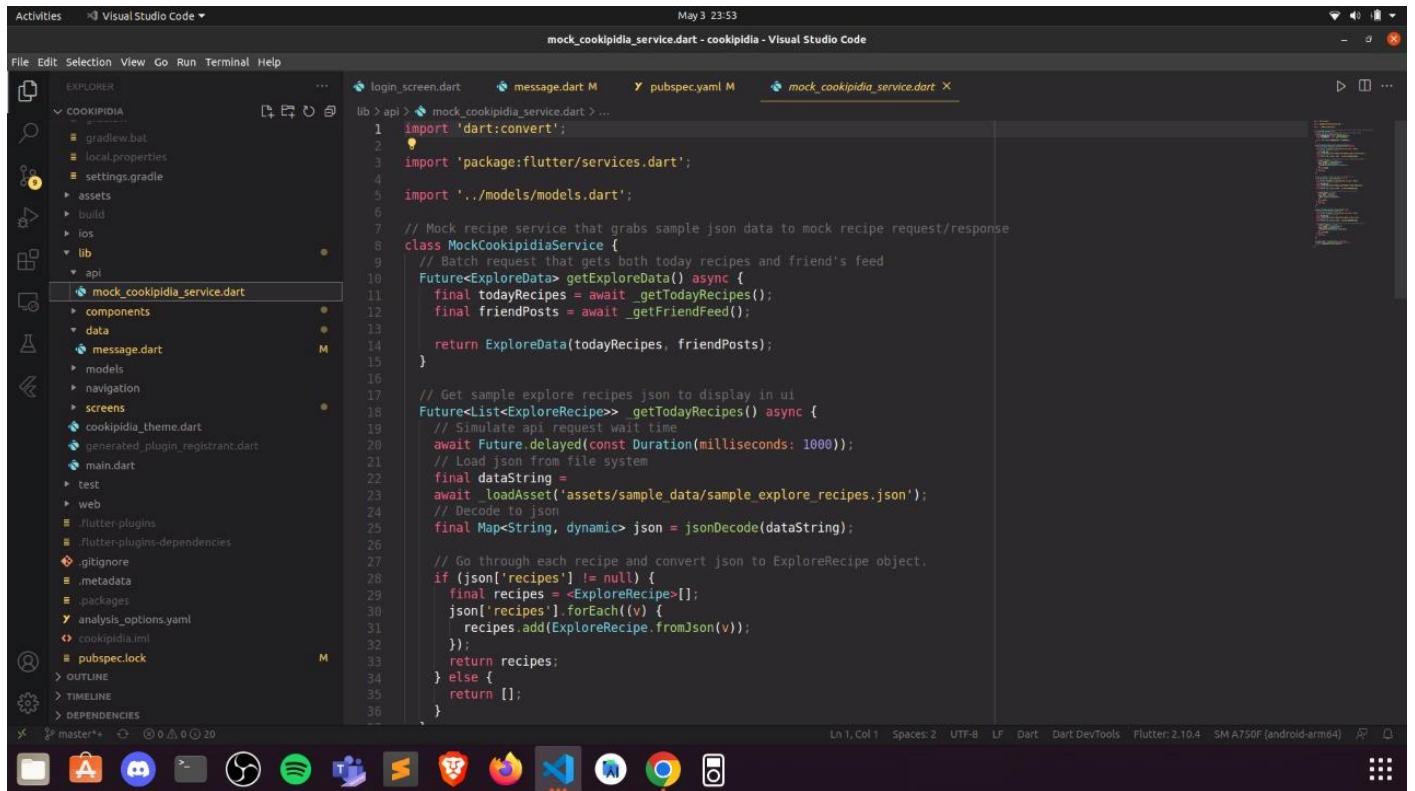


Fig. 132

Activities Visual Studio Code ▾ May 3 23:53

File Edit Selection View Go Run Terminal Help

EXPLORER COOKIPIDIA lib api components author_card.dart

login_screen.dart message.dart M pubspec.yaml M author_card.dart

```
lib > components > author_card.dart > AuthorCard > AuthorCard
1 import 'package:flutter/material.dart';
2 import '../cookipidia_theme.dart';
3 import 'circle_image.dart';
4
5 class AuthorCard extends StatefulWidget {
6   final String authorName;
7   final String title;
8   final ImageProvider? imageProvider;
9
10  const AuthorCard({
11    Key? key,
12    required this.authorName,
13    required this.title,
14    this.imageProvider,
15  }) : super(key: key);
16
17 @override
18 State<AuthorCard> createState() => _AuthorCardState();
19
20 class _AuthorCardState extends State<AuthorCard> {
21   bool _isFavorited = false;
22   @override
23   Widget build(BuildContext context) {
24     return Container(
25       padding: const EdgeInsets.all(16),
26       child: Row(
27         mainAxisSize: MainAxisSize.spaceBetween,
28         children: [
29           // 1
30           Row(
31             children: [
32               CircleImage(
33                 imageProvider: widget.imageProvider,
34                 imageRadius: 25,
35               ), // CircleImage
36             ],
37           ),
38         ],
39       ),
40     );
41   }
42 }
```

Ln 13, Col 25 Spaces:2 UTF-8 LF Dart Dart DevTools Flutter:2.10.4 SM A750F (android-arm64)

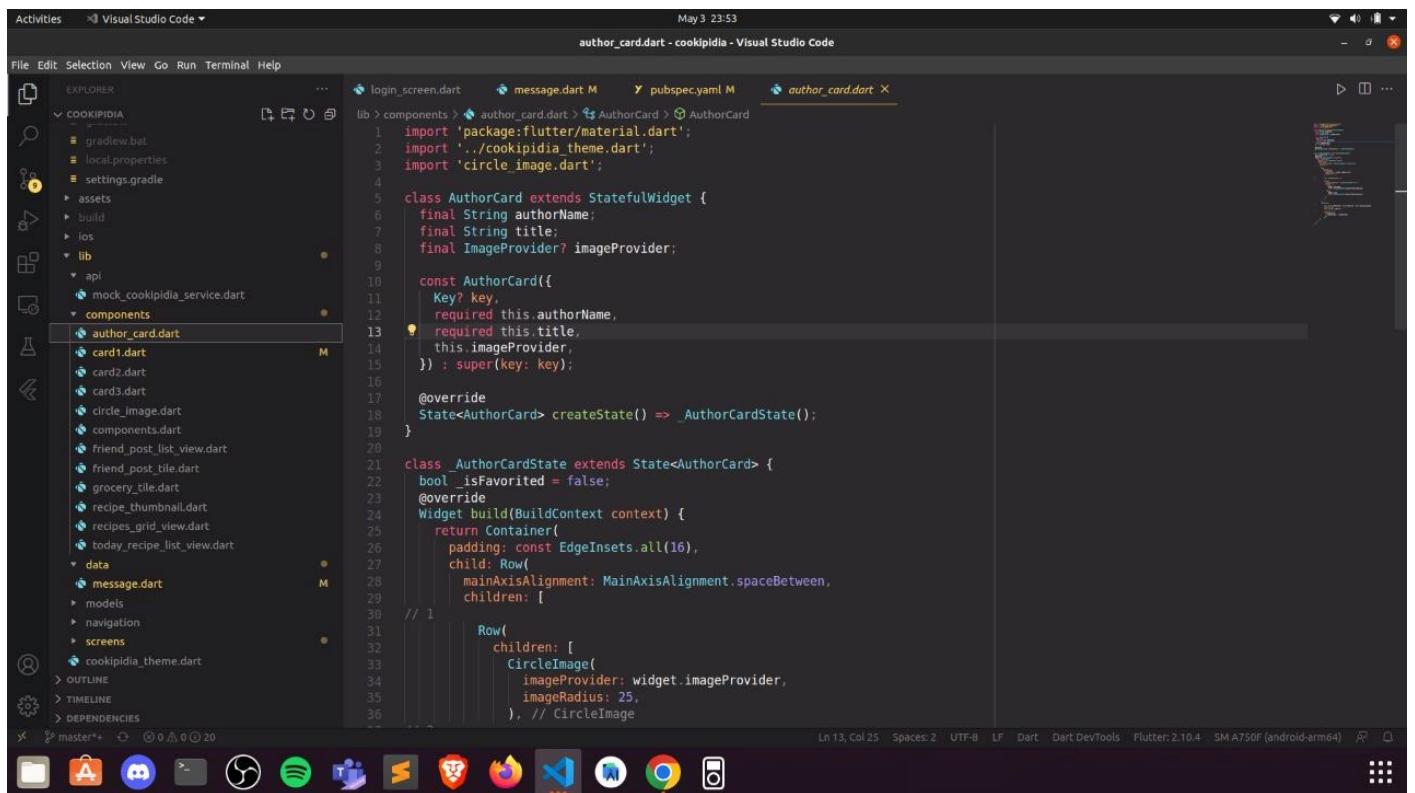


Fig. 133

The screenshot shows a Visual Studio Code interface with the following details:

- Activity Bar:** Activities, Visual Studio Code, May 3 23:53.
- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer Sidebar:** COOKIPIDIA folder structure:
 - gradlew.bat
 - local.properties
 - settings.gradle
 - assets
 - build
 - ios
 - lib
 - api
 - mock_cookipidia_service.dart
 - components
 - author_card.dart
 - card1.dart
 - card2.dart
 - card3.dart
 - circle_image.dart
 - components.dart
 - friend_post_list_view.dart
 - Friend_post_tile.dart
 - grocery_tile.dart
 - recipe_thumbnail.dart
 - recipes_grid_view.dart
 - today_recipe_list_view.dart
 - data
 - message.dart
 - models
 - navigation
 - screens
 - cookipidia_theme.dart
- Code Editor:** card1.dart - cookipidia - Visual Studio Code. The code defines a StatelessWidget named Card1 that displays a centered Container with a Stack containing Text and Positioned children for subtitle and title.
- Bottom Bar:** master+, OUTLINE, TIMELINE, DEPENDENCIES, Ln 9, Col 19, Spaces: 2, UTF-8, LF, Dart, Dart DevTools, Flutter: 2.10.4, SM A750F (android-arm64), and a set of icons for file operations.

Fig. 134

Activities Visual Studio Code - May 3, 23:53

File Edit Selection View Go Run Terminal Help

EXPLORER

COOKIPIDIA

- gradlew.bat
- local.properties
- settings.gradle
- assets
- build
- ios
- lib
 - api
 - mock_cookipidia_service.dart
 - components
 - author_card.dart
 - card1.dart M
 - card2.dart
 - card3.dart
 - circle_image.dart
 - components.dart
 - Friend_post_list_view.dart
 - friend_post_tile.dart
 - grocery_tile.dart
 - recipe_thumbnail.dart
 - recipes_grid_view.dart
 - today_recipe_list_view.dart
- data
- message.dart M
- models
- navigation
- screens
- cookipidia_theme.dart

OUTLINE

TIMELINE

DEPENDENCIES

master* 0 0 0 0 20

login_screen.dart message.dart M pubspec.yaml M card1.dart M

```
lib > components > card1.dart > Card1
  style: CookipidiaTheme.darkTextTheme.bodyText1,
    ), // Text
    bottom: 10,
    right: 0,
  ) // Positioned
),
), // Stack
padding: const EdgeInsets.all(16),
constraints: const BoxConstraints.expand(
  width: 350,
  height: 450,
), // BoxConstraints.expand
decoration: BoxDecoration(
  image: DecorationImage(
    image: AssetImage(recipe.backgroundImage),
    fit: BoxFit.cover,
  ), // DecorationImage
  borderRadius: const BorderRadius.all(Radius.circular(10.0)),
), // BoxDecoration
), // Container
); // Center
}
```

Ln 9, Col 19 Spaces: 2 UTF-8 LF Dart Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64)

Fig. 135

Activities Visual Studio Code May 3 23:53 card2.dart - cookipidia - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER COOKIDIA lib > components > card2.dart > ...

```
30   AuthorCard( 31     authorName: recipe.authorName, 32     title: recipe.role, 33     imageProvider: AssetImage(recipe.profileImage), 34   ), // AuthorCard 35   Expanded( 36     child: Stack( 37       children: [ 38         Positioned( 39           bottom: 16, 40           right: 16, 41           child: Text( 42             recipe.title, 43             style: CookipidiaTheme.lightTextTheme.headline1, 44           ), // Text 45         Positioned( 46           bottom: 70, 47           left: 16, 48           child: RotatedBox( 49             quarterTurns: 3, 50             child: Text( 51               recipe.subtitle, 52               style: CookipidiaTheme.lightTextTheme.headline1, 53             ), // Text 54           ), // RotatedBox 55         ), // Positioned 56       ], 57     ), // Stack 58   ), // Expanded 59   ], // Column 60   ), // Container 61   ); // Center 62 } // Center 63 ); // Center 64 } // Center 65 }
```

master+ 0 0 0 0 20

login_screen.dart message.dart M pubspec.yaml M card2.dart

Line 1, Col 1 Spaces: 2 UTF-8 LF Dart Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64)

Fig. 136

Activities Visual Studio Code May 3 23:53 card3.dart - cookipidia - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER COOKIDIA lib > components > card3.dart > ...

```
27   } @override 28   Widget build(BuildContext context) { 29     return Center( 30       child: Container( 31         constraints: const BoxConstraints.expand( 32           width: 350, 33           height: 450, 34         ), // BoxConstraints.expand 35         decoration: BoxDecoration( 36           image: DecorationImage( 37             image: AssetImage(recipe.backgroundImage), 38             fit: BoxFit.cover, 39           ), // DecorationImage 40           borderRadius: const BorderRadius.all( 41             Radius.circular(10.0), 42           ), // BorderRadius.all 43         ), // BoxDecoration 44         child: Stack( 45           children: [ 46             Container( 47               decoration: BoxDecoration( 48                 color: Colors.black.withOpacity(0.6), 49                 borderRadius: const BorderRadius.all(Radius.circular(10.0)), 50               ), // BoxDecoration 51             Container( 52               padding: const EdgeInsets.all(16), 53               child: Column( 54                 crossAxisAlignment: CrossAxisAlignment.start, 55                 children: [ 56                   const Icon( 57                     Icons.book, 58                     color: Colors.white, 59                     size: 40, 60                   ), // Icon 61                   const SizedBox(height: 8) 62                 ], // children 63               ), // Column 64             ), // Container 65           ], // children 66         ), // Stack 67       ), // Center 68     ); // Center 69   } // build 70 }
```

master+ 0 0 0 0 20

login_screen.dart message.dart M pubspec.yaml M card3.dart

Line 1, Col 1 Spaces: 2 UTF-8 LF Dart Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64)

Fig. 137

A screenshot of the Visual Studio Code interface. The title bar shows "Activities" and "Visual Studio Code". The status bar at the bottom right indicates "May 3 23:53", "Ln 15, Col 25", "Spaces: 2", "UTF-8", "LF", "Dart", "Dart DevTools", "Flutter: 2.10.4", and "SM A750F (android-arm64)". The left sidebar has sections for "EXPLORER", "OUTLINE", "TIMELINE", and "DEPENDENCIES". The main editor area displays the Dart file "circle_image.dart" with the following code:

```
lib > components > circle_image.dart > CircleImage > build
1 import 'package:flutter/material.dart';
2 class CircleImage extends StatelessWidget {
3   const CircleImage({
4     Key? key,
5     this.imageProvider,
6     this.imageRadius = 20,
7   }) : super(key: key);
8
9   final double imageRadius;
10  final ImageProvider? imageProvider;
11
12  @override
13  Widget build(BuildContext context) {
14    // 3
15    return CircleAvatar(
16      backgroundColor: Colors.white,
17      radius: imageRadius,
18    ); // CircleAvatar
19
20    // 4
21    child: CircleAvatar(
22      radius: imageRadius - 5,
23      backgroundImage: imageProvider,
24    ); // CircleAvatar
25  }
26
27
```

Fig. 138

A screenshot of the Visual Studio Code interface. The title bar shows "Activities" and "Visual Studio Code". The status bar at the bottom right indicates "May 3 23:53", "Ln 1, Col 1", "Spaces: 2", "UTF-8", "LF", "Dart", "Dart DevTools", "Flutter: 2.10.4", and "SM A750F (android-arm64)". The left sidebar has sections for "EXPLORER", "OUTLINE", "TIMELINE", and "DEPENDENCIES". The main editor area displays the Dart file "friend_post_list_view.dart" with the following code:

```
lib > components > friend_post_list_view.dart > ...
9 const FriendPostListView({
10   Key? key,
11   required this.friendPosts,
12 }) : super(key: key);
13
14 @override
15 Widget build(BuildContext context) {
16   // 2
17   return Padding(
18     padding: const EdgeInsets.only(
19       left: 16,
20       right: 16,
21       top: 0,
22     ), // EdgeInsets.only
23   // 3
24   child: Column(
25     crossAxisAlignment: CrossAxisAlignment.start,
26     children: [
27     // 4
28     Text('Social Chefs ! ', style: Theme.of(context).textTheme.headline1),
29     // 5
30     const SizedBox(height: 16),
31     // TODO: Add PostlistView here
32     ListView.separated(
33       // 2
34       primary: false,
35       // 3
36       physics: const NeverScrollableScrollPhysics(),
37       // 4
38       shrinkWrap: true,
39       scrollDirection: Axis.vertical,
40       itemCount: friendPosts.length,
41       itemBuilder: (context, index) {
42         // 5
43         final post = friendPosts[index];
44         return FriendPostTile(post: post);
45       }
46     );
47   );
48 }
```

Fig. 139

The screenshot shows a Visual Studio Code interface with the following details:

- Activity Bar:** Activities, Visual Studio Code, May 3 23:53.
- File Explorer:** Shows the project structure under 'COOKIPIDIA'. The 'friend_post_tile.dart' file is selected and highlighted in the tree view.
- Code Editor:** Displays the 'friend_post_tile.dart' file content. The code defines a StatelessWidget named 'FriendPostTile' that extends 'StatelessWidget'. It uses 'CircleImage' and 'Text' widgets to display a post's profile image and comment respectively.
- Bottom Bar:** Includes icons for GitHub, Atom, Docker, Spotify, Trello, Jenkins, Firefox, VS Code, Microsoft Edge, and Instagram.
- Status Bar:** L1, Col 1 | Spaces: 2 | UTF-8 | LF | Dart | Dart DevTools | Flutter: 2.10.4 | SM A750F (android-arm64) | ?

Fig. 140

Fig. 141

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Bar:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for 'COOKIPIDIA'. The 'lib' folder contains several files and subfolders like 'components', 'models', 'screens', etc.
- Code Editor:** The file 'recipe_thumbnail.dart' is open. The code defines a StatelessWidget named RecipeThumbnail that builds a Container with a Column containing an Expanded widget with a ClipRRect child and an Image.asset child.
- Bottom Bar:** Includes tabs for 'login_screen.dart', 'message.dart', 'pubspec.yaml', and 'recipe_thumbnail.dart'. Status bar at the bottom right shows 'Flutter: 2.10.4' and 'SM A750F (android-arm64)'.

Fig. 142

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Bar:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for 'COOKIPIDIA_TEST_1'. The 'assets' folder contains 'recipelist.json'.
- Code Editor:** The file 'recipelist.json' is open. It is a JSON object with properties like 'q', 'from', 'to', 'more', 'count', and 'hits'. The 'hits' array contains objects representing recipes, with one entry shown in detail.
- Bottom Bar:** Includes tabs for 'main.dart' and 'recipelist.json'. Status bar at the bottom right shows 'bash + x' and 'utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1\$'.

Fig. 143

```
May 3 23:51
recipes1.json - cookipidia_test_1 - Visual Studio Code

File Edit Selection View Go Run Terminal Help
File Explorer Problems Output Debug Console Terminal
utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1$ bash + x

main.dart recipes1.json
assets > recipes1.json > hits > 0 > recipe > dietLabels > 0
  30   "2 large russet potatoes, peeled and cut into chunks",
  31   "1 3-4 pound chicken, cut into 8 pieces (or 3 pound chicken legs)",
  32   "3/4 cup white wine",
  33   "3/4 cup chicken stock",
  34   "3 tablespoons chopped parsley",
  35   "1 tablespoon dried oregano",
  36   "Salt and pepper",
  37   "1 cup frozen peas, thawed"
],
"ingredients": [
  {
    "text": "1/2 cup olive oil",
    "weight": 108.0
  },
  {
    "text": "5 cloves garlic, peeled",
    "weight": 15.0
  },
  {
    "text": "2 large russet potatoes, peeled and cut into chunks",
    "weight": 738.0
  },
  {
    "text": "1 3-4 pound chicken, cut into 8 pieces (or 3 pound chicken legs)",
    "weight": 1587.573295000001
  },
  {
    "text": "3/4 cup white wine",
    "weight": 176.3999999999998
  },
  {
    "text": "3/4 cup chicken stock",
    "weight": 108.0
  }
]
Ln 18, Col 21 Spaces: 2 UTF-8 LF JSON Dart DevTools SM A750F (android-arm64) P D
```

Fig. 144

```
May 3 23:51
recipes2.json - cookipidia_test_1 - Visual Studio Code

File Edit Selection View Go Run Terminal Help
File Explorer Problems Output Debug Console Terminal
utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1$ bash + x

main.dart recipes2.json
lib > main.dart > MyApp > build
  24   Logger.root.level = Level.ALL;
  25   Logger.root.onRecord.listen((rec) {
  26     print('${rec.level.name}: ${rec.time}: ${rec.message}');
  27   });
  28 }
  29
  30 class MyApp extends StatelessWidget {
  31   final Repository repository;
  32   const MyApp({
  33     Key? key,
  34     required this.repository,
  35   }) : super(key: key);
  36
  37   // This widget is the root of your application.
  38   @override
  39   Widget build(BuildContext context) {
  40     return MultiProvider(
  41       providers: [
  42         // 1
  43         Provider<Repository>(
  44           lazy: false,
  45           create: () => repository,
  46           // 2
  47           dispose: (_, Repository repository) => repository.close(),
  48           // 3
  49           ), // Provider
  50           Provider<ServiceInterface>(
  51             create: () => RecipeService.create(),
  52             lazy: false,
  53             ), // Provider
  54           ), // Provider
  55         ],
  56       );
  57     }
  58   }
  59 }

Ln 41, Col 5 Spaces: 2 UTF-8 LF Dart Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64) P D
```

Fig. 145

```

File Edit Selection View Go Run Terminal Help
File Explorer Terminal
main.dart pubspec.yaml
COOKIPIDIA_TEST_1
  ingredient.dart
  models.dart
  recipe.dart
  moor
    moor_db.dart
    moor_db.g.dart
    moor_repository.dart
  sqlite
  memory_repository.dart
  repository.dart
  mock_service
  network
  ui
  generated_plugin_registrant.dart
  main.dart
  test
  web
  .flutter-plugins
  .flutter-plugins-dependencies
  .gitignore
  .metadata
  .packages
  analysis_options.yaml
  cookipidia_test_1.iml
  pubspec.lock
  pubspec.yaml
  README.md
  OUTLINE
  TIMELINE
  DEPENDENCIES
  3 2.0 36
  bash + x
  PROBLEMS 38 OUTPUT DEBUG CONSOLE TERMINAL
  utkarsh@utkarsh-nitro:~/AndroidStudioProjects/cookipidia_test_1$ Ln 68, Col 27 Spaces: 2 UTF-8 LF YAML Dart DevTools Flutter: 2.10.4 SM A750F (android-arm64) P

```

The screenshot shows the Visual Studio Code interface with the file `pubspec.yaml` open. The code defines several dependencies:

```

dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  cached_network_image: ^3.1.0
  flutter_slidable: ^0.6.0
  flutter_svg: ^0.22.0
  shared_preferences: ^2.0.13
  json_annotation: ^4.4.0
  chopper: ^4.0.5
  logging: ^1.0.2
  provider: ^6.0.2
  equatable: ^2.0.3

  sqflite: ^2.0.2
  path_provider: ^2.0.9
  synchronized: ^3.0.0+2
  sqflite: ^2.2.0

  moor_flutter: ^4.1.0

  dev_dependencies:
    flutter_test:
      sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the "analysis_options.yaml" file located at the root of your
  # project.
  flutter_lints: ^1.0.0

```

Fig. 146

```

File Edit Selection View Go Run Terminal Help
File Explorer Terminal
manifest.json
RAYCHAT
  ui
    login.dart
    message_list.dart
    message_widget.dart
    generated_plugin_registrant.dart
    main.dart
  test
  web
    Icons
      Icon-192.png
      Icon-512.png
      Icon-maskable-192.png
      Icon-maskable-512.png
      favicon.png
    index.html
  manifest.json
  windows
  .flutter-plugins
  .flutter-plugins-dependencies
  .gitignore
  .metadata
  .packages
  analysis_options.yaml
  pubspec.lock
  pubspec.yaml
  raychat.iml
  README.md
  OUTLINE
  TIMELINE
  DEPENDENCIES
  master 1 0 12
  bash + x
  PROBLEMS 13 OUTPUT DEBUG CONSOLE TERMINAL
  utkarsh@utkarsh-nitro:~/AndroidStudioProjects/raychat$ Ln 1, Col 1 Spaces: 4 UTF-8 LF JSON Dart DevTools SM A750F (android-arm64) P

```

The screenshot shows the Visual Studio Code interface with the file `manifest.json` open. The JSON object defines a maskable icon:

```

{
  "web": {
    "manifest": {
      "icons": [
        {
          "src": "icons/Icon-maskable-512.png",
          "sizes": "512x512",
          "type": "image/png",
          "purpose": "maskable"
        }
      ]
    }
  }
}

```

Fig. 147

The screenshot shows the Visual Studio Code interface with the following details:

- Activity Bar:** Activities, Visual Studio Code.
- Header:** May 4 16:26, CMakeLists.txt - raychat - Visual Studio Code.
- File Explorer:** Shows the project structure under RAYCHAT:
 - ui
 - login.dart
 - message_list.dart
 - message_widget.dart
 - generated_plugin_registrant.dart
 - main.dart- test
- web
- icons
 - Icon-192.png
 - Icon-512.png
 - Icon-maskable-192.png
 - Icon-maskable-512.png
- favicon.png
- index.html
- manifest.json
- windows
- flutter
 - generated_plugin_symlinks
- ephemeral/.plugin_symlinks
- CMakeLists.txt
- generated_plugin_registrant.cc
- generated_plugin_registrant.h
- generated_plugins.cmake
- runner
- .gitignore
- CMakeLists.txt
- .flutter-plugins
- .flutter-plugins-dependencies

- Editor:** The CMakeLists.txt file is open, showing code for a Flutter project. It includes configurations for CMake, Flutter libraries, and target definitions.
- Bottom Bar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), bash, +, Timeline, Dependencies.
- Bottom Status Bar:** utkarsh@utkarsh-nitro:~/AndroidStudioProjects/raychat\$
- Bottom Icons:** A row of icons for various tools and services.

Fig. 148

```
// Generated file. Do not edit.  
// clang-format off  
#include "generated_plugin_registrant.h"  
void RegisterPlugins(flutter::PluginRegistry* registry) {  
}
```

Fig. 149

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Bar:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for "RAYCHAT".
- Editor:** The active file is "generated_plugin_registrant.h".
- Code:**

```
1 //  
2 // Generated file. Do not edit.  
3 //  
4 // clang-format off  
5 #ifndef GENERATED_PLUGIN_REGISTRANT_  
6 #define GENERATED_PLUGIN_REGISTRANT_  
7 #include <flutter/plugin_registry.h>  
8 // Registers Flutter plugins.  
9 void RegisterPlugins(flutter::PluginRegistry* registry);  
10 //endif // GENERATED_PLUGIN_REGISTRANT_
```
- Terminal:** bash
- Bottom:** A dock with various icons including a terminal, browser, and file manager.

Fig. 150

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- File Bar:** File Edit Selection View Go Run Terminal Help
- Explorer:** Shows the project structure for "runner".
- Editor:** The active file is "CMakeLists.txt".
- Code:**

```
1 cmake_minimum_required(VERSION 3.14)  
2 project(runner LANGUAGES CXX)  
3  
4 add_executable(${BINARY_NAME} WIN32  
5 "flutter_window.cpp"  
6 "main.cpp"  
7 "utils.cpp"  
8 "win32_window.cpp"  
9 "${FLUTTER_MANAGED_DIR}/generated_plugin_registrant.cc"  
10 "Runner.rc"  
11 "runner.exe.manifest"  
12 )  
13 apply_standard_settings(${BINARY_NAME})  
14 target_compile_definitions(${BINARY_NAME} PRIVATE "NOMINMAX")  
15 target_link_libraries(${BINARY_NAME} PRIVATE flutter flutter_wrapper_app)  
16 target_include_directories(${BINARY_NAME} PRIVATE "${CMAKE_SOURCE_DIR}")  
17 add_dependencies(${BINARY_NAME} flutter assemble)
```
- Terminal:** bash
- Bottom:** A dock with various icons including a terminal, browser, and file manager.

Fig. 151

Activities Visual Studio Code May 4 16:26 flutter_window.cpp - raychat - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER RAYCHAT windows flutter resources runner

Icon-192.png Icon-512.png Icon-maskable-192.png Icon-maskable-512.png favicon.png index.html manifest.json windows flutter ephemeral/.plugin_symlinks CMakeLists.txt generated_plugin_registrant.cc generated_plugin_registrant.h generated_plugins.cmake runner resources app_icon.ico CMakeLists.txt flutter_window.cpp flutter_window.h main.cpp resource.h runner.exe.manifest Runner.rc utils.cpp utils.h win32_window.cpp win32_window.h

flutter_window.cpp

```
1 #include "flutter_window.h"
2
3 #include <optional>
4
5 #include "flutter/generated_plugin_registrant.h"
6
7 FlutterWindow::FlutterWindow(const flutter::DartProject& project)
8     : project_(project) {}
9
10 FlutterWindow::~FlutterWindow() {}
11
12 bool FlutterWindow::OnCreate() {
13     if (!Win32Window::OnCreate()) {
14         return false;
15     }
16
17     RECT frame = GetClientArea();
18
19     // The size here must match the window dimensions to avoid unnecessary surface
20     // creation / destruction in the startup path.
21     flutter_controller_ = std::make_unique<flutter::FlutterViewController>(
22         frame.right - frame.left, frame.bottom - frame.top, project_);
23     // Ensure that basic setup of the controller was successful.
24     if (!flutter_controller_->engine() || !flutter_controller_->view()) {
25         return false;
26     }
27     RegisterPlugins(flutter_controller_->engine());
28     SetChildContent(flutter_controller_->view()->GetNativeWindow());
29     return true;
30 }
31
32 void FlutterWindow::OnDestroy() {
33 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

utkarsh@utkarsh-nitro:~/AndroidStudioProjects/raychat\$

Ln 1, Col 1 Spaces: 2 UTF-8 LF C++ Dart DevTools SM A750F (android-arm64) Linux

Fig. 152

Activities Visual Studio Code May 4 16:26 flutter_window.cpp - raychat - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER RAYCHAT windows flutter resources runner

Icon-192.png Icon-512.png Icon-maskable-192.png Icon-maskable-512.png favicon.png index.html manifest.json windows flutter ephemeral/.plugin_symlinks CMakeLists.txt generated_plugin_registrant.cc generated_plugin_registrant.h generated_plugins.cmake runner resources app_icon.ico CMakeLists.txt flutter_window.cpp flutter_window.h main.cpp resource.h runner.exe.manifest Runner.rc utils.cpp utils.h win32_window.cpp win32_window.h

flutter_window.cpp

```
1 #include "flutter_window.h"
2
3 #include <optional>
4
5 #include "flutter/generated_plugin_registrant.h"
6
7 FlutterWindow::FlutterWindow(const flutter::DartProject& project)
8     : project_(project) {}
9
10 FlutterWindow::~FlutterWindow() {}
11
12 bool FlutterWindow::OnCreate() {
13     if (!Win32Window::OnCreate()) {
14         return false;
15     }
16
17     RECT frame = GetClientArea();
18
19     // The size here must match the window dimensions to avoid unnecessary surface
20     // creation / destruction in the startup path.
21     flutter_controller_ = std::make_unique<flutter::FlutterViewController>(
22         frame.right - frame.left, frame.bottom - frame.top, project_);
23     // Ensure that basic setup of the controller was successful.
24     if (!flutter_controller_->engine() || !flutter_controller_->view()) {
25         return false;
26     }
27     RegisterPlugins(flutter_controller_->engine());
28     SetChildContent(flutter_controller_->view()->GetNativeWindow());
29     return true;
30 }
31
32 void FlutterWindow::OnDestroy() {
33     if (flutter_controller_) {
34         flutter_controller_ = nullptr;
35     }
36
37     Win32Window::OnDestroy();
38 }
39
40 LRESULT
41 FlutterWindow::MessageHandler(HWND hwnd, UINT const message,
42                                 WPARAM const wparam,
43                                 LPARAM const lparam) noexcept {
44     // Give Flutter, including plugins, an opportunity to handle window messages.
45     if (flutter_controller_) {
46         std::optional<LRESULT> result =
47             flutter_controller_->HandleTopLevelWindowProc(hwnd, message, wparam,
48                                                       lparam);
49
50         if (result) {
51             return *result;
52         }
53     }
54
55     switch (message) {
56     case WM_FONTCHANGE:
57         flutter_controller_->engine()->ReloadSystemFonts();
58         break;
59     }
60 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

utkarsh@utkarsh-nitro:~/AndroidStudioProjects/raychat\$

Ln 1, Col 1 Spaces: 2 UTF-8 LF C++ Dart DevTools SM A750F (android-arm64) Linux

Fig. 153

CHAPTER 4: RESULTS ANALYSIS AND VALIDATION

The app focuses on providing users with the basic idea of cooking ingredients. While also providing various users to communicate with each other and discuss about recipes. It also provides a list to maintain grocery items for the user. The app frontend is implemented using various widgets of flutter and for the backend it uses Firestore & Firebase.

As one can observe that the app is very much ready for daily usage incorporates and families. few features that can be added to make it more productive are:

1. Maps Integration to allow user to see which shop is nearby.
2. Audio call/ video call between people who are sharing a dish.
3. Export the dish final image as PNG/JPEG file.
4. Setting a teacher mode so that a professional chef can guide us properly how to cook.
5. Column for Invention of different kinds of recipes.

4.1. STEEPLE Analysis:

- Social
 - Increase social interaction and solidarity
 - Meet new people and make new friends through the Recipe Forum
- Technology
 - Smartphone penetration is increasing day after day
 - The application is accessible from anywhere using a smartphone
 - Real time communication between actors
- Environmental
 - Increase of awareness regarding healthier foods and recipes.
- Economical
 - My Cookery Master will enable users to try good and less costly recipes
- Political
 - My Cookery Master may help the government in sharing good and useful information through ads.
- Legal
 - Insurances of users.
- Ethical
 - Client confidentiality should be kept: all information related to users should only be communicated to their respective users.

4.2. Output of My Cookery Master

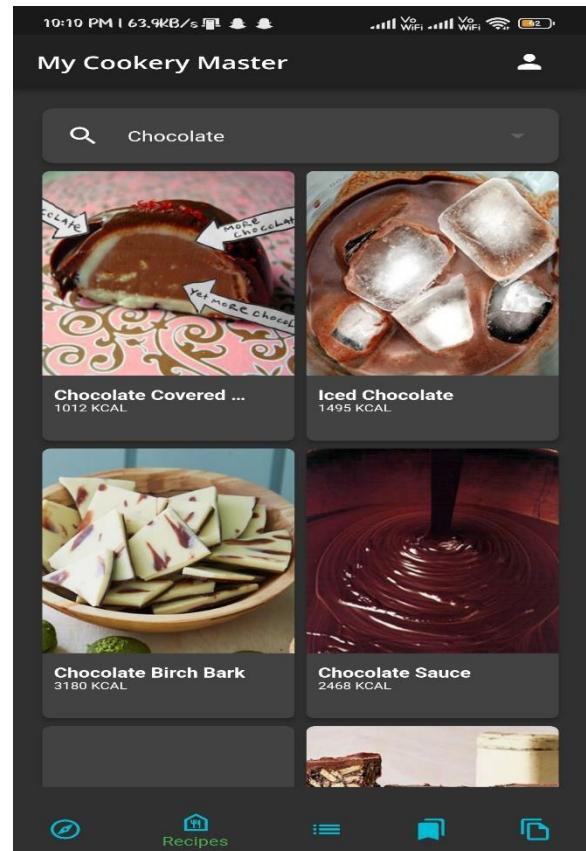
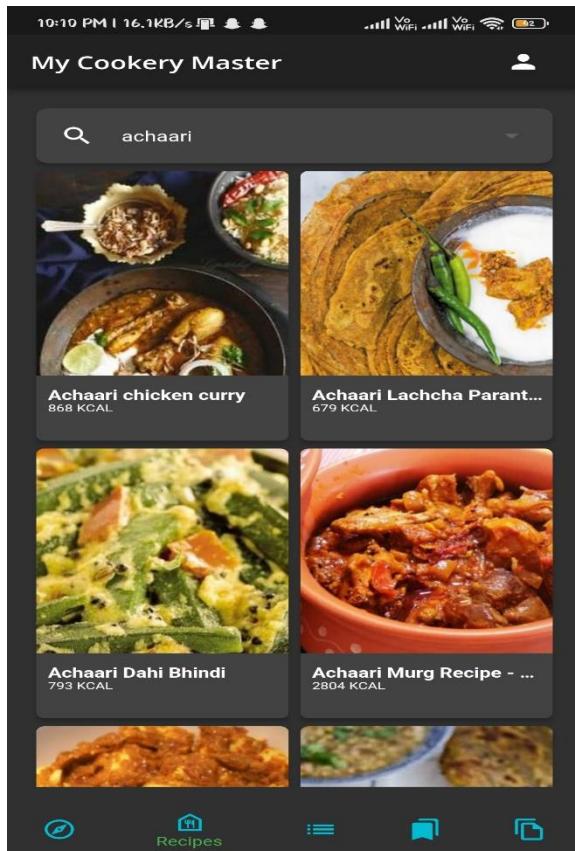


Fig. 154 & 155: Recipe Page (My Cookery Master)

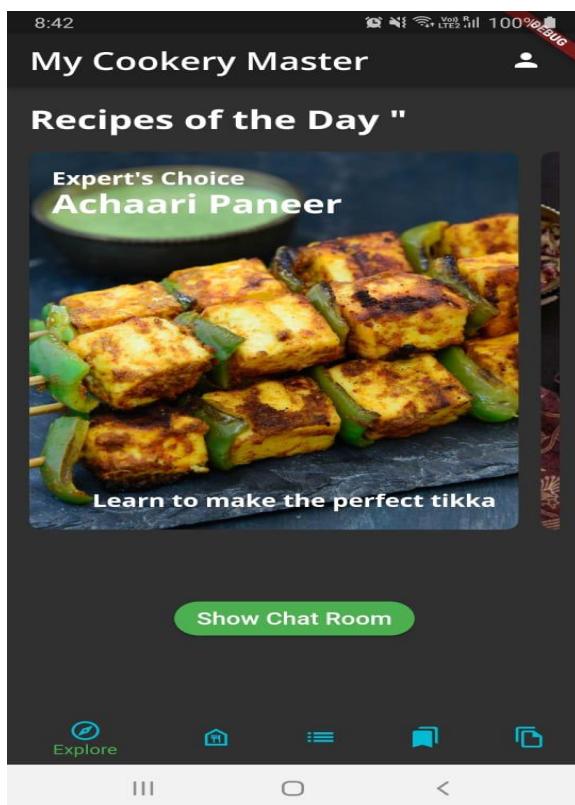


Fig. 156: Explore Page

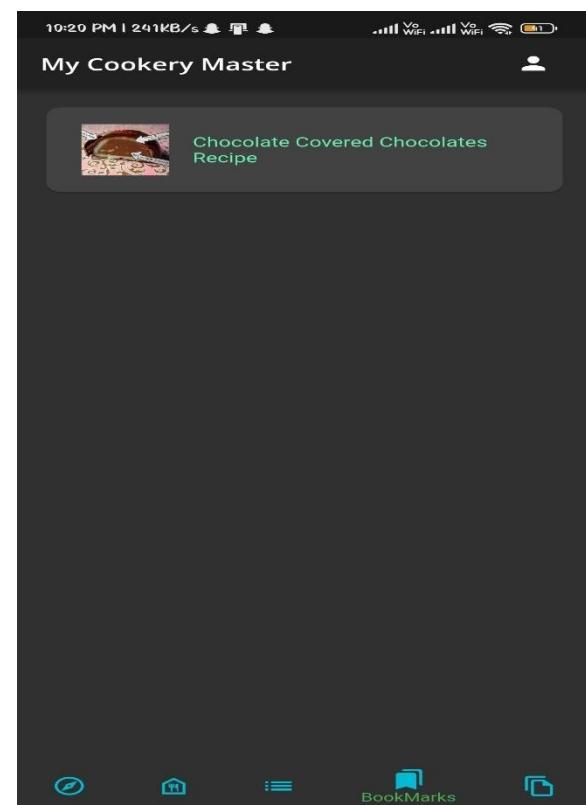


Fig. 157: Bookmarks Page

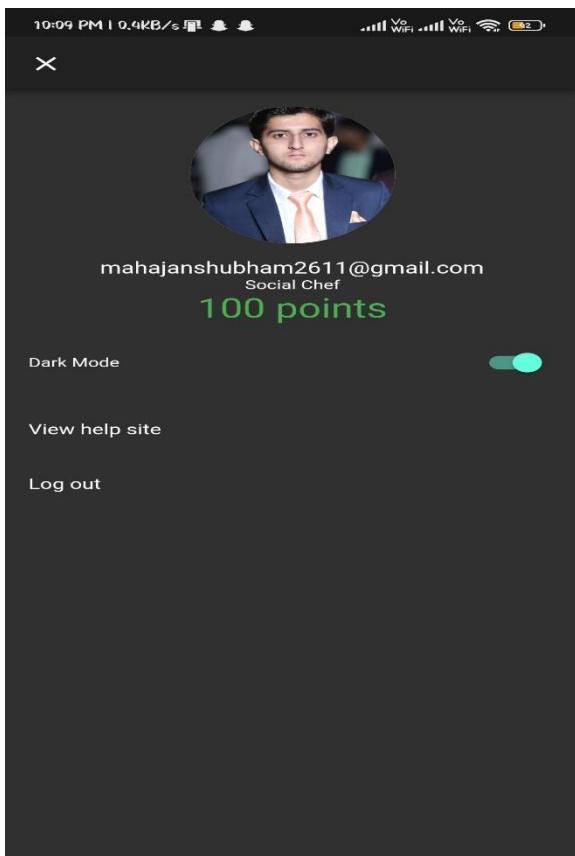


Fig. 158: Profile Page

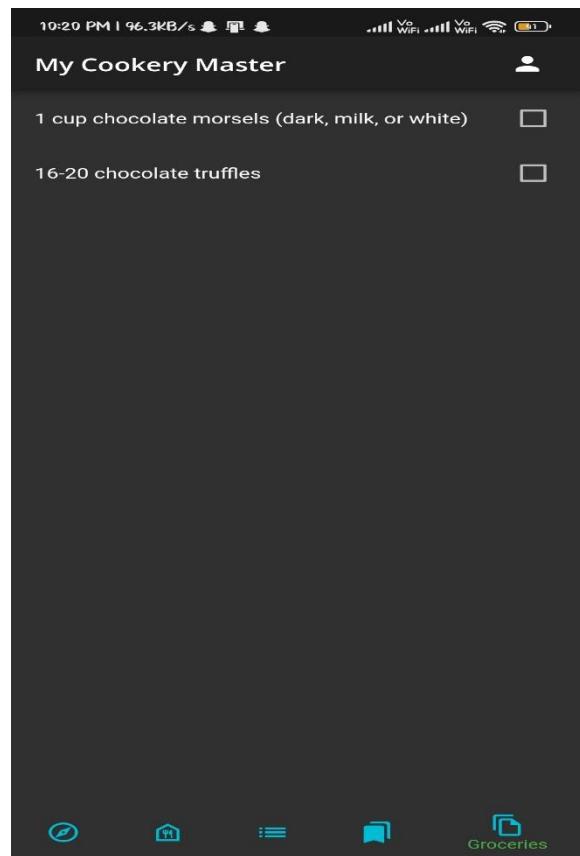


Fig. 159: Grocery Page

CHAPTER 5: CONCLUSION AND FUTURE WORKS

5.1. Conclusion: My Cookery Master application meets with the enterprise class application principles. It is designed to be performing, scalable, extensible, and highly available. It also ensures the privacy of the users' data and secures its access. Given that it may be improved in many ways, the application is also easily maintainable.

This document summarizes the work that has been done since the beginning of this semester. Indeed, it starts by giving an overview about the project specification and requirements. The document also states the methodology followed and which consists of 5 main parts: The first part will be devoted to data gathering and software requirements specification. Consequently, I will have a look at different mobile apps which target the same goal. They are plenty of Recipe Organizer apps. Each one has some various features. The second part will be dedicated to the design phase, including the app and the database. Also, in this phase, the software tools to be used will be specified. For example, the IDE, the database Server, the modelling language for the design, and finally the software testing tools. The third part will be the implementation phase, here, the design will be converted to code in order to develop the targeted app. The fourth step will be devoted to testing the app. In this phase, two testing methods will be used, namely: Black Box testing and White box testing. The last phase will be the deployment phase.

5.2. Future Works: As a future work, I am planning to persist in developing more mobile apps and entering deeply the world of Android development. My Cookery Master has helped me to gain a lot of development skills and enrich my background, as I spent the previous 4 months searching for every tiny detail that concerns the development of android application. Thankfully, I have built a good knowledge. Therefore, any upcoming project of mobile application development will undoubtedly be within my reach. In addition, as to the future of My Cookery Master, I will deploy it in Google Play Store and update the app from time to time if necessary. We have already started the deployment phase by buying a host for my database in HostGator server. I will export my database soon, and then buying an account in Google Play Store to publish the app. Also, we will monetize my application using Ad mob.

REFERENCES

1. Android Developer guide, from
<http://developer.android.com/> , accessed in March 2015
2. Android Tutorial, from
<http://www.tutorialspoint.com/android/>, accessed in March 2015
3. Android Activity Lifecycle Diagram, from
<http://www.javatpoint.com/images/androidimages/Android-Activity-Lifecycle.png> accessed in April 2015
4. Android Service Lifecycle Diagram, from
http://www.tutorialspoint.com/android/images/android_service.lifecycle.jpg, accessed in April 2015
5. Android Tutorial, from
<http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html> , accessed in March 2015
6. Marko Gargenta. Learning Android, O'Reilly Media, Inc, March 2011.
http://aiti.mit.edu/media/programs/indonesia-summer-2013/materials/gargenta_-_2011_-learning_android.pdf , accessed in March 2022
7. Android Developers Blog, from
<http://android-developers.blogspot.com/> , accessed in March 2015 [8] Food2Fork , from
http://food2fork.com/ , accessed in March 2022.
8. Gilad Bracha, Alex Buckley, James Gosling, Bill Joy, Guy Steele. 2015. The Java Language Specifications Java SE 8 Edition. Oracle America Inc., Redwood City, CA
9. Tim Bray, Eve Maler, Jean Paoli, C.M. Sperberg-McQueen, Francois Yergeau. 2008. Extensible Markup Language (XML) 1.0 (Fifth Edition). Retrieved May 6, 2018 from
<https://www.w3.org/TR/REC-xml/>
10. Technopedia. What is NoSQL? | Definition from Technopedia. Retrieved May 7, 2019 from
<https://www.techopedia.com/definition/27689/nosql-database>.
11. Technopedia. What is Concurrency? | Definition from Technopedia. Retrieved May 11, 2018 from
<https://www.techopedia.com/definition/25146/concurrency-programming>.
12. Navathe. Elmasri, "Fundamentals of Database Systems", Pearson Education, Inc. California, 2000.
13. TATLI, Ipek,"Food Recommendation System Project Report.", (2009).
14. Richard Fairley, "Software Engineering Concept", Publisher: Tata McGraw- Hill Education, 2001.
15. Roger S Pressman, "Software Engineering: A Practitioner's Approach" (first edition),1982.
16. Roger S Pressman, "Software Engineering: A beginner's guide" (1988).
17. De Almeida, Jorge Miguel Tavares Soares." Personalized Food Recommendations." (2015).
18. Lee Cheng, Teh and Yusof, Umi and Khalid, mohd nor akmal." Content-Based Filtering Algorithm for Mobile Recipe Application" 2014 8th Malaysian Software Engineering Conference, MySEC 2014.

19. Android Developers Blog, from <http://android-developers.blogspot.com/> accessed in March 2015
20. Food 2 Fork, from <http://food2fork.com/> accessed in March 2015
21. Merriam-Webster. Database | Definition of Database by Merriam-Webster. Retrieved May 4, 2018 from <https://www.merriam-webster.com/dictionary/database>
22. Firebase. Firebase Products. Retrieved May 4, 2018 from <https://firebase.google.com/products/>
23. Katherine Chou, Xavier Ducrohet, Tor Norbye. 2013. Android Developers Blog: Android Studio: An IDE built for Android. Retrieved May 5, 2018 from <https://androiddevelopers.googleblog.com/2013/05/android-studio-ide-built-for-android.html>.
24. Android Developers. Android Studio Features | Android Developers. Retrieved May 5, 2018 from <https://developer.android.com/studio/features/>
25. Jon Byous. 1999. JAVA TECHNOLOGY: THE EARLY YEARS. Internet Archive, Retrieved May 6, 2018 from <https://web.archive.org/web/20050420081440/http://java.sun.com/features/1998/05/birthday.html>
26. <https://flutter.dev/>
27. <https://dart.dev/>
28. <https://firebase.google.com/>
29. <https://www.cookinglight.com/cooking-101/12-cooking-skills-every-young-adult-should-learn>
30. <https://retrohousewifegoesgreen.com/basic-cooking-skills/>
31. <https://www.reportlinker.com/insight/americans-cooking-habits.html>
32. Android Developer guide, from <http://developer.android.com/> accessed in March 2015
33. Android Tutorial, from <http://www.tutorialspoint.com/android/> accessed in March 2015
34. Android Activity Lifecycle Diagram, from <http://www.javatpoint.com/images/androidimages/Android-Activity-Lifecycle.png> accessed in April 2015
35. Android Service Lifecycle Diagram, from http://www.tutorialspoint.com/android/images/android_service_lifecycle.jpg accessed in April 2015
36. Android Tutorial, from <http://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html> accessed in March 2015
37. Marko Gargenta Learning Android, O'Reilly Media, Inc, March 2011.
<http://aiti.mit.edu/media/programs/indonesia-summer-2013/materials/gargenta - 2011 - learning android.pdf> accessed in March 2015