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

Table of Content

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 know 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:

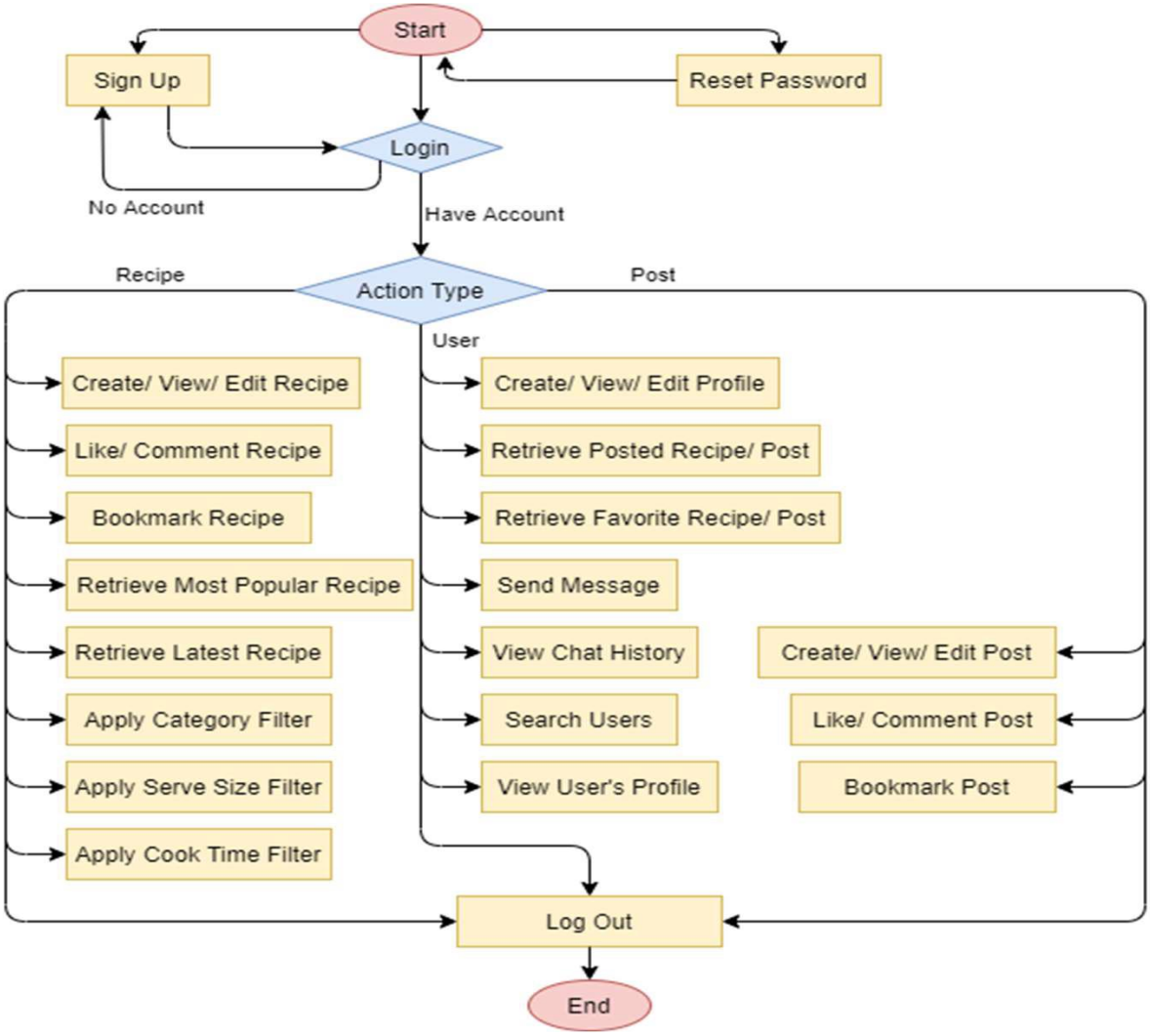


Fig. 1: System Flow Chart

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

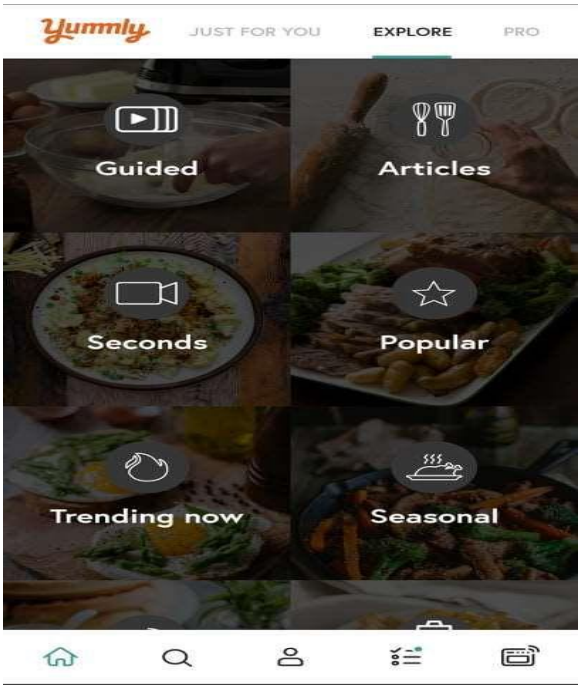


Fig.3: Explore Page

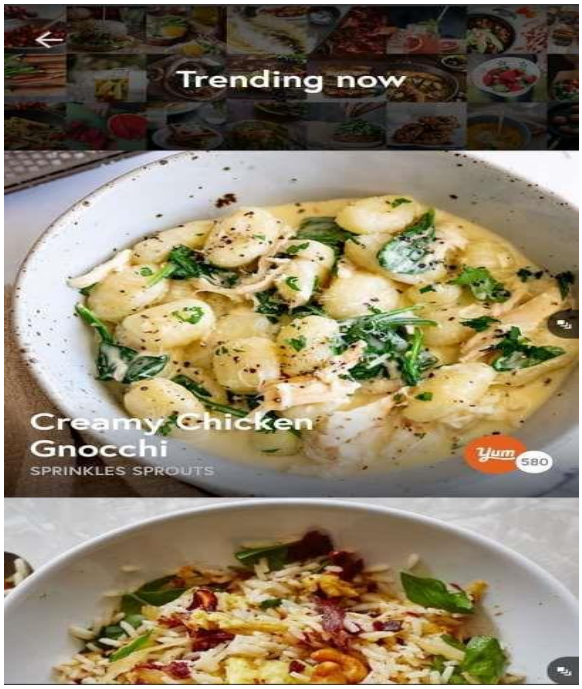


Fig. 4: Categorized Page

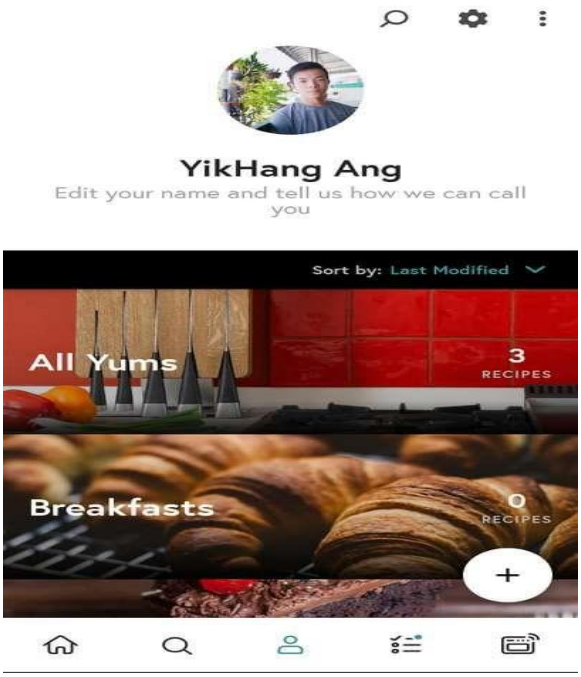


Fig. 5: Profile Page

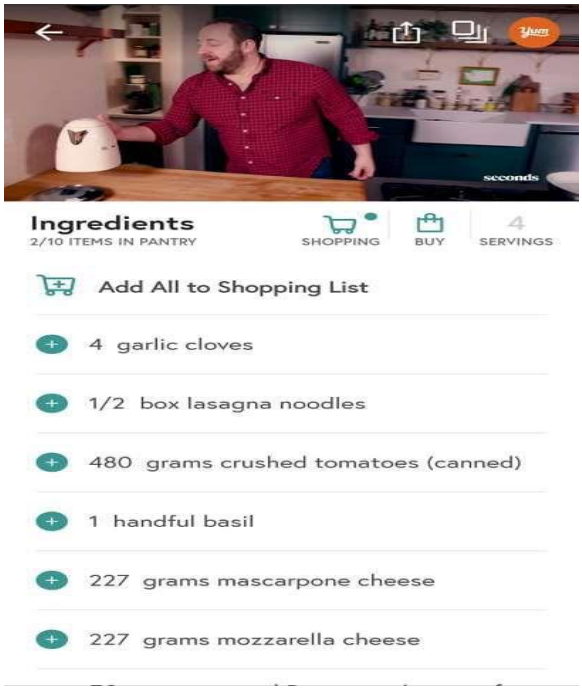


Fig. 6: Ingredient List

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.

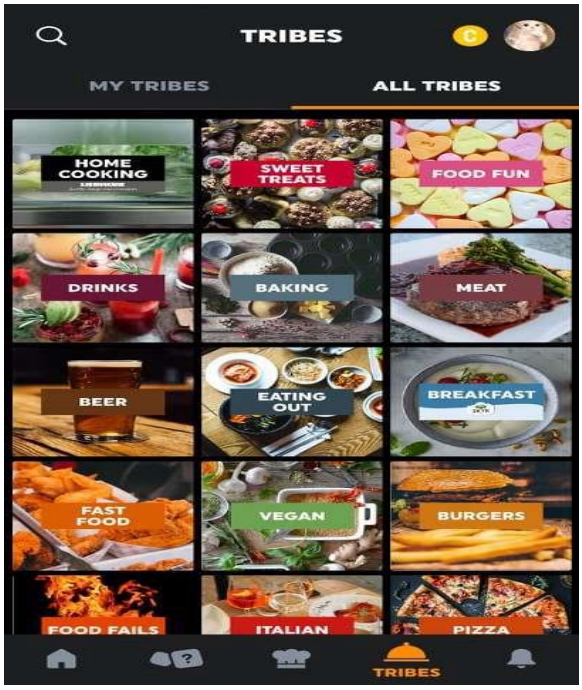


Fig. 8: Tribes Page

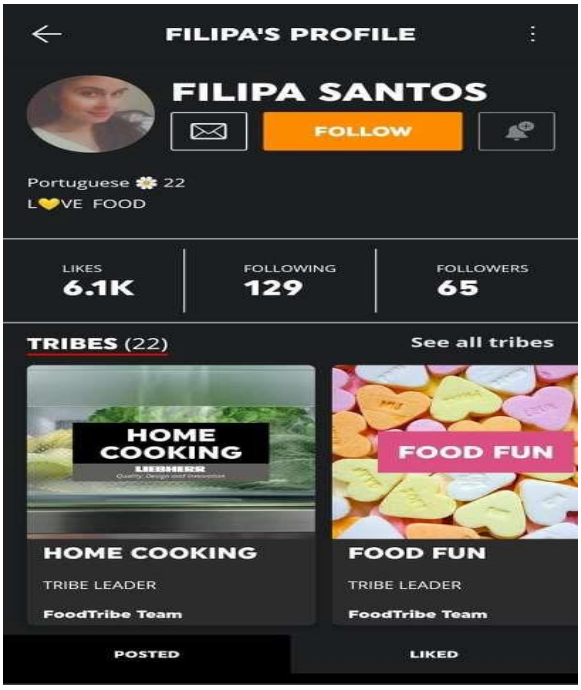


Fig. 9: User Page

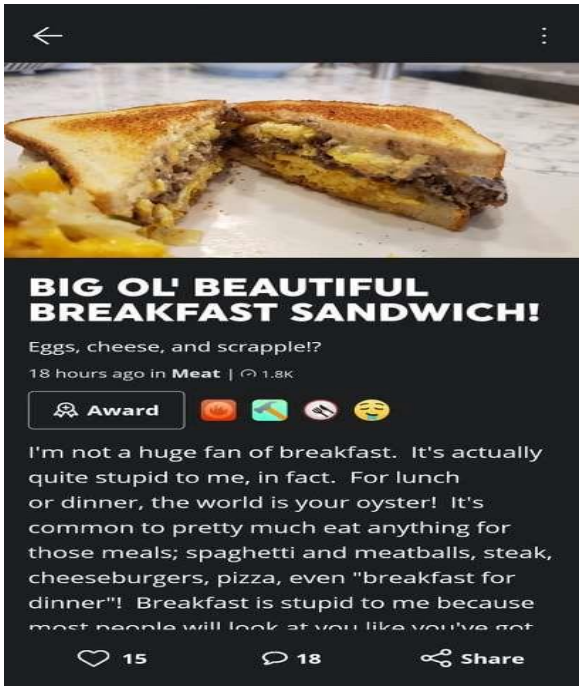


Fig. 10: Food Post

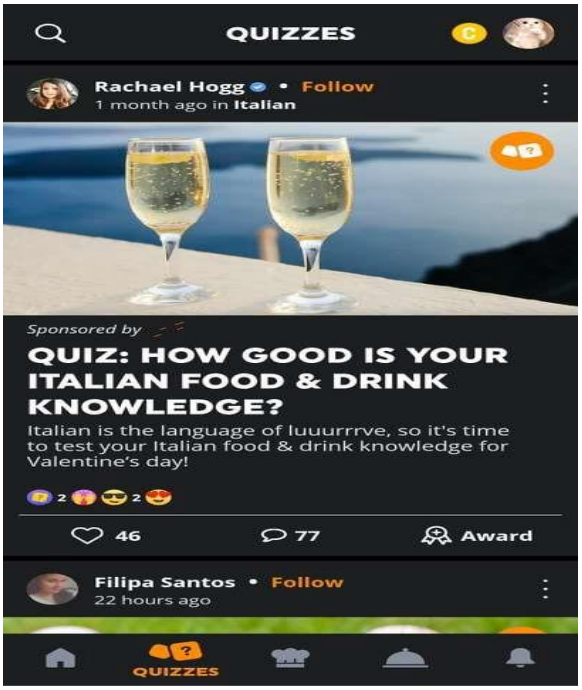


Fig. 11: Quick Page

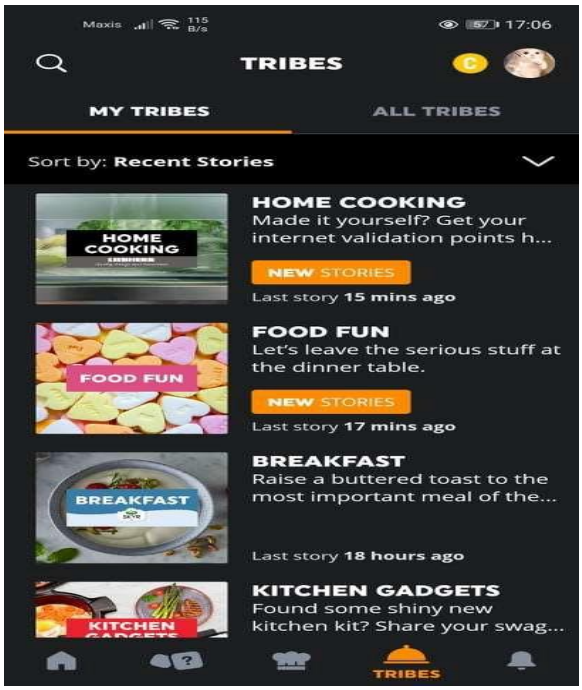


Fig. 12: My Tribe (FoodTribe)

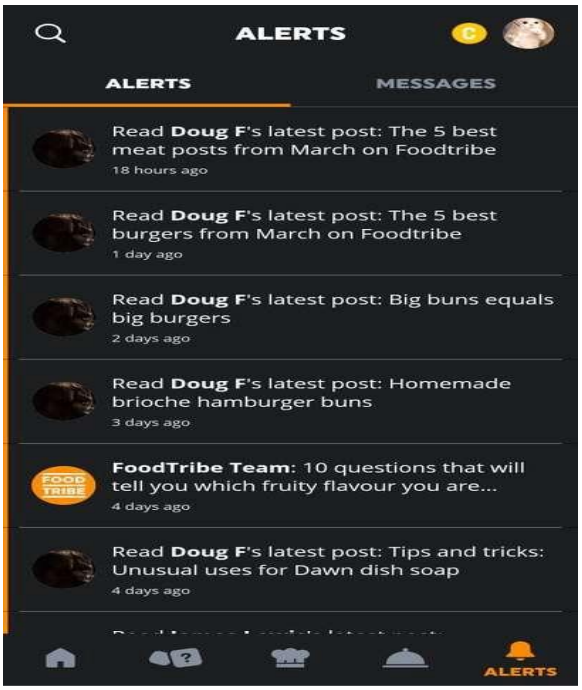


Fig. 13: Notification

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.

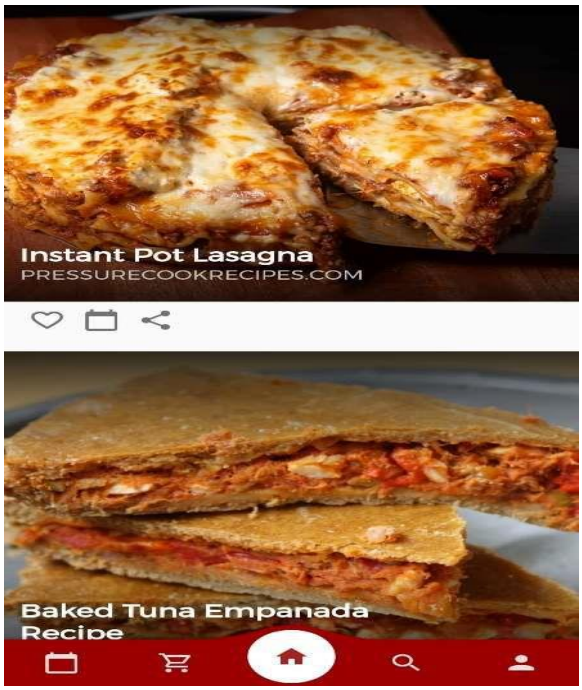


Fig. 14: Home Page

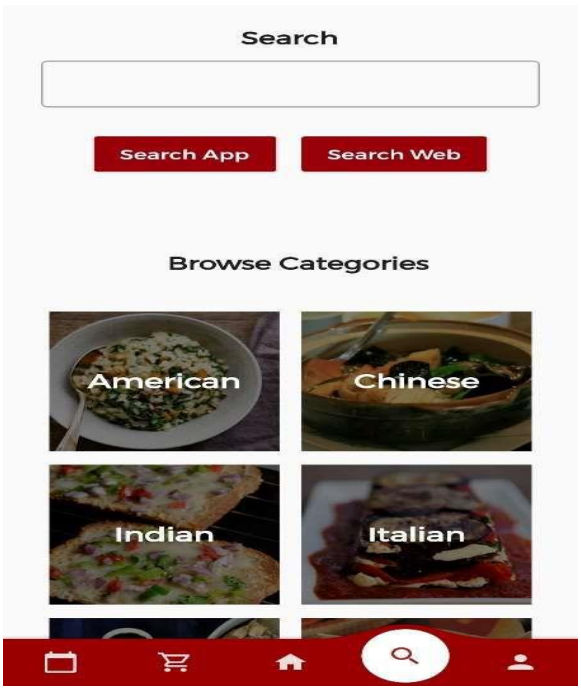


Fig. 15: Search Page

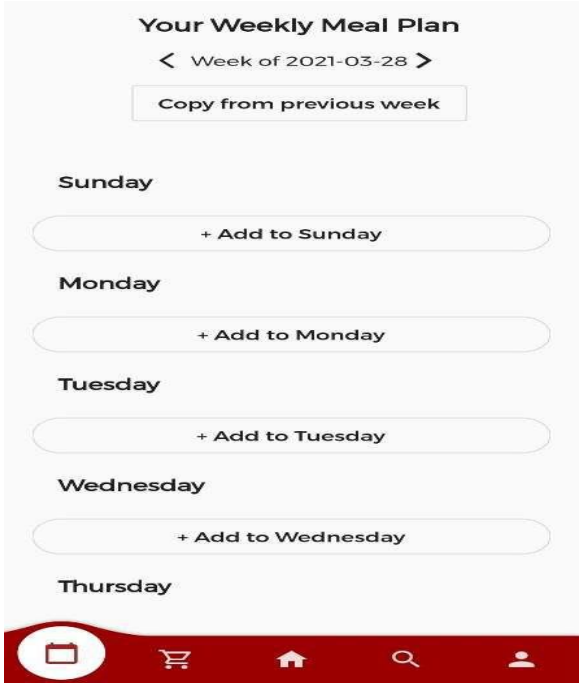


Fig. 16: Meal Plan

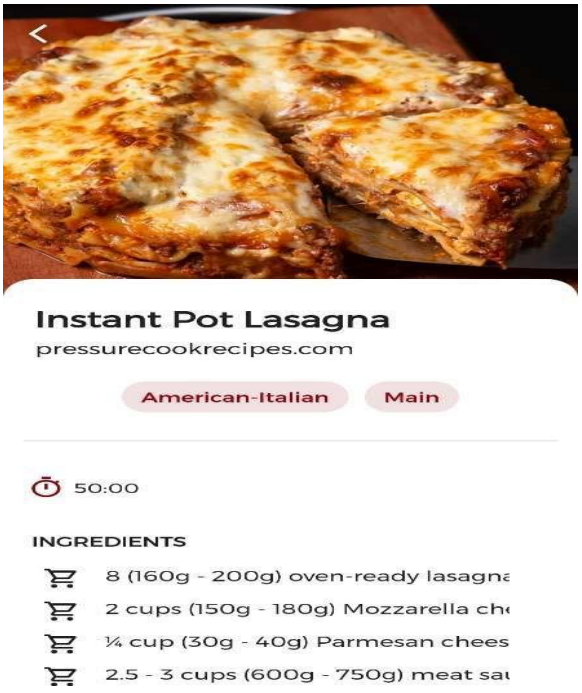


Fig. 17: Recipe Page

Fig. 18: Ingredient List

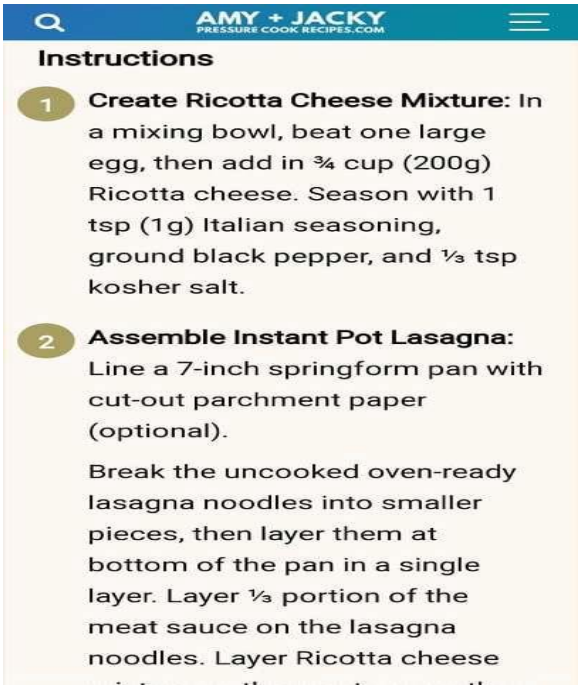


Fig. 19: Direction Details

CHAPTER 3: DESIGN FLOW/PROCESS

3.1. Planning & Timeline:

Gantt Chart

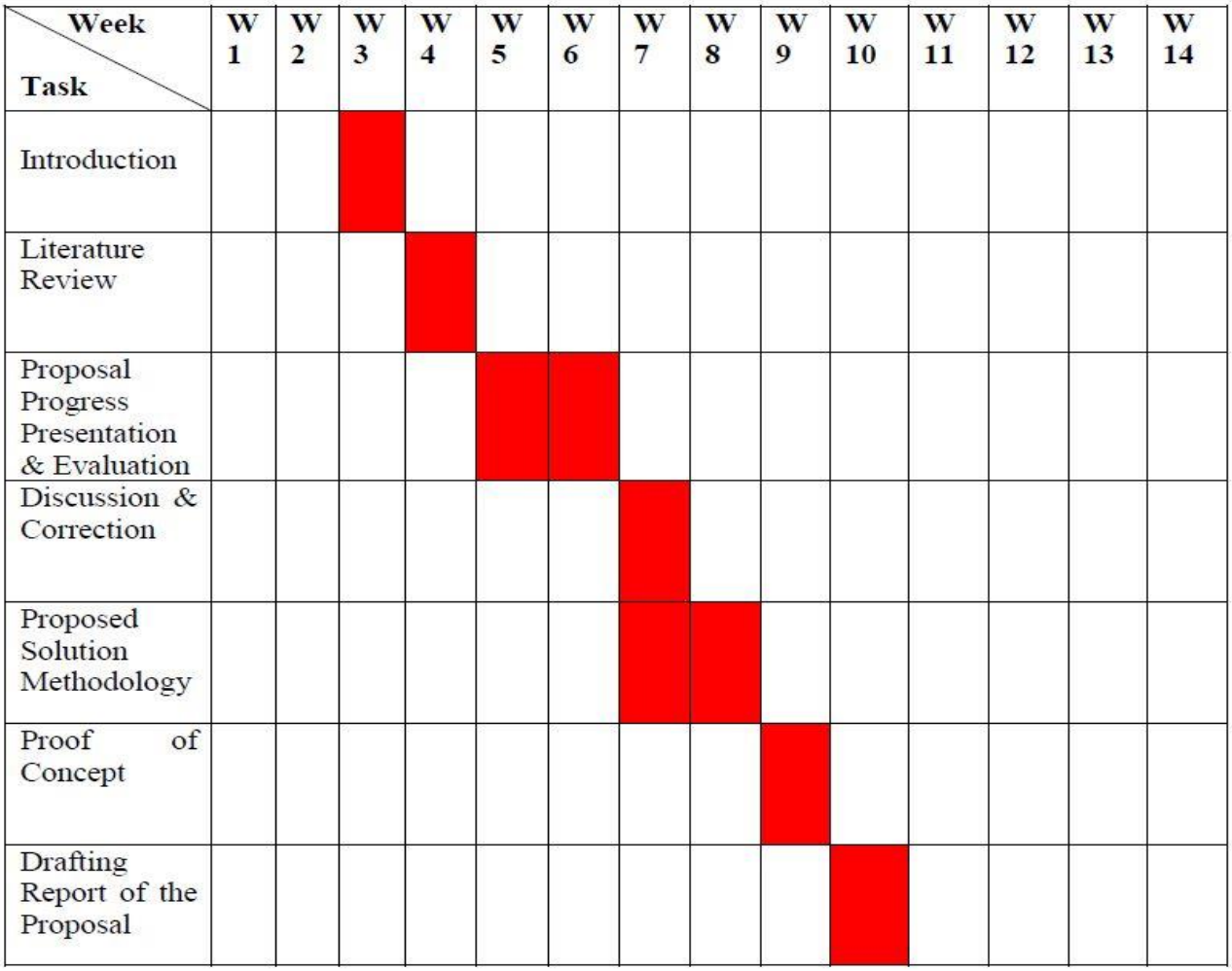


Fig 20. Gant Chart for Project Processes and Completion

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.

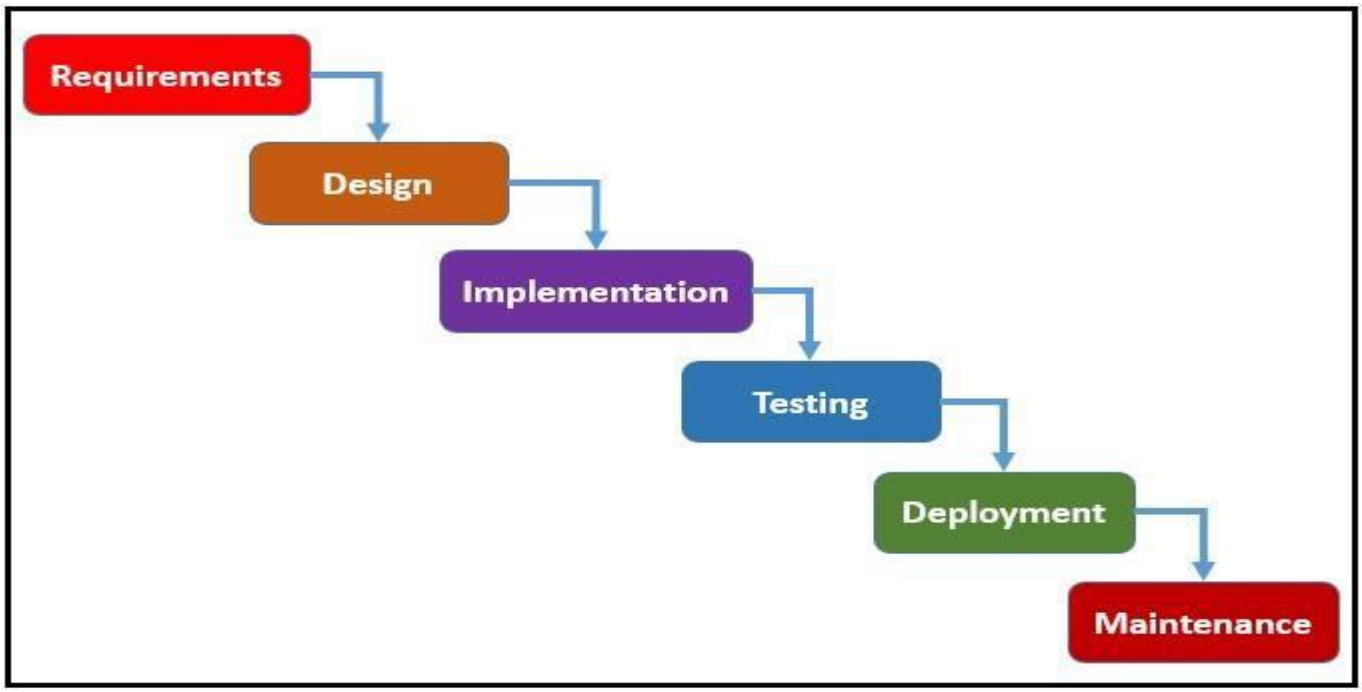


Fig 21. Waterfall Model Methodology

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.

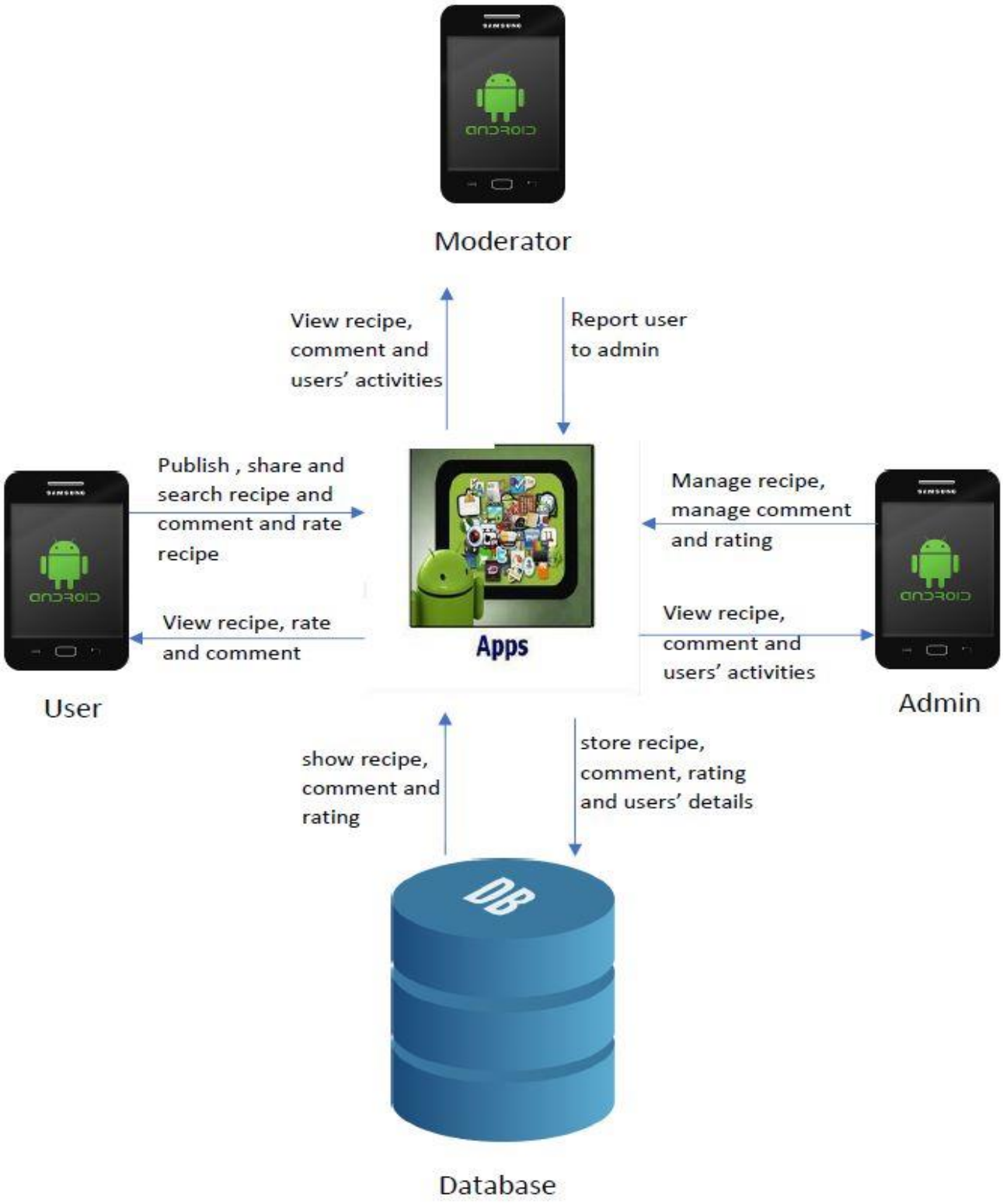


Fig. 24: Framework of the My Cookery Master App

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.

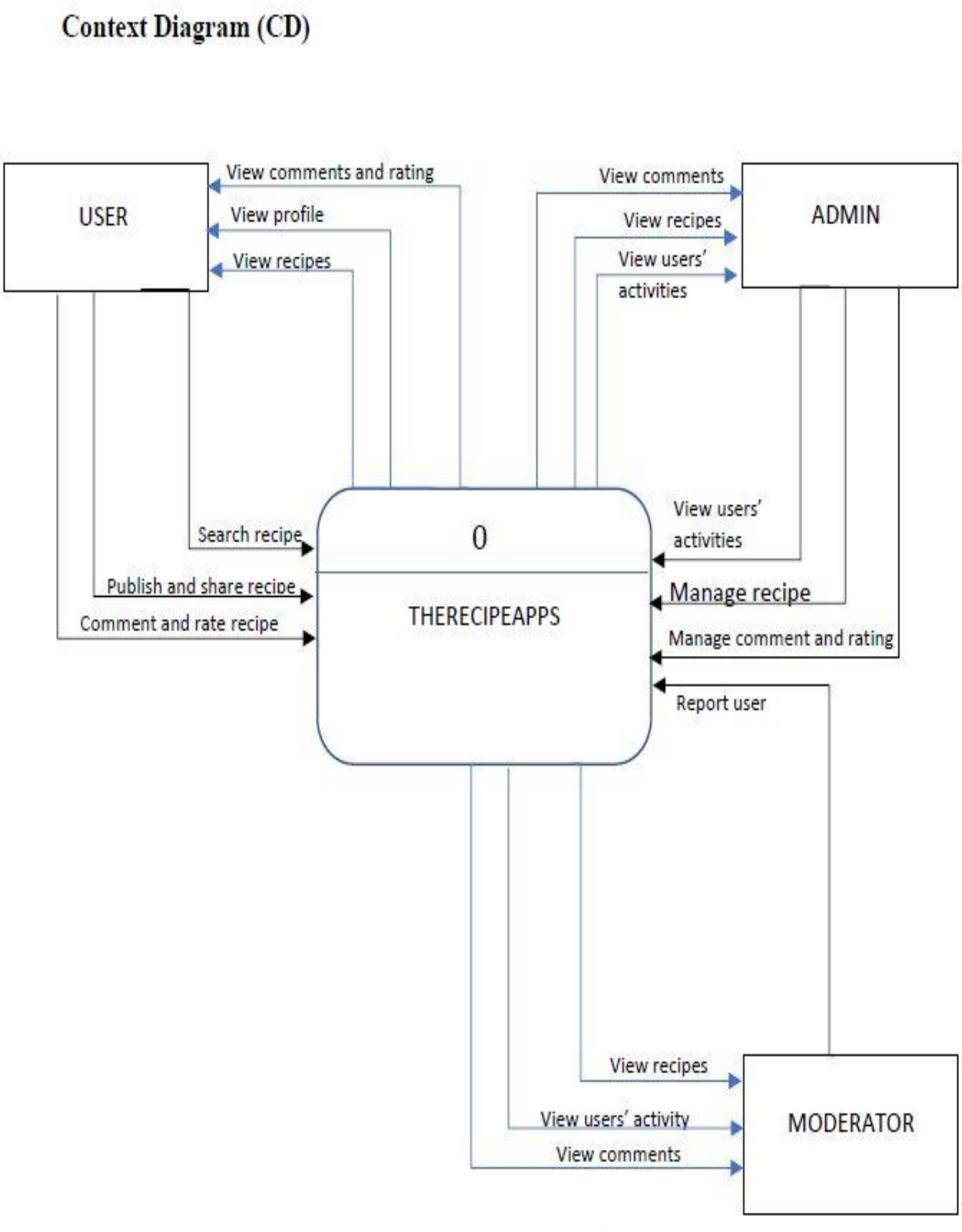


Fig. 25: Context Diagram for My Cookery Master

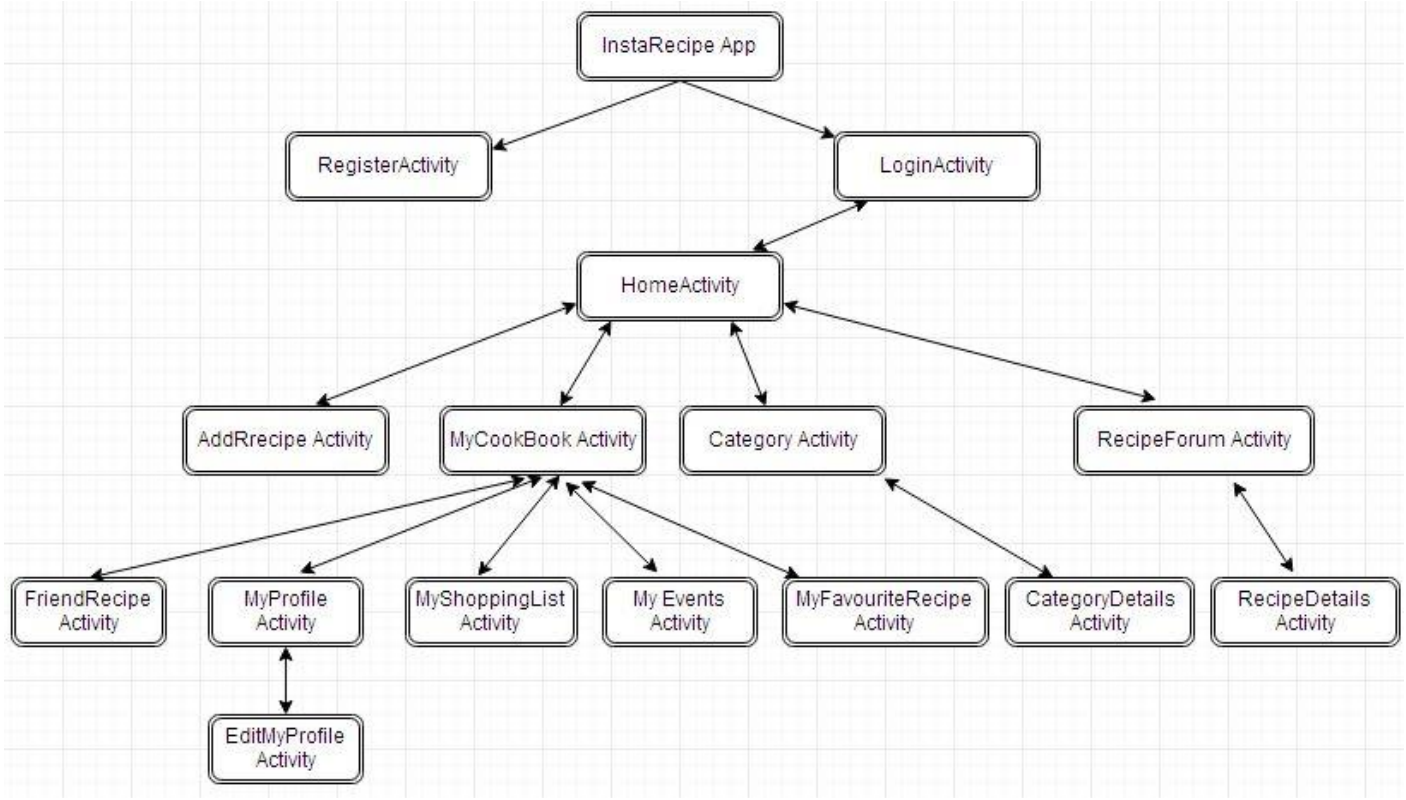


Fig. 26: Activity Diagram

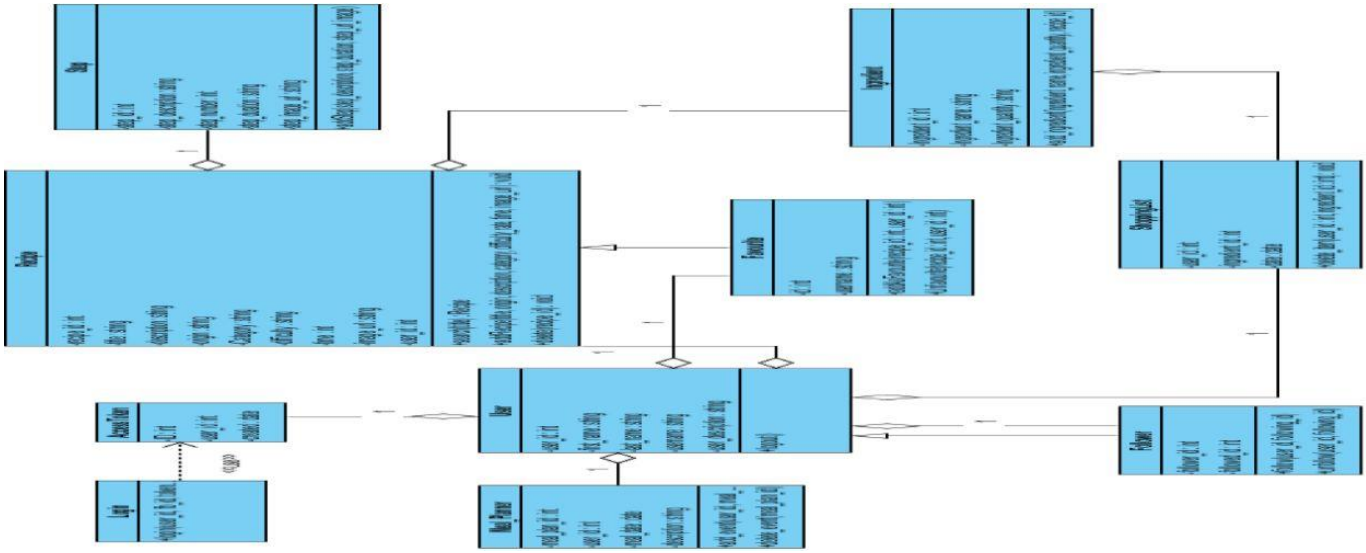


Fig. 27: Class Diagram for Server

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.

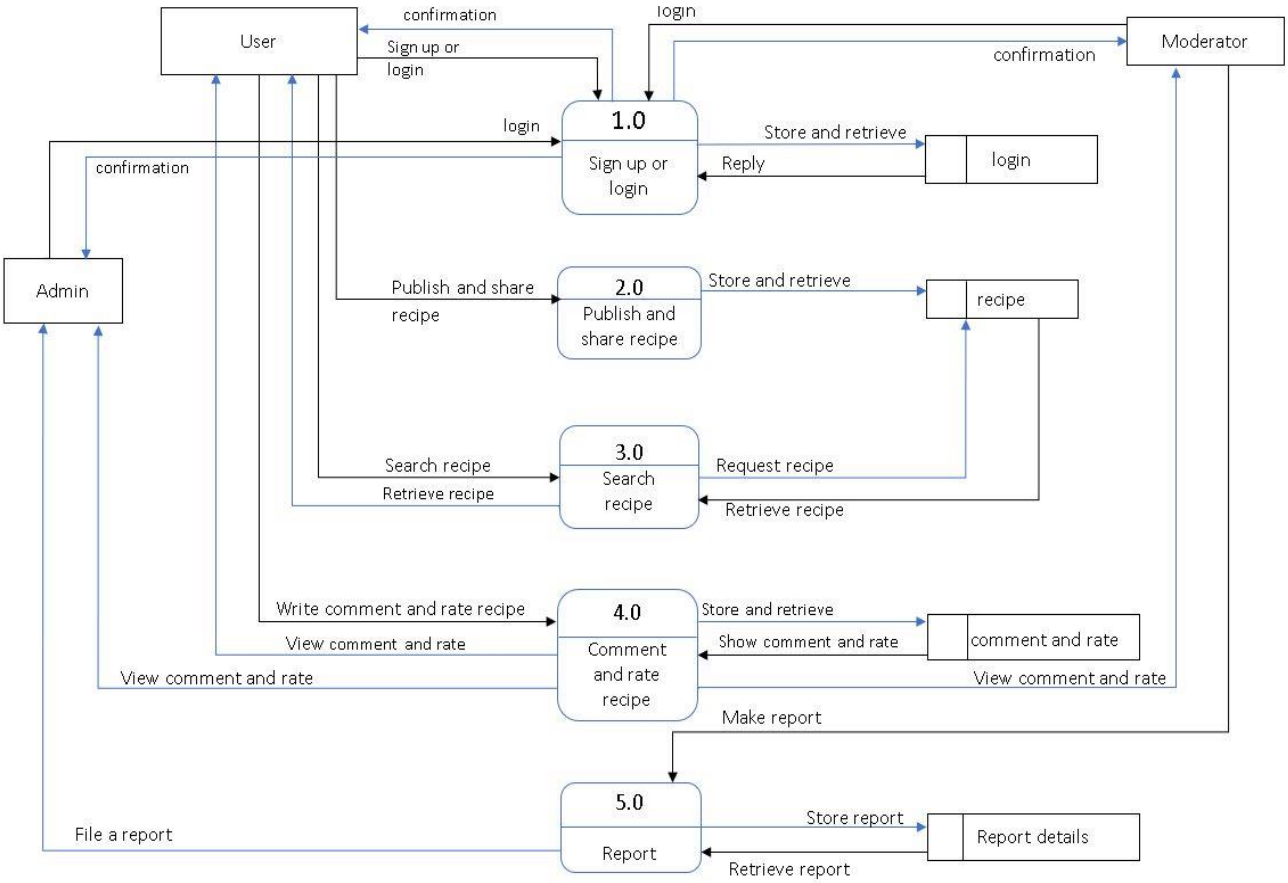


Fig. 28: DFD for My Cookery Master Application

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 inferenced 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.

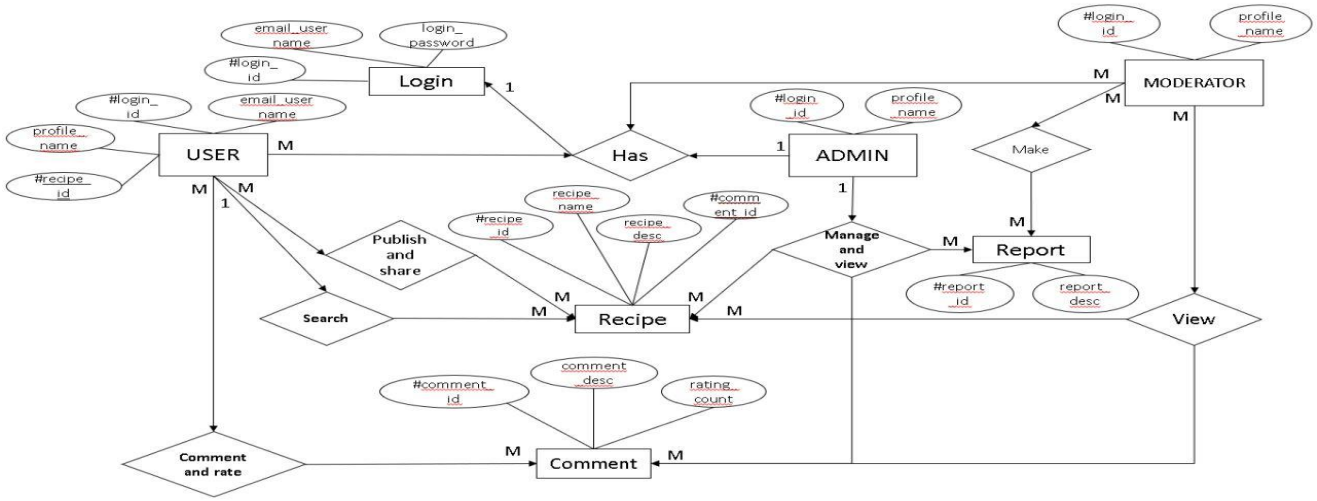


Fig. 29: ER Diagram for My Cookery Master Application

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.

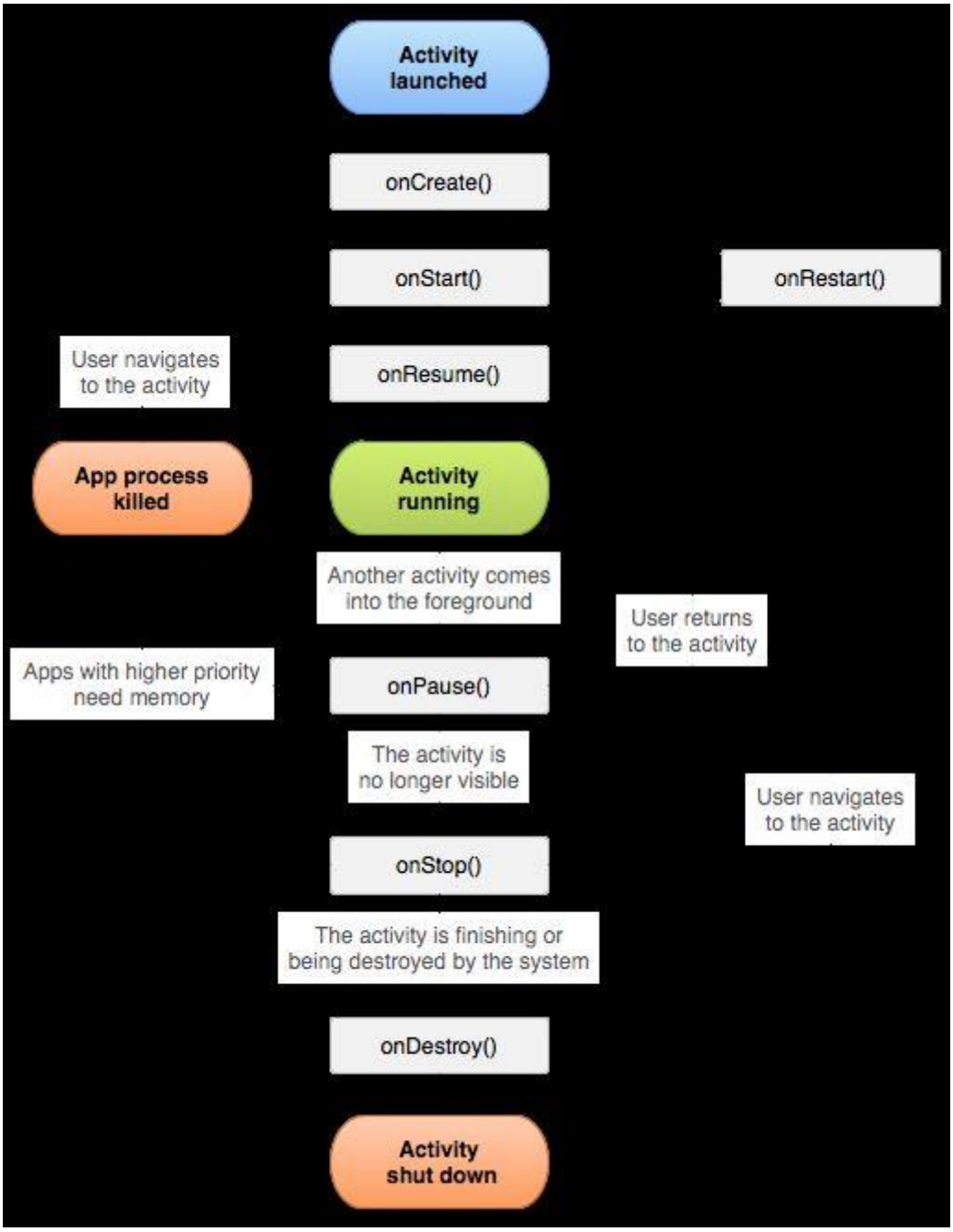


Fig. 30: Activity Lifecycle

The callback methods for activity are as follows which are shown in Figure 10. onCreate(): Thisis 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].

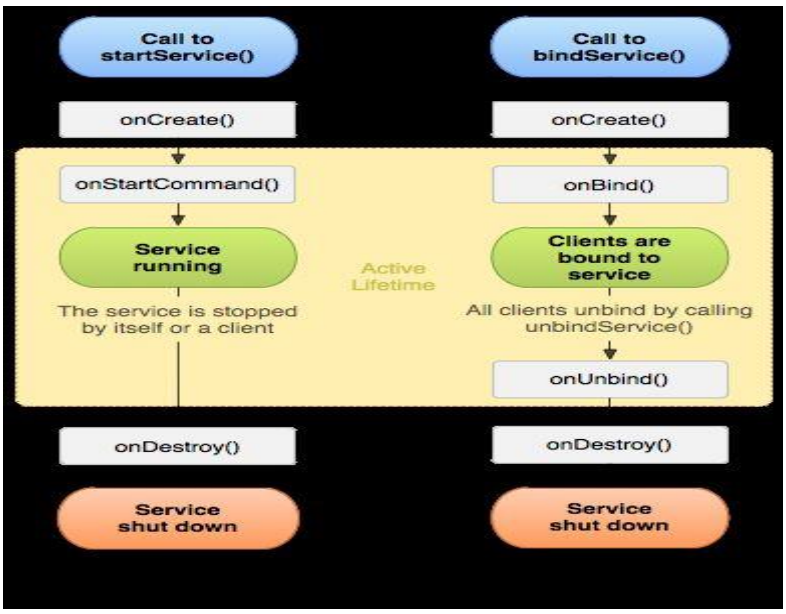


Fig. 31: Service Lifecycle

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.

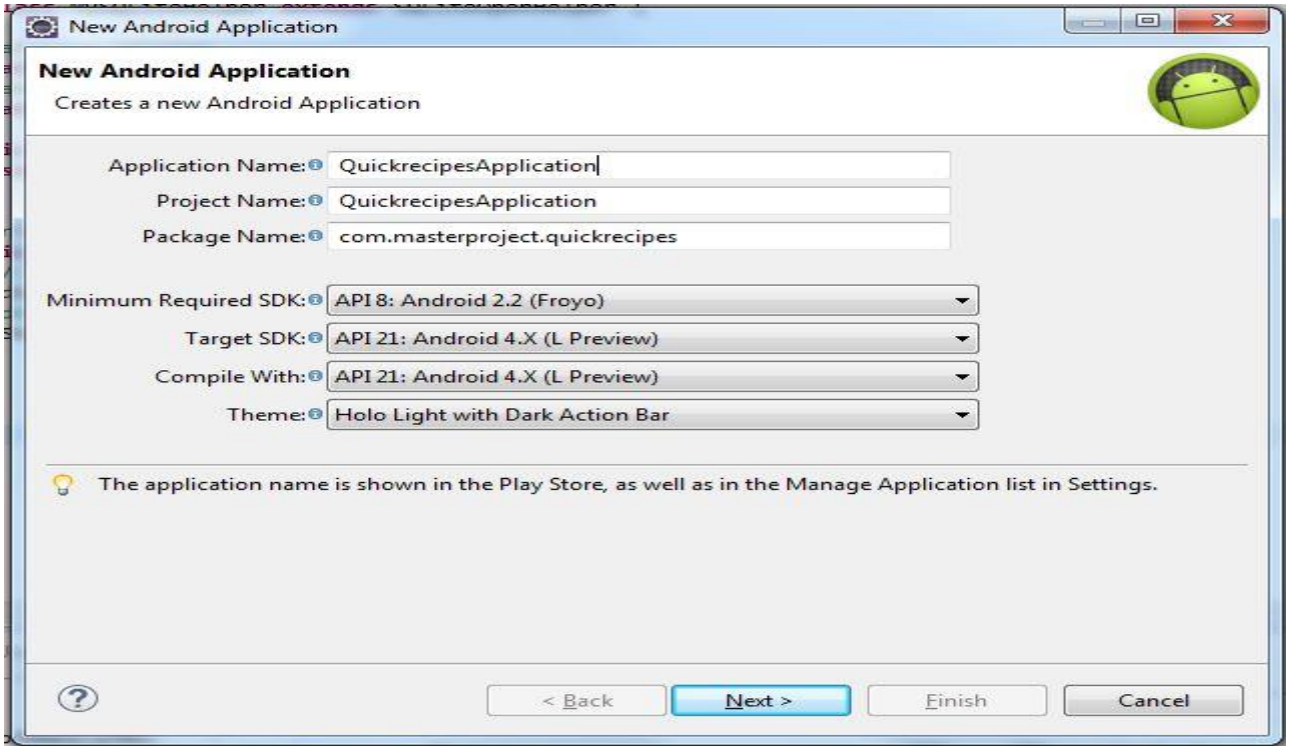


Fig. 32: Creating New Android Application

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.

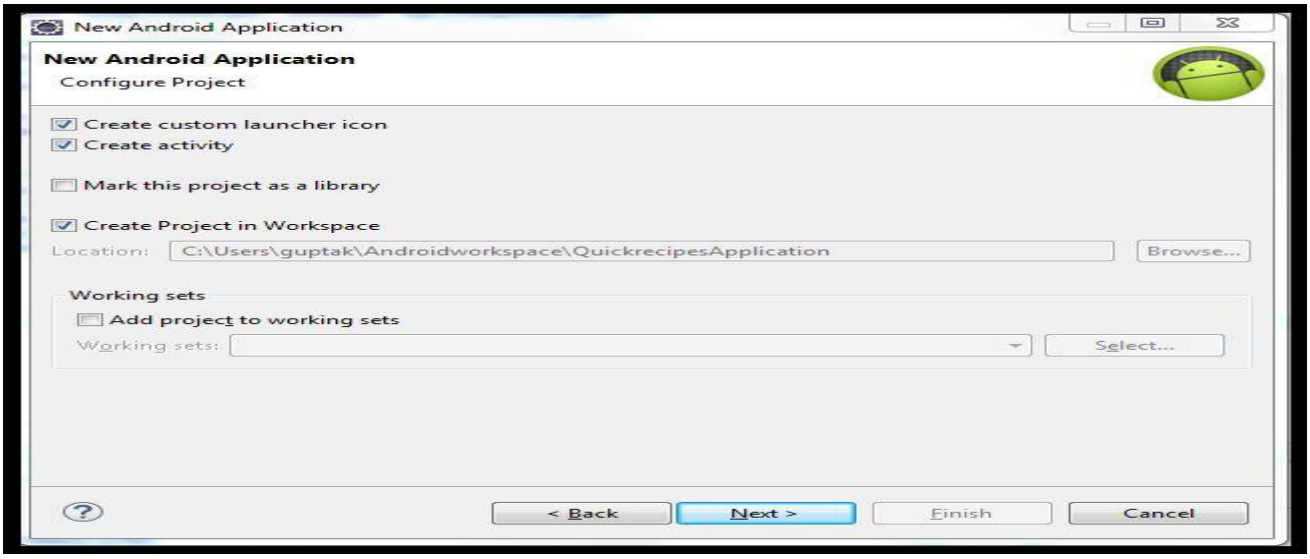


Fig. 33: Configure Project

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:

a. 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

b. 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.

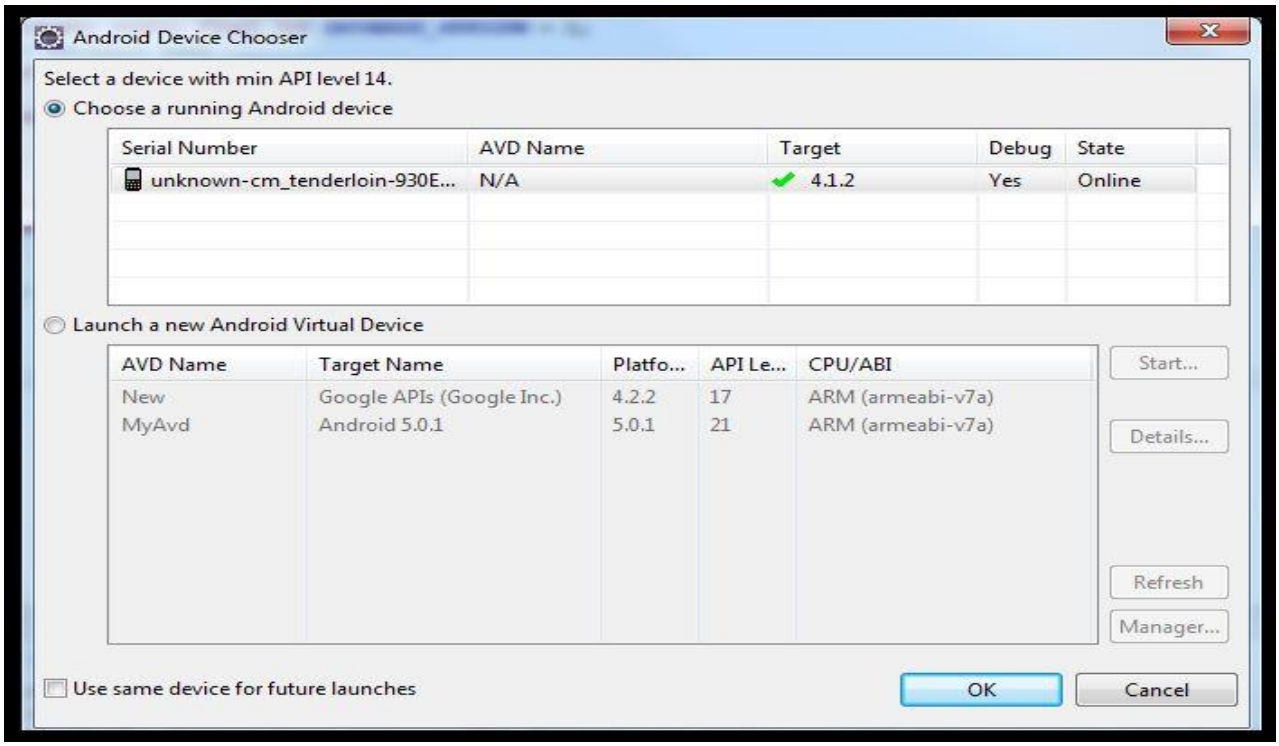


Fig. 34: Android Device Chooser

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.

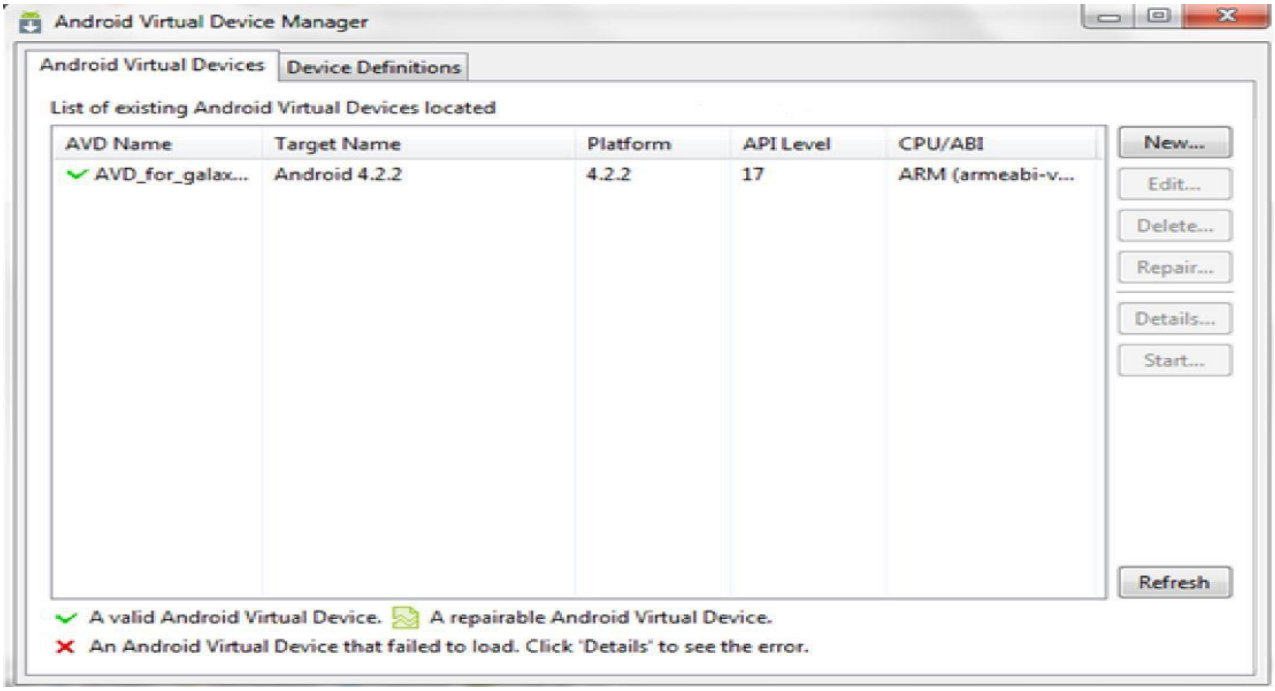


Fig. 35: Android Virtual Device Manager

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.

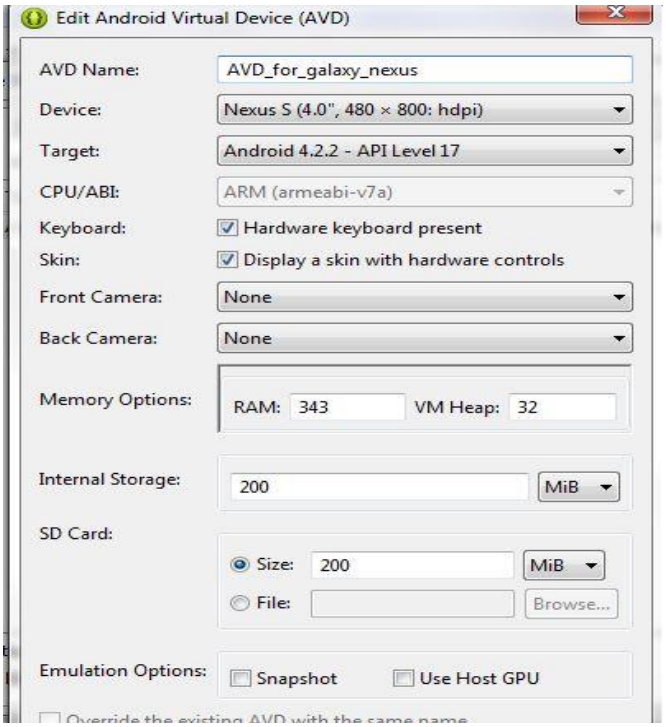


Fig. 36: Android Virtual Device Edit.

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:

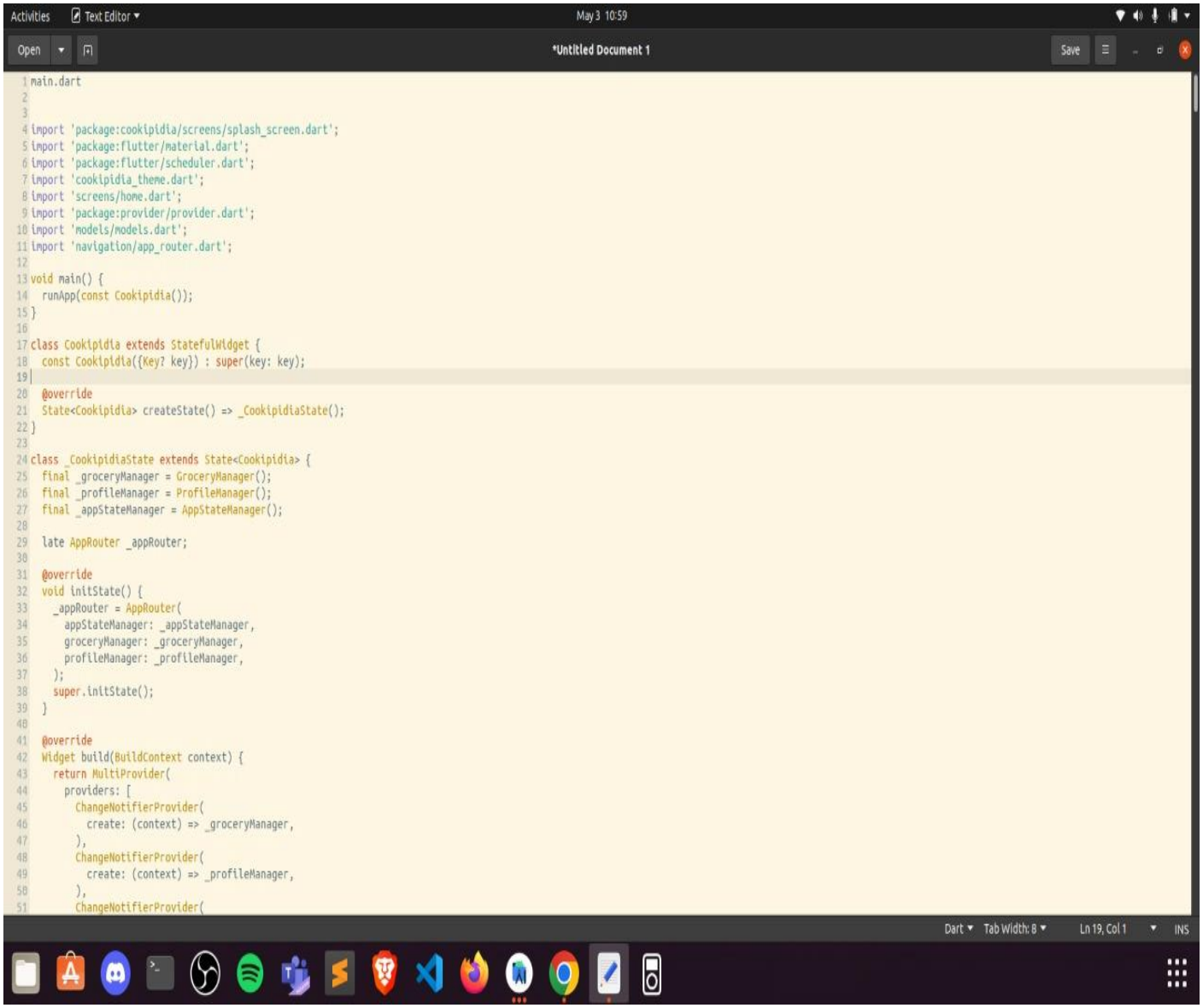
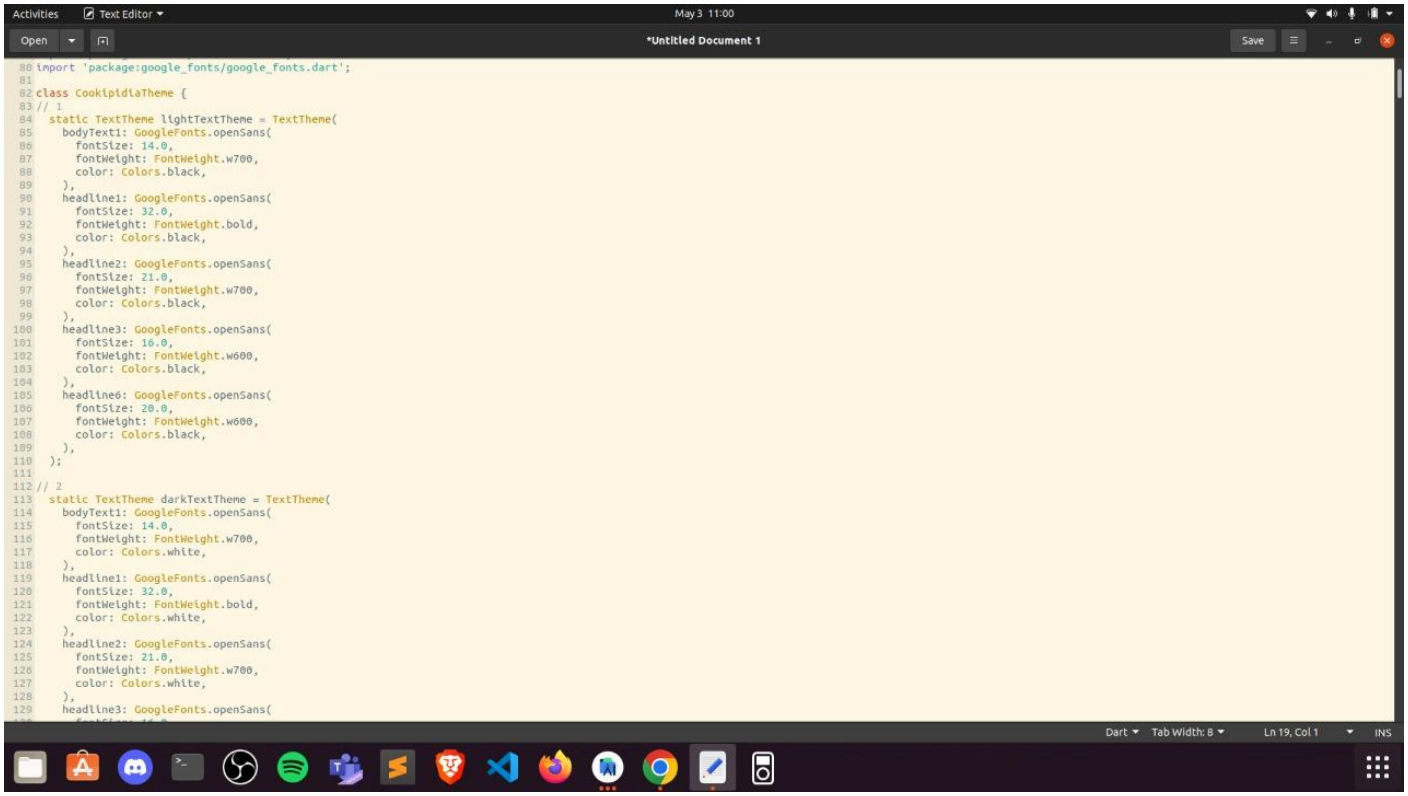
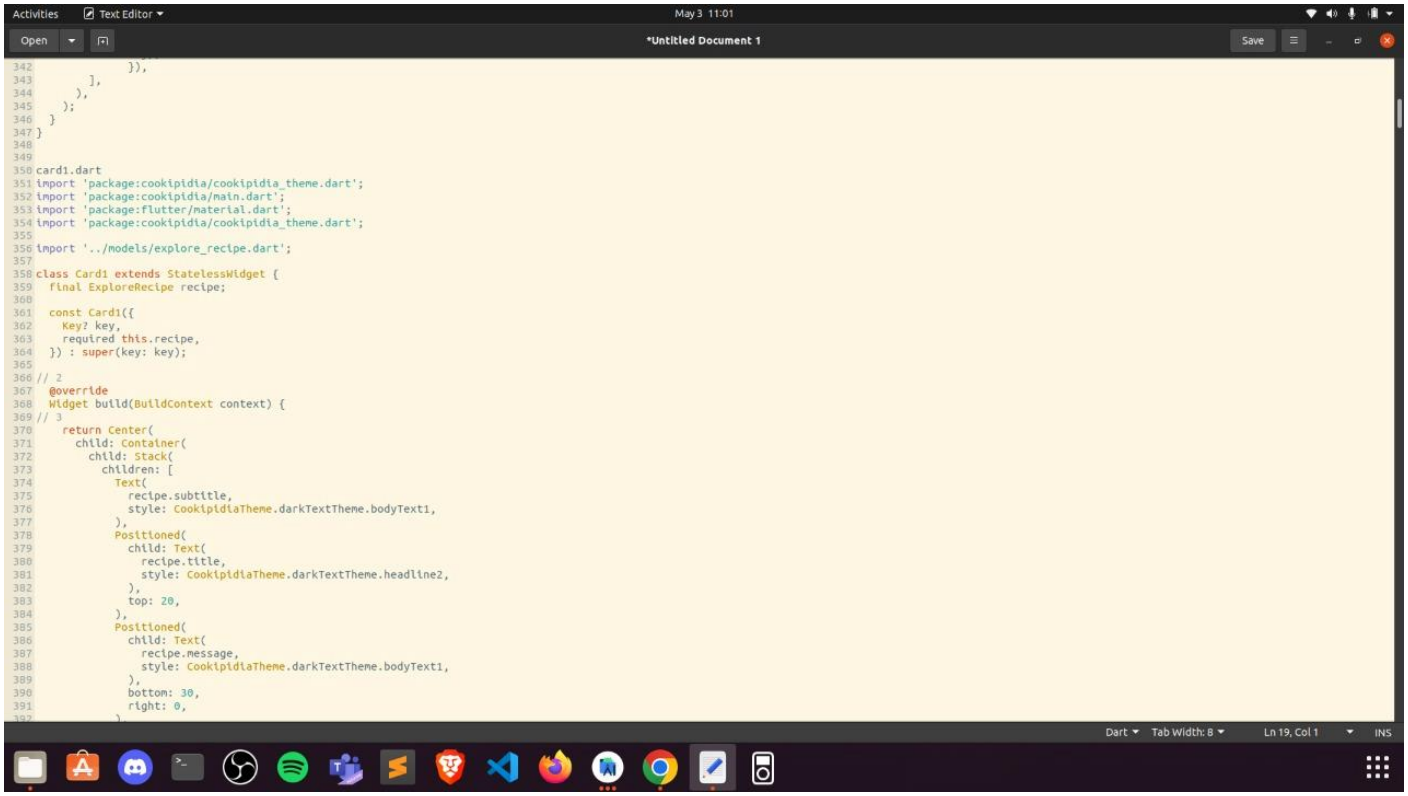


Fig. 37: Code for Home Screen for My Cookery Master




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

class CookipidiaTheme {
// 1
  static TextTheme lightTextTheme = TextTheme(
    bodyText1: GoogleFonts.openSans(
      fontSize: 14.0,
      fontWeight: FontWeight.w700,
      color: Colors.black,
    ),
    headline1: GoogleFonts.openSans(
      fontSize: 32.0,
      fontWeight: FontWeight.bold,
      color: Colors.black,
    ),
    headline2: GoogleFonts.openSans(
      fontSize: 21.0,
      fontWeight: FontWeight.w700,
      color: Colors.black,
    ),
    headline3: GoogleFonts.openSans(
      fontSize: 16.0,
      fontWeight: FontWeight.w600,
      color: Colors.black,
    ),
    headline6: GoogleFonts.openSans(
      fontSize: 20.0,
      fontWeight: FontWeight.w600,
      color: Colors.black,
    ),
  );

// 2
  static TextTheme darkTextTheme = TextTheme(
    bodyText1: GoogleFonts.openSans(
      fontSize: 14.0,
      fontWeight: FontWeight.w700,
      color: Colors.white,
    ),
    headline1: GoogleFonts.openSans(
      fontSize: 32.0,
      fontWeight: FontWeight.bold,
      color: Colors.white,
    ),
    headline2: GoogleFonts.openSans(
      fontSize: 21.0,
      fontWeight: FontWeight.w700,
      color: Colors.white,
    ),
    headline3: GoogleFonts.openSans(
```

Fig. 38




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

card1.dart
import 'package:cookipidia/cookipidia_theme.dart';
import 'package:cookipidia/main.dart';
import 'package:flutter/material.dart';
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);

// 2
  @override
  Widget build(BuildContext context) {
// 3
    return Center(
      child: Container(
        child: Stack(
          children: [
            Text(
              recipe.subtitle,
              style: CookipidiaTheme.darkTextTheme.bodyText1,
            ),
            Positioned(
              child: Text(
                recipe.title,
                style: CookipidiaTheme.darkTextTheme.headline2,
              ),
              top: 20,
            ),
            Positioned(
              child: Text(
                recipe.message,
                style: CookipidiaTheme.darkTextTheme.bodyText1,
              ),
              bottom: 30,
              right: 0,
            )
```

Fig. 39

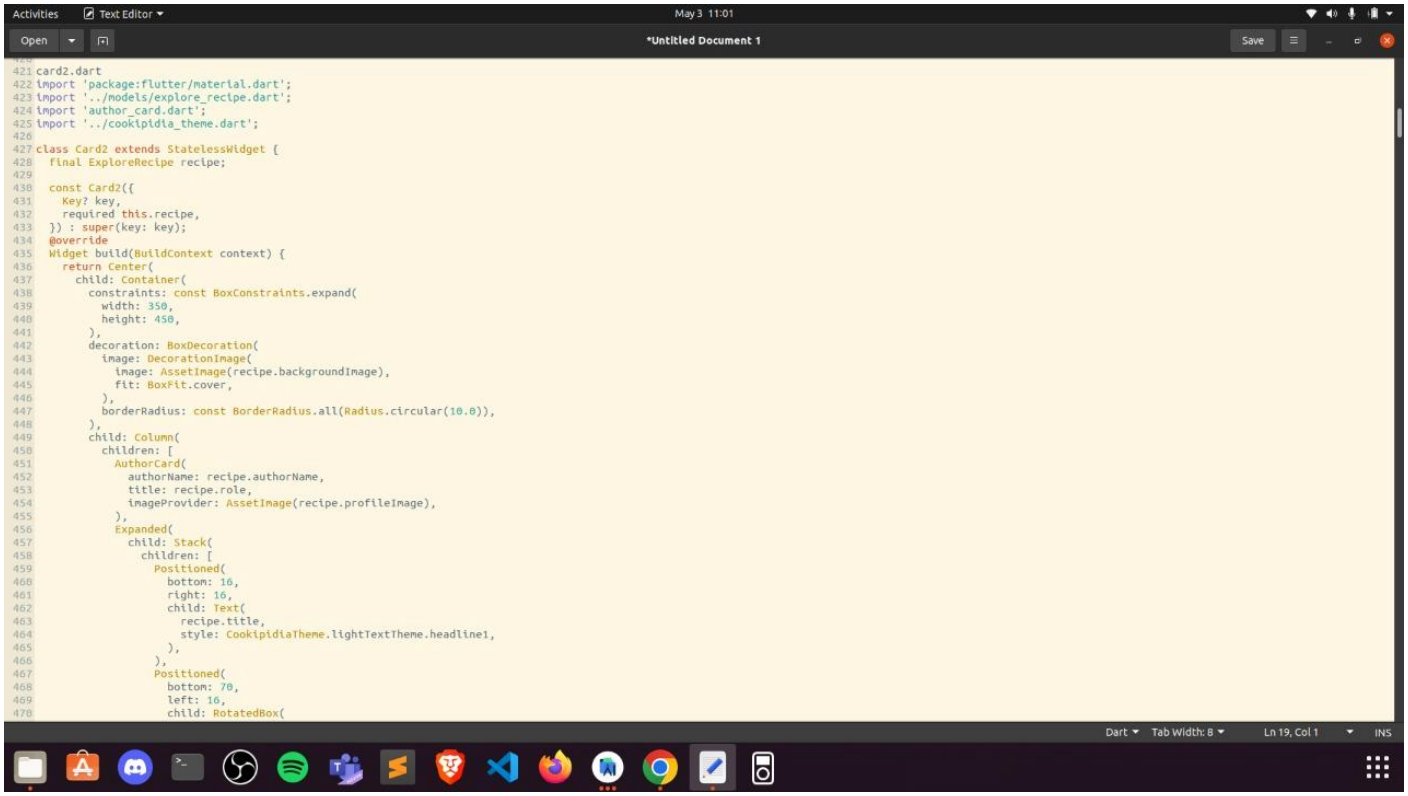

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
card2.dart
import 'package:flutter/material.dart';
import '../models/explore_recipe.dart';
import 'author_card.dart';
import '../cookipidia_theme.dart';
class Card2 extends StatelessWidget {
final ExploreRecipe recipe;
const Card2({
Key? key,
required this.recipe,
}) : super(key: key);
@override
Widget build(BuildContext context) {
return Center(
child: Container(
constraints: const BoxConstraints.expand(
width: 350,
height: 450,
),
decoration: BoxDecoration(
image: DecorationImage(
image: AssetImage(recipe.backgroundImage),
fit: BoxFit.cover,
),
borderRadius: const BorderRadius.all(Radius.circular(10.0)),
),
child: Column(
children: [
AuthorCard(
authorName: recipe.authorName,
title: recipe.role,
imageProvider: AssetImage(recipe.profileImage),
),
Expanded(
child: Stack(
children: [
Positioned(
bottom: 16,
right: 16,
child: Text(
recipe.title,
style: CookipidiaTheme.lightTextTheme.headline1,
),
),
Positioned(
bottom: 70,
left: 16,
child: RotatedBox(
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 40

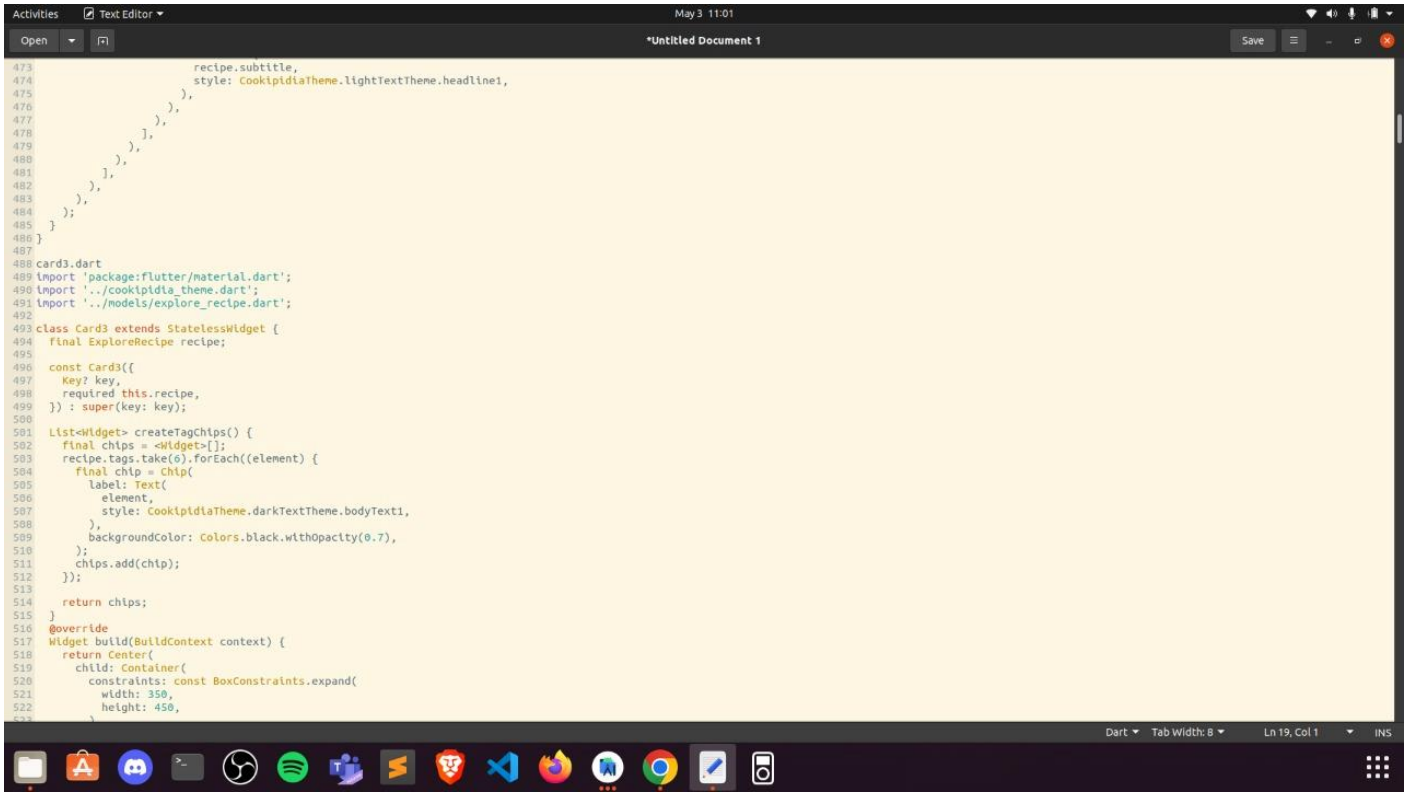

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
recipe.subtitle,
style: CookipidiaTheme.lightTextTheme.headline1,
card3.dart
import 'package:flutter/material.dart';
import '../cookipidia_theme.dart';
import '../models/explore_recipe.dart';
class Card3 extends StatelessWidget {
final ExploreRecipe recipe;
const Card3({
Key? key,
required this.recipe,
}) : super(key: key);
List<Widget> createTagChips() {
final chips = <Widget>[];
recipe.tags.take(6).forEach((element) {
final chip = Chip(
label: Text(
element,
style: CookipidiaTheme.darkTextTheme.bodyText1,
),
backgroundColor: Colors.black.withOpacity(0.7),
);
chips.add(chip);
});
return chips;
}
@override
Widget build(BuildContext context) {
return Center(
child: Container(
constraints: const BoxConstraints.expand(
width: 350,
height: 450,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 41

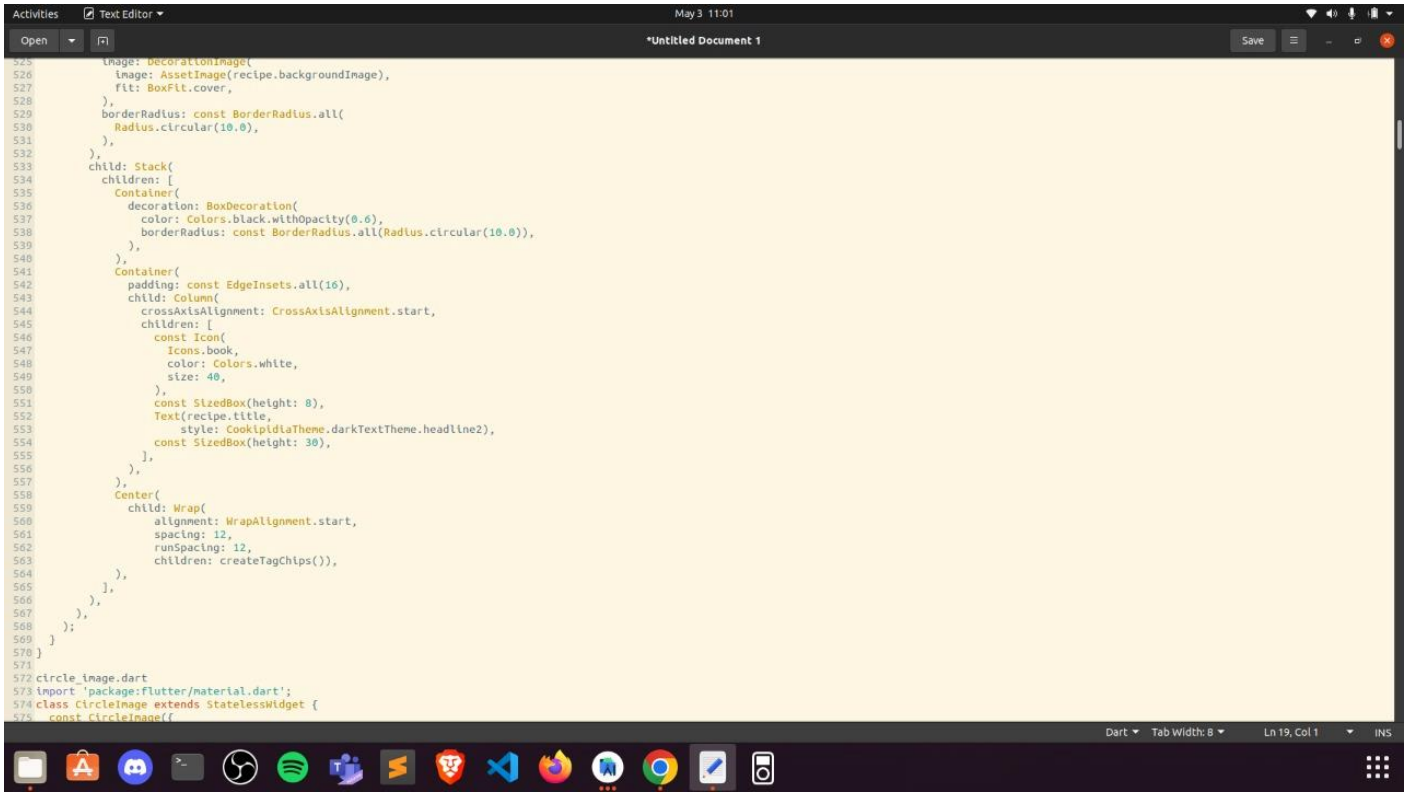

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
image: DecorationImage(
image: AssetImage(recipe.backgroundImage),
fit: BoxFit.cover,
),
borderRadius: const BorderRadius.all(
Radius.circular(10.0),
),
),
child: Stack(
children: [
Container(
decoration: BoxDecoration(
color: Colors.black.withOpacity(0.6),
borderRadius: const BorderRadius.all(Radius.circular(10.0)),
),
),
Container(
padding: const EdgeInsets.all(16),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
const Icon(
Icons.book,
color: Colors.white,
size: 40,
),
const SizedBox(height: 8),
Text(recipe.title,
style: CookipidiaTheme.darkTextTheme.headline2),
const SizedBox(height: 30),
],
),
),
Center(
child: Wrap(
alignment: WrapAlignment.start,
spacing: 12,
runSpacing: 12,
children: createTagChips()),
),
],
),
),
);
}
}
circle_image.dart
import 'package:flutter/material.dart';
class CircleImage extends StatelessWidget {
const CircleImage({
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 42

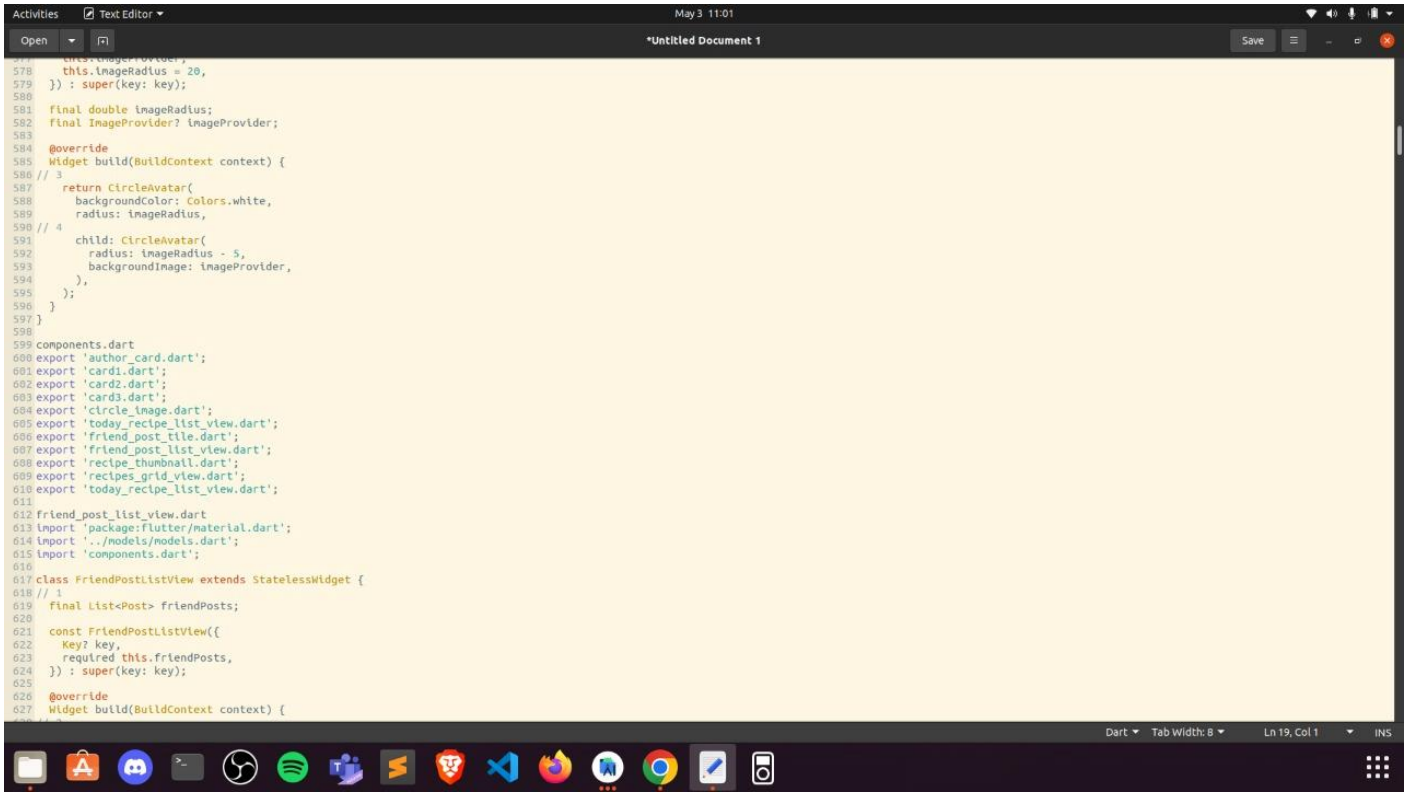

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
this.imageRadius = 20,
}) : super(key: key);
final double imageRadius;
final ImageProvider? imageProvider;
@override
Widget build(BuildContext context) {
// 3
return CircleAvatar(
backgroundColor: Colors.white,
radius: imageRadius,
// 4
child: CircleAvatar(
radius: imageRadius - 5,
backgroundImage: imageProvider,
),
);
}
}
components.dart
export 'author_card.dart';
export 'card1.dart';
export 'card2.dart';
export 'card3.dart';
export 'circle_image.dart';
export 'today_recipe_list_view.dart';
export 'friend_post_tile.dart';
export 'friend_post_list_view.dart';
export 'recipe_thumbnail.dart';
export 'recipes_grid_view.dart';
export 'today_recipe_list_view.dart';
friend_post_list_view.dart
import 'package:flutter/material.dart';
import '../models/models.dart';
import 'components.dart';
class FriendPostListView extends StatelessWidget {
// 1
final List<Post> friendPosts;
const FriendPostListView({
Key? key,
required this.friendPosts,
}) : super(key: key);
@override
Widget build(BuildContext context) {
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 43

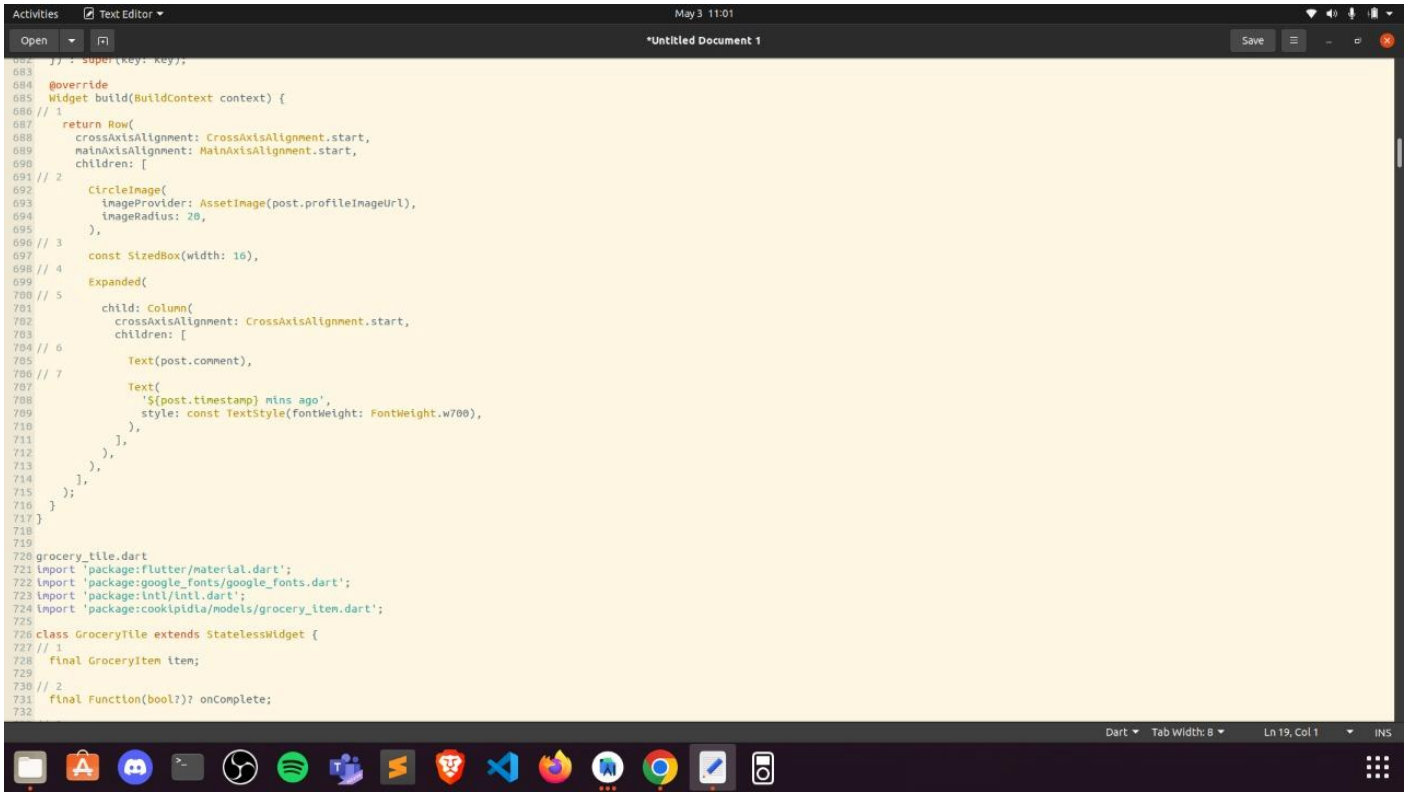

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
@override
Widget build(BuildContext context) {
return Row(
crossAxisAlignment: CrossAxisAlignment.start,
mainAxisAlignment: MainAxisAlignment.start,
children: [
CircleImage(
imageProvider: AssetImage(post.profileImageUrl),
imageRadius: 20,
),
const SizedBox(width: 16),
Expanded(
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(post.comment),
Text(
'${post.timestamp} mins ago',
style: const TextStyle(fontWeight: FontWeight.w700),
),
],
),
),
],
);
}
}
grocery_tile.dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:intl/intl.dart';
import 'package:cookipidia/models/grocery_item.dart';
class GroceryTile extends StatelessWidget {
final GroceryItem item;
final Function(bool?)? onComplete;
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 44

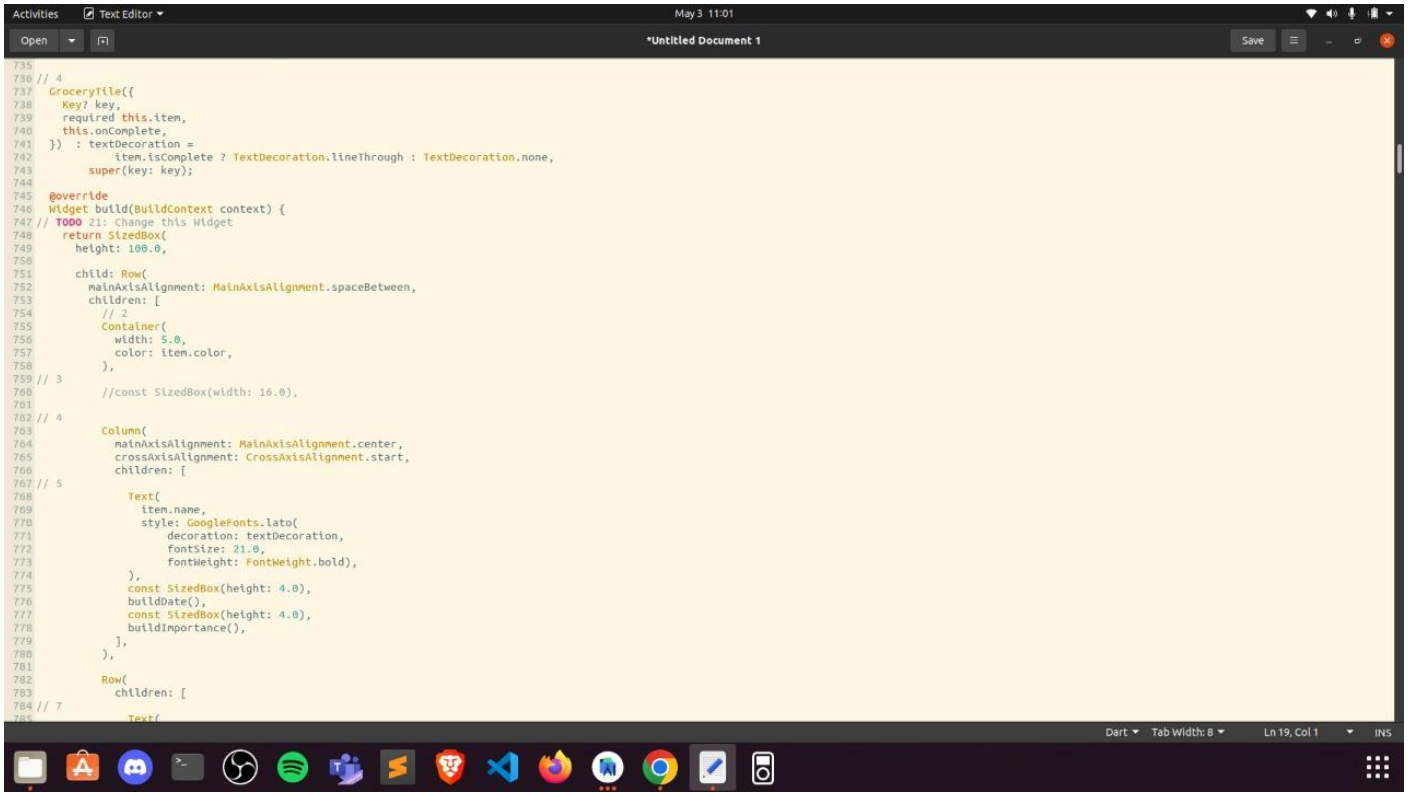

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
GroceryTile({
Key? key,
required this.item,
this.onComplete,
}) : textDecoration =
item.isComplete ? TextDecoration.lineThrough : TextDecoration.none,
super(key: key);
@override
Widget build(BuildContext context) {
// TODO 21: Change this Widget
return SizedBox(
height: 100.0,
child: Row(
mainAxisAlignment: MainAxisAlignment.spaceBetween,
children: [
// 2
Container(
width: 5.0,
color: item.color,
),
//const SizedBox(width: 16.0),
Column(
mainAxisAlignment: MainAxisAlignment.center,
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
item.name,
style: GoogleFonts.lato(
decoration: textDecoration,
fontSize: 21.0,
fontWeight: FontWeight.bold),
),
const SizedBox(height: 4.0),
buildDate(),
const SizedBox(height: 4.0),
buildImportance(),
],
),
Row(
children: [
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 45

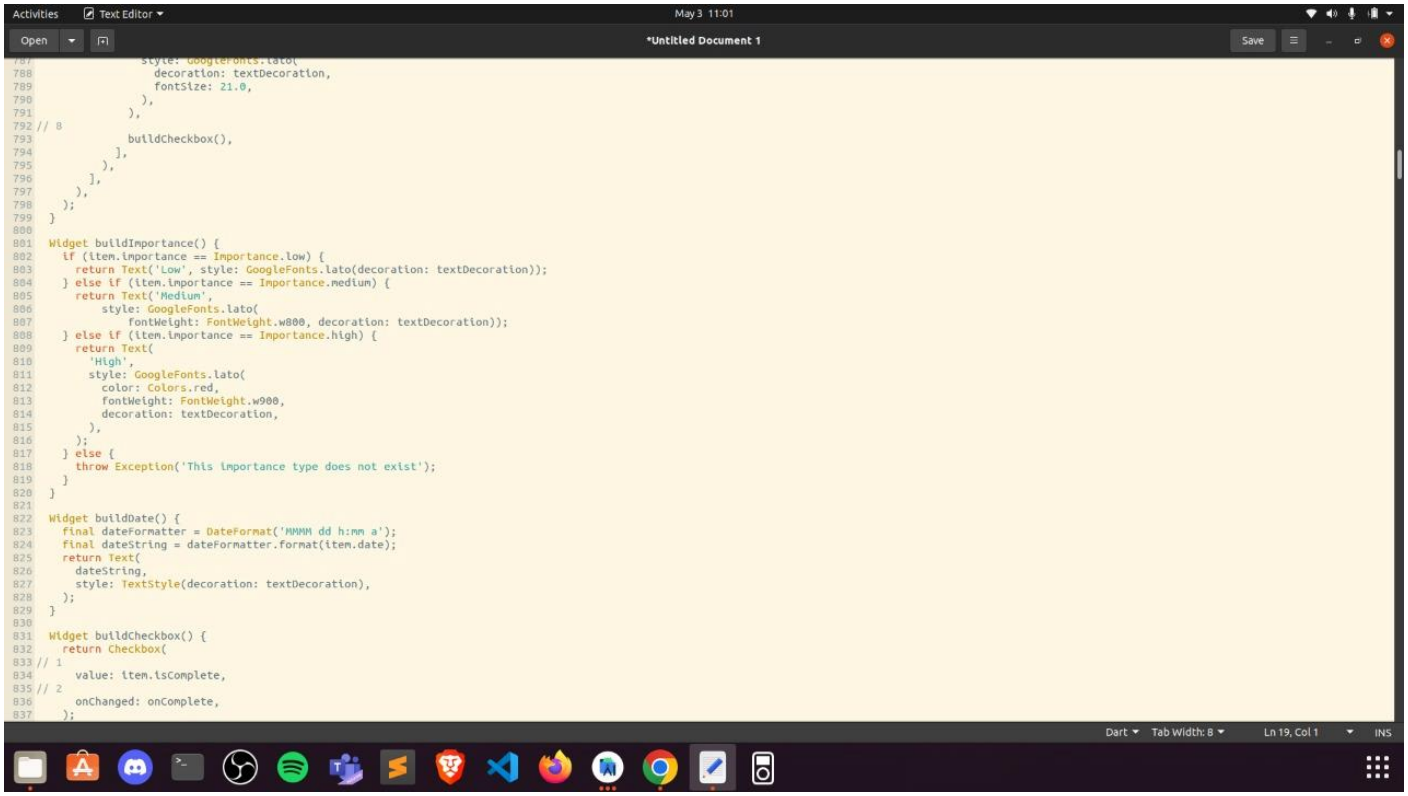

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 46

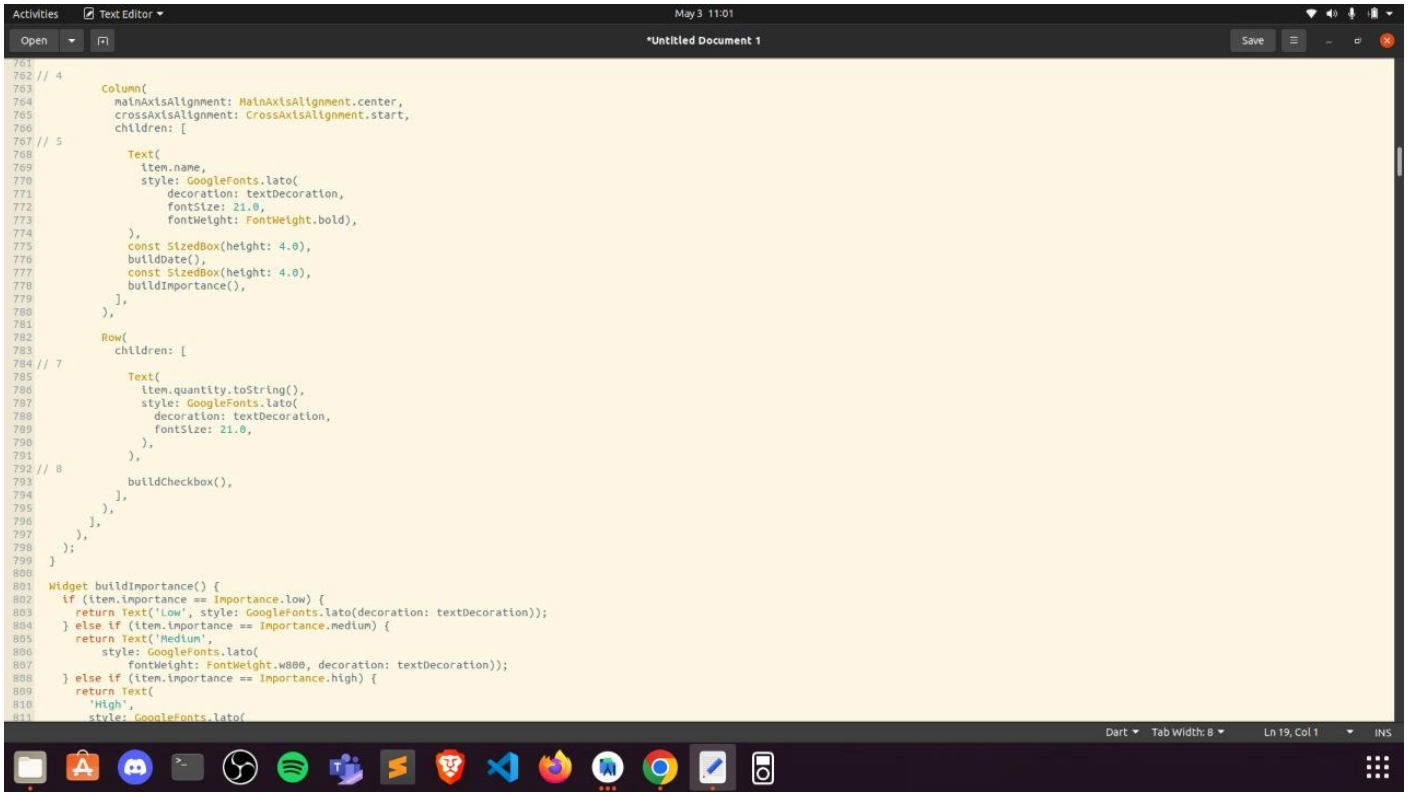

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 47

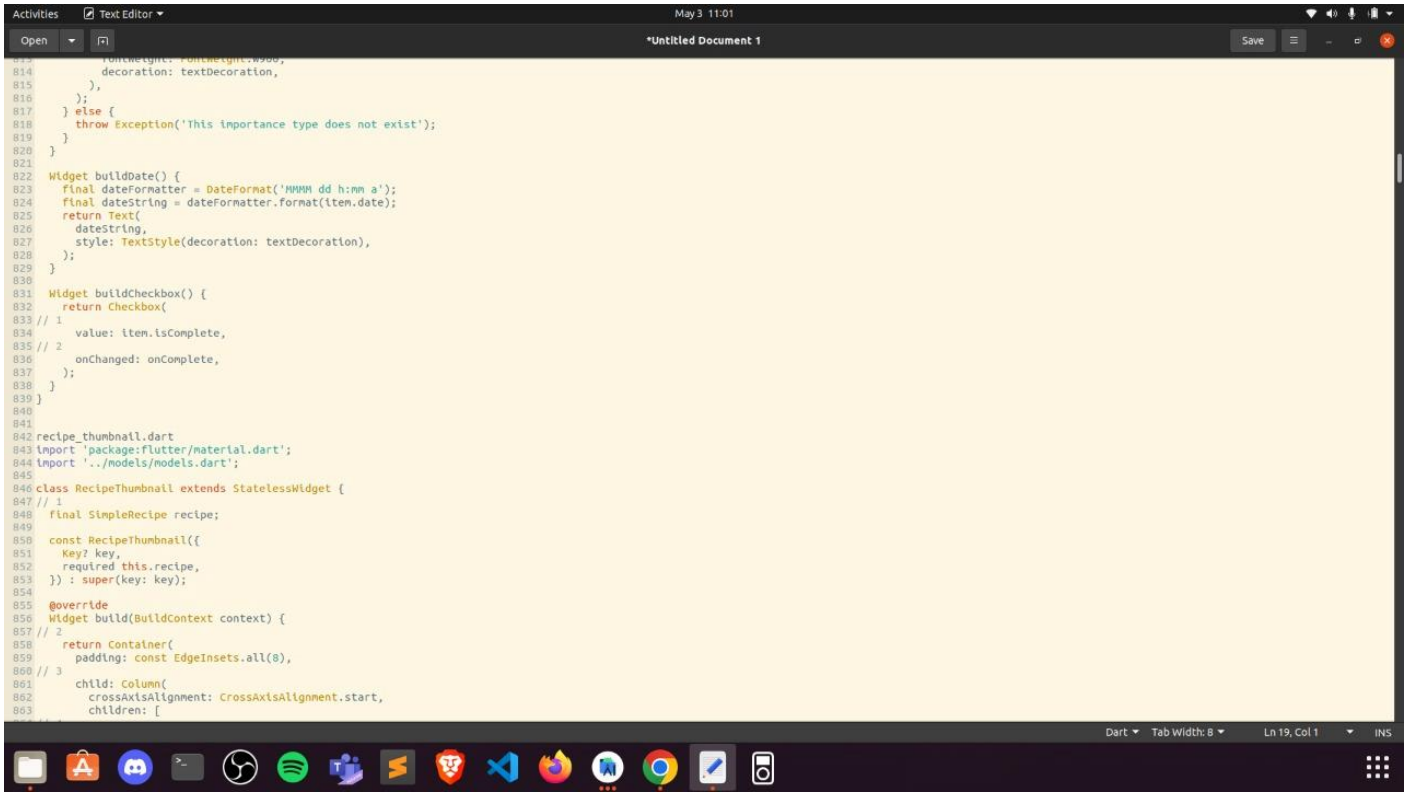

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
decoration: textDecoration,
),
);
} else {
throw Exception('This importance type does not exist');
}
}
Widget buildDate() {
final dateFormatter = DateFormat('MMMM dd h:mm a');
final dateString = dateFormatter.format(item.date);
return Text(
dateString,
style: TextStyle(decoration: textDecoration),
);
}
Widget buildCheckbox() {
return Checkbox(
// 1
value: item.isComplete,
// 2
onChanged: onComplete,
);
}
}
recipe_thumbnail.dart
import 'package:flutter/material.dart';
import '../models/models.dart';
class RecipeThumbnail extends StatelessWidget {
// 1
final SimpleRecipe recipe;
const RecipeThumbnail({
Key? key,
required this.recipe,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 2
return Container(
padding: const EdgeInsets.all(8),
// 3
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 48

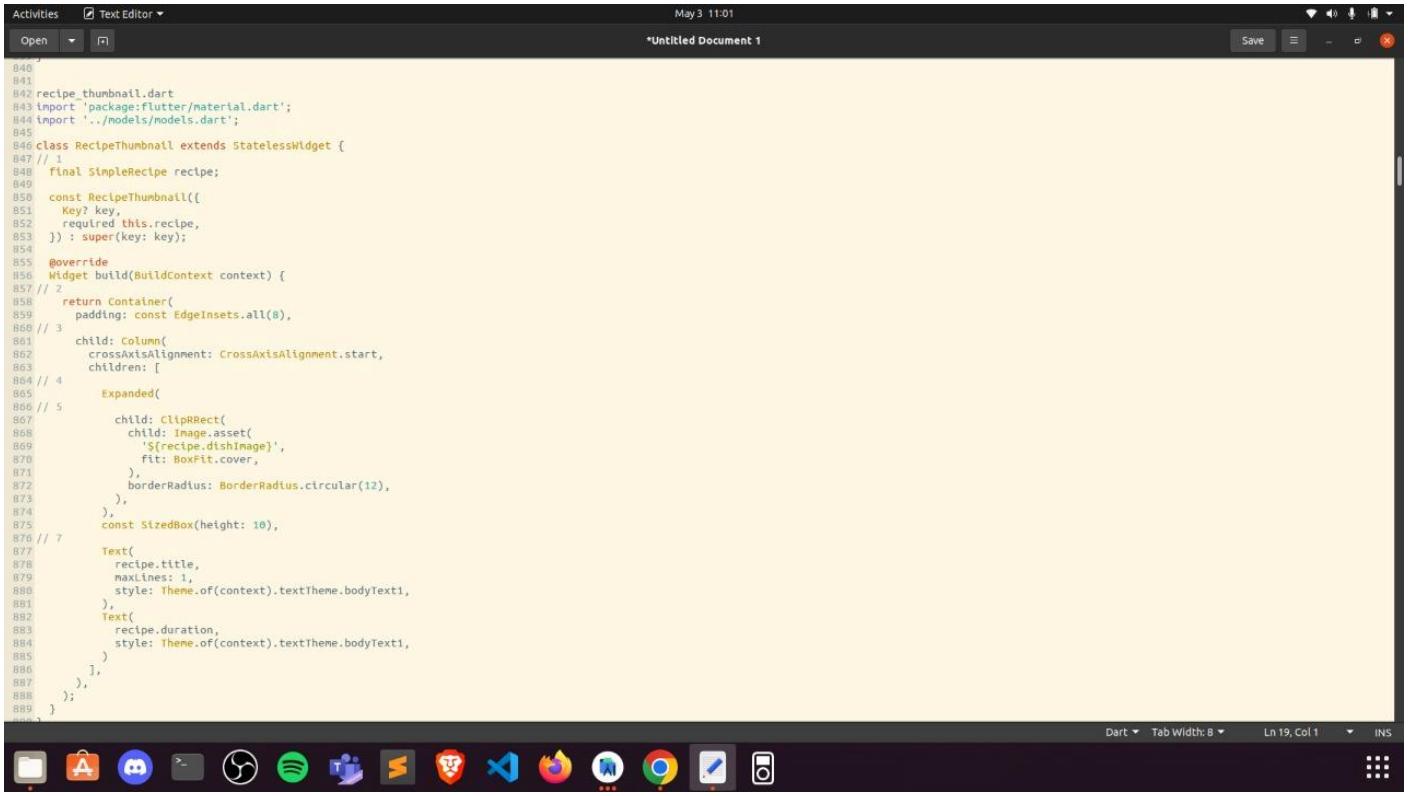

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
recipe_thumbnail.dart
import 'package:flutter/material.dart';
import '../models/models.dart';
class RecipeThumbnail extends StatelessWidget {
// 1
final SimpleRecipe recipe;
const RecipeThumbnail({
Key? key,
required this.recipe,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 2
return Container(
padding: const EdgeInsets.all(8),
// 3
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
// 4
Expanded(
// 5
child: ClipRRect(
child: Image.asset(
'${recipe.dishImage}',
fit: BoxFit.cover,
),
borderRadius: BorderRadius.circular(12),
),
),
const SizedBox(height: 10),
// 7
Text(
recipe.title,
maxLines: 1,
style: Theme.of(context).textTheme.bodyText1,
),
Text(
recipe.duration,
style: Theme.of(context).textTheme.bodyText1,
)
],
),
);
}
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 49

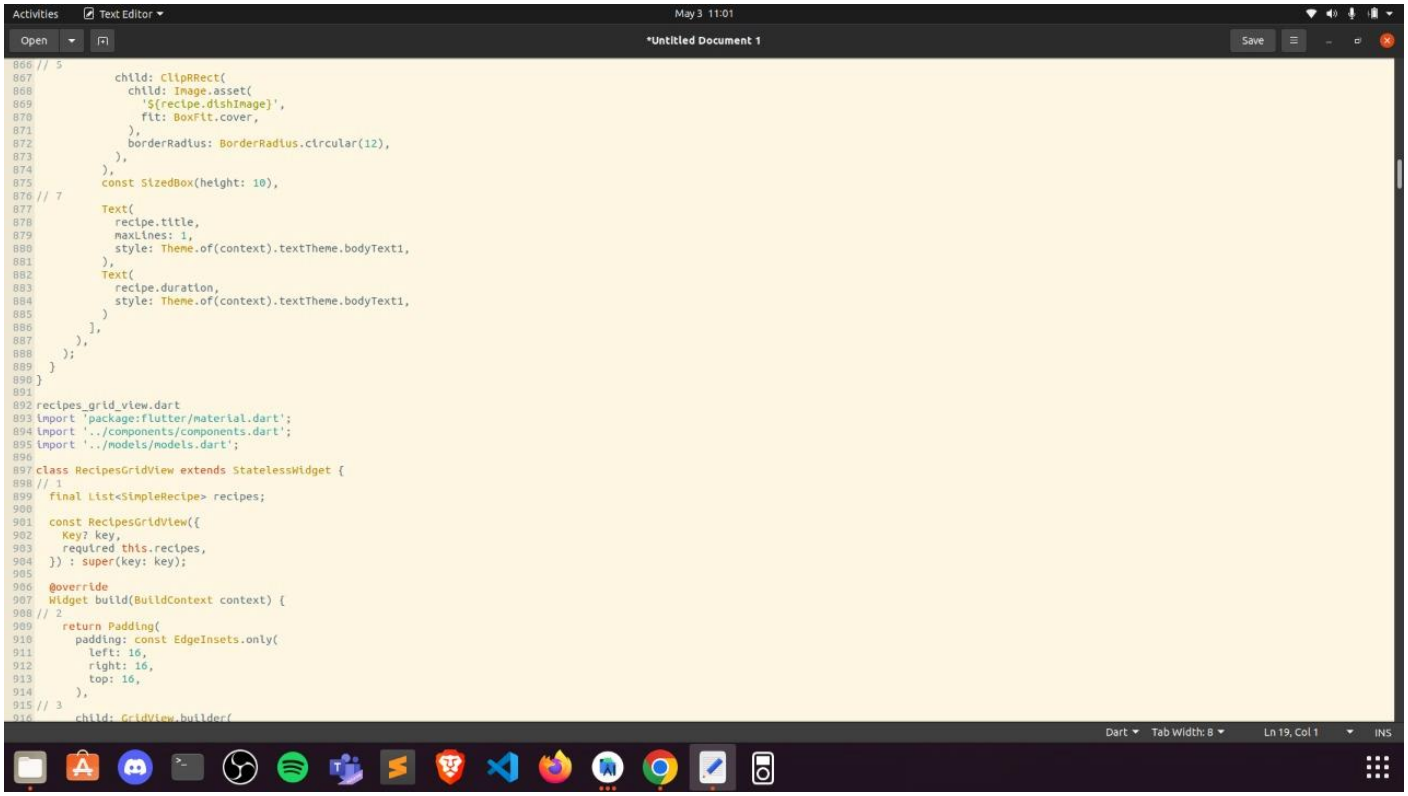

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
// 5
child: ClipRRect(
child: Image.asset(
'${recipe.dishImage}',
fit: BoxFit.cover,
),
borderRadius: BorderRadius.circular(12),
),
),
const SizedBox(height: 10),
// 7
Text(
recipe.title,
maxLines: 1,
style: Theme.of(context).textTheme.bodyText1,
),
Text(
recipe.duration,
style: Theme.of(context).textTheme.bodyText1,
)
],
),
);
}
}
recipes_grid_view.dart
import 'package:flutter/material.dart';
import '../components/components.dart';
import '../models/models.dart';
class RecipesGridView extends StatelessWidget {
// 1
final List<SimpleRecipe> recipes;
const RecipesGridView({
Key? key,
required this.recipes,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 2
return Padding(
padding: const EdgeInsets.only(
left: 16,
right: 16,
top: 16,
),
// 3
child: GridView.builder(
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 50

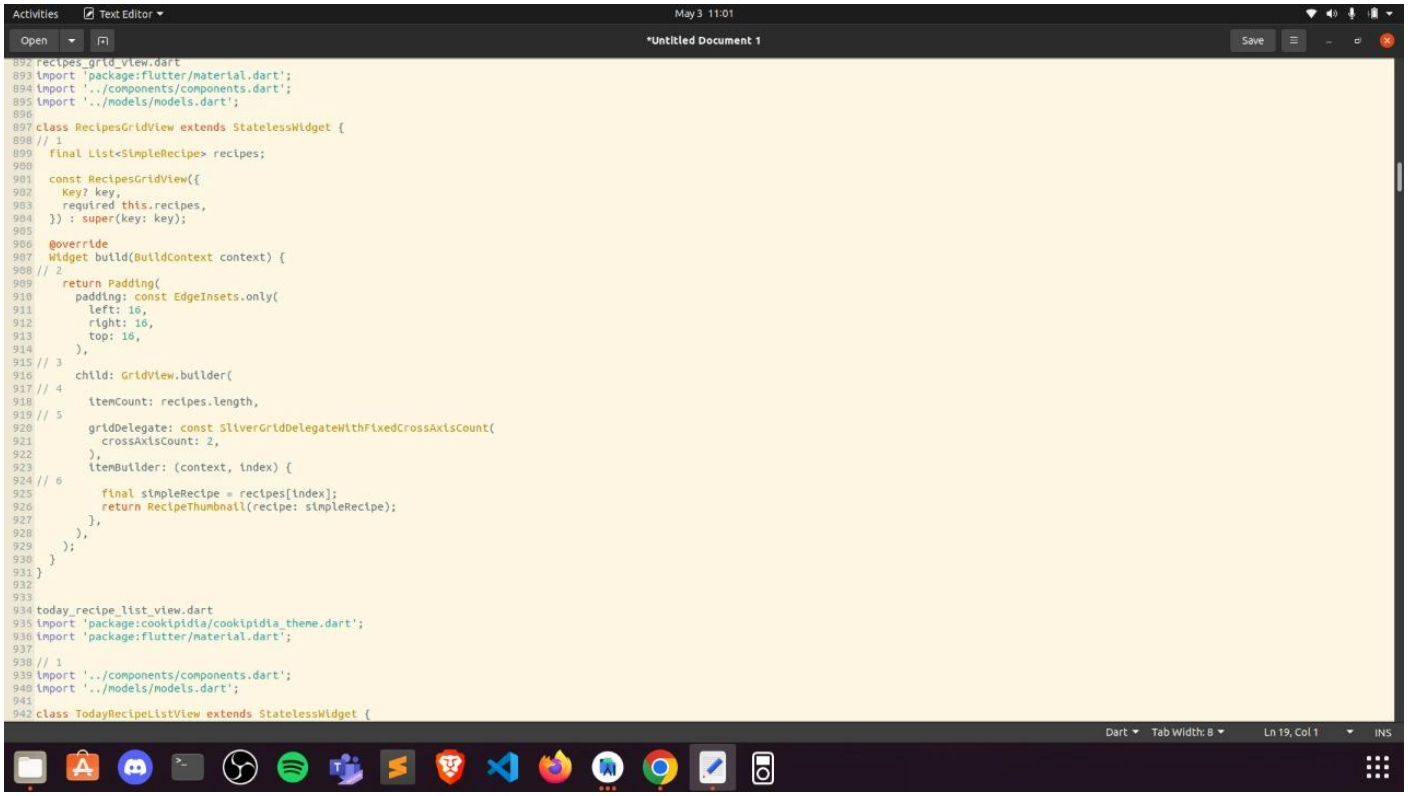

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
recipes_grid_view.dart
import 'package:flutter/material.dart';
import '../components/components.dart';
import '../models/models.dart';
class RecipesGridView extends StatelessWidget {
// 1
final List<SimpleRecipe> recipes;
const RecipesGridView({
Key? key,
required this.recipes,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 2
return Padding(
padding: const EdgeInsets.only(
left: 16,
right: 16,
top: 16,
),
// 3
child: GridView.builder(
// 4
itemCount: recipes.length,
// 5
gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
crossAxisCount: 2,
),
itemBuilder: (context, index) {
// 6
final simpleRecipe = recipes[index];
return RecipeThumbnail(recipe: simpleRecipe);
},
),
);
}
}
today_recipe_list_view.dart
import 'package:cookipidia/cookipidia_theme.dart';
import 'package:flutter/material.dart';
// 1
import '../components/components.dart';
import '../models/models.dart';
class TodayRecipeListView extends StatelessWidget {
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 51

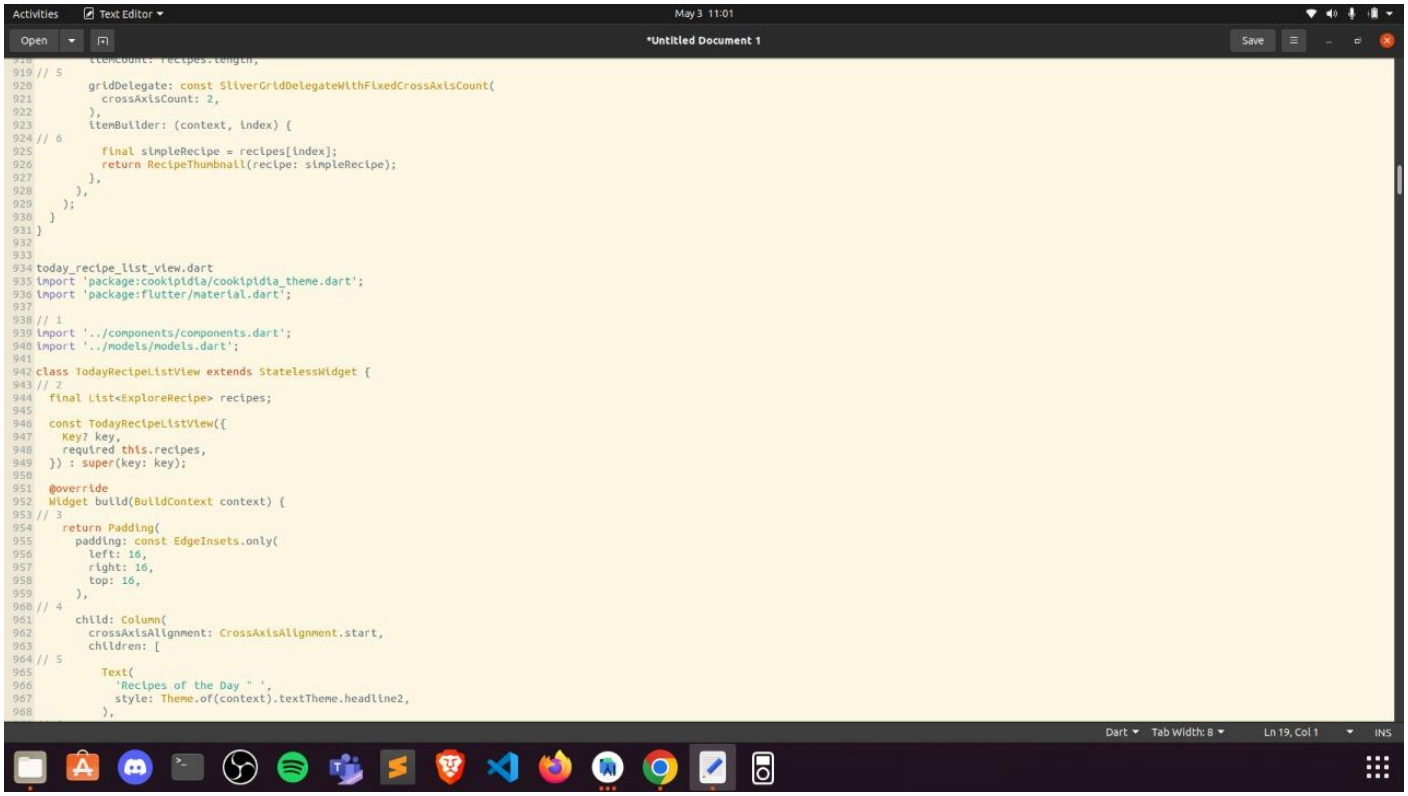

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
itemCount: recipes.length,
// 5
gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
crossAxisCount: 2,
),
itemBuilder: (context, index) {
// 6
final simpleRecipe = recipes[index];
return RecipeThumbnail(recipe: simpleRecipe);
},
),
);
}
}
today_recipe_list_view.dart
import 'package:cookipidia/cookipidia_theme.dart';
import 'package:flutter/material.dart';
// 1
import '../components/components.dart';
import '../models/models.dart';
class TodayRecipeListView extends StatelessWidget {
// 2
final List<ExploreRecipe> recipes;
const TodayRecipeListView({
Key? key,
required this.recipes,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 3
return Padding(
padding: const EdgeInsets.only(
left: 16,
right: 16,
top: 16,
),
// 4
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
// 5
Text(
'Recipes of the Day ',
style: Theme.of(context).textTheme.headline2,
),
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 52

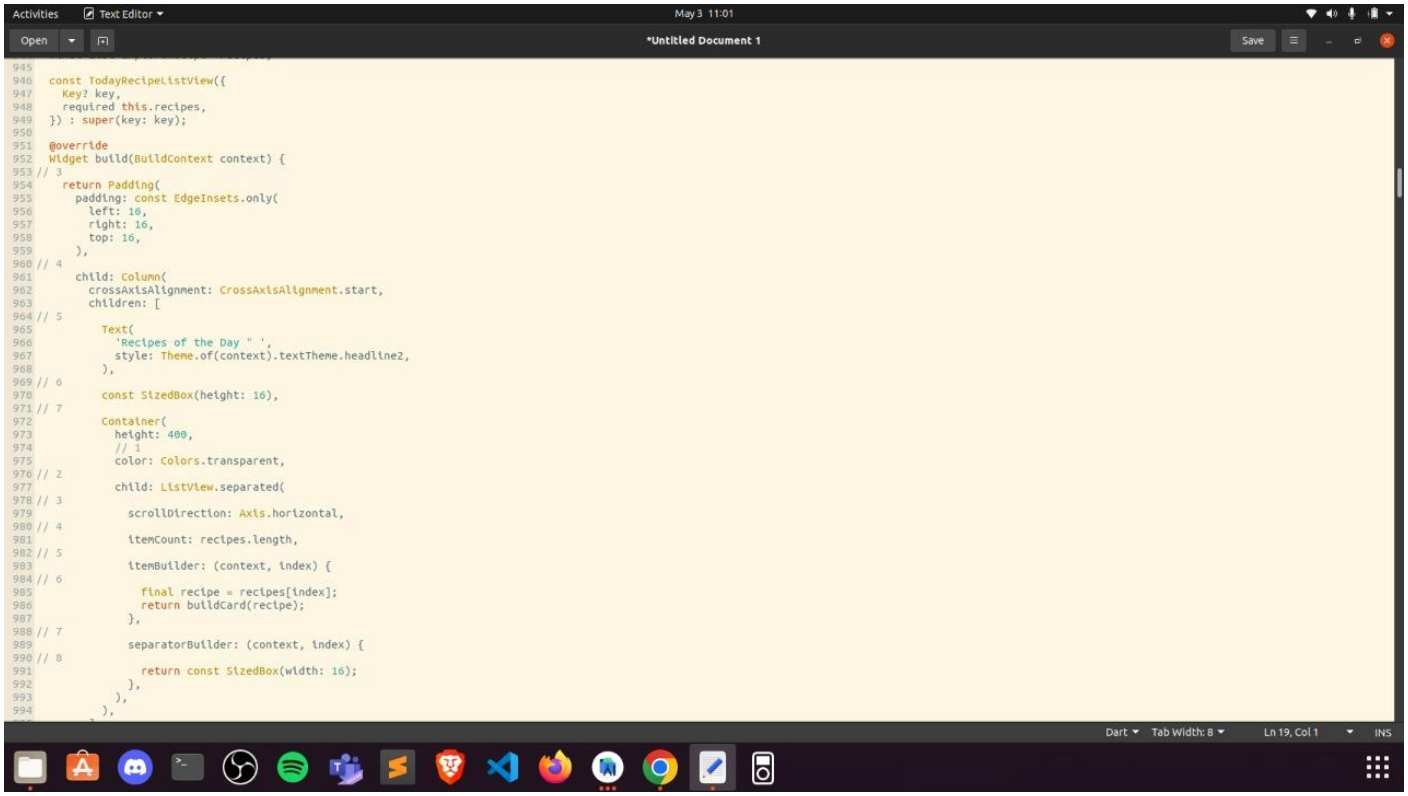

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
const TodayRecipeListView({
Key? key,
required this.recipes,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 3
return Padding(
padding: const EdgeInsets.only(
left: 16,
right: 16,
top: 16,
),
// 4
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
// 5
Text(
'Recipes of the Day ',
style: Theme.of(context).textTheme.headline2,
),
// 6
const SizedBox(height: 16),
// 7
Container(
height: 400,
// 1
color: Colors.transparent,
// 2
child: ListView.separated(
// 3
scrollDirection: Axis.horizontal,
// 4
itemCount: recipes.length,
// 5
itemBuilder: (context, index) {
// 6
final recipe = recipes[index];
return buildCard(recipe);
},
// 7
separatorBuilder: (context, index) {
// 8
return const SizedBox(width: 16);
},
),
),
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 53

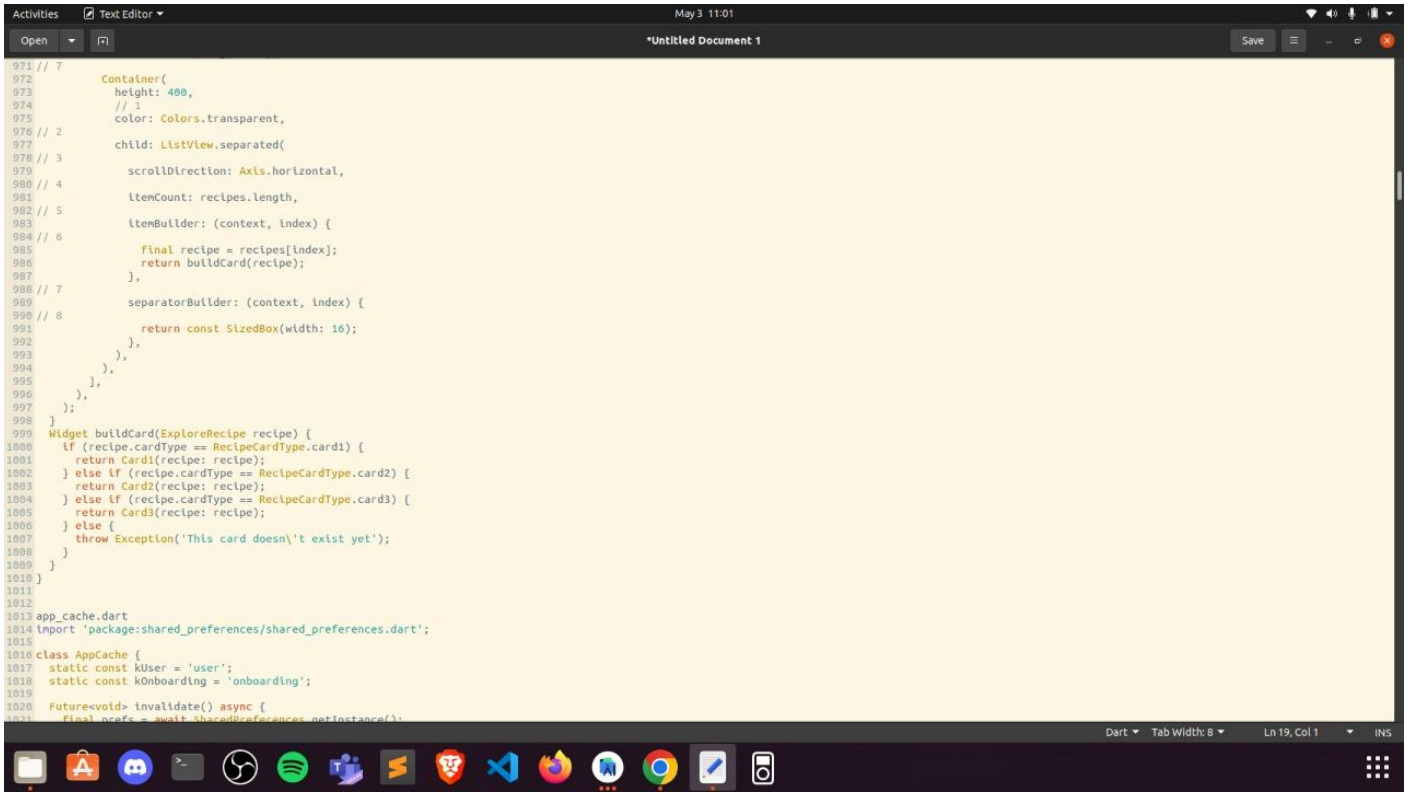

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
// 7
Container(
height: 400,
// 1
color: Colors.transparent,
// 2
child: ListView.separated(
// 3
scrollDirection: Axis.horizontal,
// 4
itemCount: recipes.length,
// 5
itemBuilder: (context, index) {
// 6
final recipe = recipes[index];
return buildCard(recipe);
},
// 7
separatorBuilder: (context, index) {
// 8
return const SizedBox(width: 16);
},
),
),
],
),
);
}
Widget buildCard(ExploreRecipe recipe) {
if (recipe.cardType == RecipeCardType.card1) {
return Card1(recipe: recipe);
} else if (recipe.cardType == RecipeCardType.card2) {
return Card2(recipe: recipe);
} else if (recipe.cardType == RecipeCardType.card3) {
return Card3(recipe: recipe);
} else {
throw Exception('This card doesn\'t exist yet');
}
}
}
app_cache.dart
import 'package:shared_preferences/shared_preferences.dart';
class AppCache {
static const kUser = 'user';
static const kOnboarding = 'onboarding';
Future<void> invalidate() async {
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 54

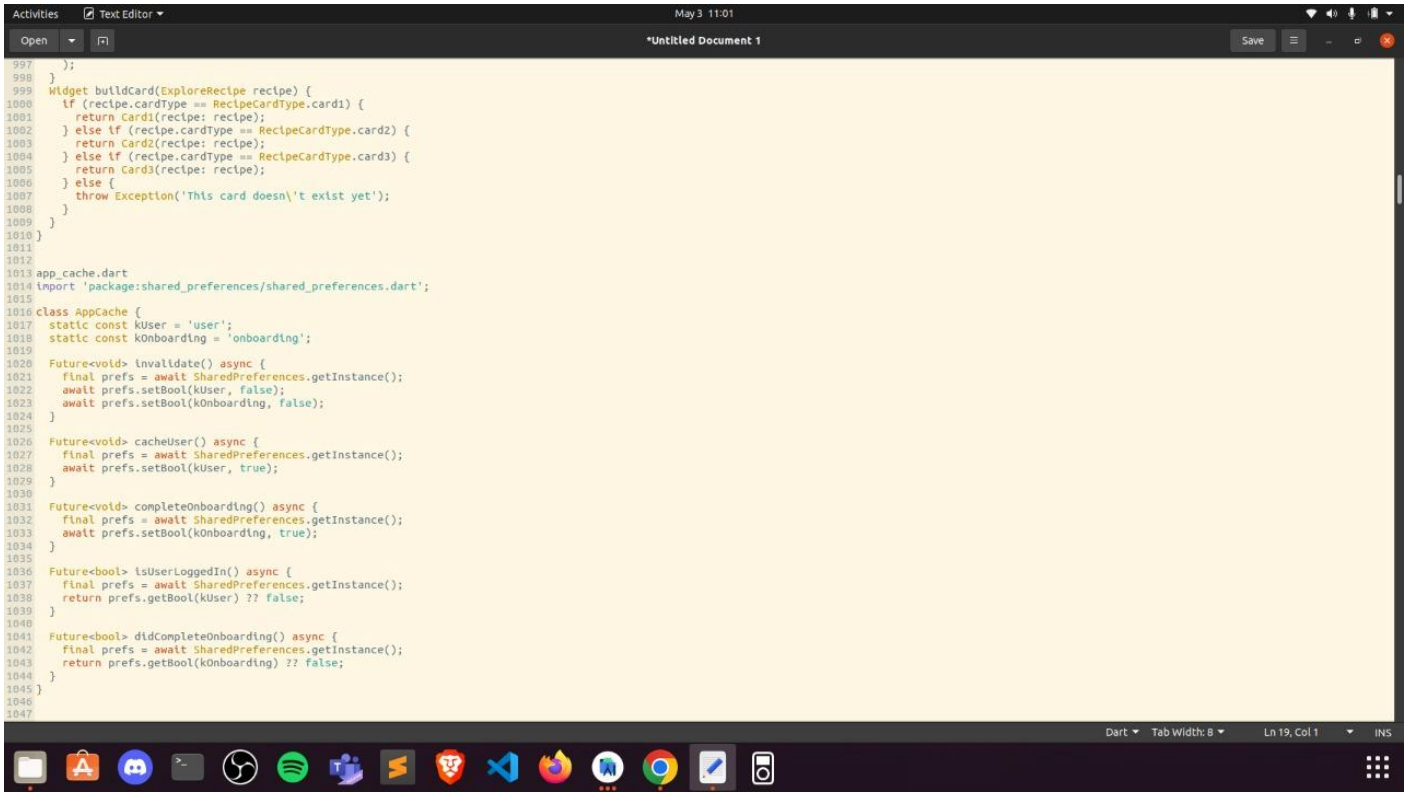

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
);
}
Widget buildCard(ExploreRecipe recipe) {
if (recipe.cardType == RecipeCardType.card1) {
return Card1(recipe: recipe);
} else if (recipe.cardType == RecipeCardType.card2) {
return Card2(recipe: recipe);
} else if (recipe.cardType == RecipeCardType.card3) {
return Card3(recipe: recipe);
} else {
throw Exception('This card doesn\'t exist yet');
}
}
}
app_cache.dart
import 'package:shared_preferences/shared_preferences.dart';
class AppCache {
static const kUser = 'user';
static const kOnboarding = 'onboarding';
Future<void> invalidate() async {
final prefs = await SharedPreferences.getInstance();
await prefs.setBool(kUser, false);
await prefs.setBool(kOnboarding, false);
}
Future<void> cacheUser() async {
final prefs = await SharedPreferences.getInstance();
await prefs.setBool(kUser, true);
}
Future<void> completeOnboarding() async {
final prefs = await SharedPreferences.getInstance();
await prefs.setBool(kOnboarding, true);
}
Future<bool> isUserLoggedIn() async {
final prefs = await SharedPreferences.getInstance();
return prefs.getBool(kUser) ?? false;
}
Future<bool> didCompleteOnboarding() async {
final prefs = await SharedPreferences.getInstance();
return prefs.getBool(kOnboarding) ?? false;
}
}
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 55

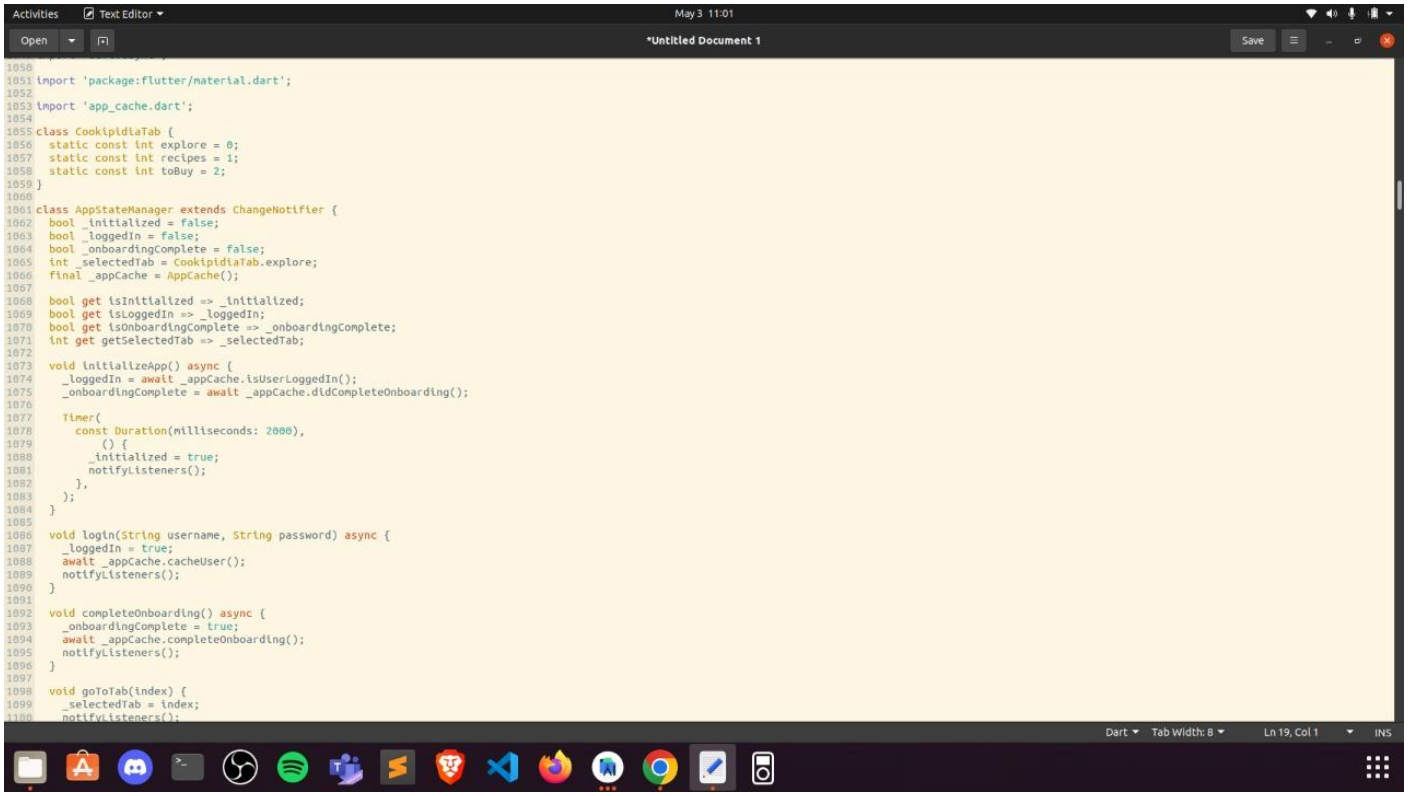

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
import 'package:flutter/material.dart';
import 'app_cache.dart';
class CookipidiaTab {
static const int explore = 0;
static const int recipes = 1;
static const int toBuy = 2;
}
class AppStateManager extends ChangeNotifier {
bool _initialized = false;
bool _loggedIn = false;
bool _onboardingComplete = false;
int _selectedTab = CookipidiaTab.explore;
final _appCache = AppCache();
bool get isInitialized => _initialized;
bool get isLoggedIn => _loggedIn;
bool get isOnboardingComplete => _onboardingComplete;
int get getSelectedTab => _selectedTab;
void initializeApp() async {
_loggedIn = await _appCache.isUserLoggedIn();
_onboardingComplete = await _appCache.didCompleteOnboarding();
Timer(
const Duration(milliseconds: 2000),
() {
_initialized = true;
notifyListeners();
},
);
}
void login(String username, String password) async {
_loggedIn = true;
await _appCache.cacheUser();
notifyListeners();
}
void completeOnboarding() async {
_onboardingComplete = true;
await _appCache.completeOnboarding();
notifyListeners();
}
void goToTab(index) {
_selectedTab = index;
notifyListeners();
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 56

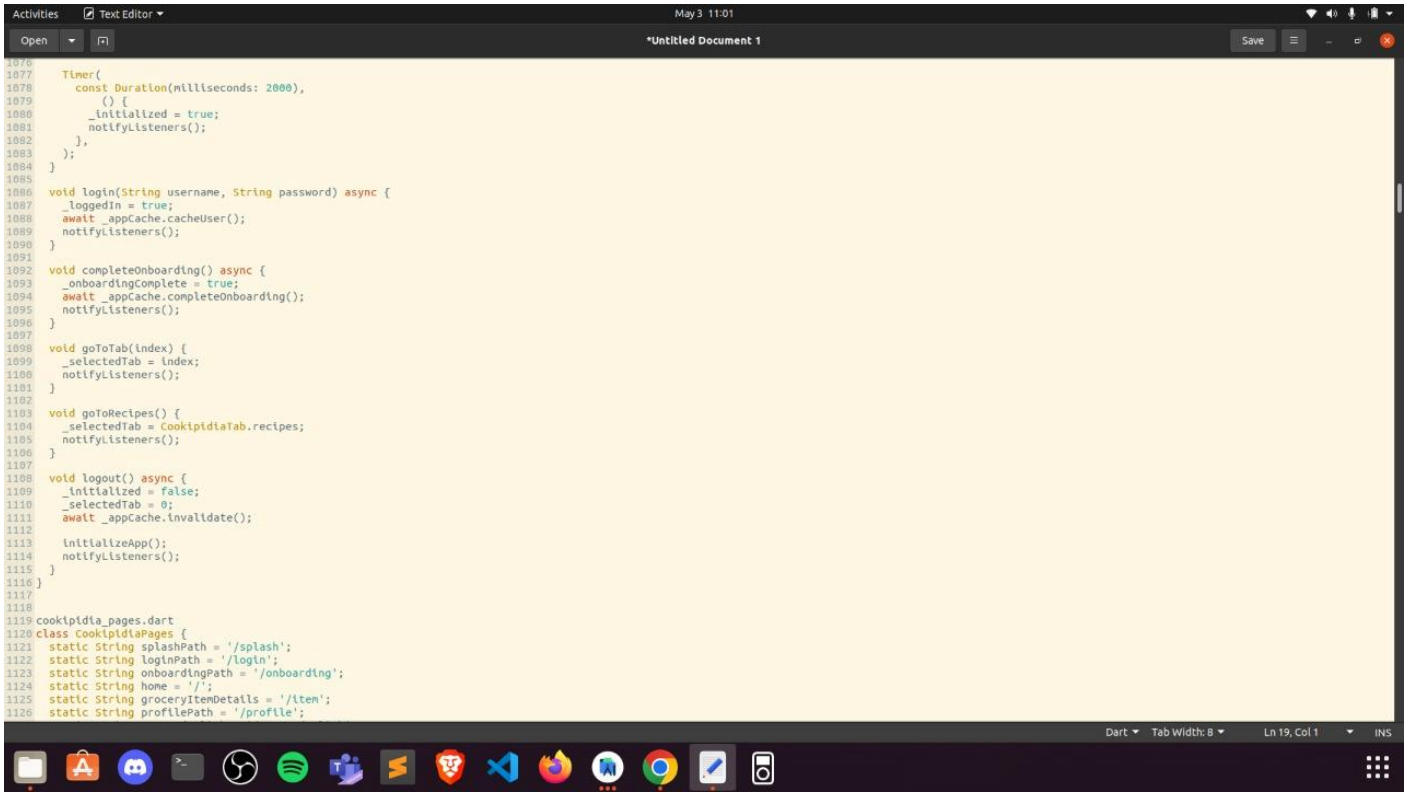

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
Timer(
const Duration(milliseconds: 2000),
() {
_initialized = true;
notifyListeners();
},
);
}
void login(String username, String password) async {
_loggedIn = true;
await _appCache.cacheUser();
notifyListeners();
}
void completeOnboarding() async {
_onboardingComplete = true;
await _appCache.completeOnboarding();
notifyListeners();
}
void goToTab(index) {
_selectedTab = index;
notifyListeners();
}
void goToRecipes() {
_selectedTab = CookipidiaTab.recipes;
notifyListeners();
}
void logout() async {
_initialized = false;
_selectedTab = 0;
await _appCache.invalidate();
initializeApp();
notifyListeners();
}
}
cookipidia_pages.dart
class CookipidiaPages {
static String splashPath = '/splash';
static String loginPath = '/login';
static String onboardingPath = '/onboarding';
static String home = '/';
static String groceryItemDetails = '/item';
static String profilePath = '/profile';
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 57

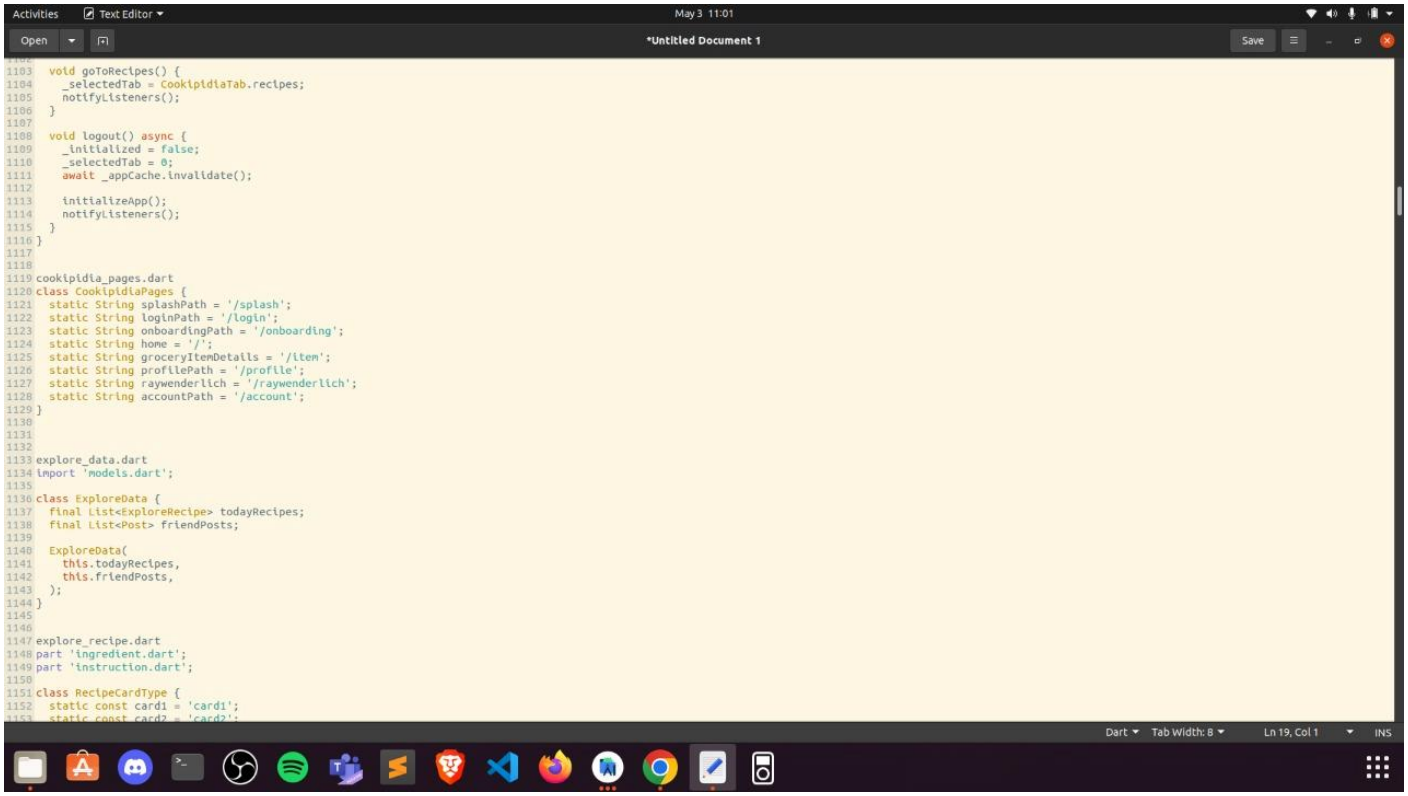

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
void goToRecipes() {
_selectedTab = CookipidiaTab.recipes;
notifyListeners();
}
void logout() async {
_initialized = false;
_selectedTab = 0;
await _appCache.invalidate();
initializeApp();
notifyListeners();
}
}
cookipidia_pages.dart
class CookipidiaPages {
static String splashPath = '/splash';
static String loginPath = '/login';
static String onboardingPath = '/onboarding';
static String home = '/';
static String groceryItemDetails = '/item';
static String profilePath = '/profile';
static String raywenderlich = '/raywenderlich';
static String accountPath = '/account';
}
explore_data.dart
import 'models.dart';
class ExploreData {
final List<ExploreRecipe> todayRecipes;
final List<Post> friendPosts;
ExploreData(
this.todayRecipes,
this.friendPosts,
);
}
explore_recipe.dart
part 'ingredient.dart';
part 'instruction.dart';
class RecipeCardType {
static const card1 = 'card1';
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 58

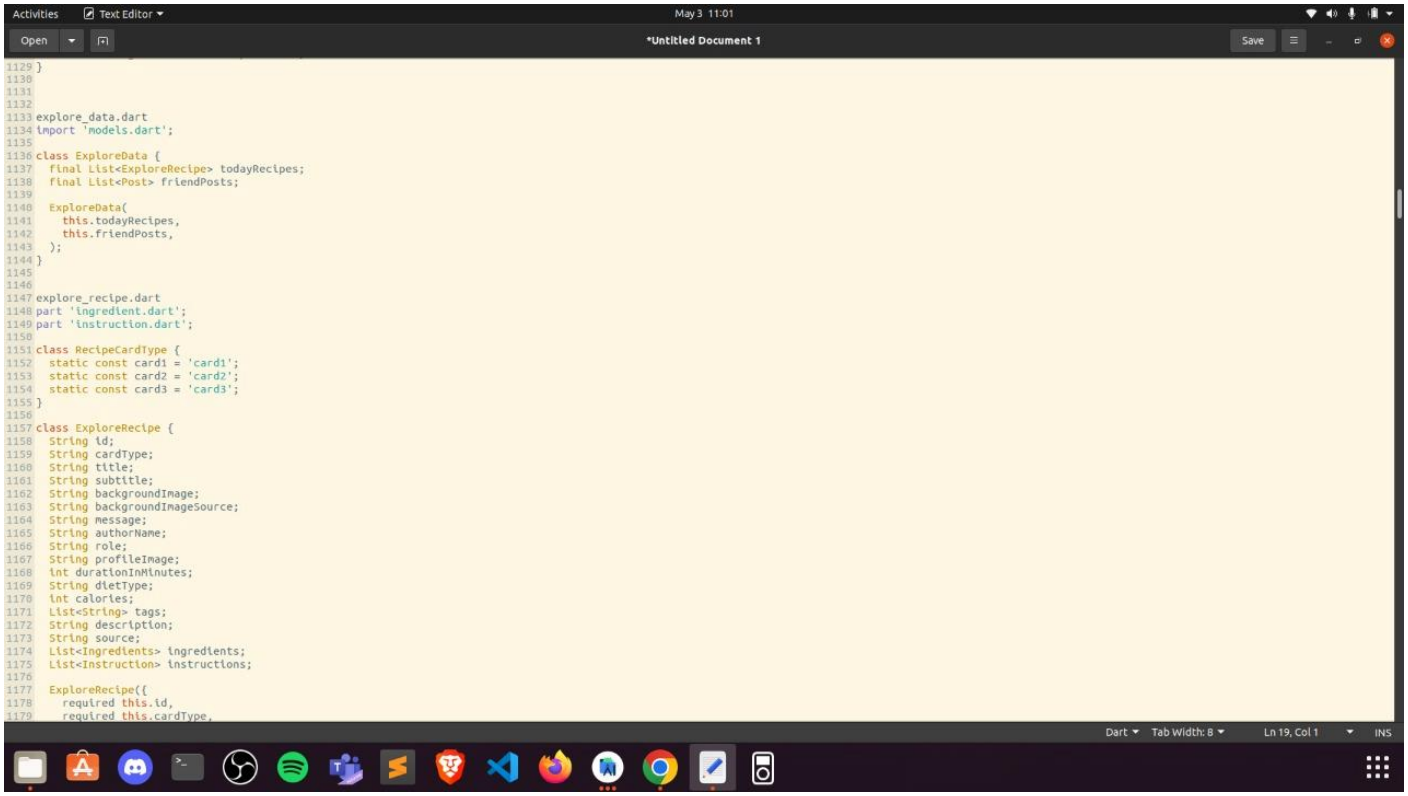

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
}
explore_data.dart
import 'models.dart';
class ExploreData {
final List<ExploreRecipe> todayRecipes;
final List<Post> friendPosts;
ExploreData(
this.todayRecipes,
this.friendPosts,
);
}
explore_recipe.dart
part 'ingredient.dart';
part 'instruction.dart';
class RecipeCardType {
static const card1 = 'card1';
static const card2 = 'card2';
static const card3 = 'card3';
}
class ExploreRecipe {
String id;
String cardType;
String title;
String subtitle;
String backgroundImage;
String backgroundImageSource;
String message;
String authorName;
String role;
String profileImage;
int durationInMinutes;
String dietType;
int calories;
List<String> tags;
String description;
String source;
List<Ingredients> ingredients;
List<Instruction> instructions;
ExploreRecipe({
required this.id,
required this.cardType,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 59

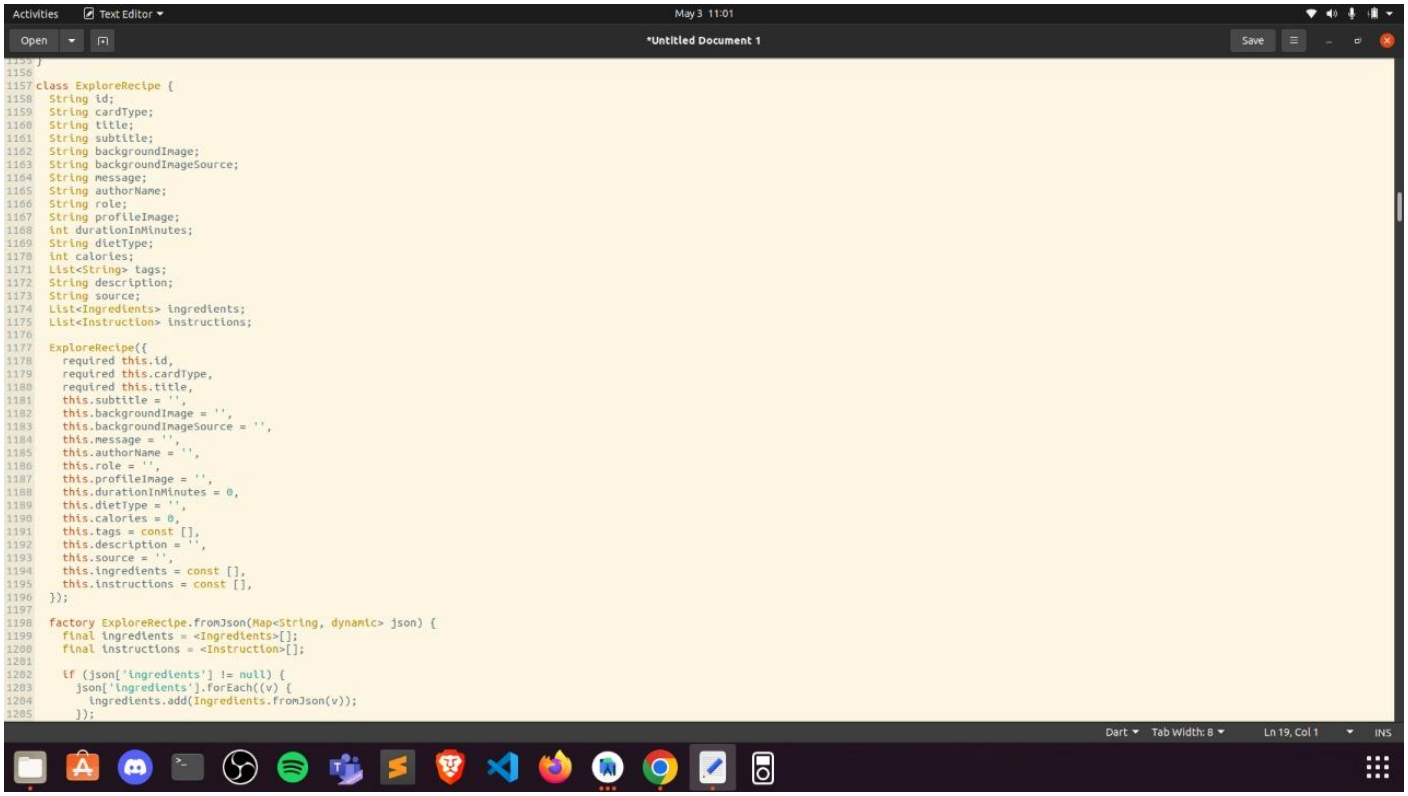

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
class ExploreRecipe {
String id;
String cardType;
String title;
String subtitle;
String backgroundImage;
String backgroundImageSource;
String message;
String authorName;
String role;
String profileImage;
int durationInMinutes;
String dietType;
int calories;
List<String> tags;
String description;
String source;
List<Ingredients> ingredients;
List<Instruction> instructions;
ExploreRecipe({
required this.id,
required this.cardType,
required this.title,
this.subtitle = '',
this.backgroundImage = '',
this.backgroundImageSource = '',
this.message = '',
this.authorName = '',
this.role = '',
this.profileImage = '',
this.durationInMinutes = 0,
this.dietType = '',
this.calories = 0,
this.tags = const [],
this.description = '',
this.source = '',
this.ingredients = const [],
this.instructions = const [],
});
factory ExploreRecipe.fromJson(Map<String, dynamic> json) {
final ingredients = <Ingredients>[];
final instructions = <Instruction>[];
if (json['ingredients'] != null) {
json['ingredients'].forEach((v) {
ingredients.add(Ingredients.fromJson(v));
});
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 60

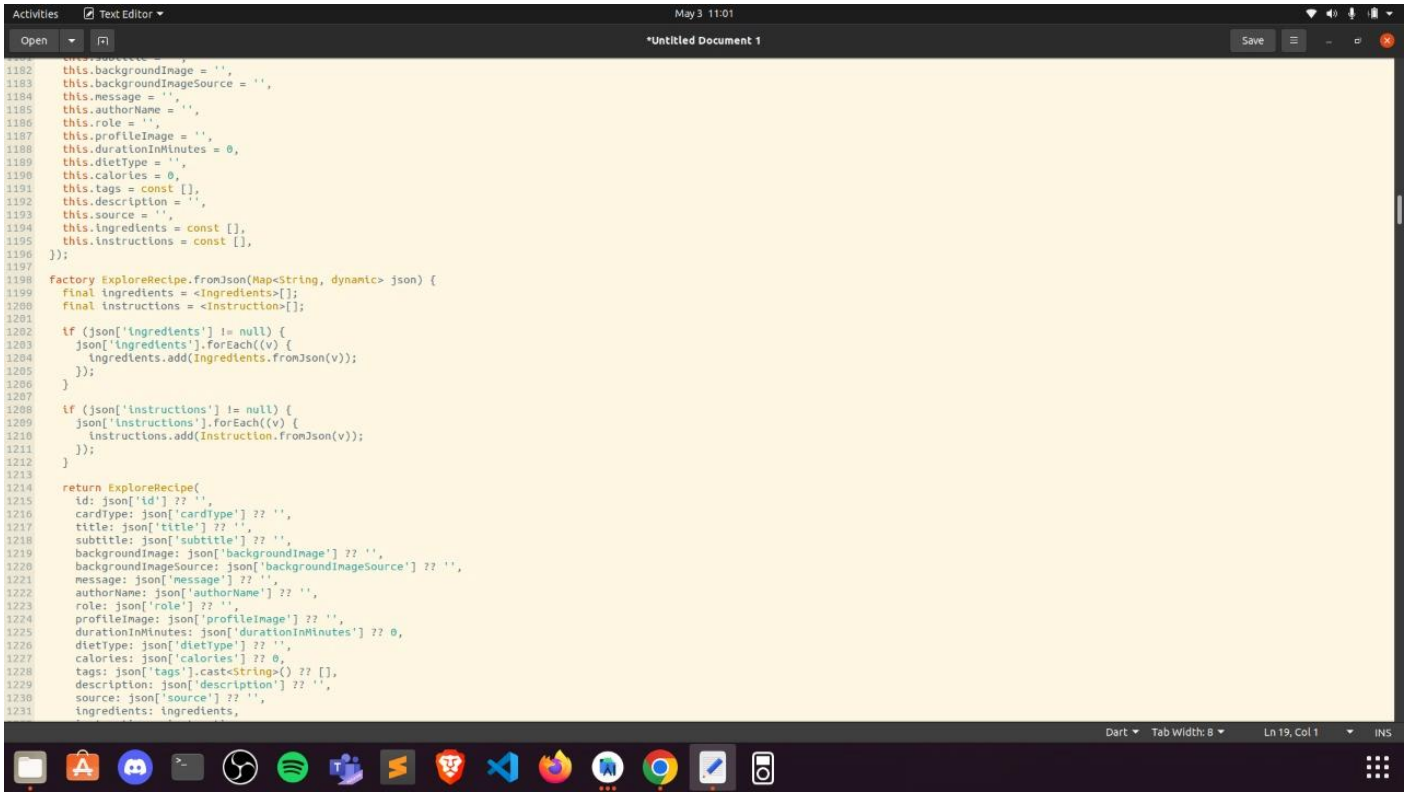

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
this.backgroundImage = '',
this.backgroundImageSource = '',
this.message = '',
this.authorName = '',
this.role = '',
this.profileImage = '',
this.durationInMinutes = 0,
this.dietType = '',
this.calories = 0,
this.tags = const [],
this.description = '',
this.source = '',
this.ingredients = const [],
this.instructions = const [],
});
factory ExploreRecipe.fromJson(Map<String, dynamic> json) {
final ingredients = <Ingredients>[];
final instructions = <Instruction>[];
if (json['ingredients'] != null) {
json['ingredients'].forEach((v) {
ingredients.add(Ingredients.fromJson(v));
});
}
if (json['instructions'] != null) {
json['instructions'].forEach((v) {
instructions.add(Instruction.fromJson(v));
});
}
return ExploreRecipe(
id: json['id'] ?? '',
cardType: json['cardType'] ?? '',
title: json['title'] ?? '',
subtitle: json['subtitle'] ?? '',
backgroundImage: json['backgroundImage'] ?? '',
backgroundImageSource: json['backgroundImageSource'] ?? '',
message: json['message'] ?? '',
authorName: json['authorName'] ?? '',
role: json['role'] ?? '',
profileImage: json['profileImage'] ?? '',
durationInMinutes: json['durationInMinutes'] ?? 0,
dietType: json['dietType'] ?? '',
calories: json['calories'] ?? 0,
tags: json['tags'].cast<String>() ?? [],
description: json['description'] ?? '',
source: json['source'] ?? '',
ingredients: ingredients,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 61

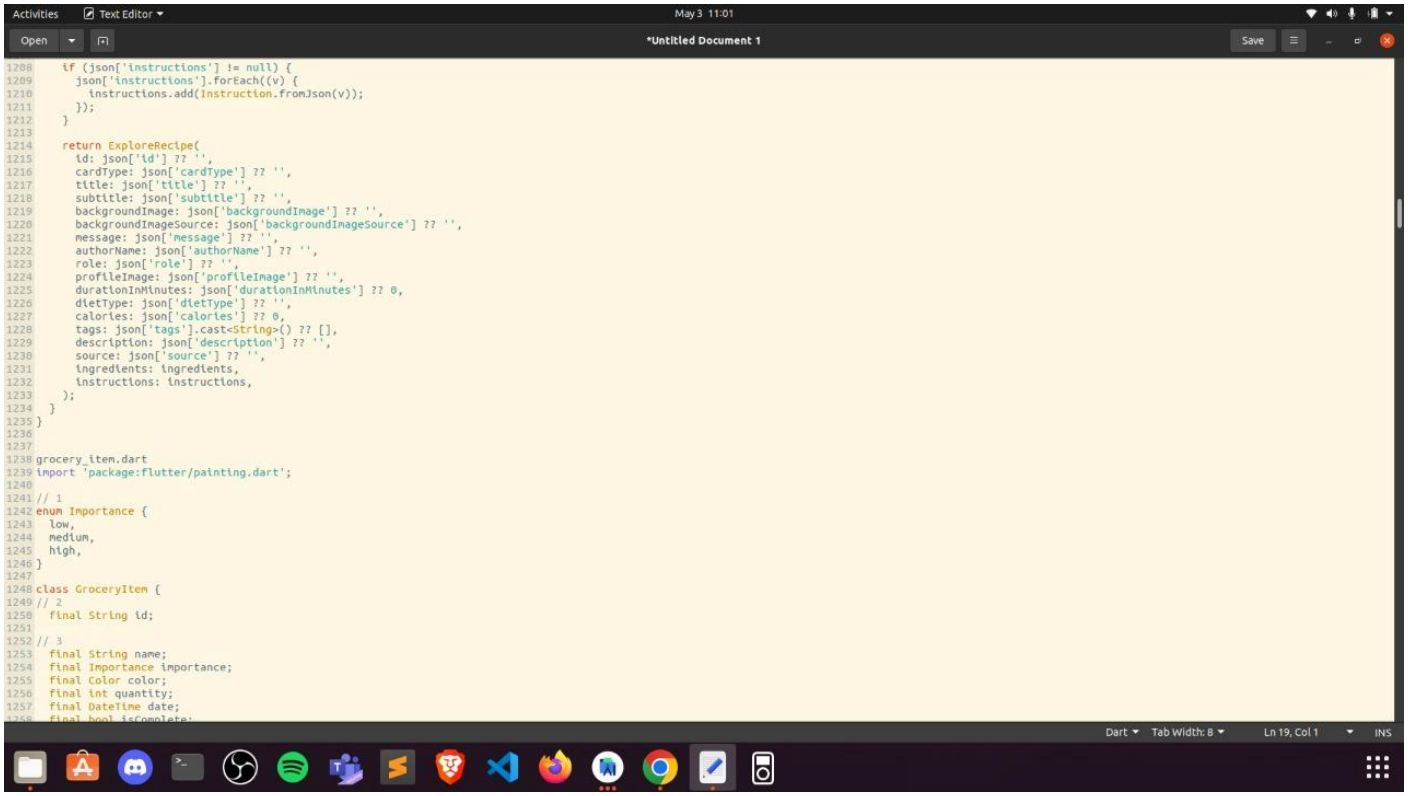

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
if (json['instructions'] != null) {
json['instructions'].forEach((v) {
instructions.add(Instruction.fromJson(v));
});
}
return ExploreRecipe(
id: json['id'] ?? '',
cardType: json['cardType'] ?? '',
title: json['title'] ?? '',
subtitle: json['subtitle'] ?? '',
backgroundImage: json['backgroundImage'] ?? '',
backgroundImageSource: json['backgroundImageSource'] ?? '',
message: json['message'] ?? '',
authorName: json['authorName'] ?? '',
role: json['role'] ?? '',
profileImage: json['profileImage'] ?? '',
durationInMinutes: json['durationInMinutes'] ?? 0,
dietType: json['dietType'] ?? '',
calories: json['calories'] ?? 0,
tags: json['tags'].cast<String>() ?? [],
description: json['description'] ?? '',
source: json['source'] ?? '',
ingredients: ingredients,
instructions: instructions,
);
}
}
grocery_item.dart
import 'package:flutter/painting.dart';
// 1
enum Importance {
low,
medium,
high,
}
class GroceryItem {
// 2
final String id;
// 3
final String name;
final Importance importance;
final Color color;
final int quantity;
final DateTime date;
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 62

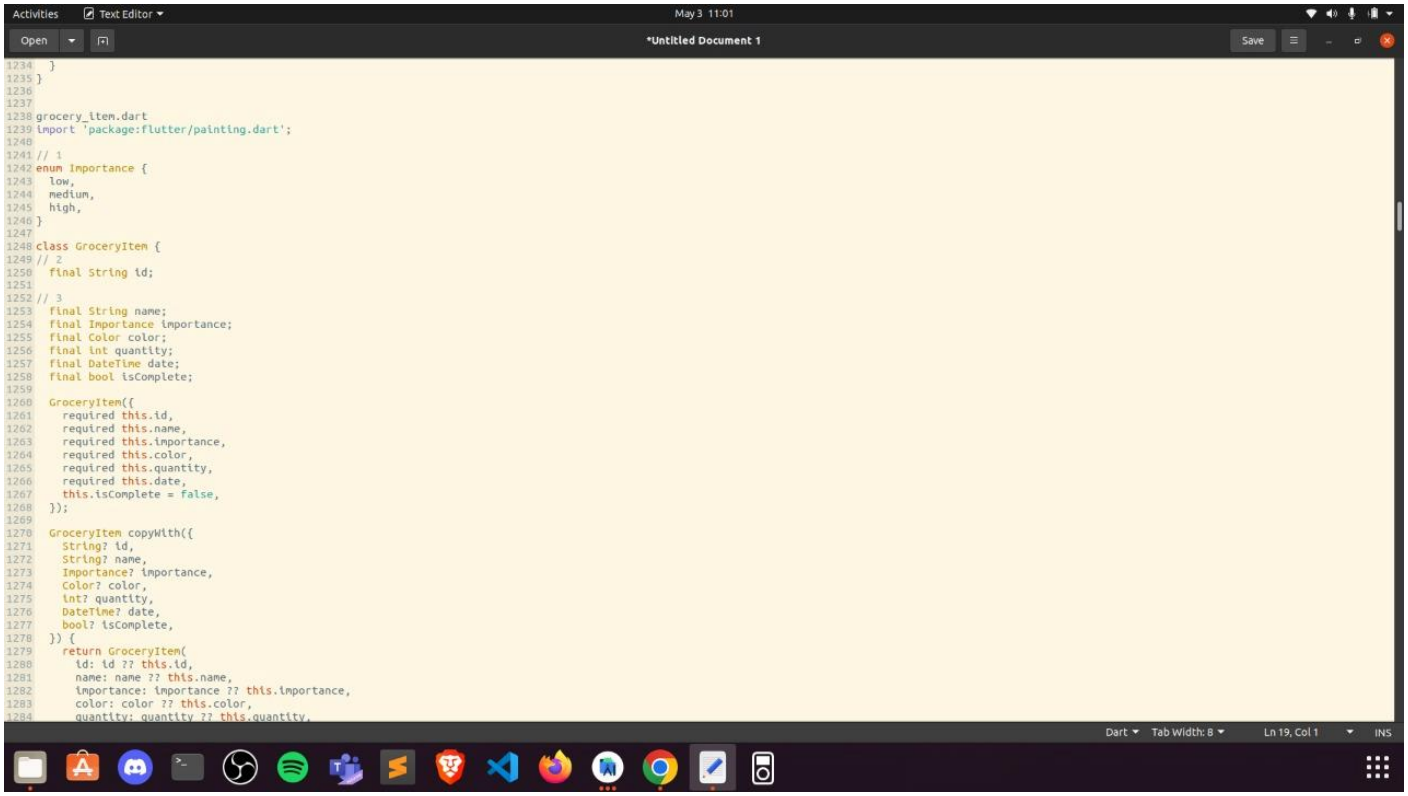
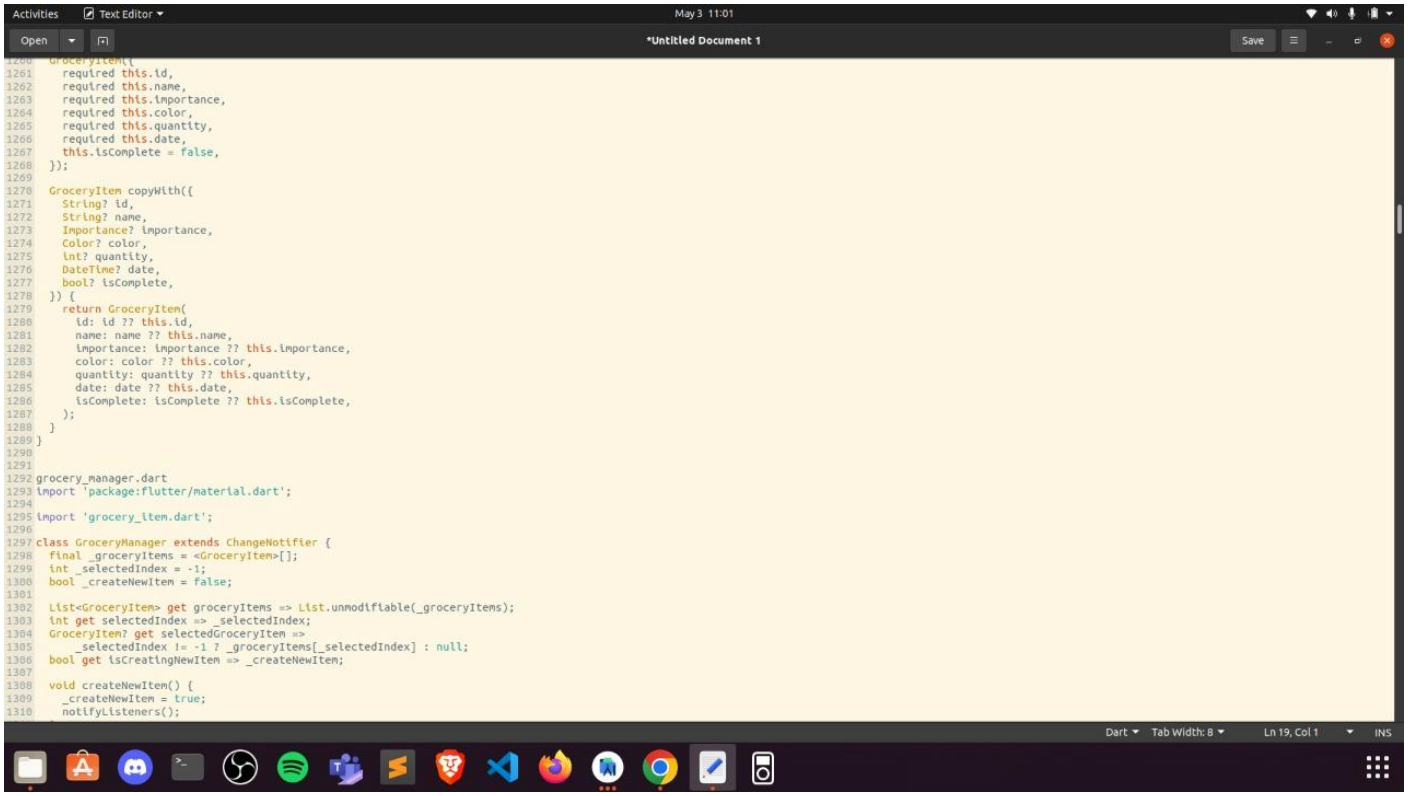
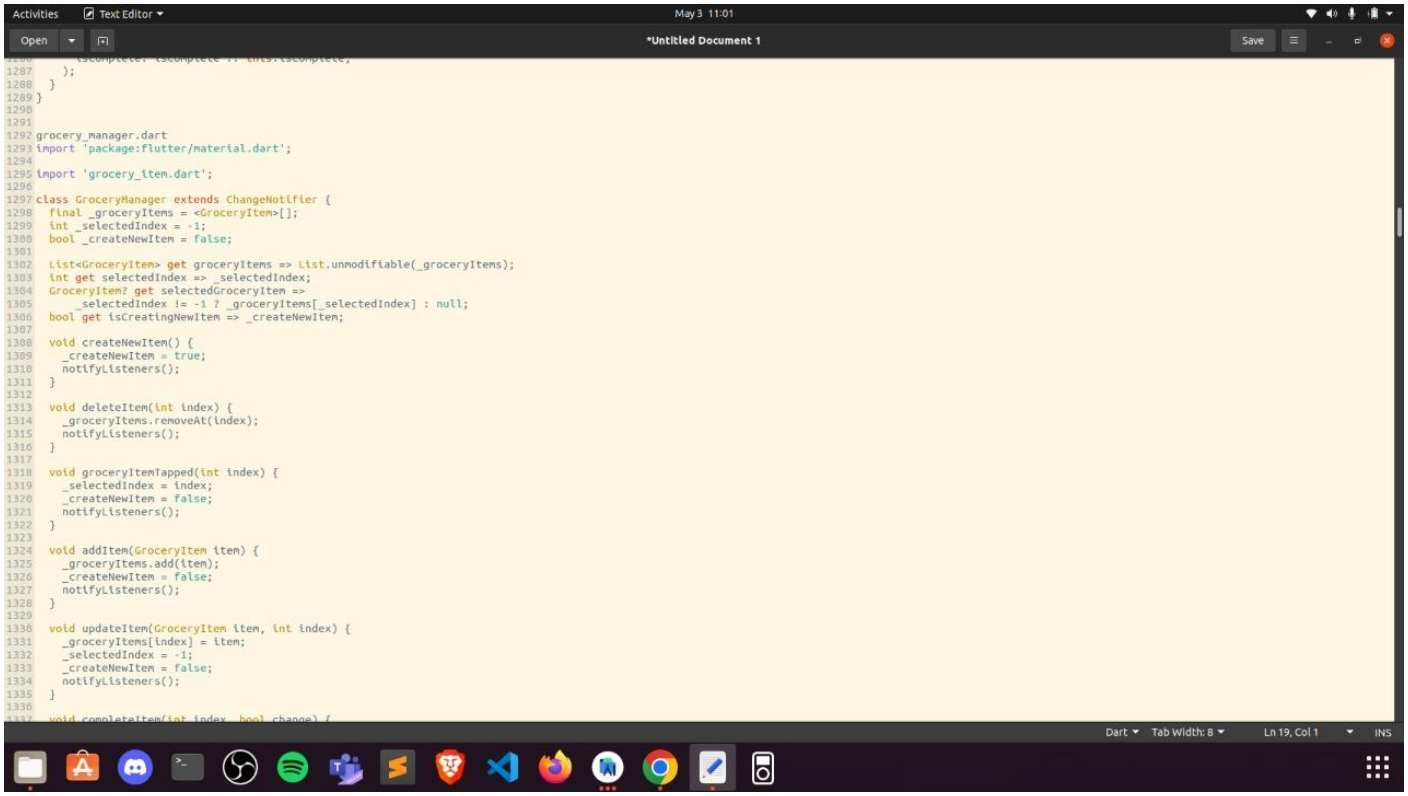

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
}
}
grocery_item.dart
import 'package:flutter/painting.dart';
// 1
enum Importance {
low,
medium,
high,
}
class GroceryItem {
// 2
final String id;
// 3
final String name;
final Importance importance;
final Color color;
final int quantity;
final DateTime date;
final bool isComplete;
GroceryItem({
required this.id,
required this.name,
required this.importance,
required this.color,
required this.quantity,
required this.date,
this.isComplete = false,
});
GroceryItem copyWith({
String? id,
String? name,
Importance? importance,
Color? color,
int? quantity,
DateTime? date,
bool? isComplete,
}) {
return GroceryItem(
id: id ?? this.id,
name: name ?? this.name,
importance: importance ?? this.importance,
color: color ?? this.color,
quantity: quantity ?? this.quantity,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 63




```
  GroceryItem({
    required this.id,
    required this.name,
    required this.importance,
    required this.color,
    required this.quantity,
    required this.date,
    this.isComplete = false,
  });

  GroceryItem copyWith({
    String? id,
    String? name,
    Importance? importance,
    Color? color,
    int? quantity,
    DateTime? date,
    bool? isComplete,
  }) {
    return GroceryItem(
      id: id ?? this.id,
      name: name ?? this.name,
      importance: importance ?? this.importance,
      color: color ?? this.color,
      quantity: quantity ?? this.quantity,
      date: date ?? this.date,
      isComplete: isComplete ?? this.isComplete,
    );
  }
}


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

import 'grocery_item.dart';

class GroceryManager extends ChangeNotifier {
  final _groceryItems = <GroceryItem>[];
  int _selectedIndex = -1;
  bool _createNewItem = false;

  List<GroceryItem> get groceryItems => List.unmodifiable(_groceryItems);
  int get selectedIndex => _selectedIndex;
  GroceryItem? get selectedGroceryItem =>
      _selectedIndex != -1 ? _groceryItems[_selectedIndex] : null;
  bool get isCreatingNewItem => _createNewItem;

  void createNewItem() {
    _createNewItem = true;
    notifyListeners();
```

Fig. 64




```
    );
  }
}


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

import 'grocery_item.dart';

class GroceryManager extends ChangeNotifier {
  final _groceryItems = <GroceryItem>[];
  int _selectedIndex = -1;
  bool _createNewItem = false;

  List<GroceryItem> get groceryItems => List.unmodifiable(_groceryItems);
  int get selectedIndex => _selectedIndex;
  GroceryItem? get selectedGroceryItem =>
      _selectedIndex != -1 ? _groceryItems[_selectedIndex] : null;
  bool get isCreatingNewItem => _createNewItem;

  void createNewItem() {
    _createNewItem = true;
    notifyListeners();
  }

  void deleteItem(int index) {
    _groceryItems.removeAt(index);
    notifyListeners();
  }

  void groceryItemTapped(int index) {
    _selectedIndex = index;
    _createNewItem = false;
    notifyListeners();
  }

  void addItem(GroceryItem item) {
    _groceryItems.add(item);
    _createNewItem = false;
    notifyListeners();
  }

  void updateItem(GroceryItem item, int index) {
    _groceryItems[index] = item;
    _selectedIndex = -1;
    _createNewItem = false;
    notifyListeners();
  }

  void completeItem(int index, bool change) {
```

Fig. 65

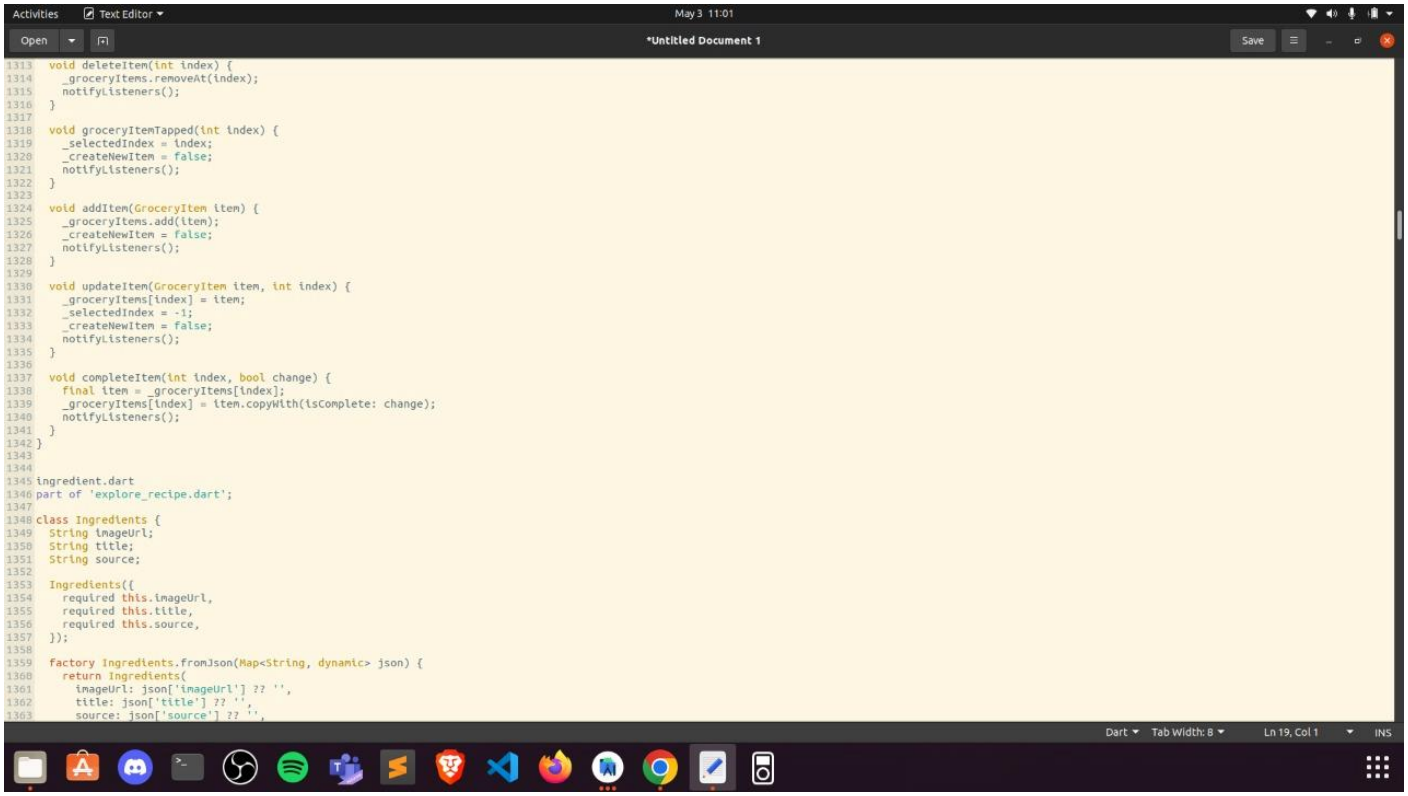

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
void deleteItem(int index) {
_groceryItems.removeAt(index);
notifyListeners();
}
void groceryItemTapped(int index) {
_selectedIndex = index;
_createNewItem = false;
notifyListeners();
}
void addItem(GroceryItem item) {
_groceryItems.add(item);
_createNewItem = false;
notifyListeners();
}
void updateItem(GroceryItem item, int index) {
_groceryItems[index] = item;
_selectedIndex = -1;
_createNewItem = false;
notifyListeners();
}
void completeItem(int index, bool change) {
final item = _groceryItems[index];
_groceryItems[index] = item.copyWith(isComplete: change);
notifyListeners();
}
}
ingredient.dart
part of 'explore_recipe.dart';
class Ingredients {
String imageUrl;
String title;
String source;
Ingredients({
required this.imageUrl,
required this.title,
required this.source,
});
factory Ingredients.fromJson(Map<String, dynamic> json) {
return Ingredients(
imageUrl: json['imageUrl'] ?? '',
title: json['title'] ?? '',
source: json['source'] ?? '',
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 66

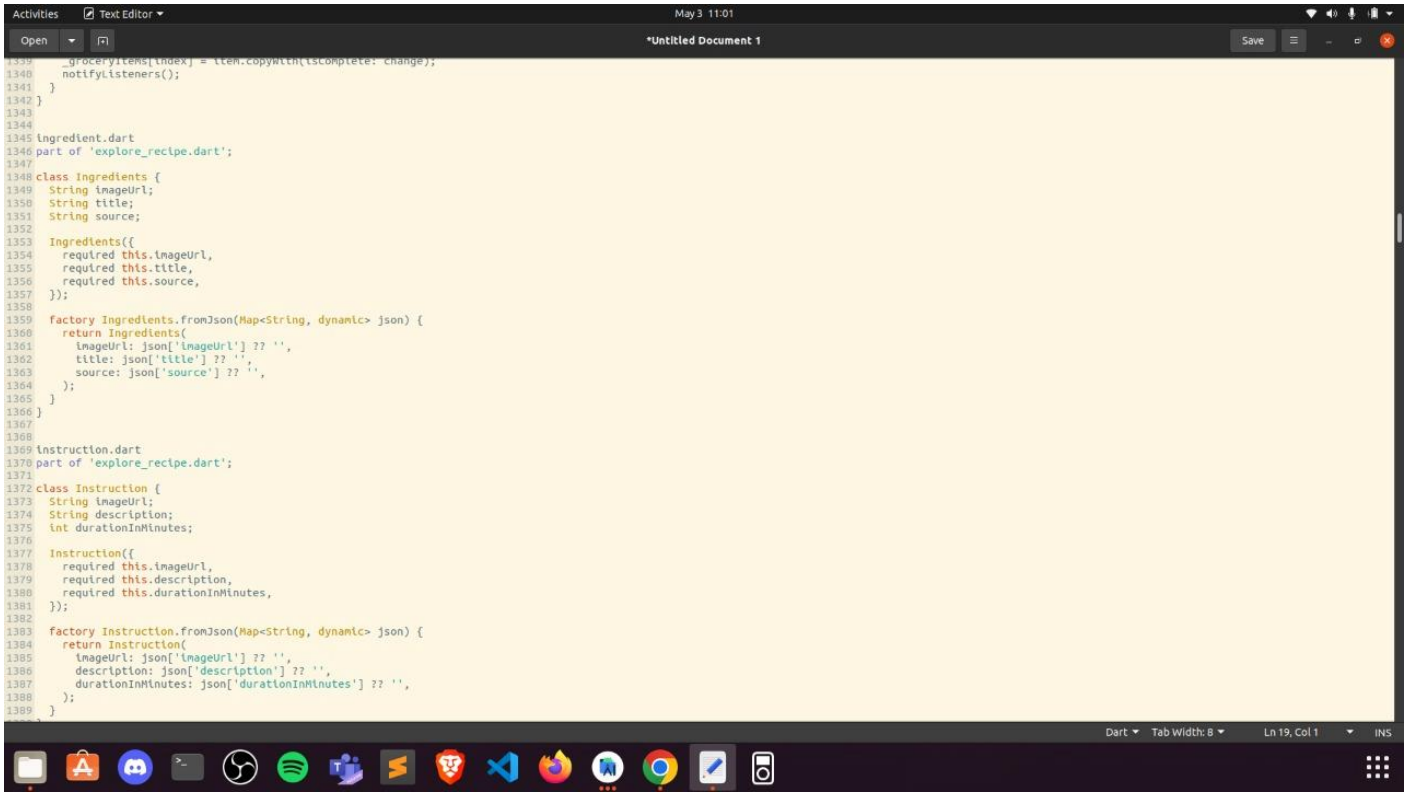

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
_groceryItems[index] = item.copyWith(isComplete: change);
notifyListeners();
}
}
ingredient.dart
part of 'explore_recipe.dart';
class Ingredients {
String imageUrl;
String title;
String source;
Ingredients({
required this.imageUrl,
required this.title,
required this.source,
});
factory Ingredients.fromJson(Map<String, dynamic> json) {
return Ingredients(
imageUrl: json['imageUrl'] ?? '',
title: json['title'] ?? '',
source: json['source'] ?? '',
);
}
}
instruction.dart
part of 'explore_recipe.dart';
class Instruction {
String imageUrl;
String description;
int durationInMinutes;
Instruction({
required this.imageUrl,
required this.description,
required this.durationInMinutes,
});
factory Instruction.fromJson(Map<String, dynamic> json) {
return Instruction(
imageUrl: json['imageUrl'] ?? '',
description: json['description'] ?? '',
durationInMinutes: json['durationInMinutes'] ?? '',
);
}
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 67

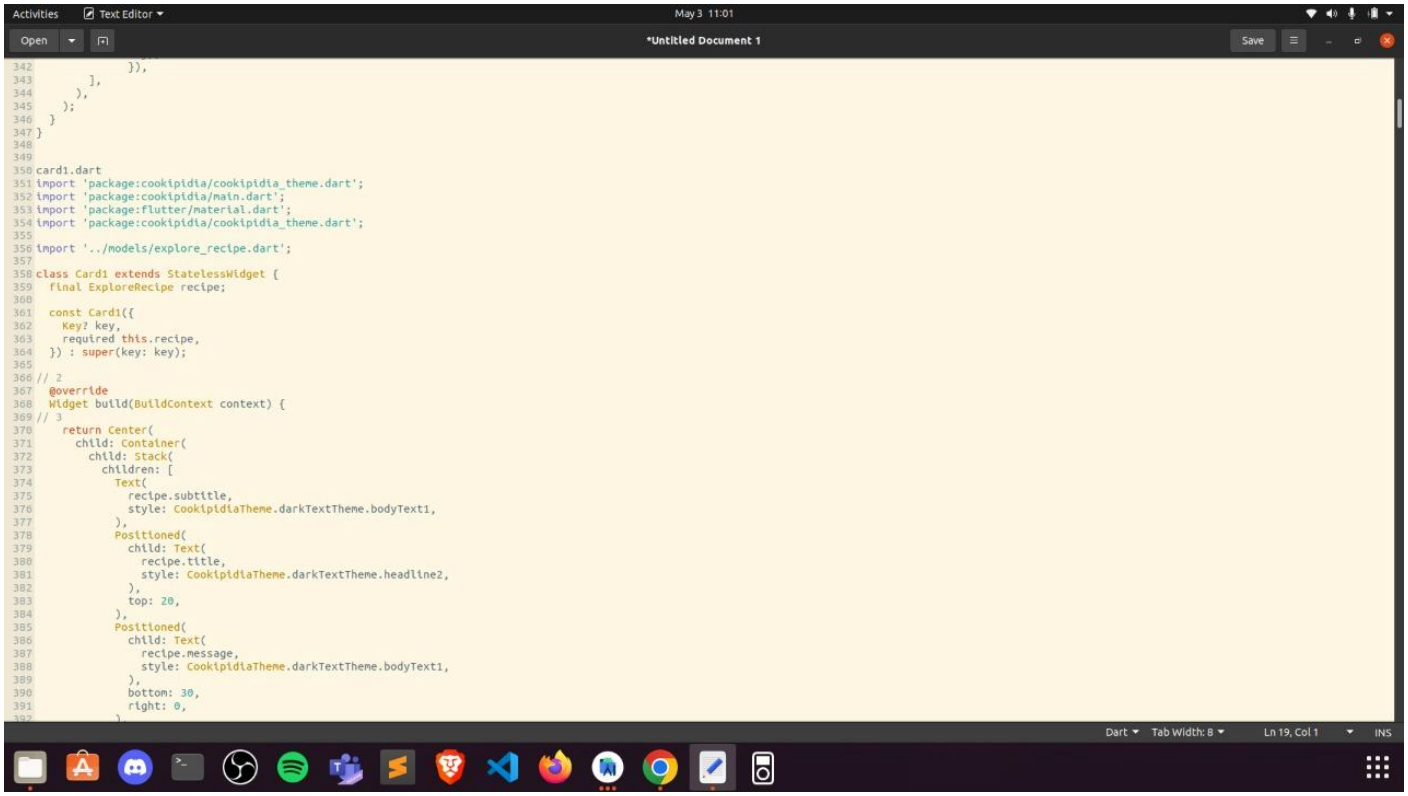

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
card1.dart
import 'package:cookipidia/cookipidia_theme.dart';
import 'package:cookipidia/main.dart';
import 'package:flutter/material.dart';
import 'package:cookipidia/cookipidia_theme.dart';
import '../models/explore_recipe.dart';
class Card1 extends StatelessWidget {
final ExploreRecipe recipe;
const Card1({
Key? key,
required this.recipe,
}) : super(key: key);
// 2
@override
Widget build(BuildContext context) {
// 3
return Center(
child: Container(
child: Stack(
children: [
Text(
recipe.subtitle,
style: CookipidiaTheme.darkTextTheme.bodyText1,
),
Positioned(
child: Text(
recipe.title,
style: CookipidiaTheme.darkTextTheme.headline2,
),
top: 20,
),
Positioned(
child: Text(
recipe.message,
style: CookipidiaTheme.darkTextTheme.bodyText1,
),
bottom: 30,
right: 0,
),
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 68

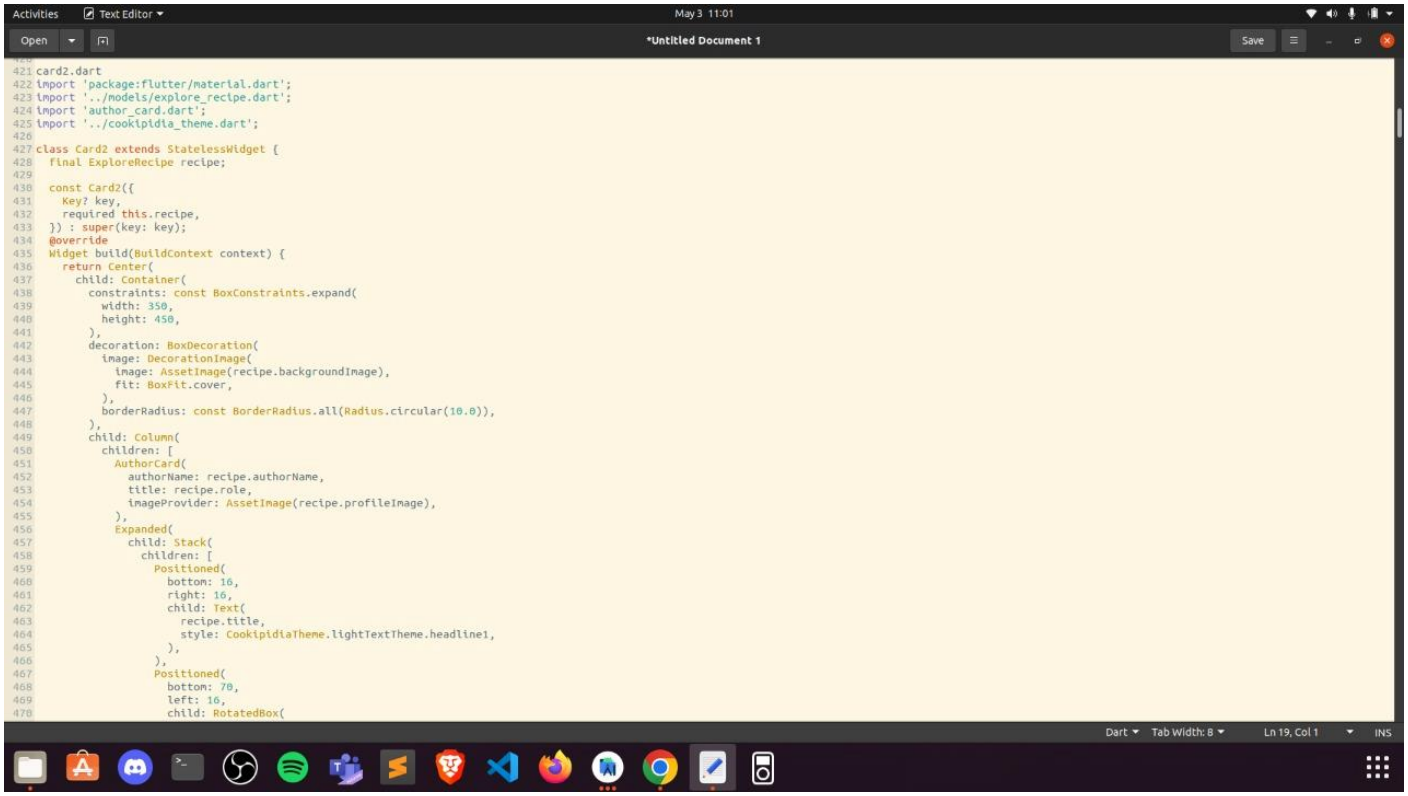

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
card2.dart
import 'package:flutter/material.dart';
import '../models/explore_recipe.dart';
import 'author_card.dart';
import '../cookipidia_theme.dart';
class Card2 extends StatelessWidget {
final ExploreRecipe recipe;
const Card2({
Key? key,
required this.recipe,
}) : super(key: key);
@override
Widget build(BuildContext context) {
return Center(
child: Container(
constraints: const BoxConstraints.expand(
width: 350,
height: 450,
),
decoration: BoxDecoration(
image: DecorationImage(
image: AssetImage(recipe.backgroundImage),
fit: BoxFit.cover,
),
borderRadius: const BorderRadius.all(Radius.circular(10.0)),
),
child: Column(
children: [
AuthorCard(
authorName: recipe.authorName,
title: recipe.role,
imageProvider: AssetImage(recipe.profileImage),
),
Expanded(
child: Stack(
children: [
Positioned(
bottom: 16,
right: 16,
child: Text(
recipe.title,
style: CookipidiaTheme.lightTextTheme.headline1,
),
),
Positioned(
bottom: 70,
left: 16,
child: RotatedBox(
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 69

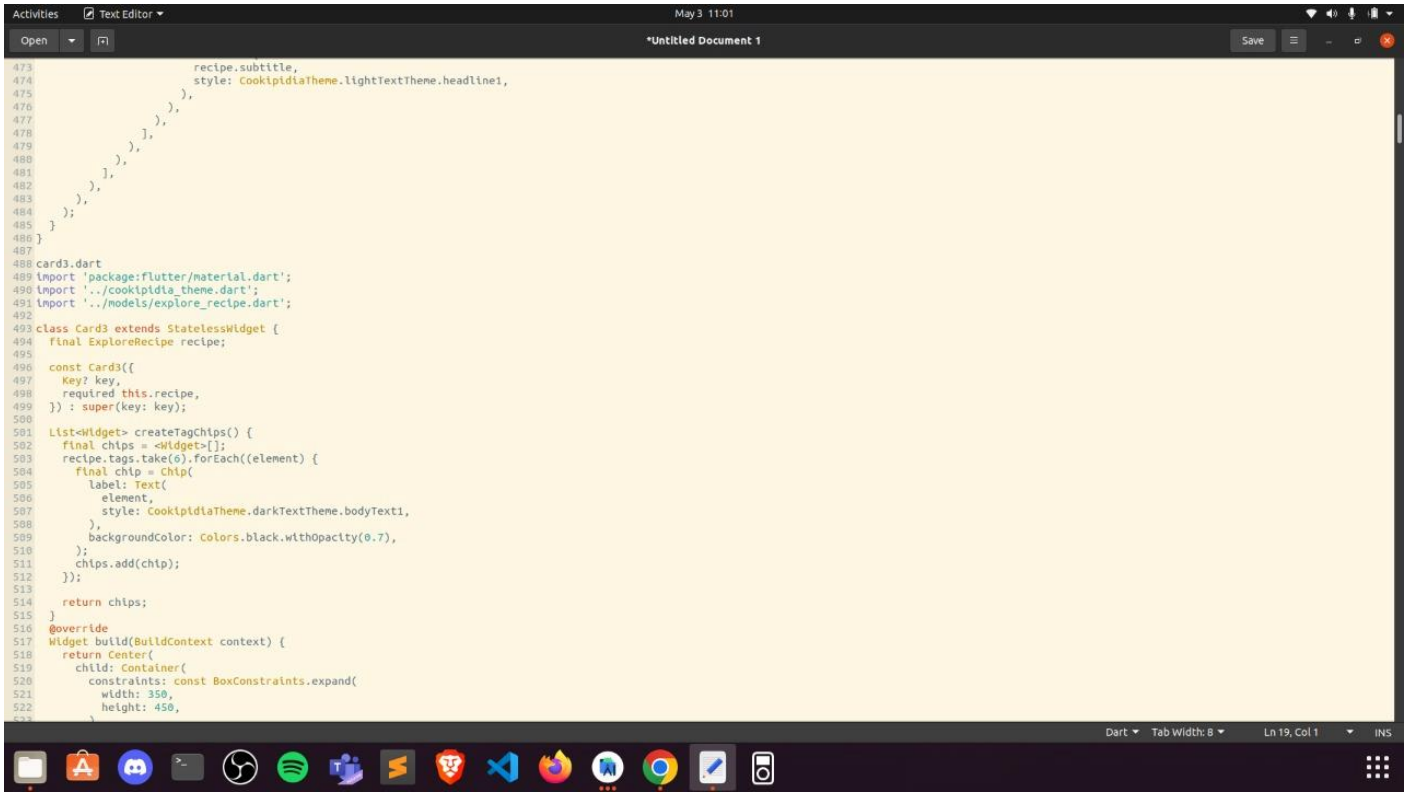

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
recipe.subtitle,
style: CookipidiaTheme.lightTextTheme.headline1,
),
),
),
),
),
),
),
),
),
),
);
}
}
card3.dart
import 'package:flutter/material.dart';
import '../cookipidia_theme.dart';
import '../models/explore_recipe.dart';
class Card3 extends StatelessWidget {
final ExploreRecipe recipe;
const Card3({
Key? key,
required this.recipe,
}) : super(key: key);
List<Widget> createTagChips() {
final chips = <Widget>[];
recipe.tags.take(6).forEach((element) {
final chip = Chip(
label: Text(
element,
style: CookipidiaTheme.darkTextTheme.bodyText1,
),
backgroundColor: Colors.black.withOpacity(0.7),
);
chips.add(chip);
});
return chips;
}
@override
Widget build(BuildContext context) {
return Center(
child: Container(
constraints: const BoxConstraints.expand(
width: 350,
height: 450,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 70

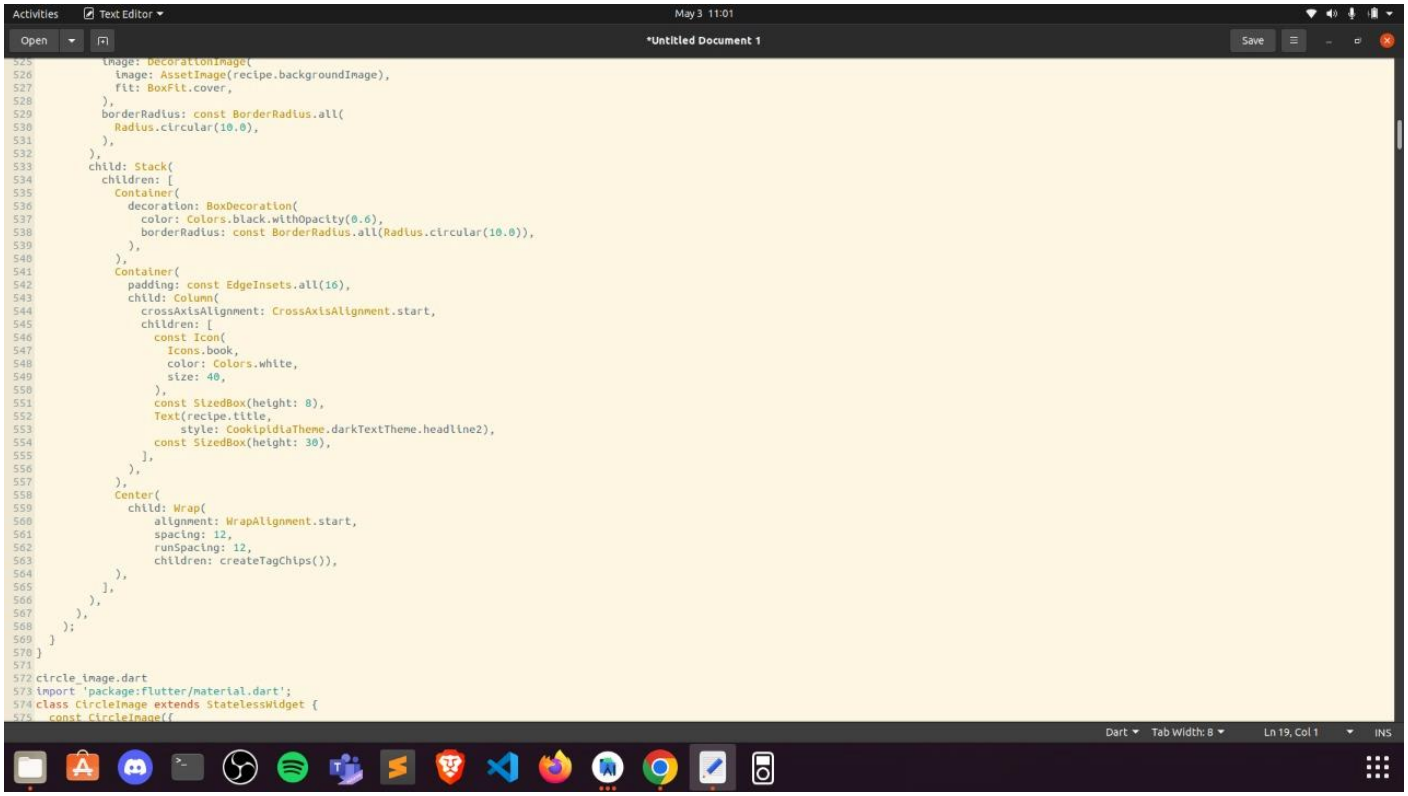

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
image: DecorationImage(
image: AssetImage(recipe.backgroundImage),
fit: BoxFit.cover,
),
borderRadius: const BorderRadius.all(
Radius.circular(10.0),
),
),
child: Stack(
children: [
Container(
decoration: BoxDecoration(
color: Colors.black.withOpacity(0.6),
borderRadius: const BorderRadius.all(Radius.circular(10.0)),
),
),
Container(
padding: const EdgeInsets.all(16),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
const Icon(
Icons.book,
color: Colors.white,
size: 40,
),
const SizedBox(height: 8),
Text(recipe.title,
style: CookipidiaTheme.darkTextTheme.headline2),
const SizedBox(height: 30),
],
),
),
Center(
child: Wrap(
alignment: WrapAlignment.start,
spacing: 12,
runSpacing: 12,
children: createTagChips()),
),
],
),
),
);
}
}
circle_image.dart
import 'package:flutter/material.dart';
class CircleImage extends StatelessWidget {
const CircleImage({
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 71

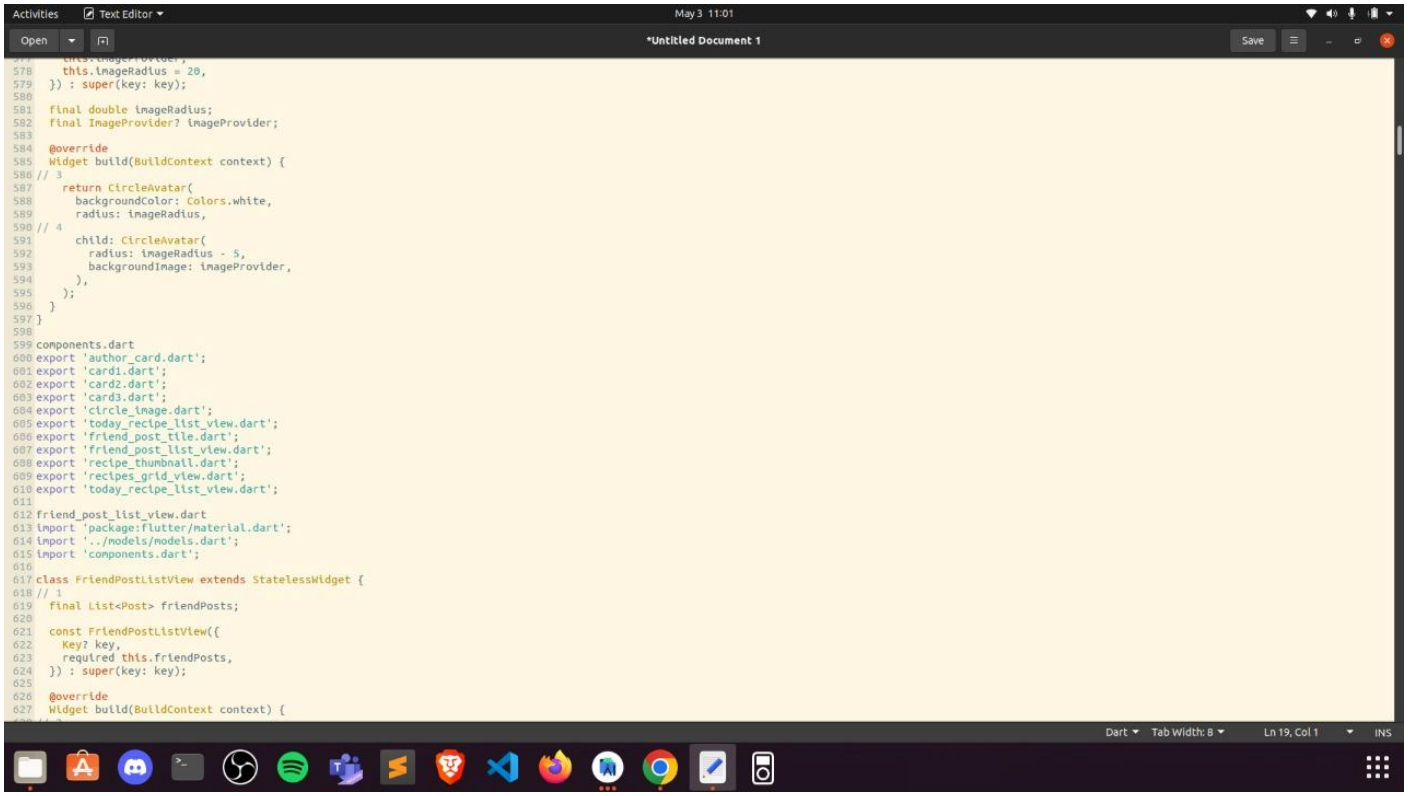

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
this.imageRadius = 20,
}) : super(key: key);
final double imageRadius;
final ImageProvider? imageProvider;
@override
Widget build(BuildContext context) {
// 3
return CircleAvatar(
backgroundColor: Colors.white,
radius: imageRadius,
// 4
child: CircleAvatar(
radius: imageRadius - 5,
backgroundImage: imageProvider,
),
);
}
}
components.dart
export 'author_card.dart';
export 'card1.dart';
export 'card2.dart';
export 'card3.dart';
export 'circle_image.dart';
export 'today_recipe_list_view.dart';
export 'friend_post_tile.dart';
export 'friend_post_list_view.dart';
export 'recipe_thumbnail.dart';
export 'recipes_grid_view.dart';
export 'today_recipe_list_view.dart';
friend_post_list_view.dart
import 'package:flutter/material.dart';
import '../models/models.dart';
import 'components.dart';
class FriendPostListView extends StatelessWidget {
// 1
final List<Post> friendPosts;
const FriendPostListView({
Key? key,
required this.friendPosts,
}) : super(key: key);
@override
Widget build(BuildContext context) {
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 72

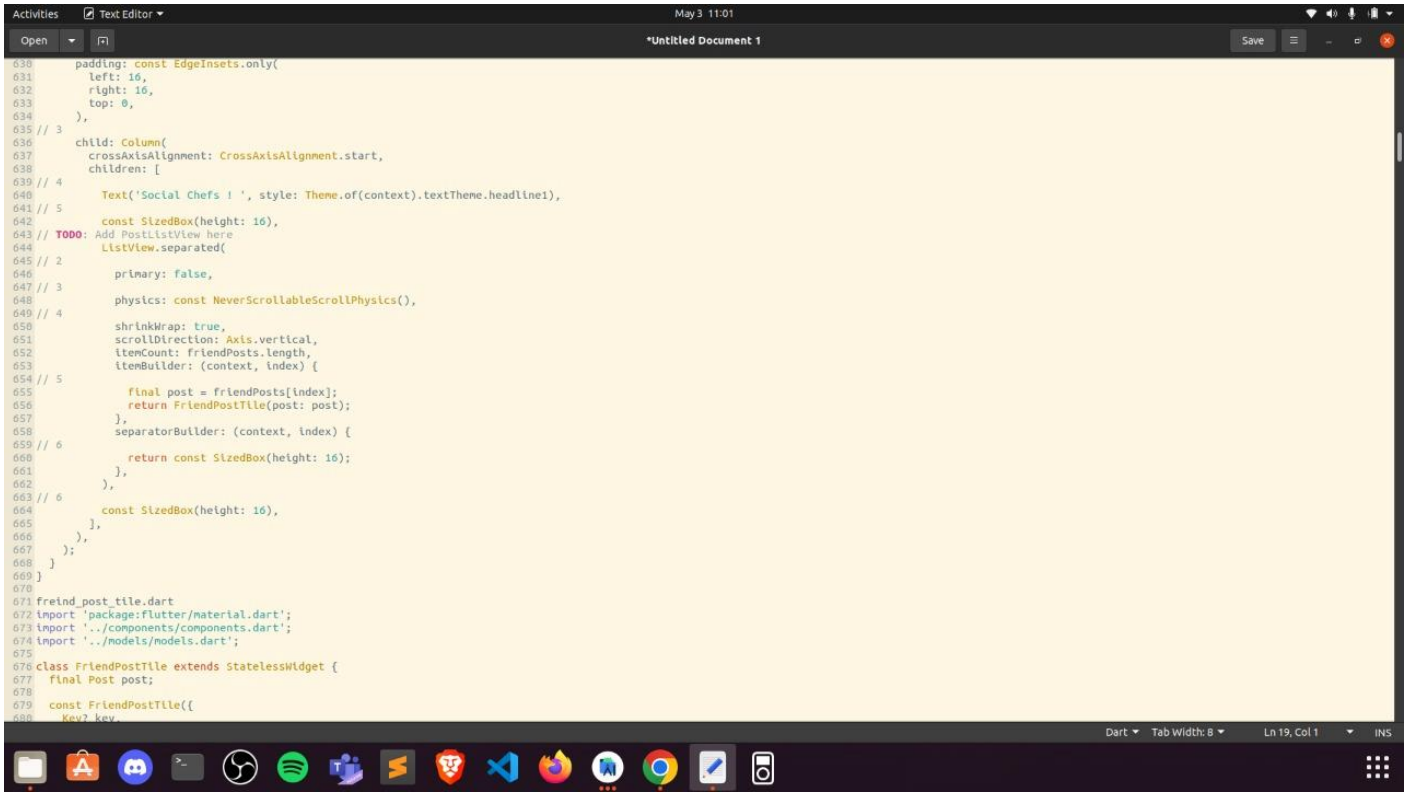

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
padding: const EdgeInsets.only(
left: 16,
right: 16,
top: 0,
),
// 3
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
// 4
Text('Social Chefs ! ', style: Theme.of(context).textTheme.headline1),
// 5
const SizedBox(height: 16),
// TODO: Add PostListView here
ListView.separated(
// 2
primary: false,
// 3
physics: const NeverScrollableScrollPhysics(),
// 4
shrinkWrap: true,
scrollDirection: Axis.vertical,
itemCount: friendPosts.length,
itemBuilder: (context, index) {
// 5
final post = friendPosts[index];
return FriendPostTile(post: post);
},
separatorBuilder: (context, index) {
// 6
return const SizedBox(height: 16);
},
),
// 6
const SizedBox(height: 16),
],
),
);
}
}
freind_post_tile.dart
import 'package:flutter/material.dart';
import '../components/components.dart';
import '../models/models.dart';
class FriendPostTile extends StatelessWidget {
final Post post;
const FriendPostTile({
Key? key,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 73

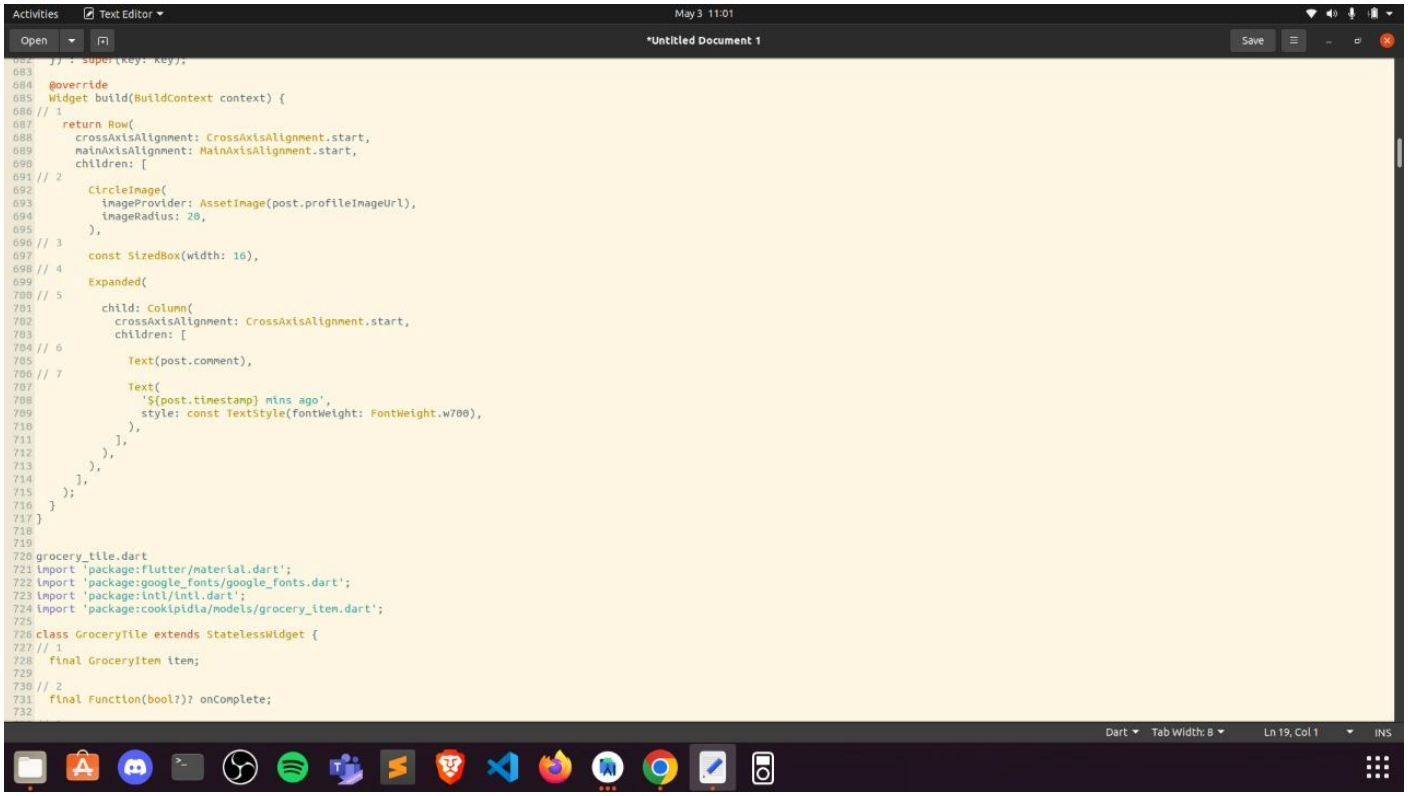

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
@override
Widget build(BuildContext context) {
return Row(
crossAxisAlignment: CrossAxisAlignment.start,
mainAxisAlignment: MainAxisAlignment.start,
children: [
CircleImage(
imageProvider: AssetImage(post.profileImageUrl),
imageRadius: 20,
),
const SizedBox(width: 16),
Expanded(
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(post.comment),
Text(
'${post.timestamp} mins ago',
style: const TextStyle(fontWeight: FontWeight.w700),
),
],
),
),
],
);
}
}
grocery_tile.dart
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:intl/intl.dart';
import 'package:cookipidia/models/grocery_item.dart';
class GroceryTile extends StatelessWidget {
final GroceryItem item;
final Function(bool?)? onComplete;
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 74

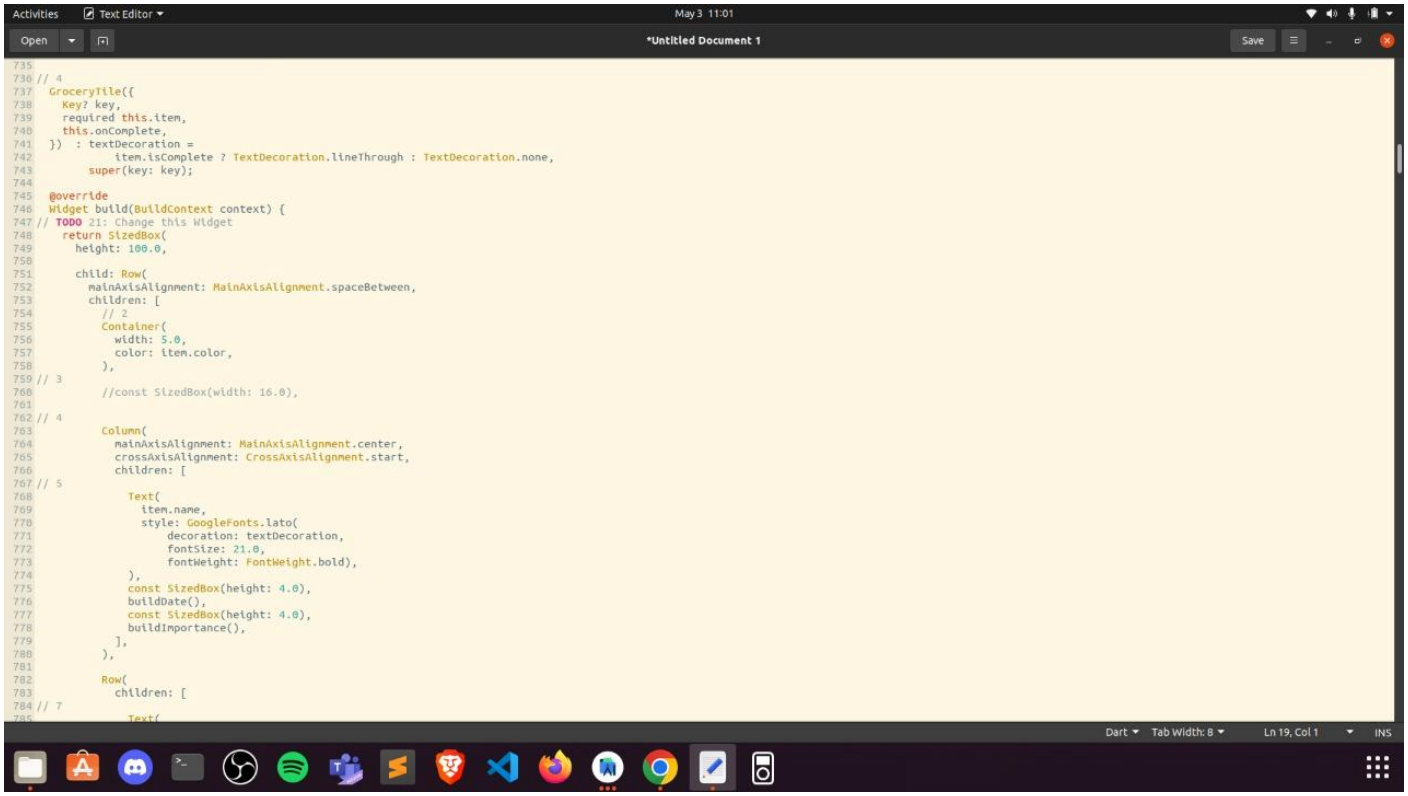

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
GroceryTile({
Key? key,
required this.item,
this.onComplete,
}) : textDecoration =
item.isComplete ? TextDecoration.lineThrough : TextDecoration.none,
super(key: key);
@override
Widget build(BuildContext context) {
// TODO 21: Change this Widget
return SizedBox(
height: 100.0,
child: Row(
mainAxisAlignment: MainAxisAlignment.spaceBetween,
children: [
// 2
Container(
width: 5.0,
color: item.color,
),
//const SizedBox(width: 16.0),
Column(
mainAxisAlignment: MainAxisAlignment.center,
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
item.name,
style: GoogleFonts.lato(
decoration: textDecoration,
fontSize: 21.0,
fontWeight: FontWeight.bold),
),
const SizedBox(height: 4.0),
buildDate(),
const SizedBox(height: 4.0),
buildImportance(),
],
),
Row(
children: [
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 75

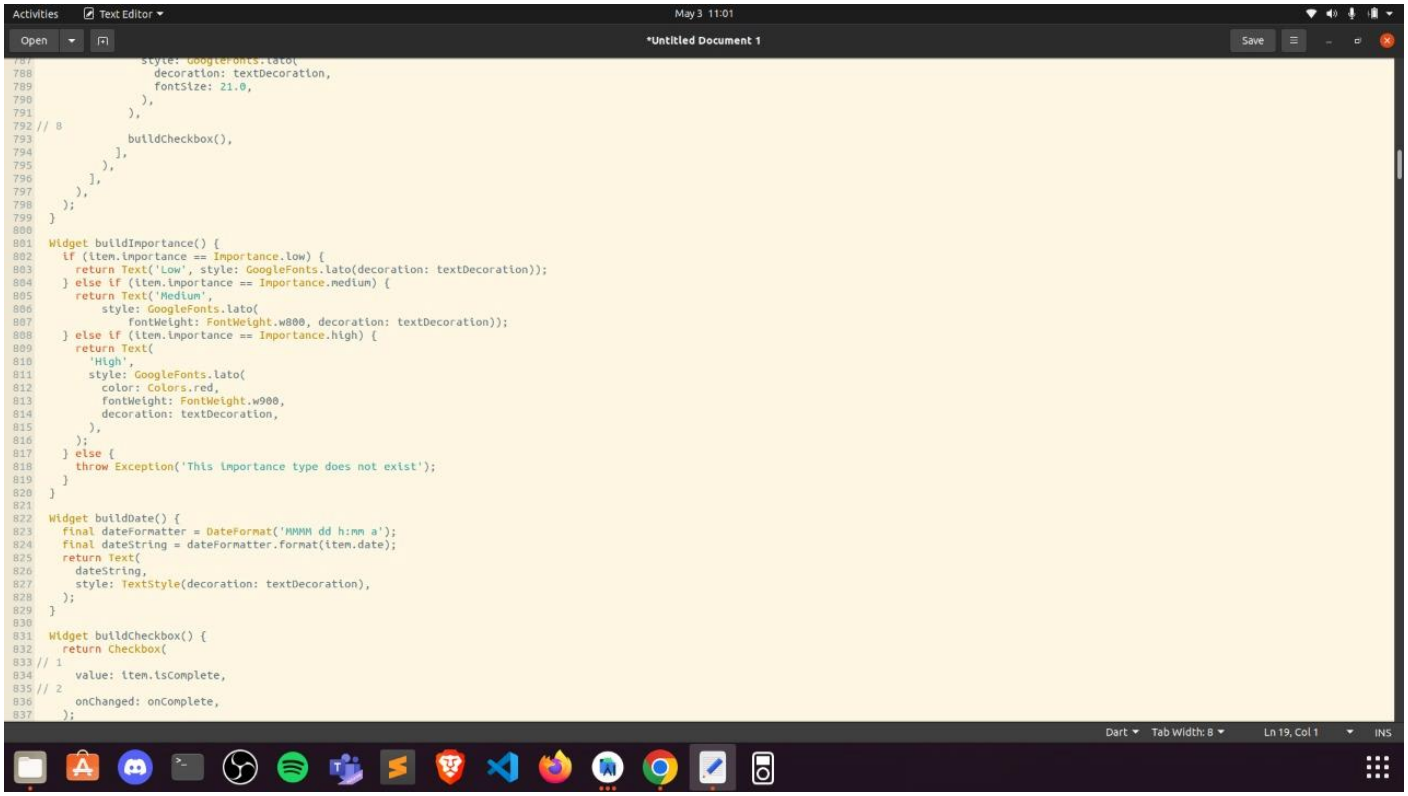

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 76

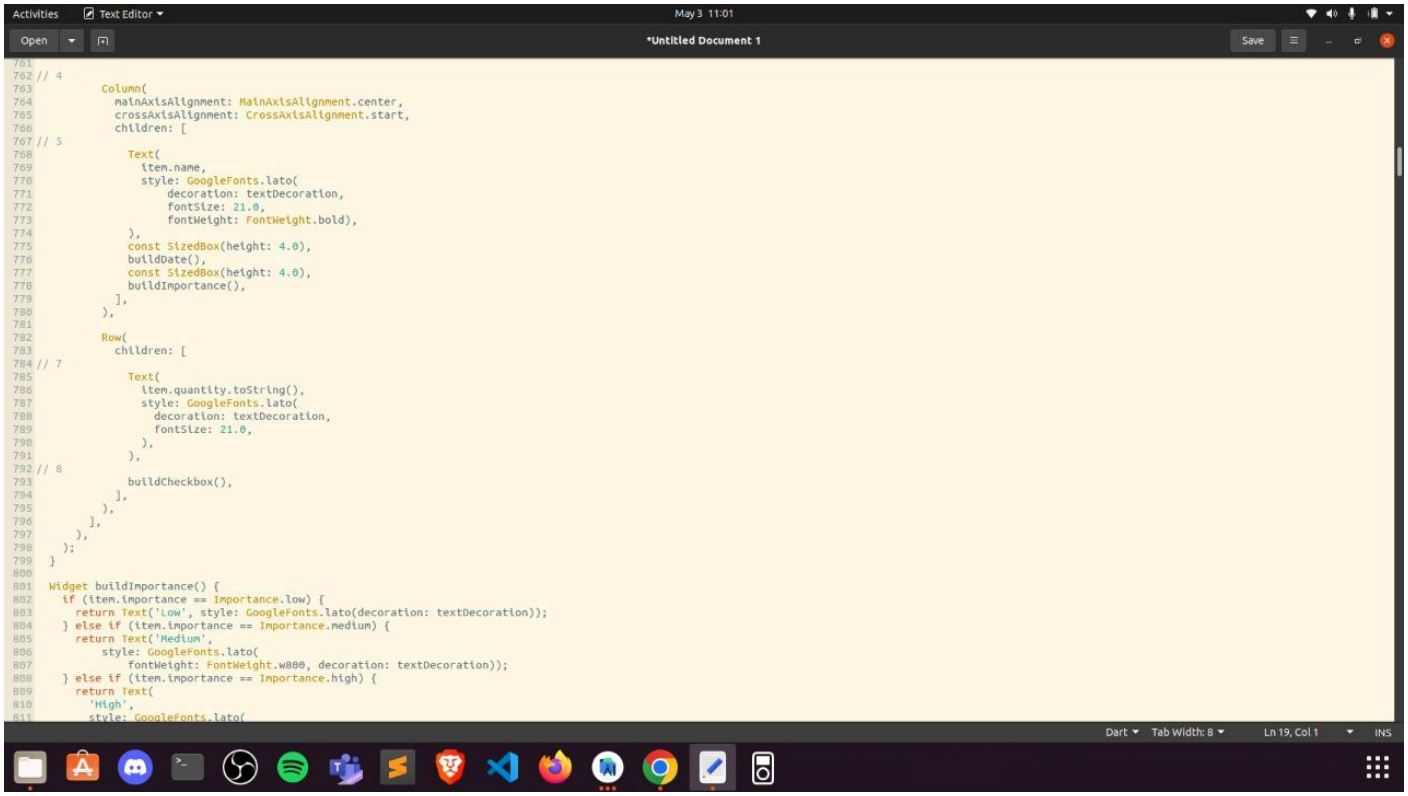

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 77

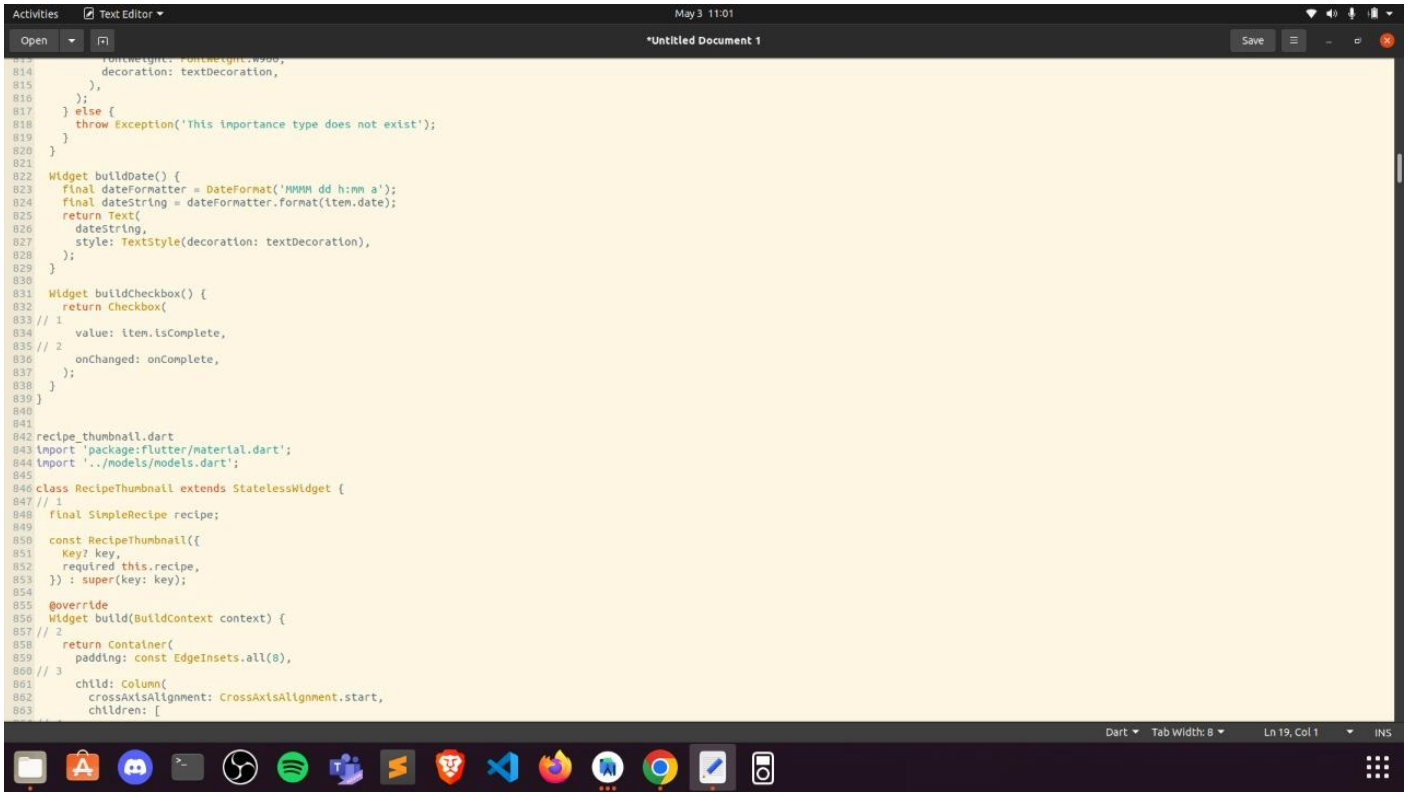

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
decoration: textDecoration,
),
);
} else {
throw Exception('This importance type does not exist');
}
}
Widget buildDate() {
final dateFormatter = DateFormat('MMMM dd h:mm a');
final dateString = dateFormatter.format(item.date);
return Text(
dateString,
style: TextStyle(decoration: textDecoration),
);
}
Widget buildCheckbox() {
return Checkbox(
// 1
value: item.isComplete,
// 2
onChanged: onComplete,
);
}
}
recipe_thumbnail.dart
import 'package:flutter/material.dart';
import '../models/models.dart';
class RecipeThumbnail extends StatelessWidget {
// 1
final SimpleRecipe recipe;
const RecipeThumbnail({
Key? key,
required this.recipe,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 2
return Container(
padding: const EdgeInsets.all(8),
// 3
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 78

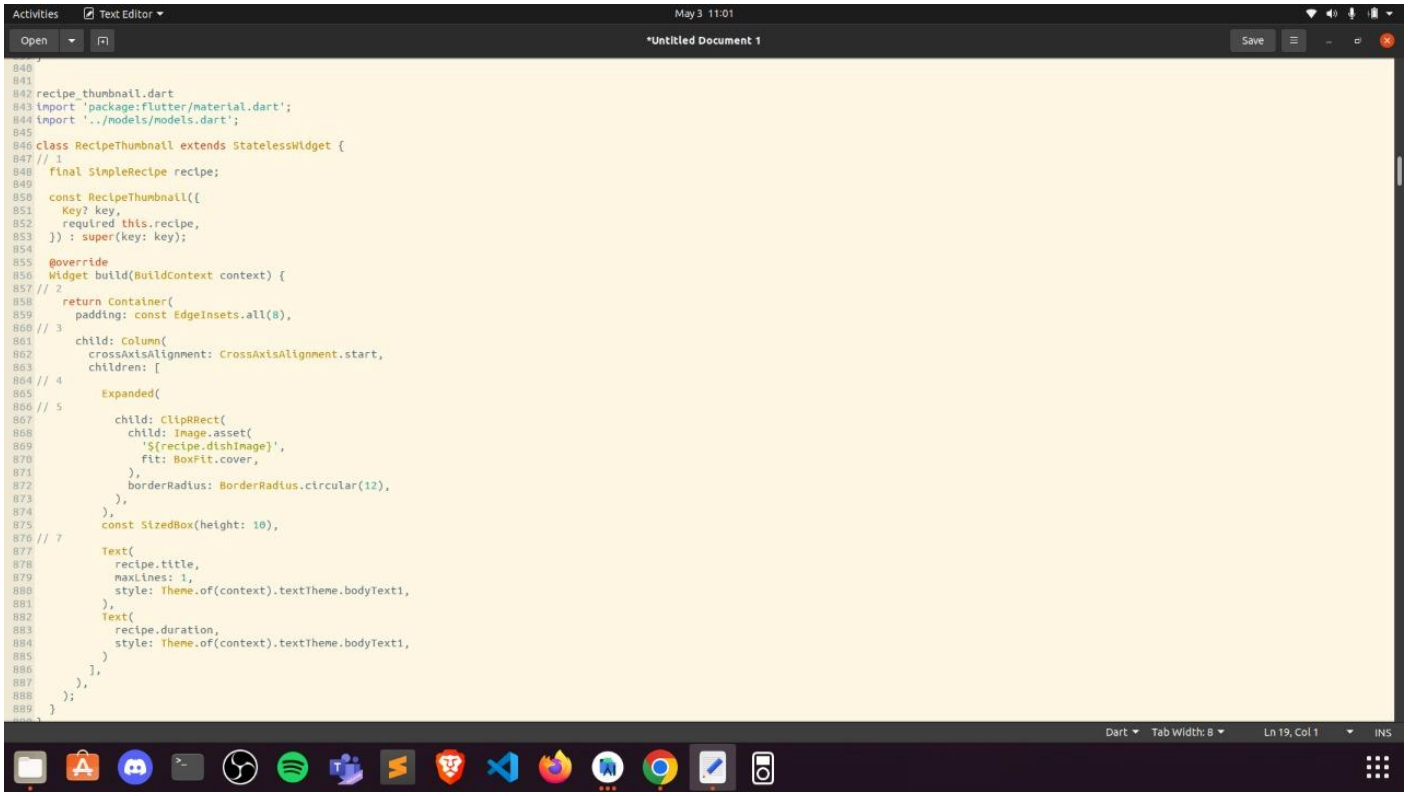
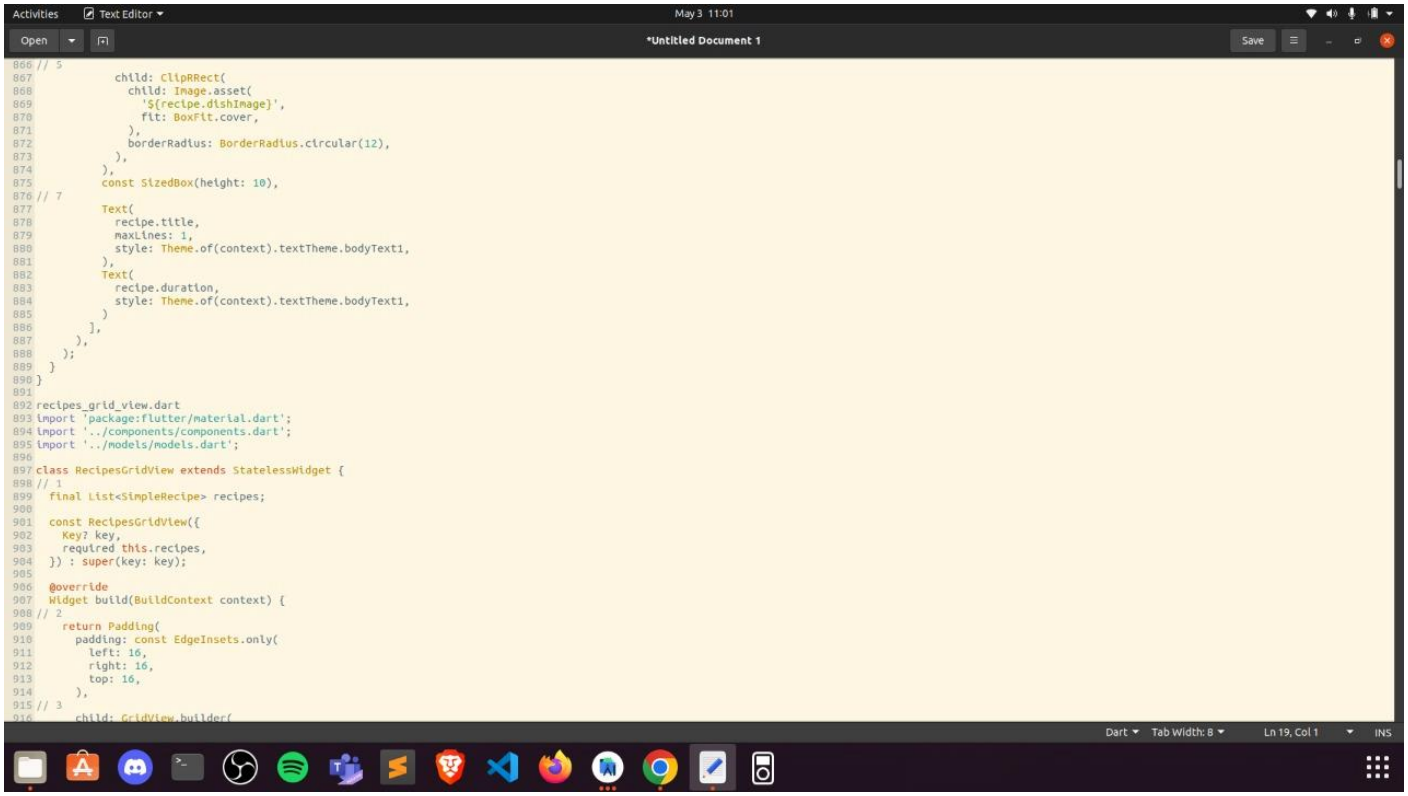
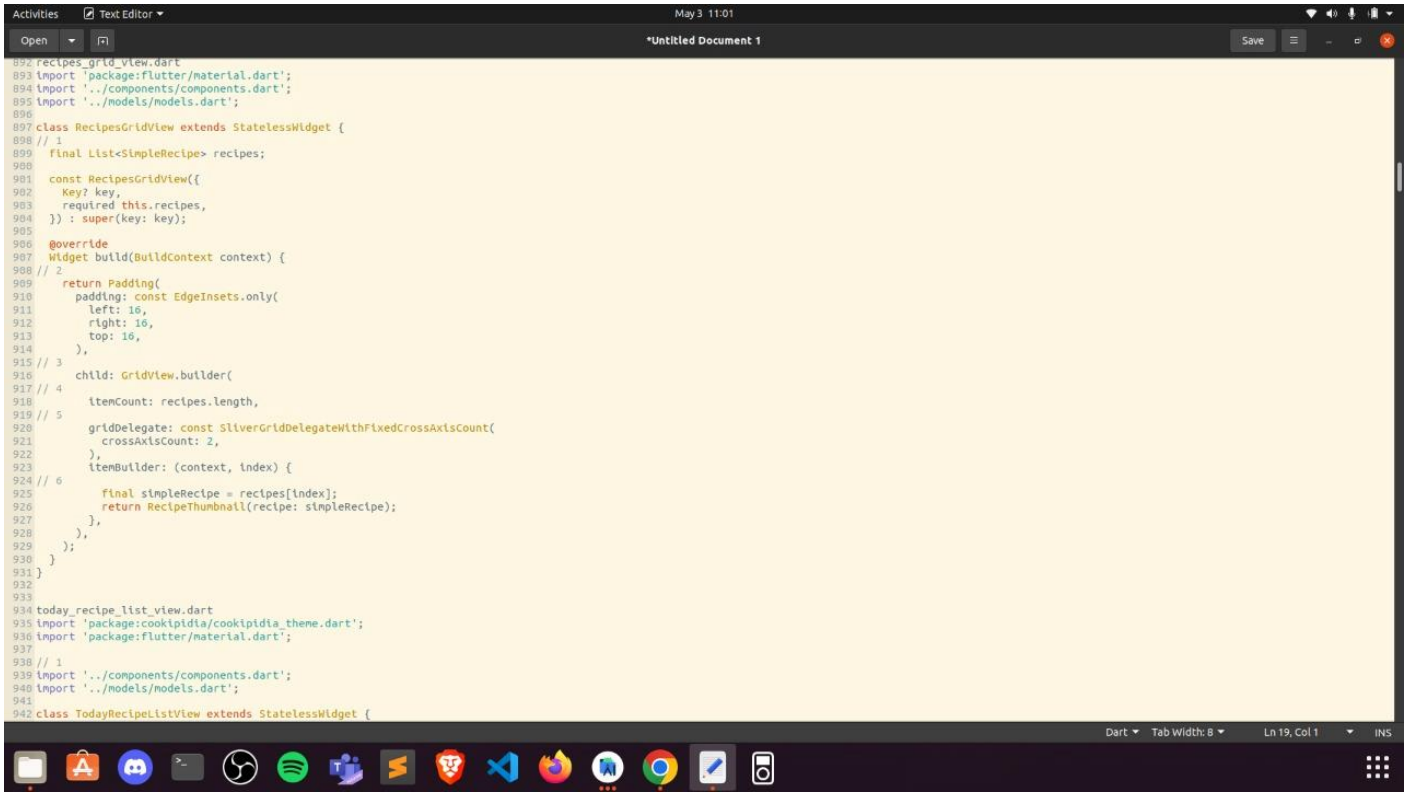

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
recipe_thumbnail.dart
import 'package:flutter/material.dart';
import '../models/models.dart';
class RecipeThumbnail extends StatelessWidget {
// 1
final SimpleRecipe recipe;
const RecipeThumbnail({
Key? key,
required this.recipe,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 2
return Container(
padding: const EdgeInsets.all(8),
// 3
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
// 4
Expanded(
// 5
child: ClipRRect(
child: Image.asset(
'${recipe.dishImage}',
fit: BoxFit.cover,
),
borderRadius: BorderRadius.circular(12),
),
),
const SizedBox(height: 10),
// 7
Text(
recipe.title,
maxLines: 1,
style: Theme.of(context).textTheme.bodyText1,
),
Text(
recipe.duration,
style: Theme.of(context).textTheme.bodyText1,
)
],
),
);
}
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 79




```
// 5
            child: ClipRRect(
              child: Image.asset(
                '${recipe.dishImage}',
                fit: BoxFit.cover,
              ),
              borderRadius: BorderRadius.circular(12),
            ),
          ),
          const SizedBox(height: 10),
// 7
          Text(
            recipe.title,
            maxLines: 1,
            style: Theme.of(context).textTheme.bodyText1,
          ),
          Text(
            recipe.duration,
            style: Theme.of(context).textTheme.bodyText1,
          )
        ],
      ),
    );
  }
}

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

class RecipesGridView extends StatelessWidget {
// 1
  final List<SimpleRecipe> recipes;

  const RecipesGridView({
    Key? key,
    required this.recipes,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
// 2
    return Padding(
      padding: const EdgeInsets.only(
        left: 16,
        right: 16,
        top: 16,
      ),
// 3
      child: GridView.builder(
```

Fig. 80




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

class RecipesGridView extends StatelessWidget {
// 1
  final List<SimpleRecipe> recipes;

  const RecipesGridView({
    Key? key,
    required this.recipes,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
// 2
    return Padding(
      padding: const EdgeInsets.only(
        left: 16,
        right: 16,
        top: 16,
      ),
// 3
      child: GridView.builder(
// 4
        itemCount: recipes.length,
// 5
        gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
          crossAxisCount: 2,
        ),
        itemBuilder: (context, index) {
// 6
          final simpleRecipe = recipes[index];
          return RecipeThumbnail(recipe: simpleRecipe);
        },
      ),
    );
  }
}


today_recipe_list_view.dart
import 'package:cookipidia/cookipidia_theme.dart';
import 'package:flutter/material.dart';

// 1
import '../components/components.dart';
import '../models/models.dart';

class TodayRecipeListView extends StatelessWidget {
```

Fig. 81

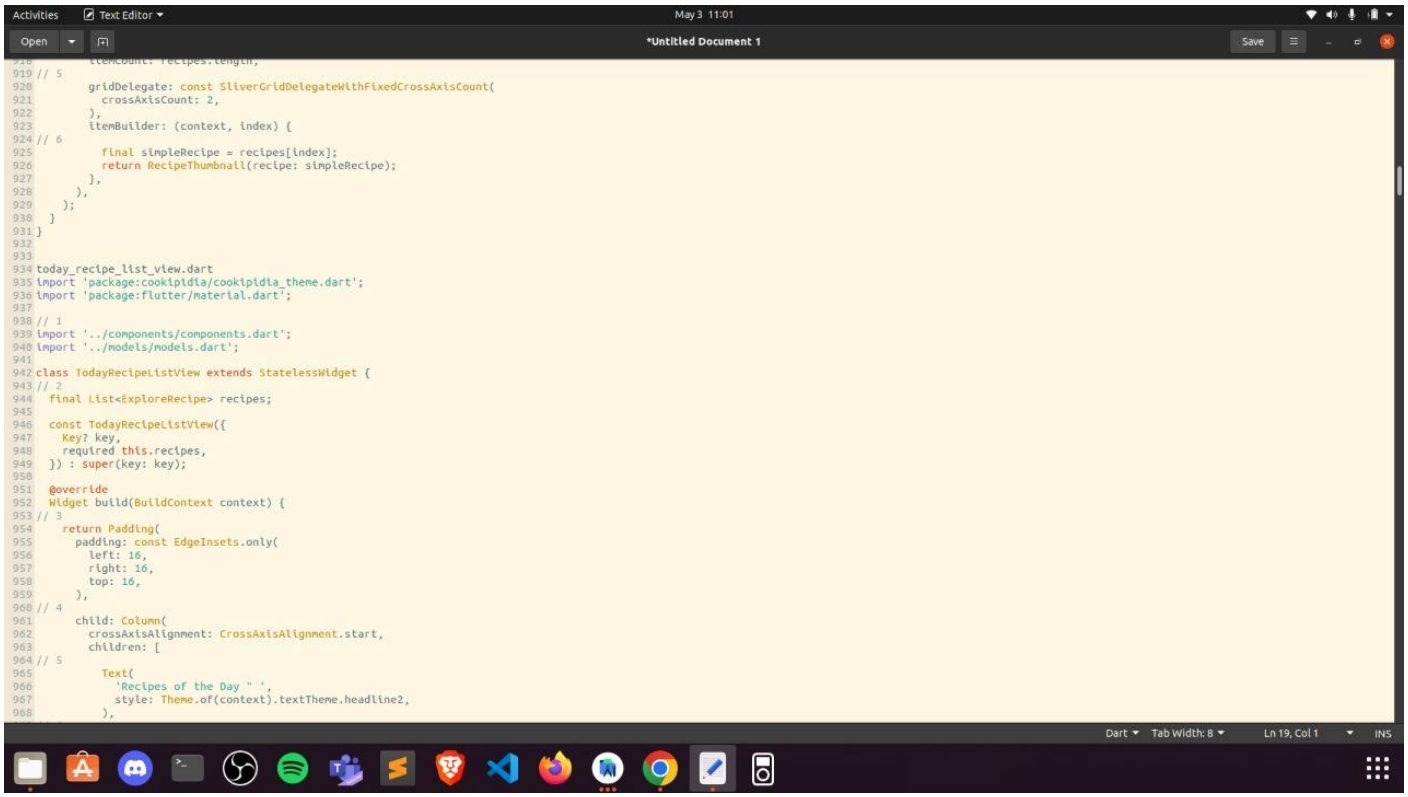

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
itemCount: recipes.length,
// 5
gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
crossAxisCount: 2,
),
itemBuilder: (context, index) {
// 6
final simpleRecipe = recipes[index];
return RecipeThumbnail(recipe: simpleRecipe);
},
),
);
}
}
today_recipe_list_view.dart
import 'package:cookipidia/cookipidia_theme.dart';
import 'package:flutter/material.dart';
// 1
import '../components/components.dart';
import '../models/models.dart';
class TodayRecipeListView extends StatelessWidget {
// 2
final List<ExploreRecipe> recipes;
const TodayRecipeListView({
Key? key,
required this.recipes,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 3
return Padding(
padding: const EdgeInsets.only(
left: 16,
right: 16,
top: 16,
),
// 4
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
// 5
Text(
'Recipes of the Day ',
style: Theme.of(context).textTheme.headline2,
),
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 82

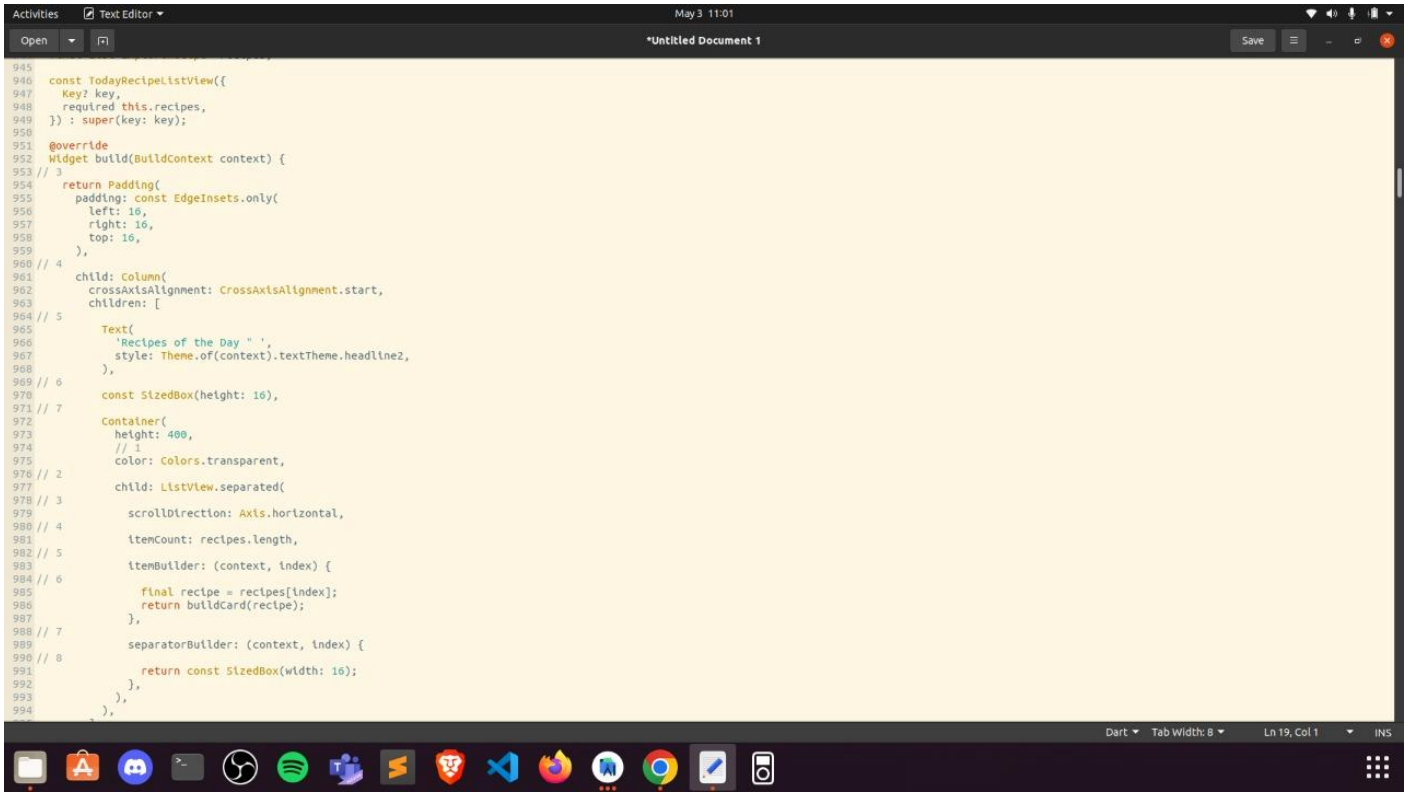
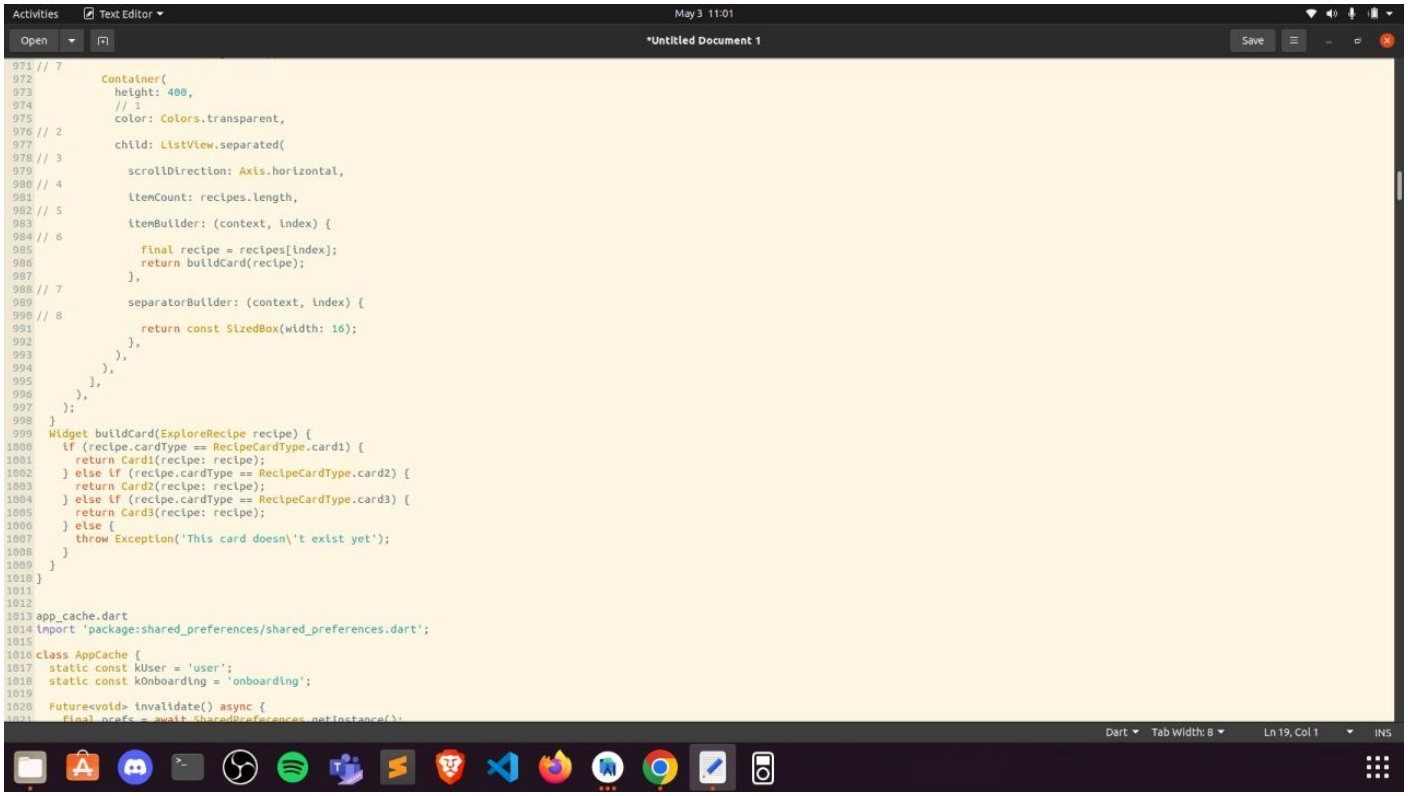
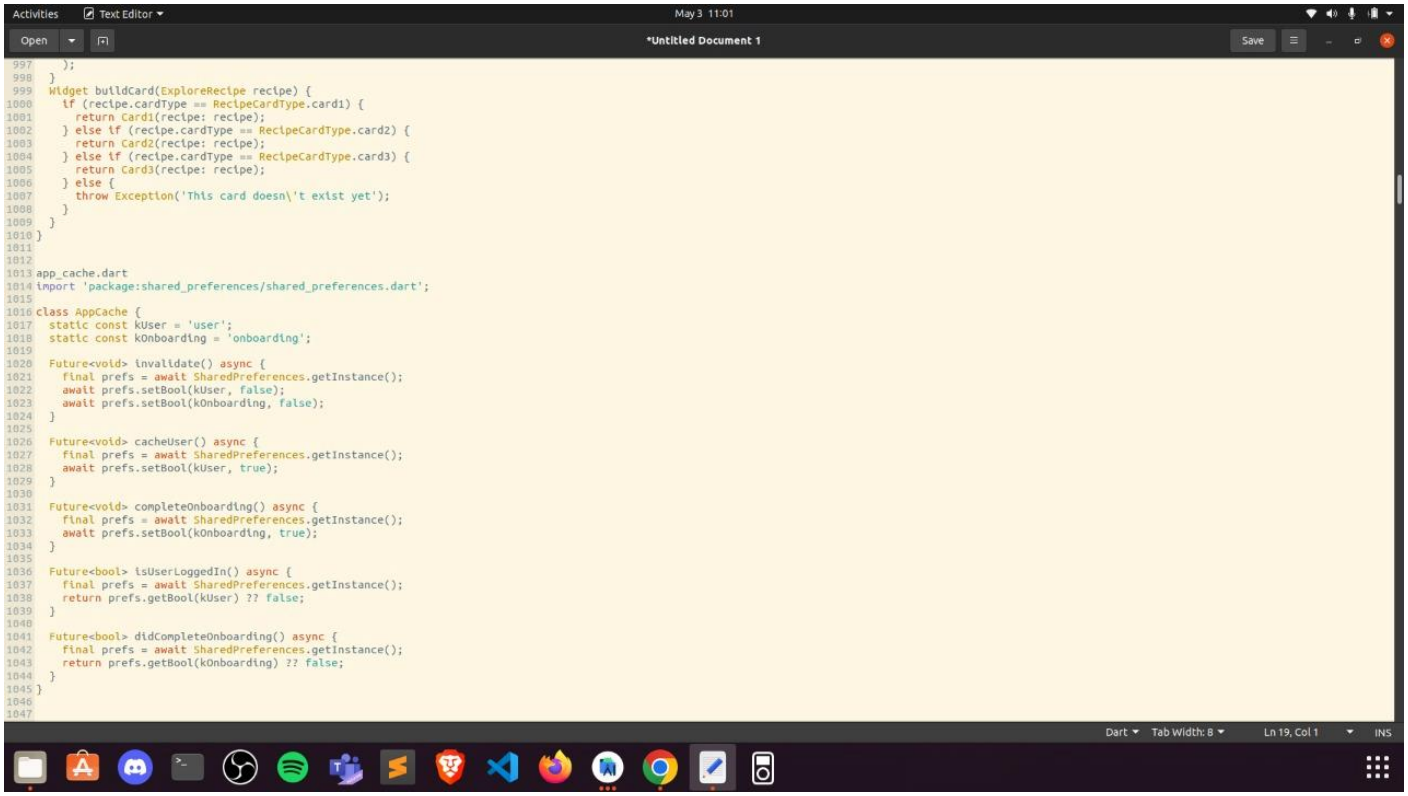

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
const TodayRecipeListView({
Key? key,
required this.recipes,
}) : super(key: key);
@override
Widget build(BuildContext context) {
// 3
return Padding(
padding: const EdgeInsets.only(
left: 16,
right: 16,
top: 16,
),
// 4
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
// 5
Text(
'Recipes of the Day ',
style: Theme.of(context).textTheme.headline2,
),
// 6
const SizedBox(height: 16),
// 7
Container(
height: 400,
// 1
color: Colors.transparent,
// 2
child: ListView.separated(
// 3
scrollDirection: Axis.horizontal,
// 4
itemCount: recipes.length,
// 5
itemBuilder: (context, index) {
// 6
final recipe = recipes[index];
return buildCard(recipe);
},
// 7
separatorBuilder: (context, index) {
// 8
return const SizedBox(width: 16);
},
),
),
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 83




```
// 7
          Container(
            height: 400,
            // 1
            color: Colors.transparent,
// 2
            child: ListView.separated(
// 3
              scrollDirection: Axis.horizontal,
// 4
              itemCount: recipes.length,
// 5
              itemBuilder: (context, index) {
// 6
                final recipe = recipes[index];
                return buildCard(recipe);
              },
// 7
              separatorBuilder: (context, index) {
// 8
                return const SizedBox(width: 16);
              },
            ),
          ),
        ],
      ),
    );
  }
  Widget buildCard(ExploreRecipe recipe) {
    if (recipe.cardType == RecipeCardType.card1) {
      return Card1(recipe: recipe);
    } else if (recipe.cardType == RecipeCardType.card2) {
      return Card2(recipe: recipe);
    } else if (recipe.cardType == RecipeCardType.card3) {
      return Card3(recipe: recipe);
    } else {
      throw Exception('This card doesn\'t exist yet');
    }
  }
}


app_cache.dart
import 'package:shared_preferences/shared_preferences.dart';

class AppCache {
  static const kUser = 'user';
  static const kOnboarding = 'onboarding';

  Future<void> invalidate() async {
    final prefs = await SharedPreferences.getInstance();
```

Fig. 84




```
    );
  }
  Widget buildCard(ExploreRecipe recipe) {
    if (recipe.cardType == RecipeCardType.card1) {
      return Card1(recipe: recipe);
    } else if (recipe.cardType == RecipeCardType.card2) {
      return Card2(recipe: recipe);
    } else if (recipe.cardType == RecipeCardType.card3) {
      return Card3(recipe: recipe);
    } else {
      throw Exception('This card doesn\'t exist yet');
    }
  }
}


app_cache.dart
import 'package:shared_preferences/shared_preferences.dart';

class AppCache {
  static const kUser = 'user';
  static const kOnboarding = 'onboarding';

  Future<void> invalidate() async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.setBool(kUser, false);
    await prefs.setBool(kOnboarding, false);
  }

  Future<void> cacheUser() async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.setBool(kUser, true);
  }

  Future<void> completeOnboarding() async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.setBool(kOnboarding, true);
  }

  Future<bool> isUserLoggedIn() async {
    final prefs = await SharedPreferences.getInstance();
    return prefs.getBool(kUser) ?? false;
  }

  Future<bool> didCompleteOnboarding() async {
    final prefs = await SharedPreferences.getInstance();
    return prefs.getBool(kOnboarding) ?? false;
  }
}
```

Fig. 85

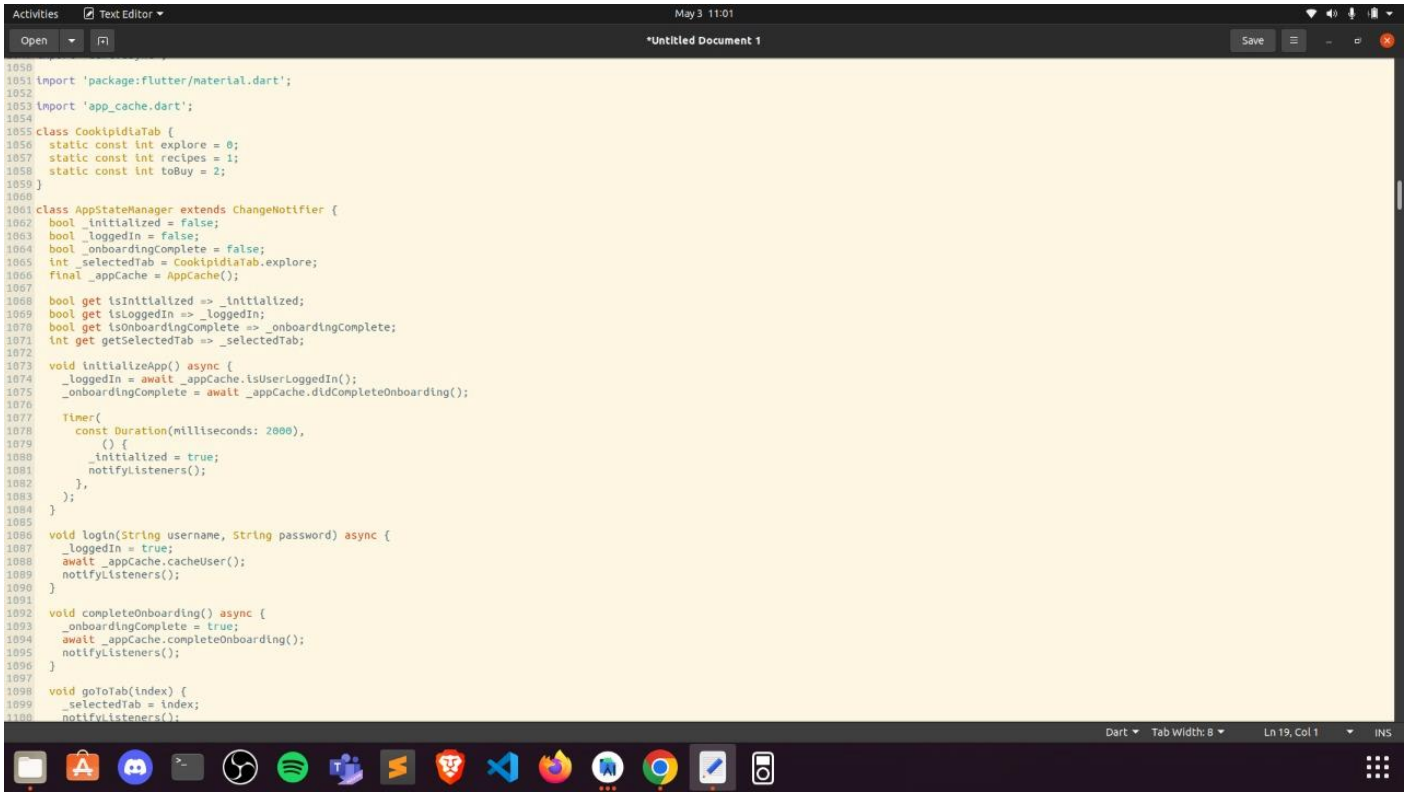

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
import 'package:flutter/material.dart';
import 'app_cache.dart';
class CookipidiaTab {
static const int explore = 0;
static const int recipes = 1;
static const int toBuy = 2;
}
class AppStateManager extends ChangeNotifier {
bool _initialized = false;
bool _loggedIn = false;
bool _onboardingComplete = false;
int _selectedTab = CookipidiaTab.explore;
final _appCache = AppCache();
bool get isInitialized => _initialized;
bool get isLoggedIn => _loggedIn;
bool get isOnboardingComplete => _onboardingComplete;
int get getSelectedTab => _selectedTab;
void initializeApp() async {
_loggedIn = await _appCache.isUserLoggedIn();
_onboardingComplete = await _appCache.didCompleteOnboarding();
Timer(
const Duration(milliseconds: 2000),
() {
_initialized = true;
notifyListeners();
},
);
}
void login(String username, String password) async {
_loggedIn = true;
await _appCache.cacheUser();
notifyListeners();
}
void completeOnboarding() async {
_onboardingComplete = true;
await _appCache.completeOnboarding();
notifyListeners();
}
void goToTab(index) {
_selectedTab = index;
notifyListeners();
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 86

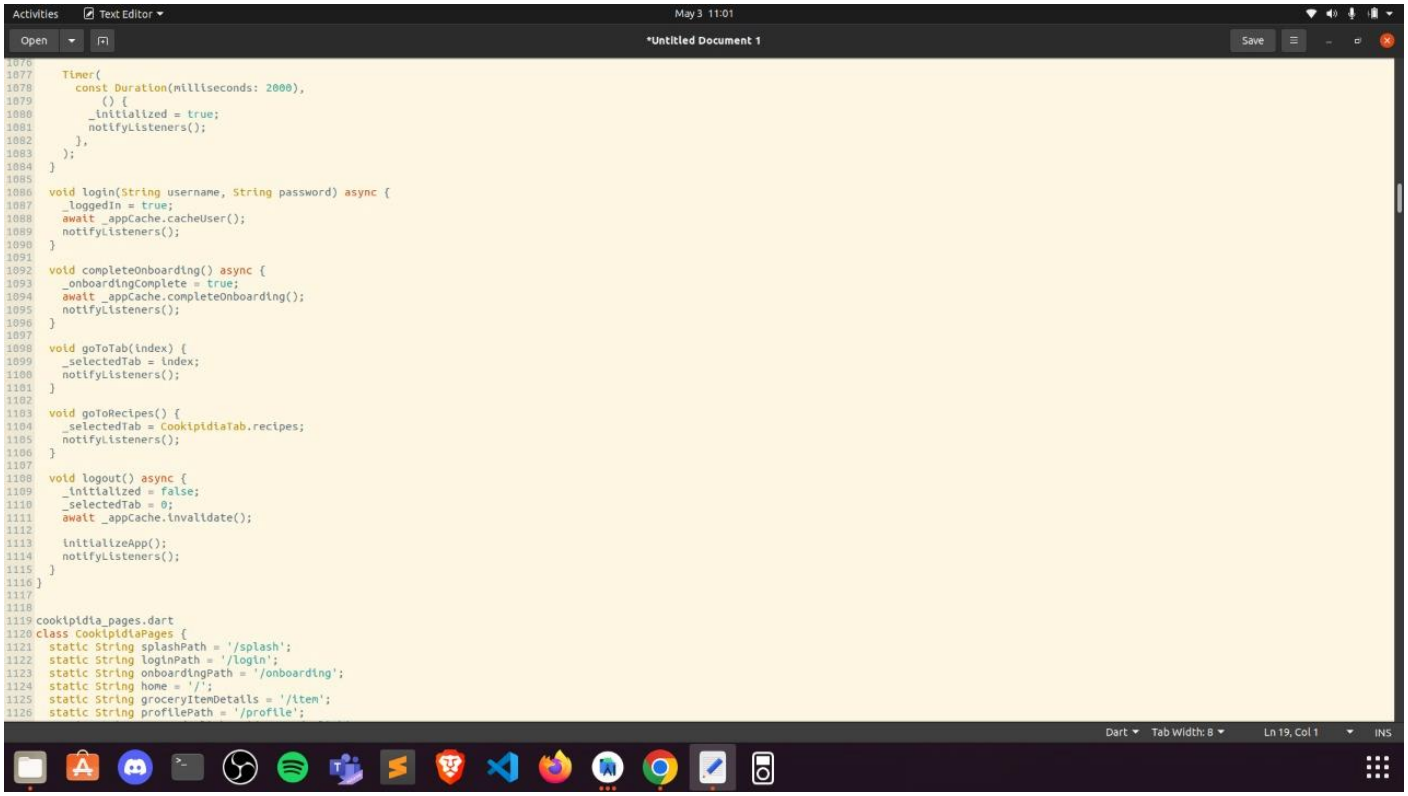

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
Timer(
const Duration(milliseconds: 2000),
() {
_initialized = true;
notifyListeners();
},
);
}
void login(String username, String password) async {
_loggedIn = true;
await _appCache.cacheUser();
notifyListeners();
}
void completeOnboarding() async {
_onboardingComplete = true;
await _appCache.completeOnboarding();
notifyListeners();
}
void goToTab(index) {
_selectedTab = index;
notifyListeners();
}
void goToRecipes() {
_selectedTab = CookipidiaTab.recipes;
notifyListeners();
}
void logout() async {
_initialized = false;
_selectedTab = 0;
await _appCache.invalidate();
initializeApp();
notifyListeners();
}
}
cookipidia_pages.dart
class CookipidiaPages {
static String splashPath = '/splash';
static String loginPath = '/login';
static String onboardingPath = '/onboarding';
static String home = '/';
static String groceryItemDetails = '/item';
static String profilePath = '/profile';
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 87

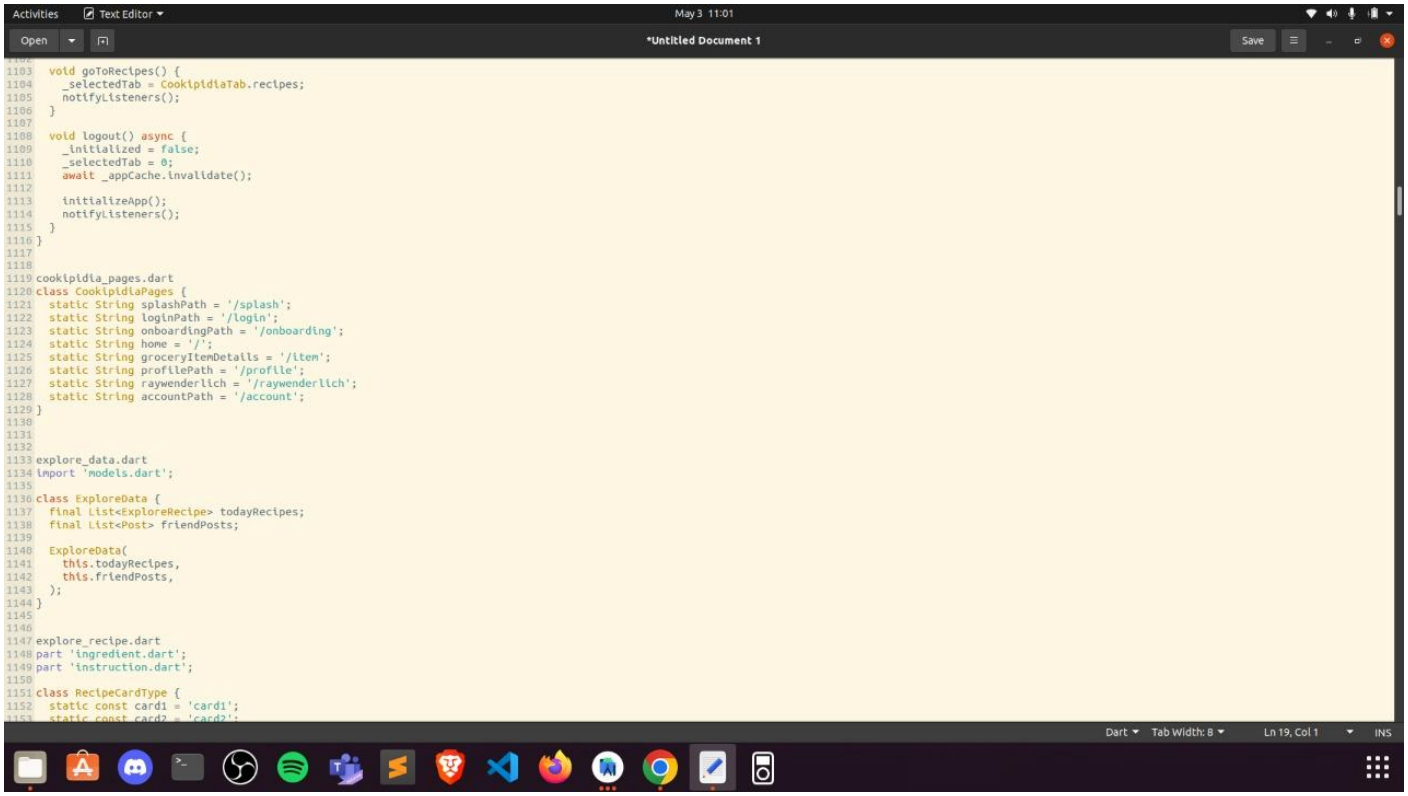

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
void goToRecipes() {
_selectedTab = CookipidiaTab.recipes;
notifyListeners();
}
void logout() async {
_initialized = false;
_selectedTab = 0;
await _appCache.invalidate();
initializeApp();
notifyListeners();
}
}
cookipidia_pages.dart
class CookipidiaPages {
static String splashPath = '/splash';
static String loginPath = '/login';
static String onboardingPath = '/onboarding';
static String home = '/';
static String groceryItemDetails = '/item';
static String profilePath = '/profile';
static String raywenderlich = '/raywenderlich';
static String accountPath = '/account';
}
explore_data.dart
import 'models.dart';
class ExploreData {
final List<ExploreRecipe> todayRecipes;
final List<Post> friendPosts;
ExploreData(
this.todayRecipes,
this.friendPosts,
);
}
explore_recipe.dart
part 'ingredient.dart';
part 'instruction.dart';
class RecipeCardType {
static const card1 = 'card1';
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 88

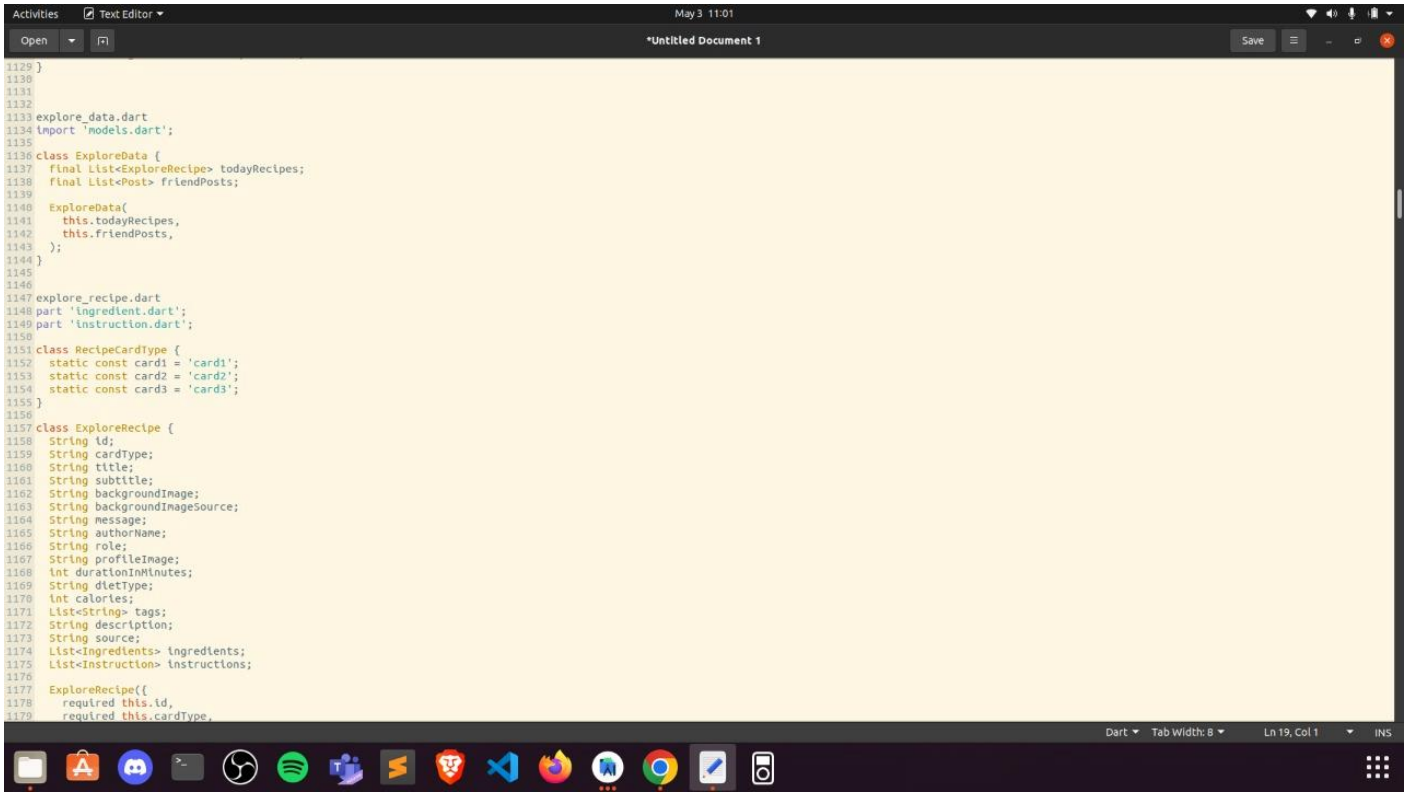

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
}
explore_data.dart
import 'models.dart';
class ExploreData {
final List<ExploreRecipe> todayRecipes;
final List<Post> friendPosts;
ExploreData(
this.todayRecipes,
this.friendPosts,
);
}
explore_recipe.dart
part 'ingredient.dart';
part 'instruction.dart';
class RecipeCardType {
static const card1 = 'card1';
static const card2 = 'card2';
static const card3 = 'card3';
}
class ExploreRecipe {
String id;
String cardType;
String title;
String subtitle;
String backgroundImage;
String backgroundImageSource;
String message;
String authorName;
String role;
String profileImage;
int durationInMinutes;
String dietType;
int calories;
List<String> tags;
String description;
String source;
List<Ingredients> ingredients;
List<Instruction> instructions;
ExploreRecipe({
required this.id,
required this.cardType,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 89

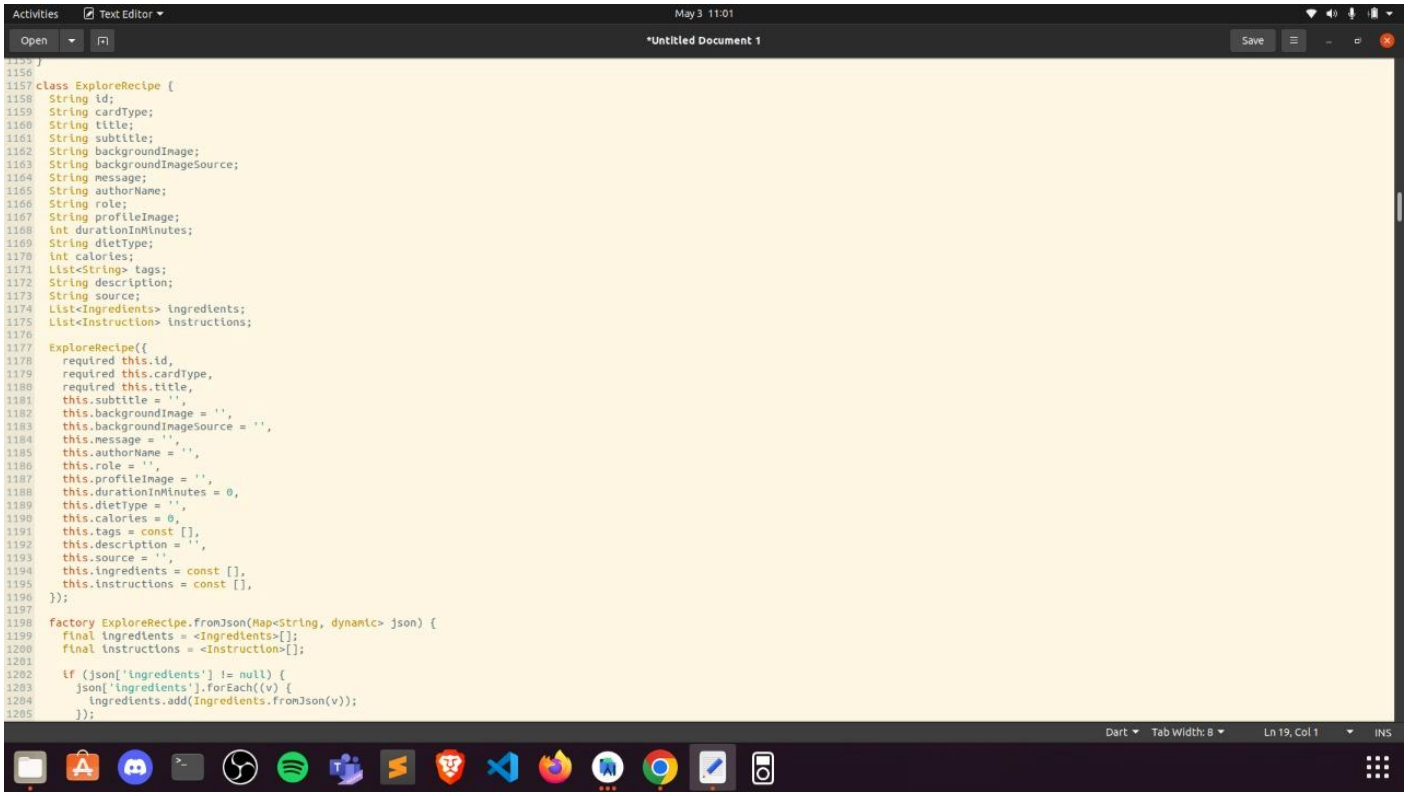

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
class ExploreRecipe {
String id;
String cardType;
String title;
String subtitle;
String backgroundImage;
String backgroundImageSource;
String message;
String authorName;
String role;
String profileImage;
int durationInMinutes;
String dietType;
int calories;
List<String> tags;
String description;
String source;
List<Ingredients> ingredients;
List<Instruction> instructions;
ExploreRecipe({
required this.id,
required this.cardType,
required this.title,
this.subtitle = '',
this.backgroundImage = '',
this.backgroundImageSource = '',
this.message = '',
this.authorName = '',
this.role = '',
this.profileImage = '',
this.durationInMinutes = 0,
this.dietType = '',
this.calories = 0,
this.tags = const [],
this.description = '',
this.source = '',
this.ingredients = const [],
this.instructions = const [],
});
factory ExploreRecipe.fromJson(Map<String, dynamic> json) {
final ingredients = <Ingredients>[];
final instructions = <Instruction>[];
if (json['ingredients'] != null) {
json['ingredients'].forEach((v) {
ingredients.add(Ingredients.fromJson(v));
});
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 90

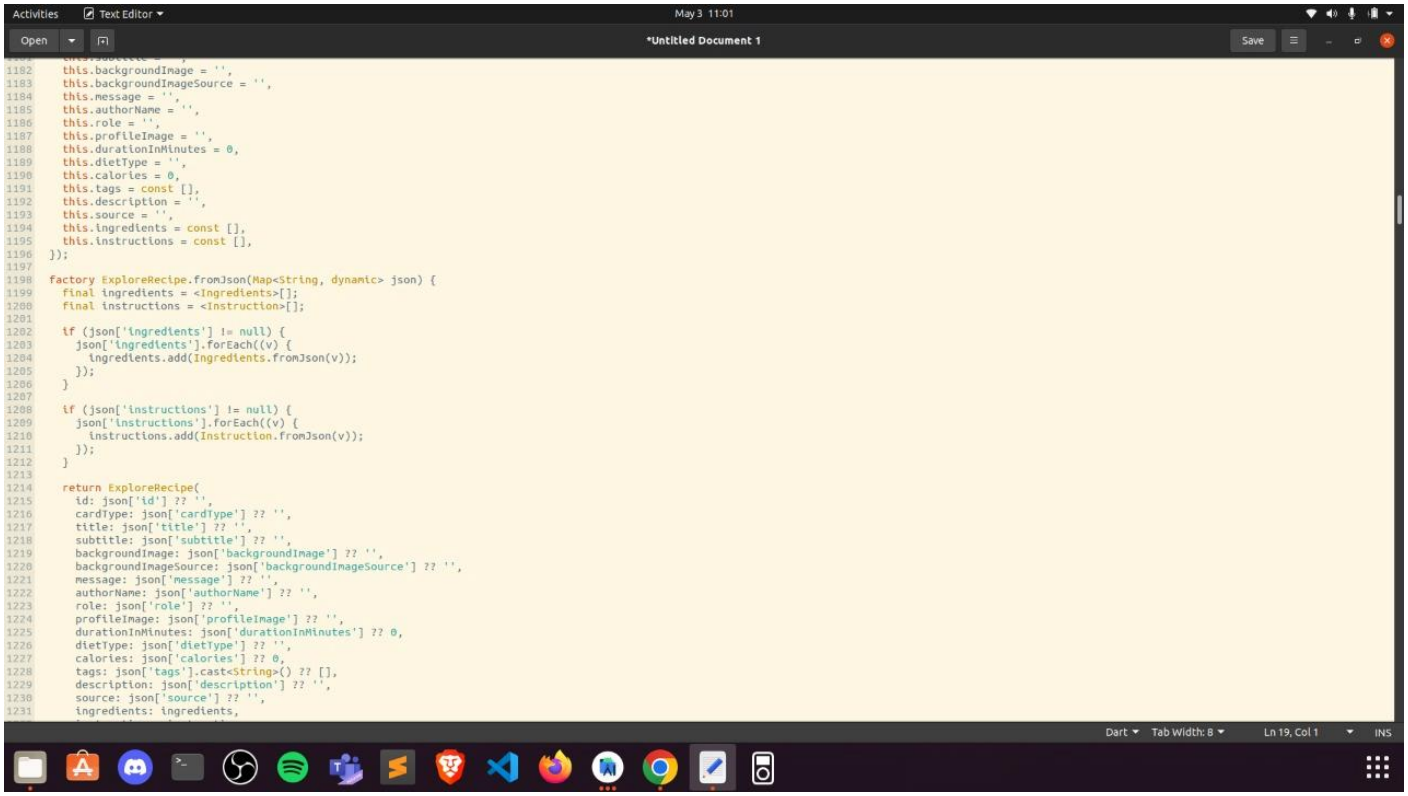

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
this.backgroundImage = '',
this.backgroundImageSource = '',
this.message = '',
this.authorName = '',
this.role = '',
this.profileImage = '',
this.durationInMinutes = 0,
this.dietType = '',
this.calories = 0,
this.tags = const [],
this.description = '',
this.source = '',
this.ingredients = const [],
this.instructions = const [],
});
factory ExploreRecipe.fromJson(Map<String, dynamic> json) {
final ingredients = <Ingredients>[];
final instructions = <Instruction>[];
if (json['ingredients'] != null) {
json['ingredients'].forEach((v) {
ingredients.add(Ingredients.fromJson(v));
});
}
if (json['instructions'] != null) {
json['instructions'].forEach((v) {
instructions.add(Instruction.fromJson(v));
});
}
return ExploreRecipe(
id: json['id'] ?? '',
cardType: json['cardType'] ?? '',
title: json['title'] ?? '',
subtitle: json['subtitle'] ?? '',
backgroundImage: json['backgroundImage'] ?? '',
backgroundImageSource: json['backgroundImageSource'] ?? '',
message: json['message'] ?? '',
authorName: json['authorName'] ?? '',
role: json['role'] ?? '',
profileImage: json['profileImage'] ?? '',
durationInMinutes: json['durationInMinutes'] ?? 0,
dietType: json['dietType'] ?? '',
calories: json['calories'] ?? 0,
tags: json['tags'].cast<String>() ?? [],
description: json['description'] ?? '',
source: json['source'] ?? '',
ingredients: ingredients,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 91

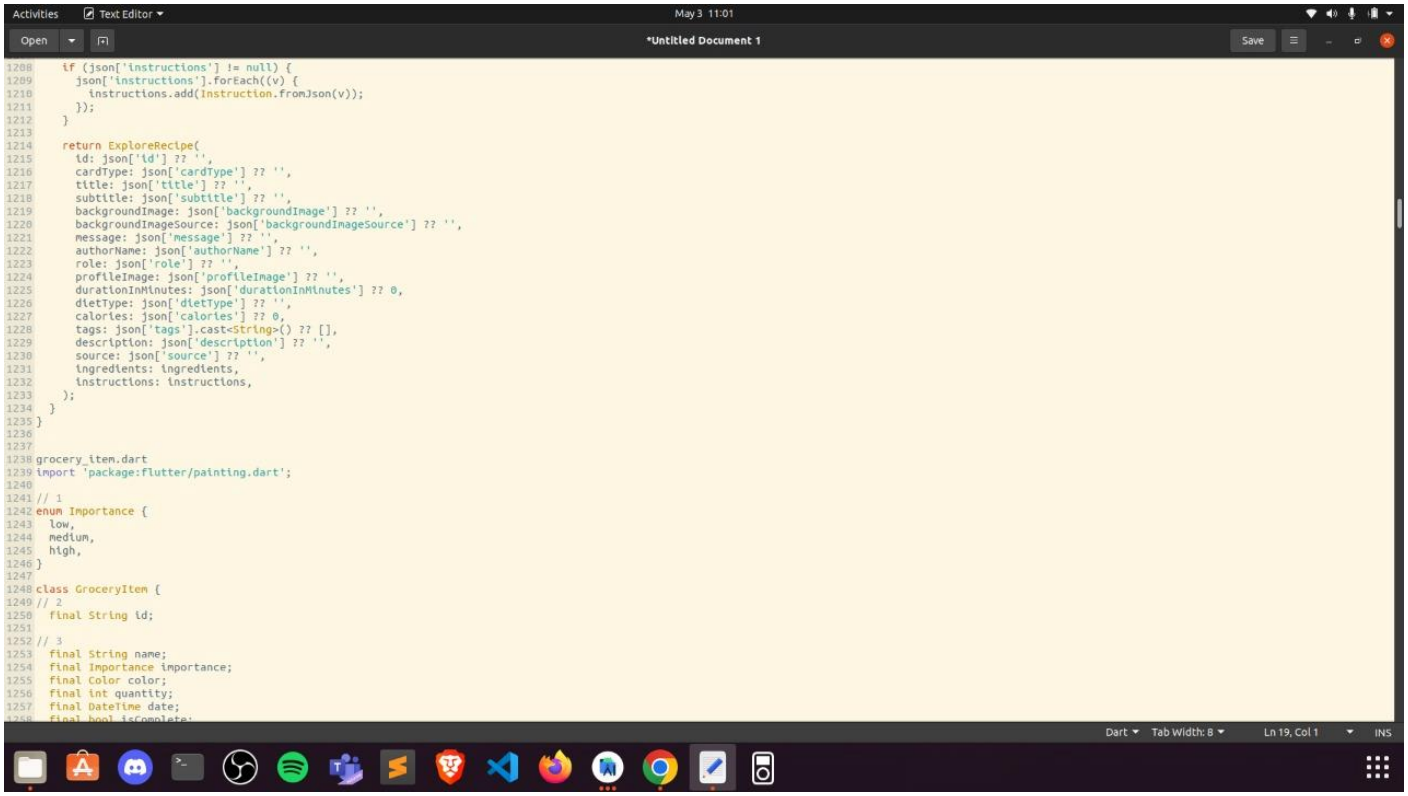

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
if (json['instructions'] != null) {
json['instructions'].forEach((v) {
instructions.add(Instruction.fromJson(v));
});
}
return ExploreRecipe(
id: json['id'] ?? '',
cardType: json['cardType'] ?? '',
title: json['title'] ?? '',
subtitle: json['subtitle'] ?? '',
backgroundImage: json['backgroundImage'] ?? '',
backgroundImageSource: json['backgroundImageSource'] ?? '',
message: json['message'] ?? '',
authorName: json['authorName'] ?? '',
role: json['role'] ?? '',
profileImage: json['profileImage'] ?? '',
durationInMinutes: json['durationInMinutes'] ?? 0,
dietType: json['dietType'] ?? '',
calories: json['calories'] ?? 0,
tags: json['tags'].cast<String>() ?? [],
description: json['description'] ?? '',
source: json['source'] ?? '',
ingredients: ingredients,
instructions: instructions,
);
}
}
grocery_item.dart
import 'package:flutter/painting.dart';
// 1
enum Importance {
low,
medium,
high,
}
class GroceryItem {
// 2
final String id;
// 3
final String name;
final Importance importance;
final Color color;
final int quantity;
final DateTime date;
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 92

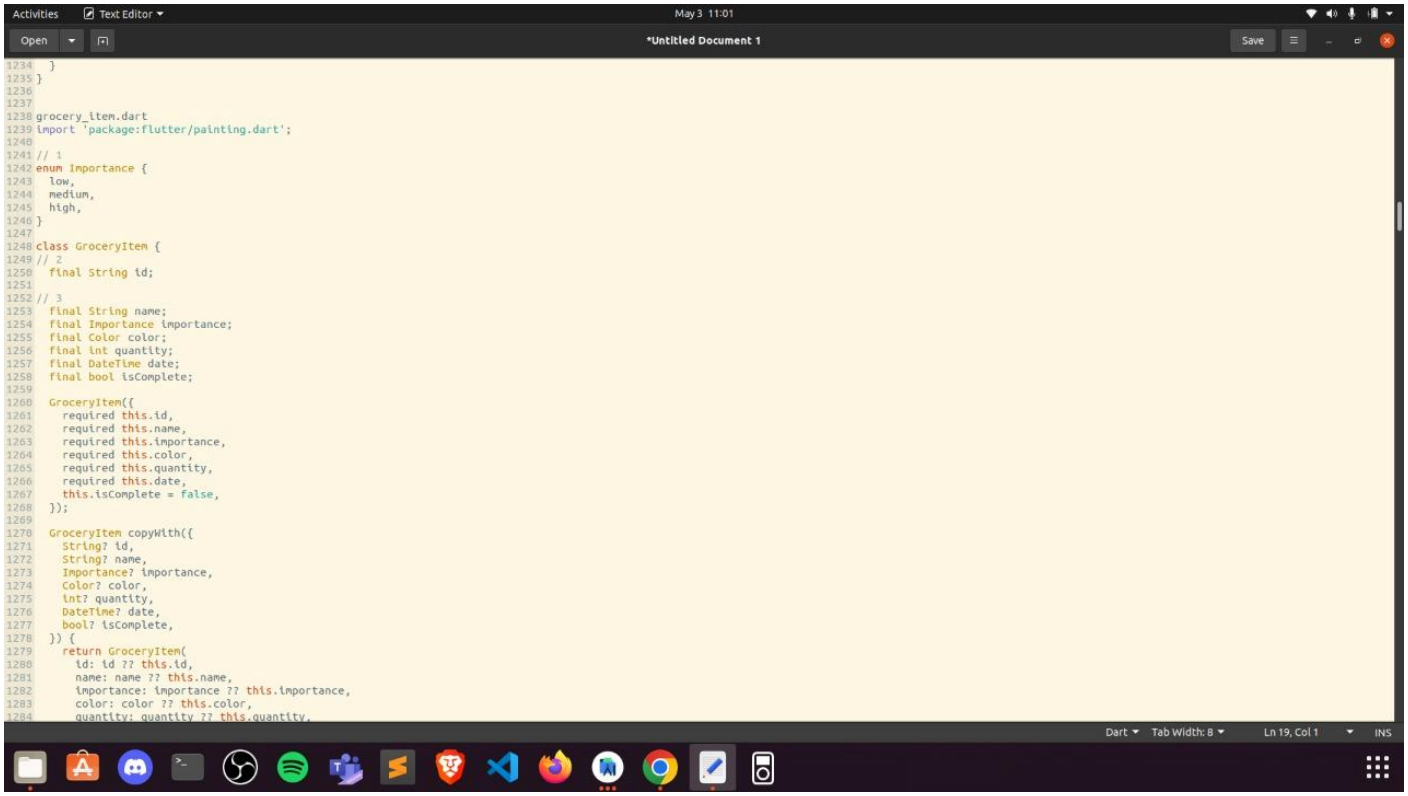

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
}
}
grocery_item.dart
import 'package:flutter/painting.dart';
// 1
enum Importance {
low,
medium,
high,
}
class GroceryItem {
// 2
final String id;
// 3
final String name;
final Importance importance;
final Color color;
final int quantity;
final DateTime date;
final bool isComplete;
GroceryItem({
required this.id,
required this.name,
required this.importance,
required this.color,
required this.quantity,
required this.date,
this.isComplete = false,
});
GroceryItem copyWith({
String? id,
String? name,
Importance? importance,
Color? color,
int? quantity,
DateTime? date,
bool? isComplete,
}) {
return GroceryItem(
id: id ?? this.id,
name: name ?? this.name,
importance: importance ?? this.importance,
color: color ?? this.color,
quantity: quantity ?? this.quantity,
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 93

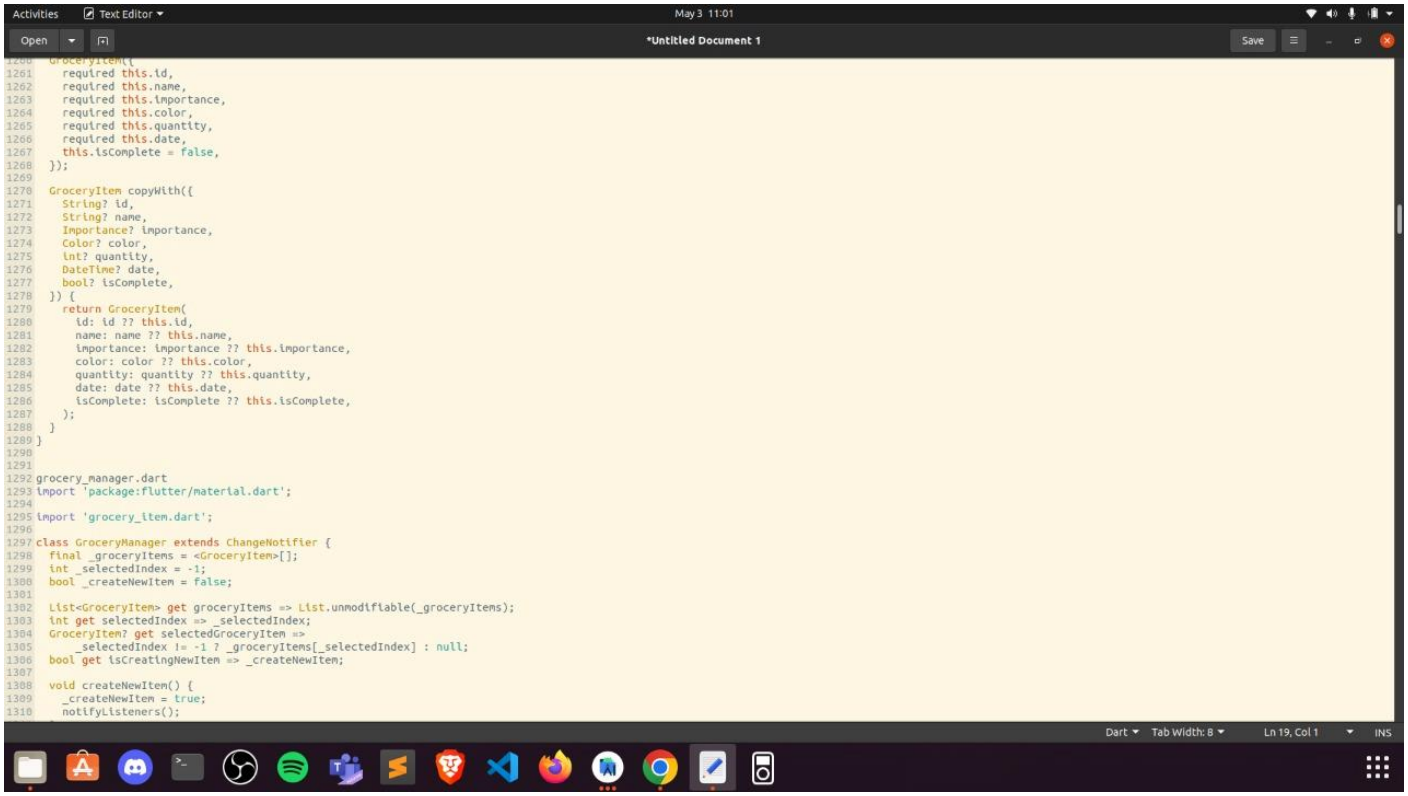

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
required this.id,
required this.name,
required this.importance,
required this.color,
required this.quantity,
required this.date,
this.isComplete = false,
});
GroceryItem copyWith({
String? id,
String? name,
Importance? importance,
Color? color,
int? quantity,
DateTime? date,
bool? isComplete,
}) {
return GroceryItem(
id: id ?? this.id,
name: name ?? this.name,
importance: importance ?? this.importance,
color: color ?? this.color,
quantity: quantity ?? this.quantity,
date: date ?? this.date,
isComplete: isComplete ?? this.isComplete,
);
}
}
grocery_manager.dart
import 'package:flutter/material.dart';
import 'grocery_item.dart';
class GroceryManager extends ChangeNotifier {
final _groceryItems = <GroceryItem>[];
int _selectedIndex = -1;
bool _createNewItem = false;
List<GroceryItem> get groceryItems => List.unmodifiable(_groceryItems);
int get selectedIndex => _selectedIndex;
GroceryItem? get selectedGroceryItem =>
_selectedIndex != -1 ? _groceryItems[_selectedIndex] : null;
bool get isCreatingNewItem => _createNewItem;
void createNewItem() {
_createNewItem = true;
notifyListeners();
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 94

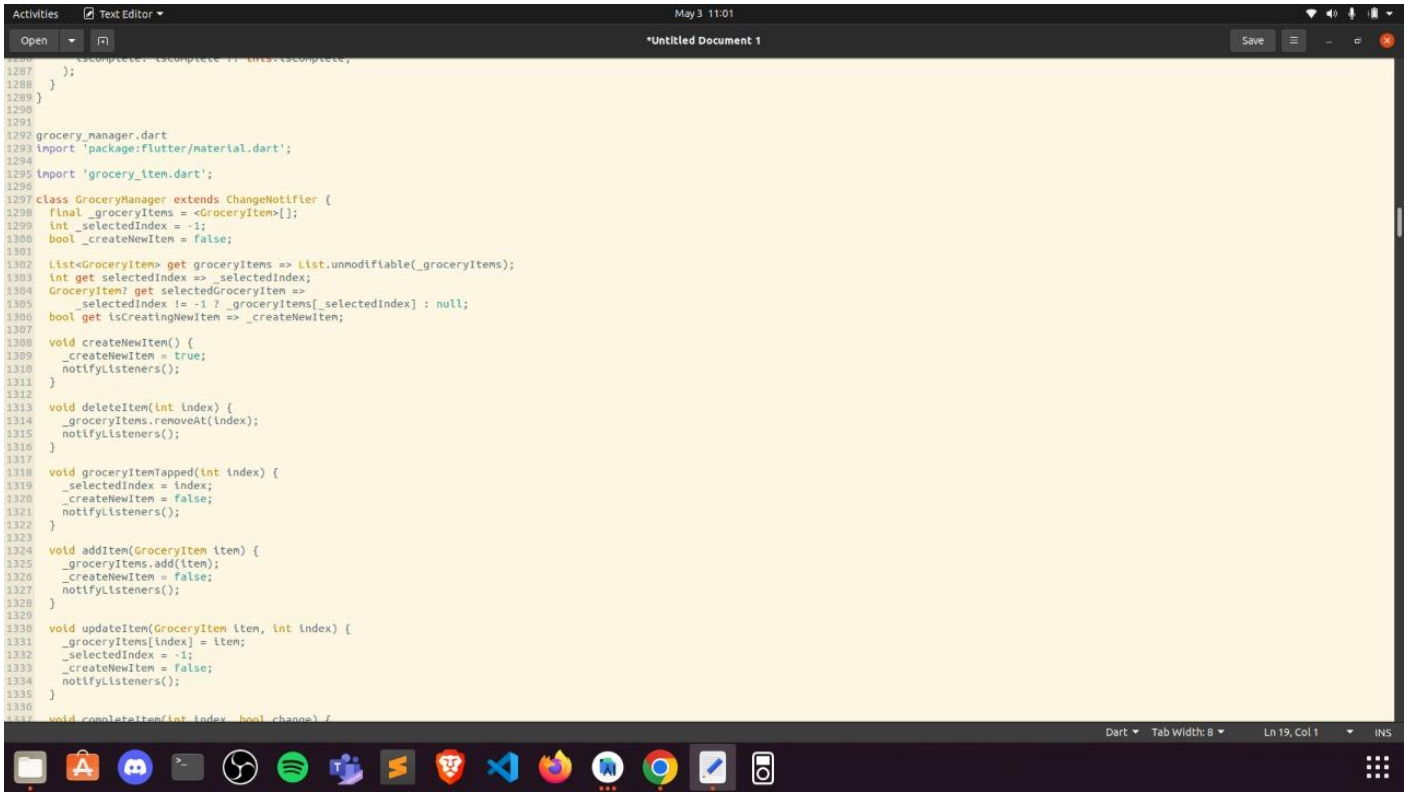
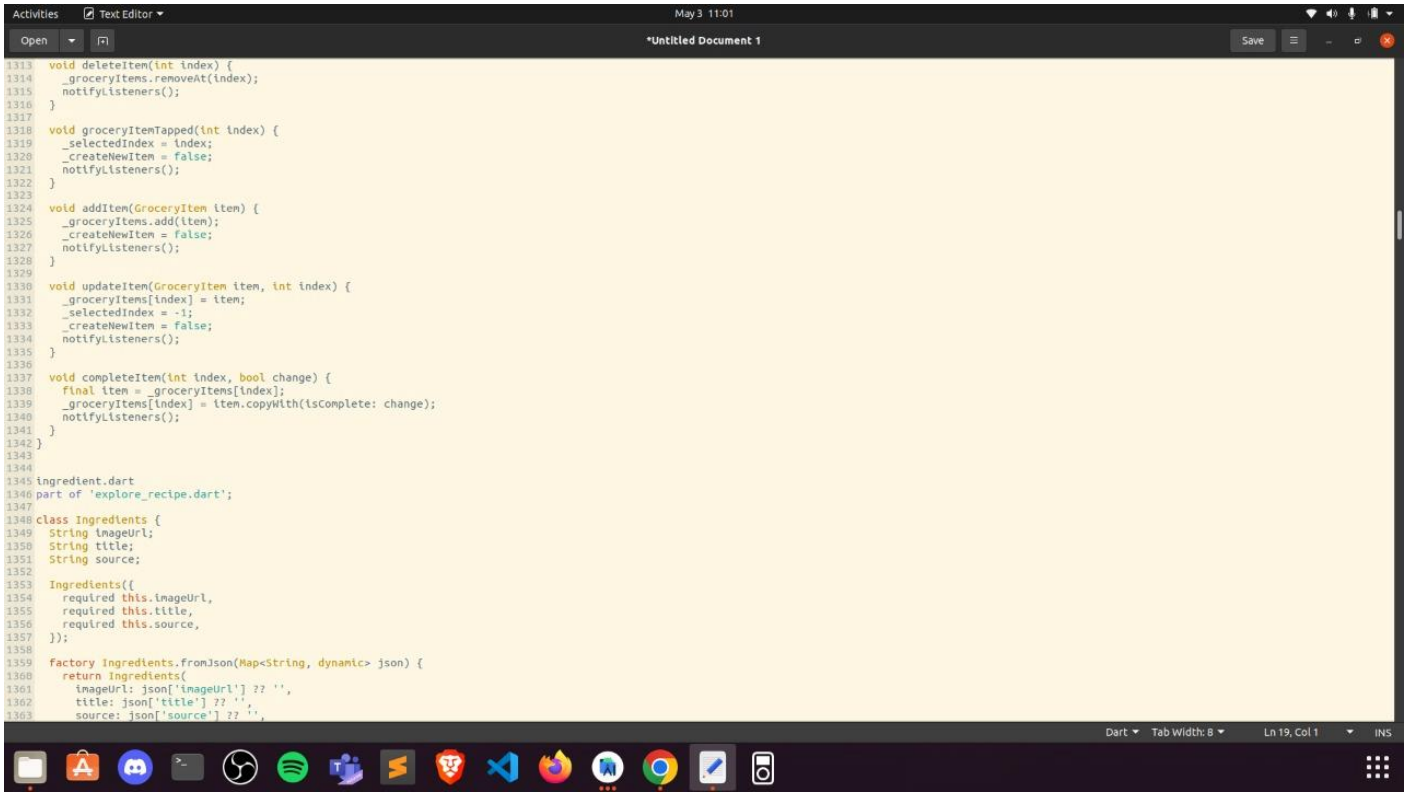
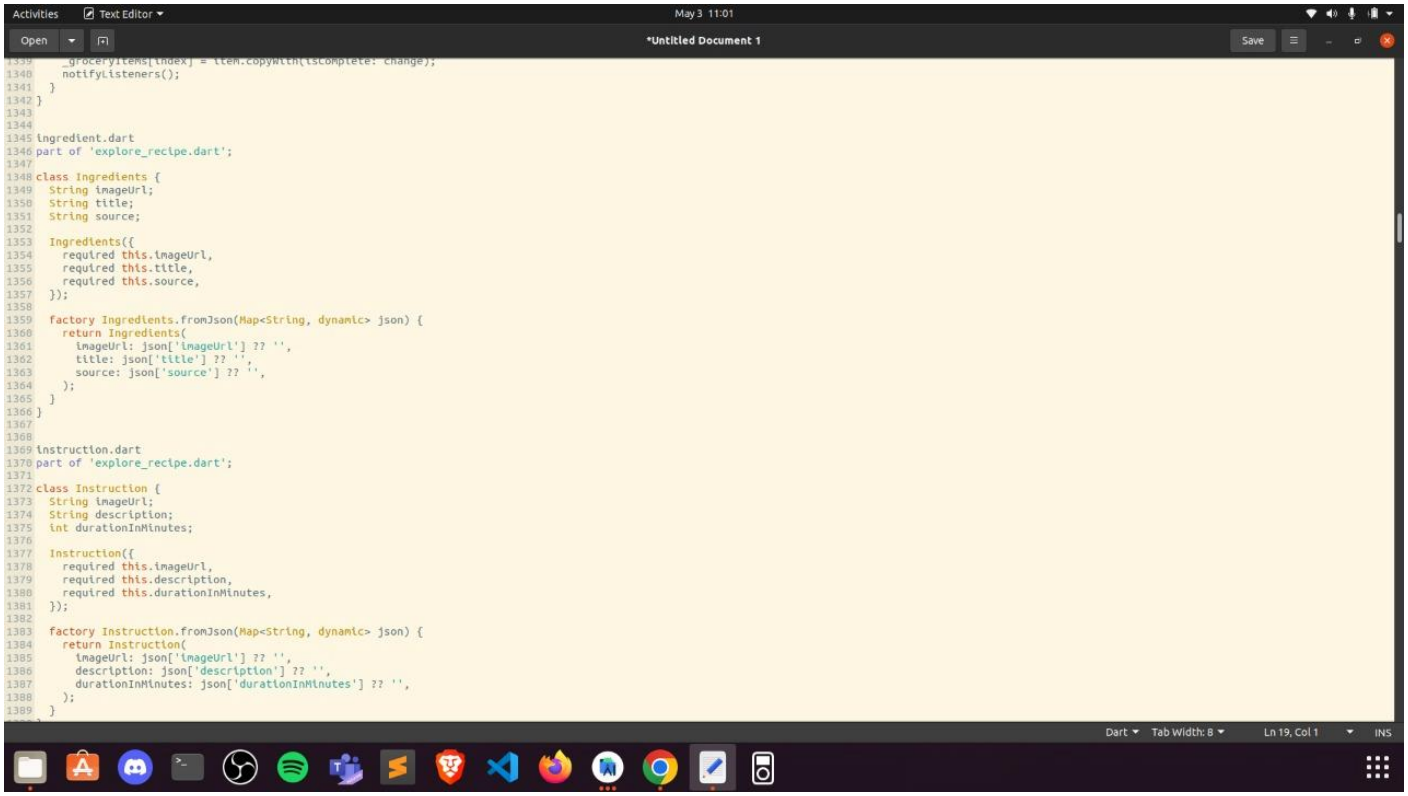

Activities
Text Editor
May 3 11:01
Open
*Untitled Document 1
Save
);
}
}
grocery_manager.dart
import 'package:flutter/material.dart';
import 'grocery_item.dart';
class GroceryManager extends ChangeNotifier {
final _groceryItems = <GroceryItem>[];
int _selectedIndex = -1;
bool _createNewItem = false;
List<GroceryItem> get groceryItems => List.unmodifiable(_groceryItems);
int get selectedIndex => _selectedIndex;
GroceryItem? get selectedGroceryItem =>
_selectedIndex != -1 ? _groceryItems[_selectedIndex] : null;
bool get isCreatingNewItem => _createNewItem;
void createNewItem() {
_createNewItem = true;
notifyListeners();
}
void deleteItem(int index) {
_groceryItems.removeAt(index);
notifyListeners();
}
void groceryItemTapped(int index) {
_selectedIndex = index;
_createNewItem = false;
notifyListeners();
}
void addItem(GroceryItem item) {
_groceryItems.add(item);
_createNewItem = false;
notifyListeners();
}
void updateItem(GroceryItem item, int index) {
_groceryItems[index] = item;
_selectedIndex = -1;
_createNewItem = false;
notifyListeners();
}
Dart
Tab Width: 8
Ln 19, Col 1
INS

Fig. 95




```
  void deleteItem(int index) {
    _groceryItems.removeAt(index);
    notifyListeners();
  }

  void groceryItemTapped(int index) {
    _selectedIndex = index;
    _createNewItem = false;
    notifyListeners();
  }

  void addItem(GroceryItem item) {
    _groceryItems.add(item);
    _createNewItem = false;
    notifyListeners();
  }

  void updateItem(GroceryItem item, int index) {
    _groceryItems[index] = item;
    _selectedIndex = -1;
    _createNewItem = false;
    notifyListeners();
  }

  void completeItem(int index, bool change) {
    final item = _groceryItems[index];
    _groceryItems[index] = item.copyWith(isComplete: change);
    notifyListeners();
  }
}


ingredient.dart
part of 'explore_recipe.dart';

class Ingredients {
  String imageUrl;
  String title;
  String source;

  Ingredients({
    required this.imageUrl,
    required this.title,
    required this.source,
  });

  factory Ingredients.fromJson(Map<String, dynamic> json) {
    return Ingredients(
      imageUrl: json['imageUrl'] ?? '',
      title: json['title'] ?? '',
      source: json['source'] ?? '',
```

Fig. 96




```
    _groceryItems[index] = item.copyWith(isComplete: change);
    notifyListeners();
  }
}


ingredient.dart
part of 'explore_recipe.dart';

class Ingredients {
  String imageUrl;
  String title;
  String source;

  Ingredients({
    required this.imageUrl,
    required this.title,
    required this.source,
  });

  factory Ingredients.fromJson(Map<String, dynamic> json) {
    return Ingredients(
      imageUrl: json['imageUrl'] ?? '',
      title: json['title'] ?? '',
      source: json['source'] ?? '',
    );
  }
}


instruction.dart
part of 'explore_recipe.dart';

class Instruction {
  String imageUrl;
  String description;
  int durationInMinutes;

  Instruction({
    required this.imageUrl,
    required this.description,
    required this.durationInMinutes,
  });

  factory Instruction.fromJson(Map<String, dynamic> json) {
    return Instruction(
      imageUrl: json['imageUrl'] ?? '',
      description: json['description'] ?? '',
      durationInMinutes: json['durationInMinutes'] ?? '',
    );
  }
```

Fig. 97

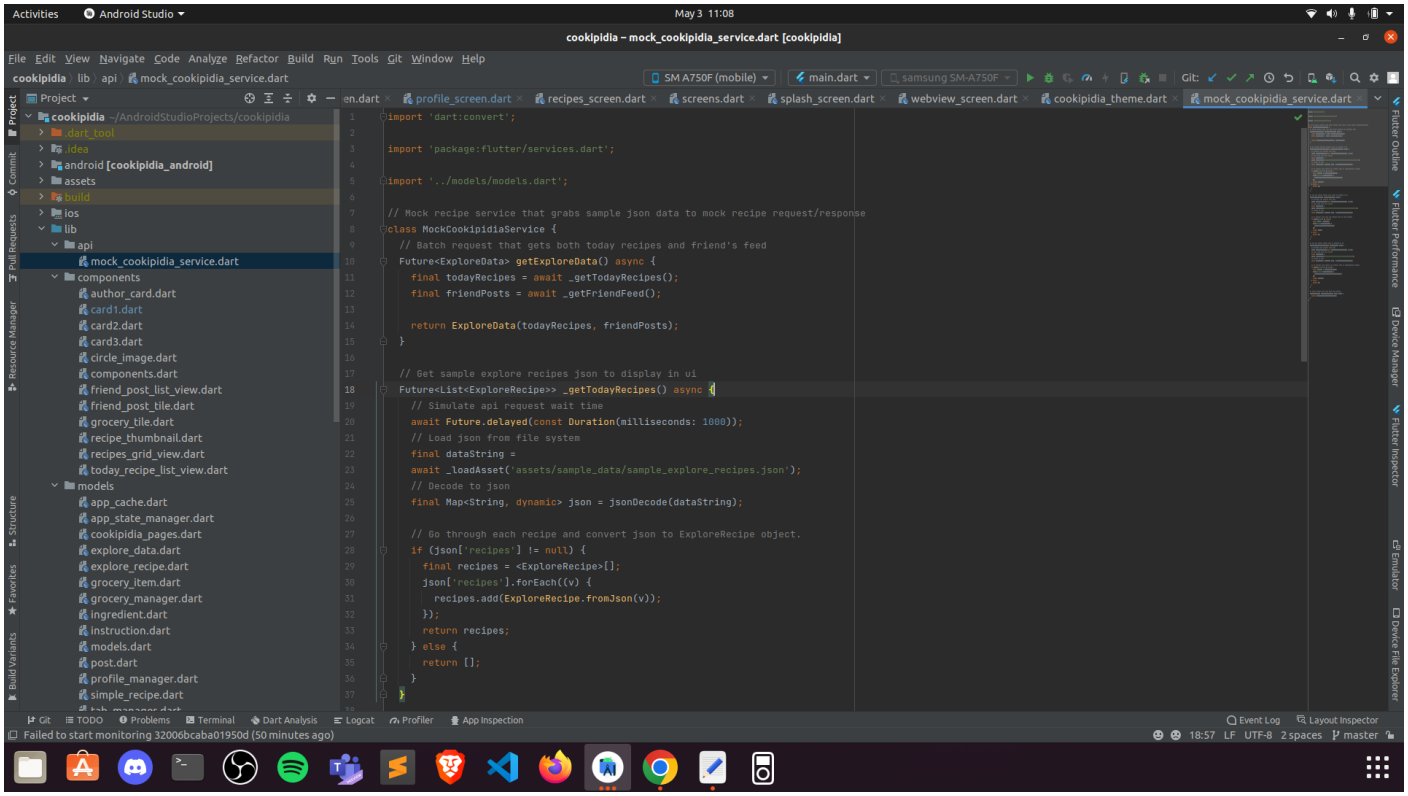

Activities
Android Studio
May 3 11:08
cookipidia – mock_cookipidia_service.dart [cookipidia]
File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help
cookipidia lib api mock_cookipidia_service.dart
SM A750F (mobile)
main.dart
samsung SM-A750F
Git:
Project
cookipidia ~/AndroidStudioProjects/cookipidia
.dart_tool
.idea
android [cookipidia_android]
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
models
app_cache.dart
app_state_manager.dart
cookipidia_pages.dart
explore_data.dart
explore_recipe.dart
grocery_item.dart
grocery_manager.dart
ingredient.dart
instruction.dart
models.dart
post.dart
profile_manager.dart
simple_recipe.dart
en.dart
profile_screen.dart
recipes_screen.dart
screens.dart
splash_screen.dart
webview_screen.dart
cookipidia_theme.dart
mock_cookipidia_service.dart
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 dataString =
await _loadAsset('assets/sample_data/sample_explore_recipes.json');
// Decode to json
final Map<String, dynamic> json = jsonDecode(dataString);
// 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 [];
}
}
Git
TODO
Problems
Terminal
Dart Analysis
Logcat
Profiler
App Inspection
Event Log
Layout Inspector
Failed to start monitoring 32006bcaba01950d (50 minutes ago)
18:57 LF UTF-8 2 spaces master

Fig. 98

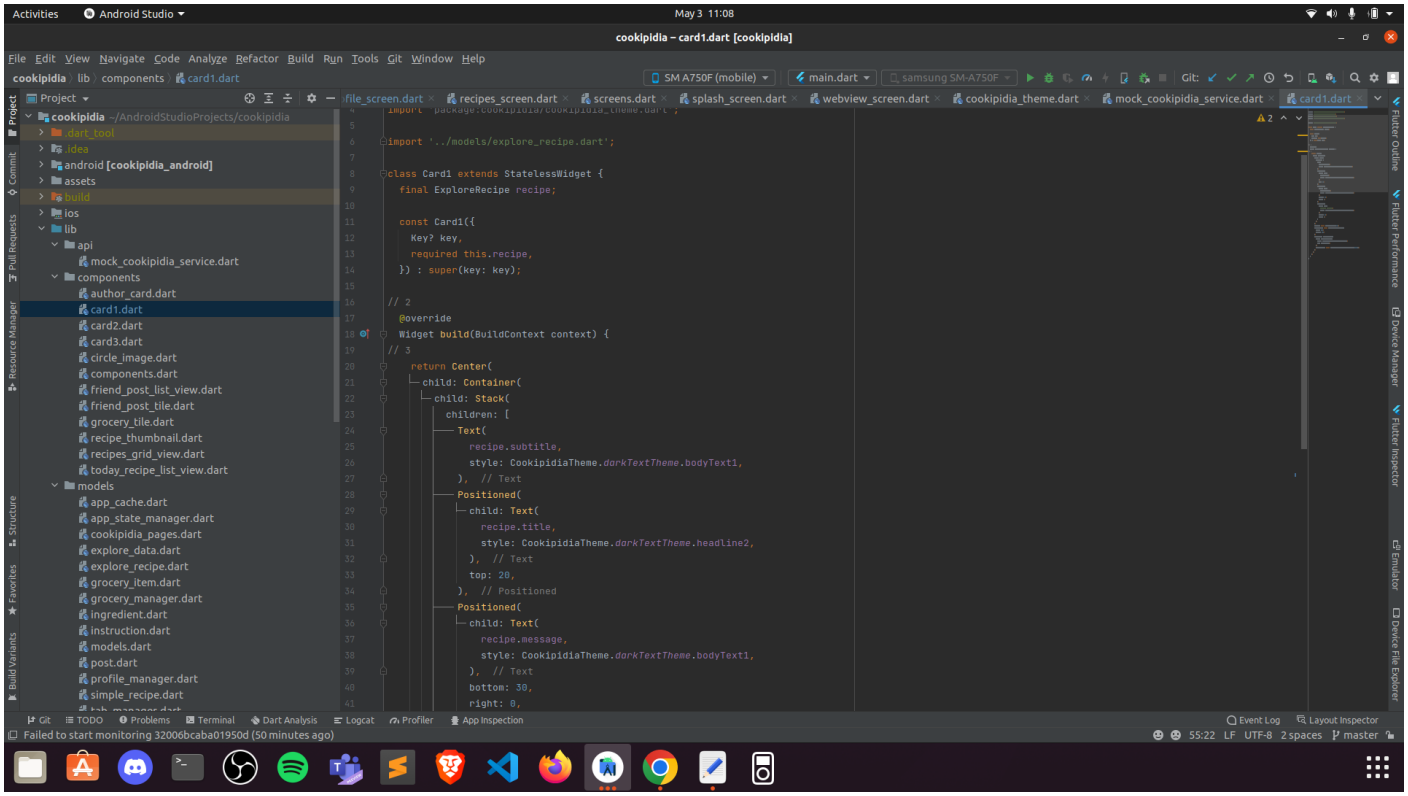

Activities
Android Studio
May 3 11:08
cookipidia – card1.dart [cookipidia]
File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help
cookipidia lib components card1.dart
SM A750F (mobile)
main.dart
samsung SM-A750F
Git:
Project
cookipidia ~/AndroidStudioProjects/cookipidia
.dart_tool
.idea
android [cookipidia_android]
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
models
app_cache.dart
app_state_manager.dart
cookipidia_pages.dart
explore_data.dart
explore_recipe.dart
grocery_item.dart
grocery_manager.dart
ingredient.dart
instruction.dart
models.dart
post.dart
profile_manager.dart
simple_recipe.dart
import '../models/explore_recipe.dart';
class Card1 extends StatelessWidget {
final ExploreRecipe recipe;
const Card1({
Key? key,
required this.recipe,
}) : super(key: key);
// 2
@override
Widget build(BuildContext context) {
// 3
return Center(
child: Container(
child: Stack(
children: [
Text(
recipe.subtitle,
style: CookipidiaTheme.darkTextTheme.bodyText1,
), // Text
Positioned(
child: Text(
recipe.title,
style: CookipidiaTheme.darkTextTheme.headline2,
), // Text
top: 20,
), // Positioned
Positioned(
child: Text(
recipe.message,
style: CookipidiaTheme.darkTextTheme.bodyText1,
), // Text
bottom: 30,
right: 8,
Git
TODO
Problems
Terminal
Dart Analysis
Logcat
Profiler
App Inspection
Event Log
Layout Inspector
Failed to start monitoring 32006bcaba01950d (50 minutes ago)
55:22 LF UTF-8 2 spaces master

Fig. 99

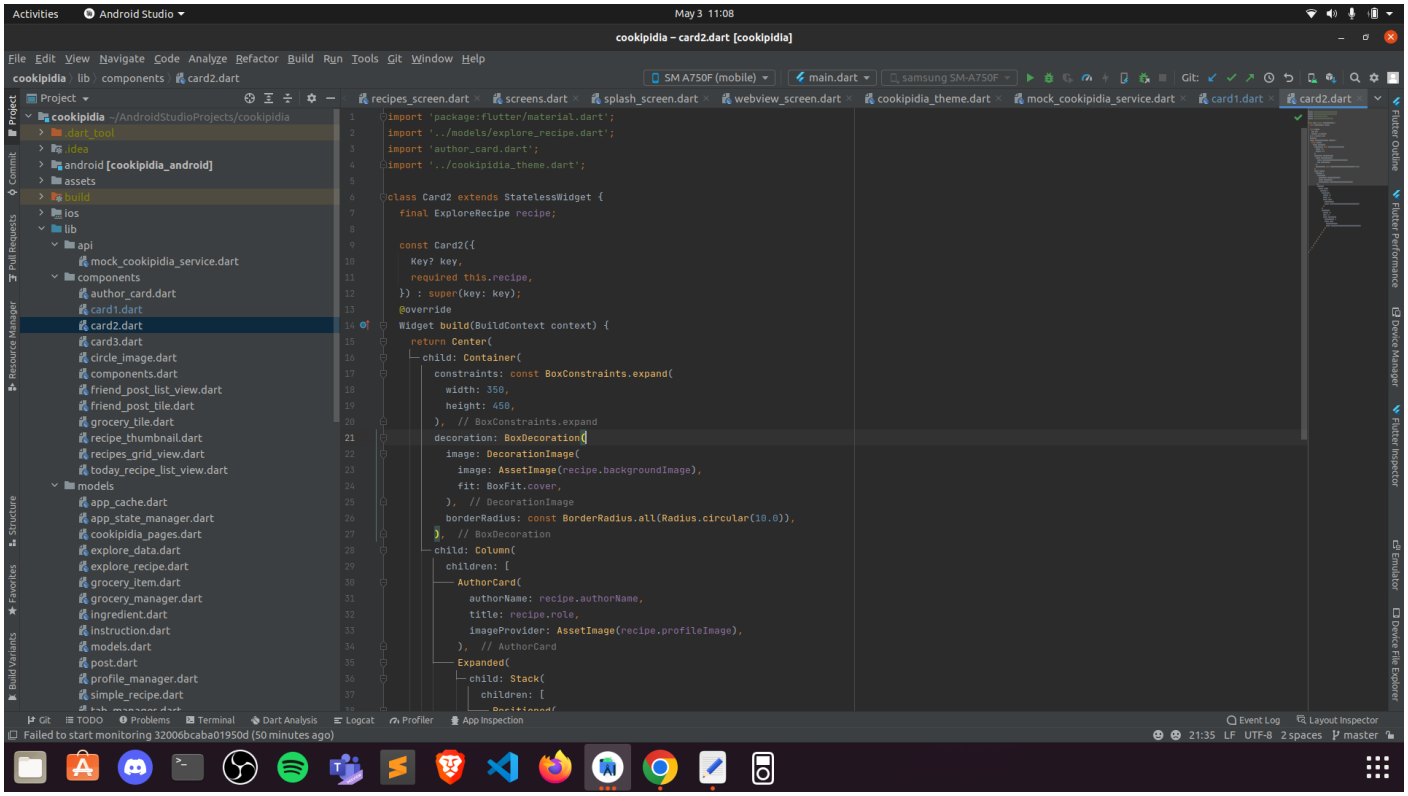


Fig. 100

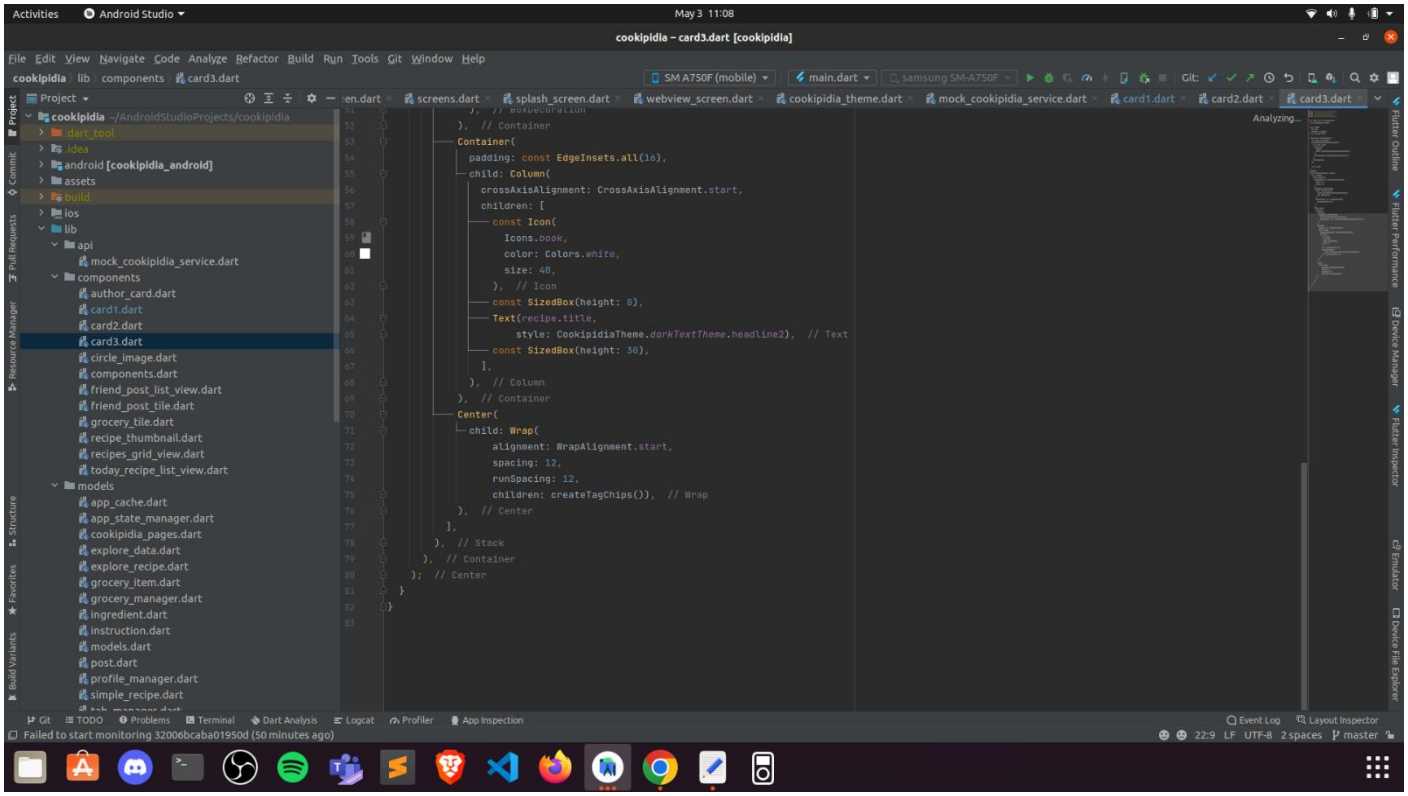


Fig. 101

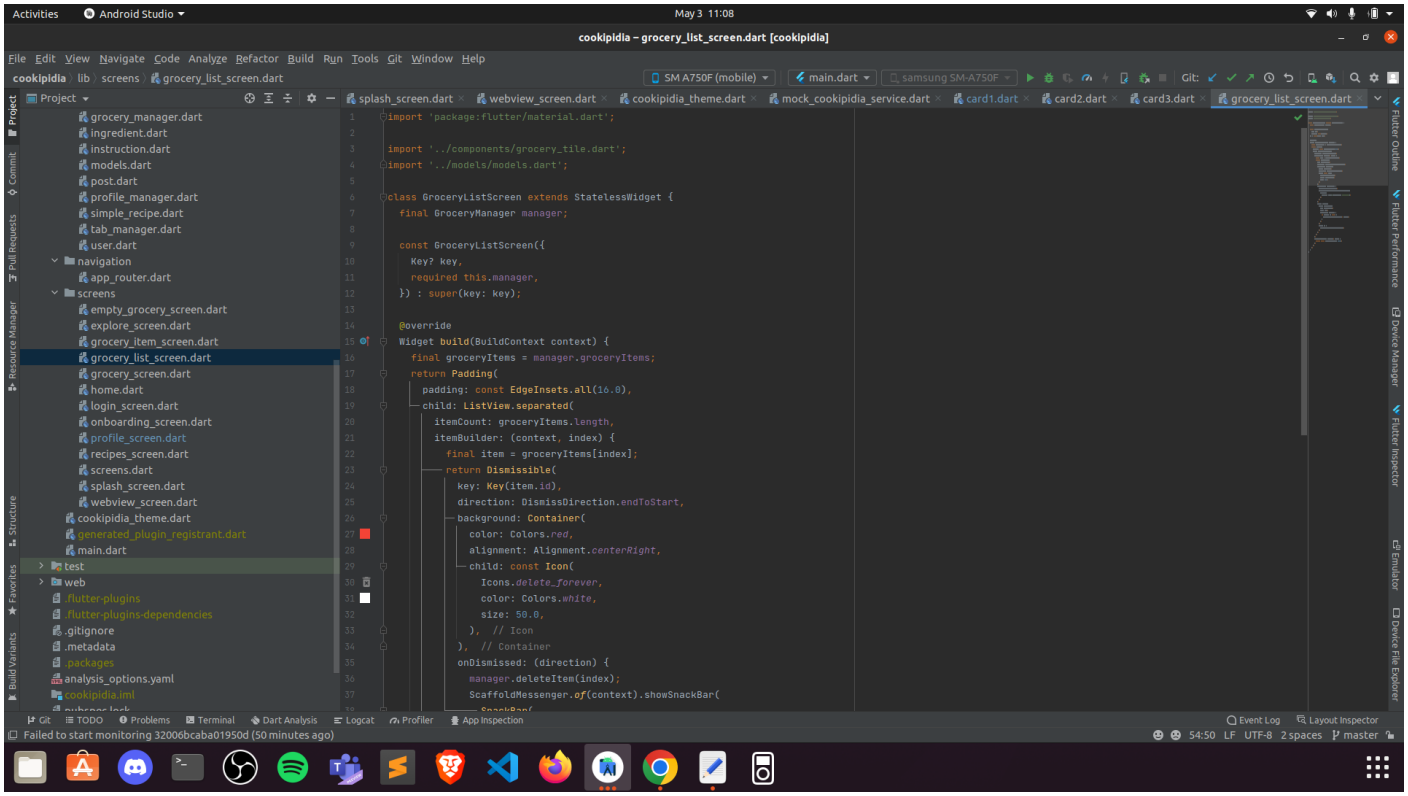


Fig. 102

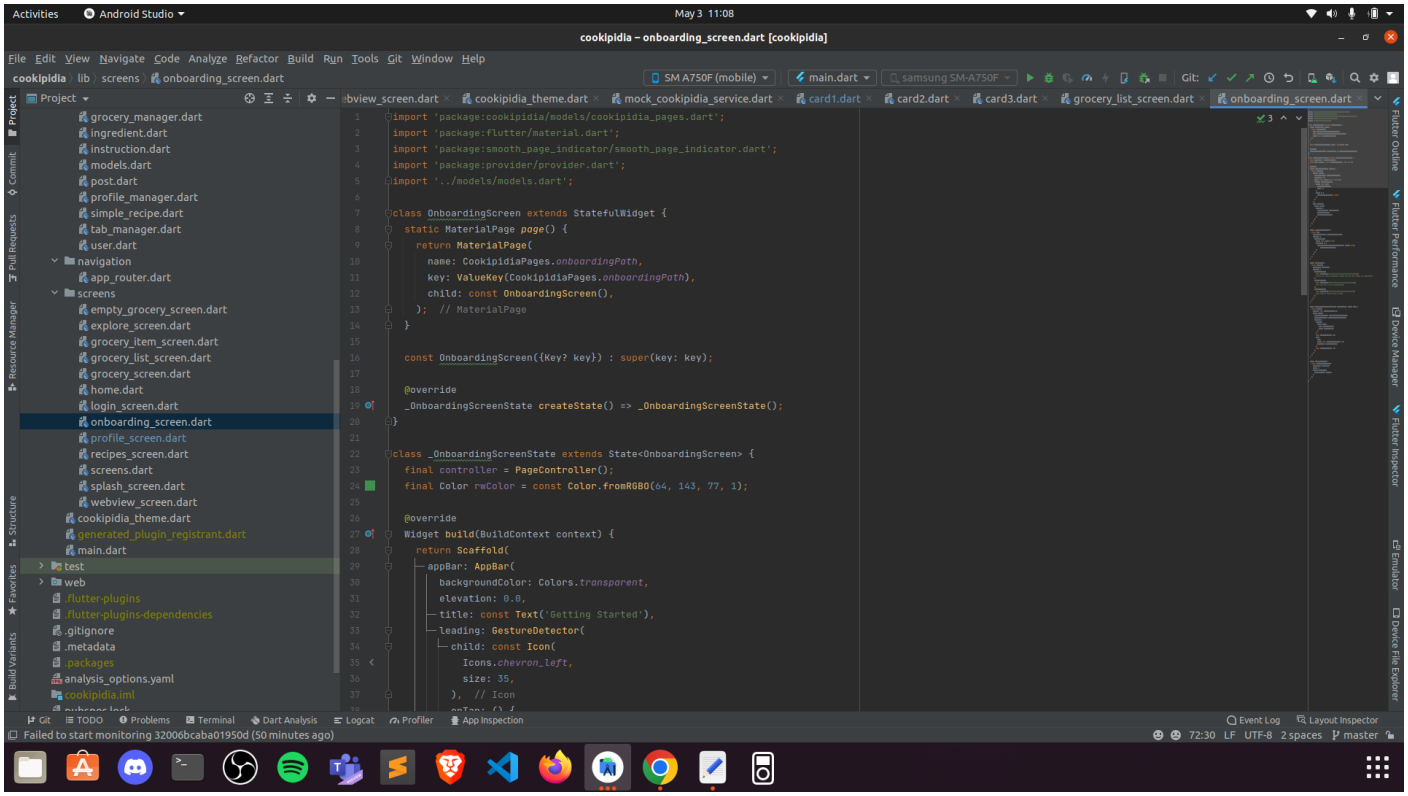


Fig. 103

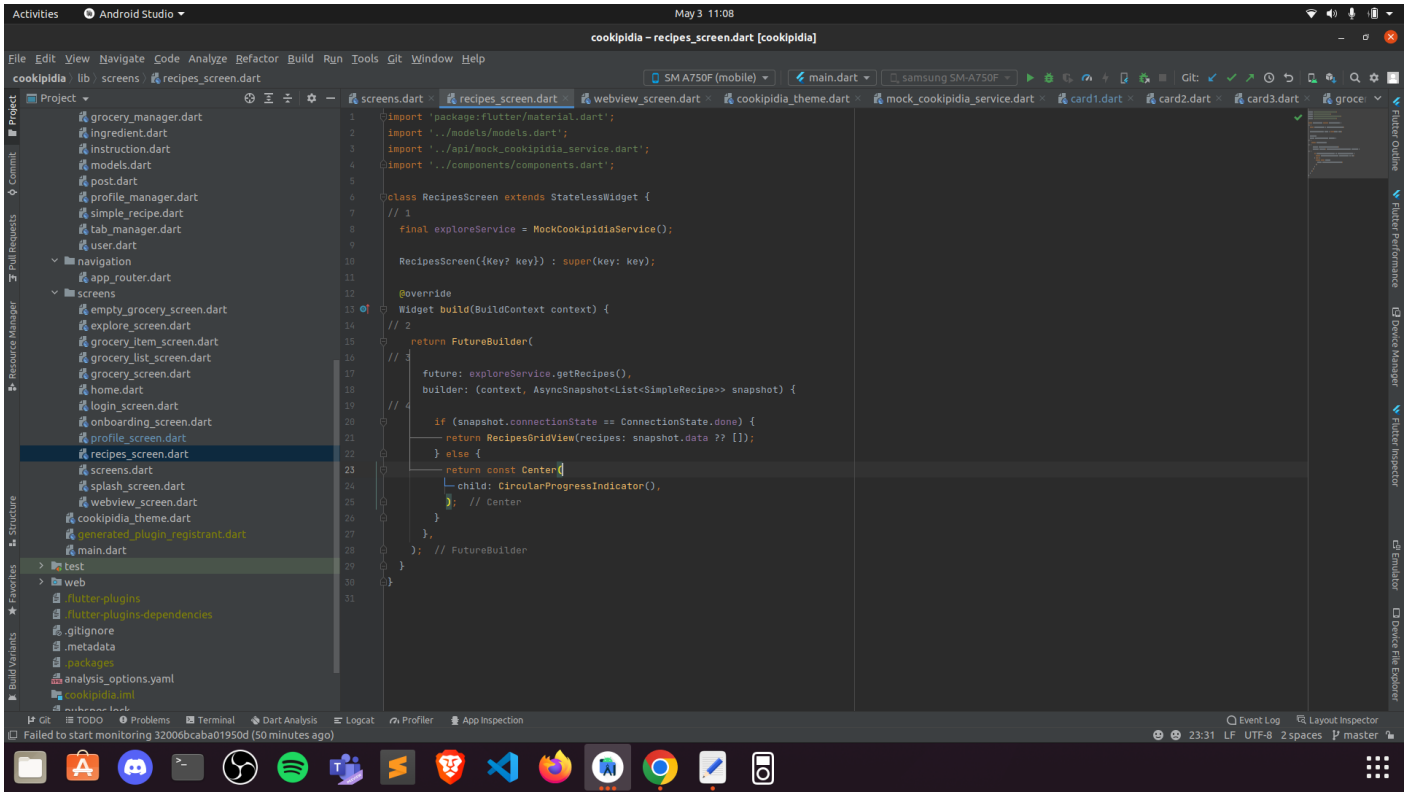


Fig. 104

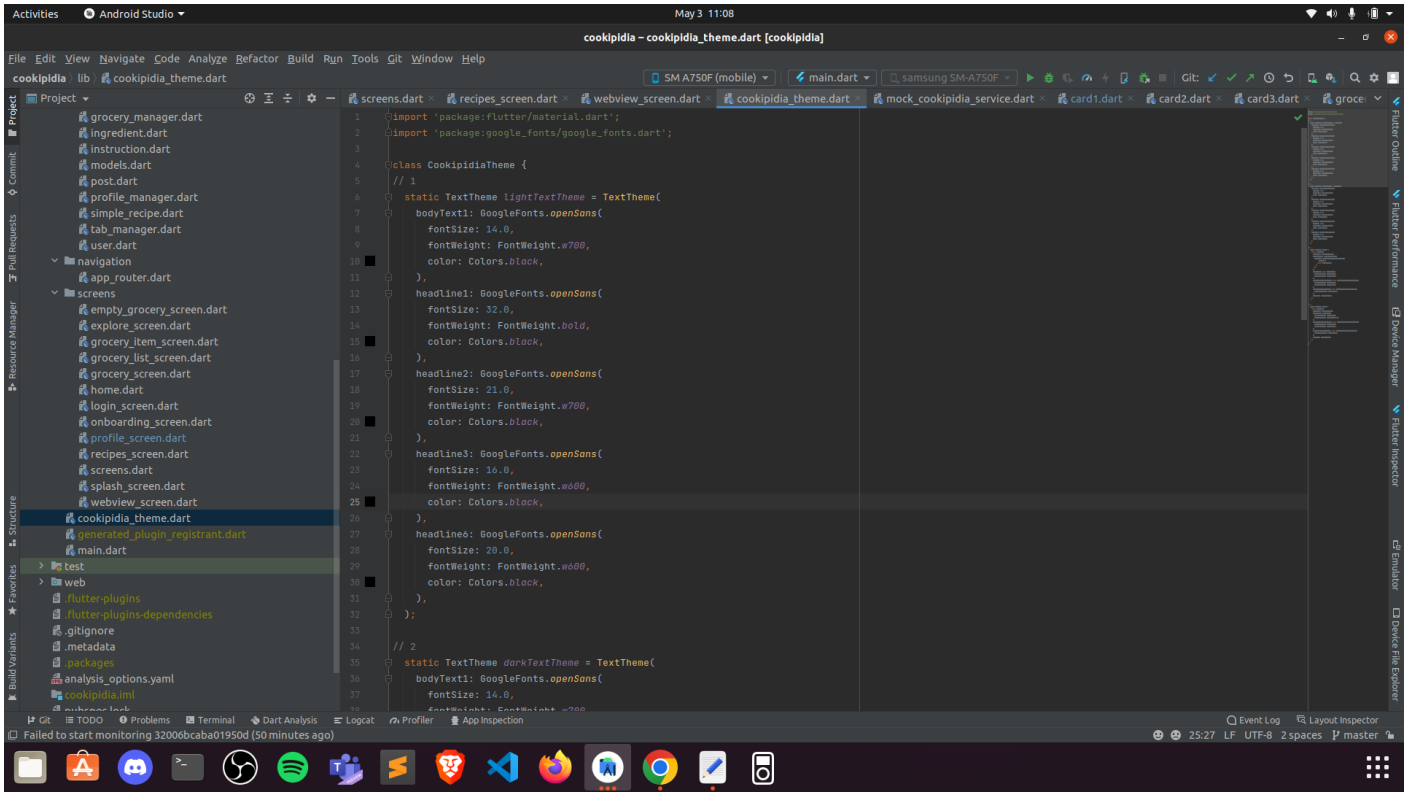


Fig. 105

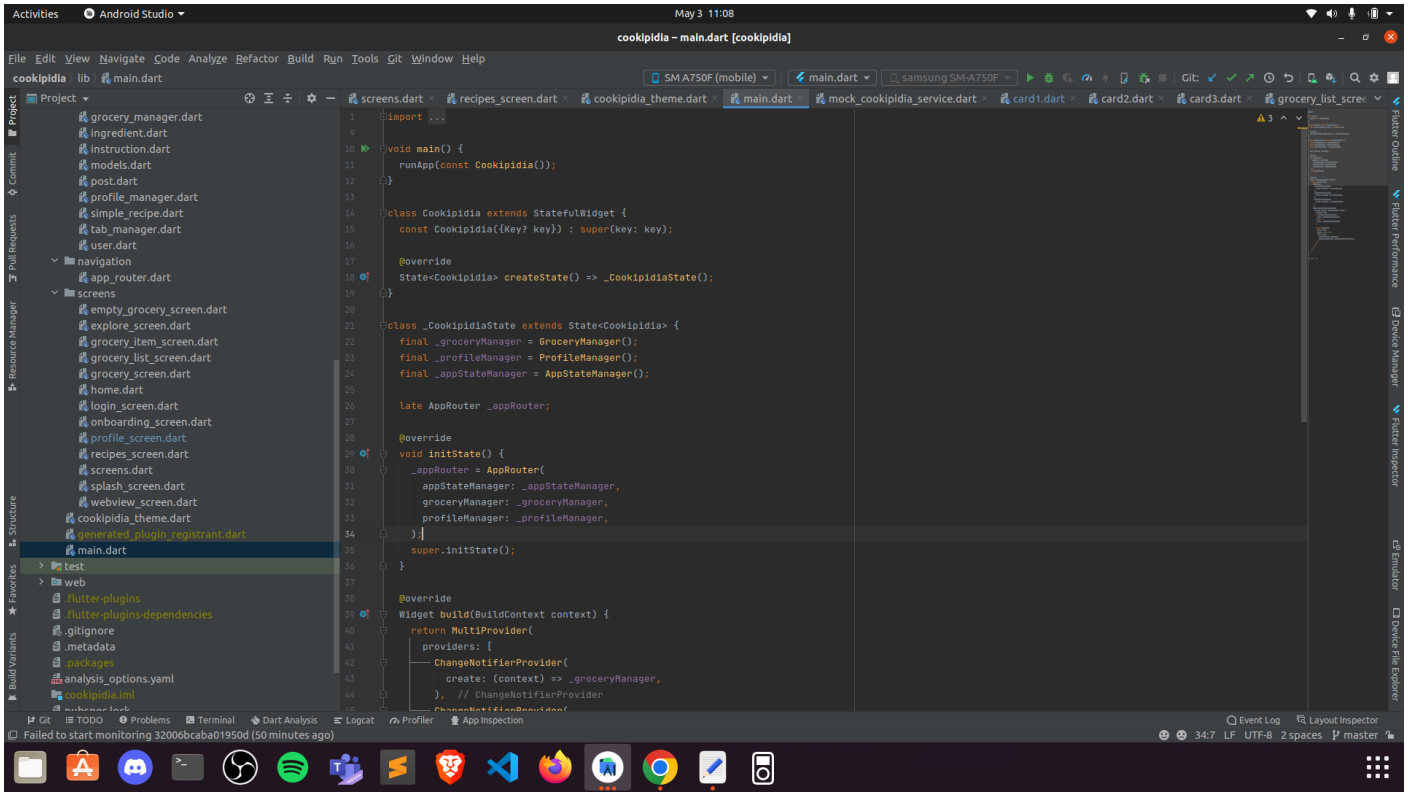


Fig. 106

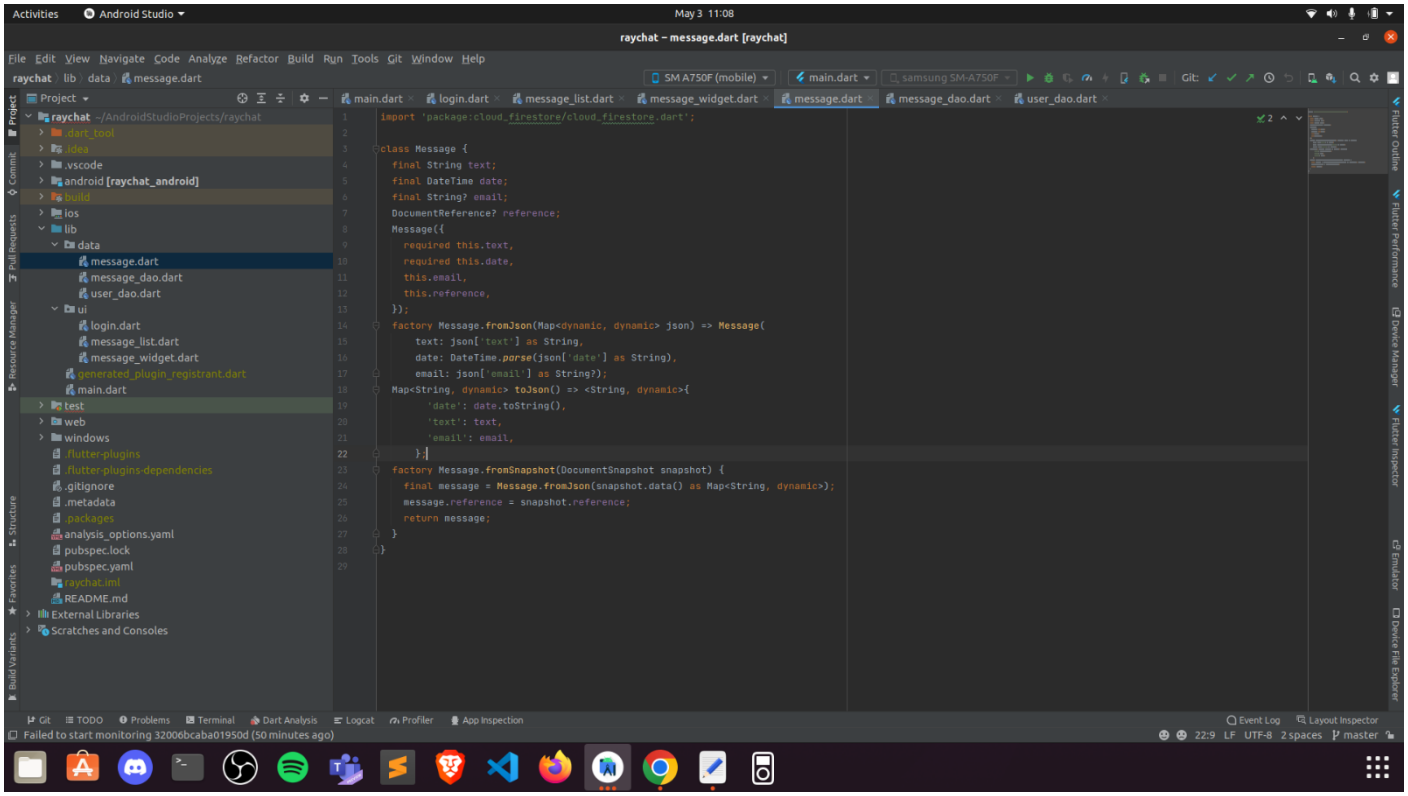


Fig. 107

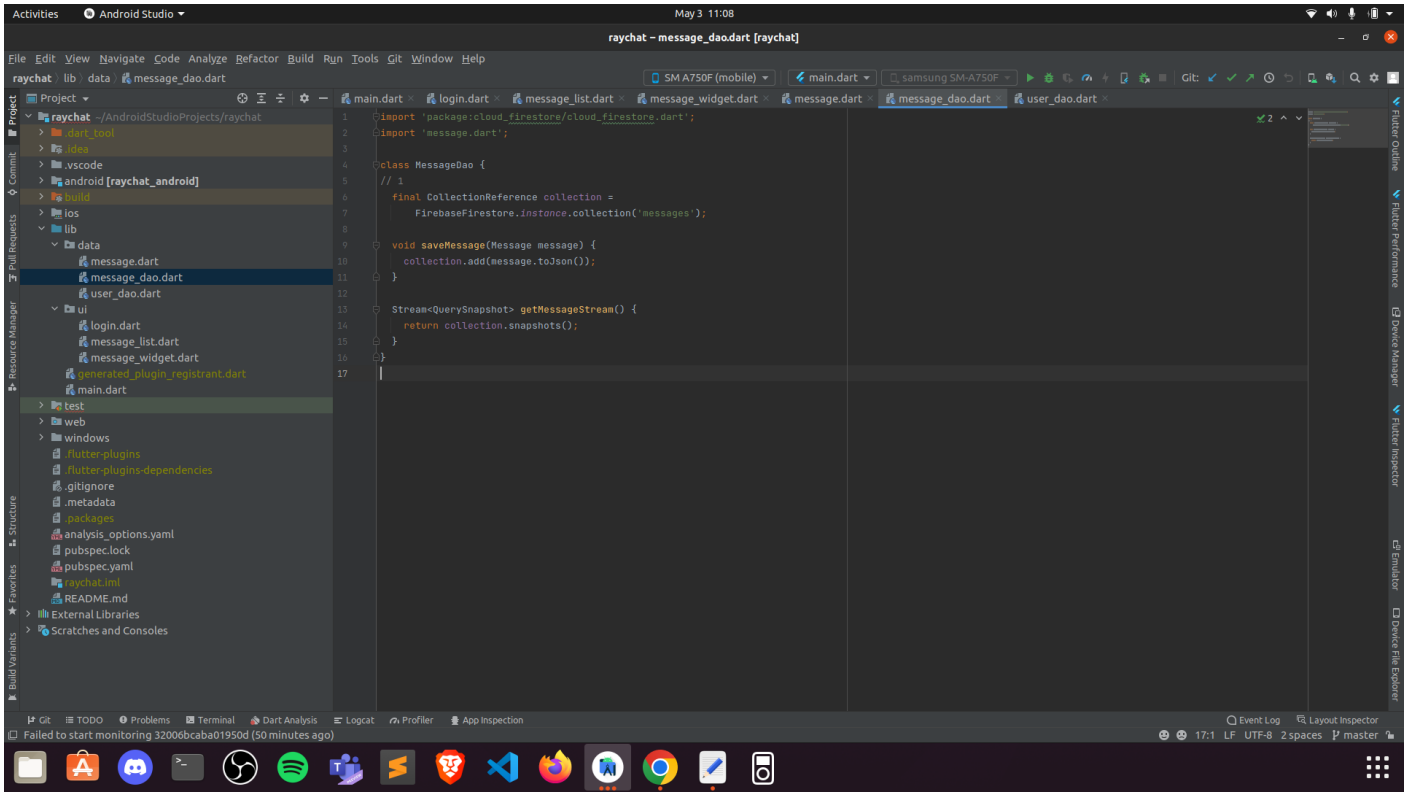


Fig. 108

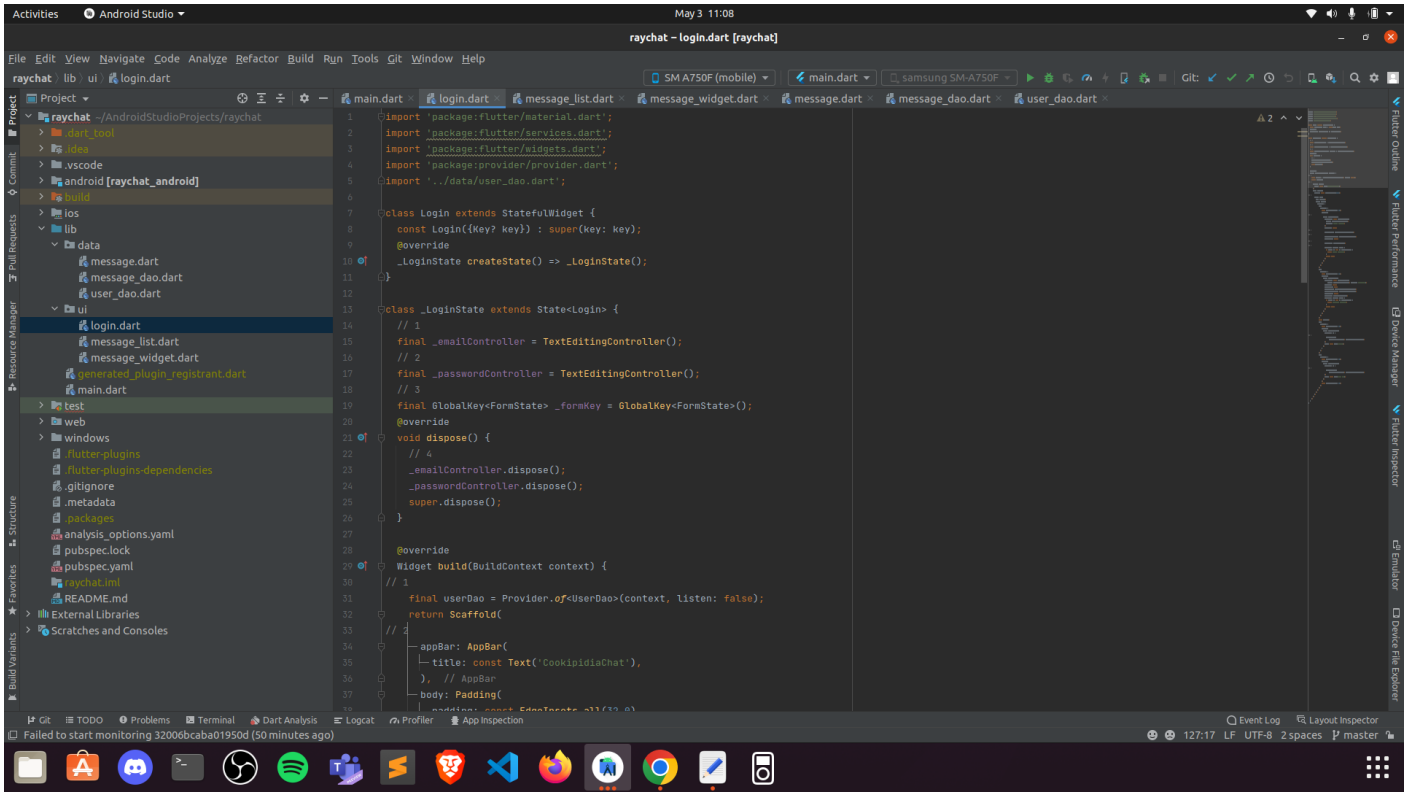


Fig. 109

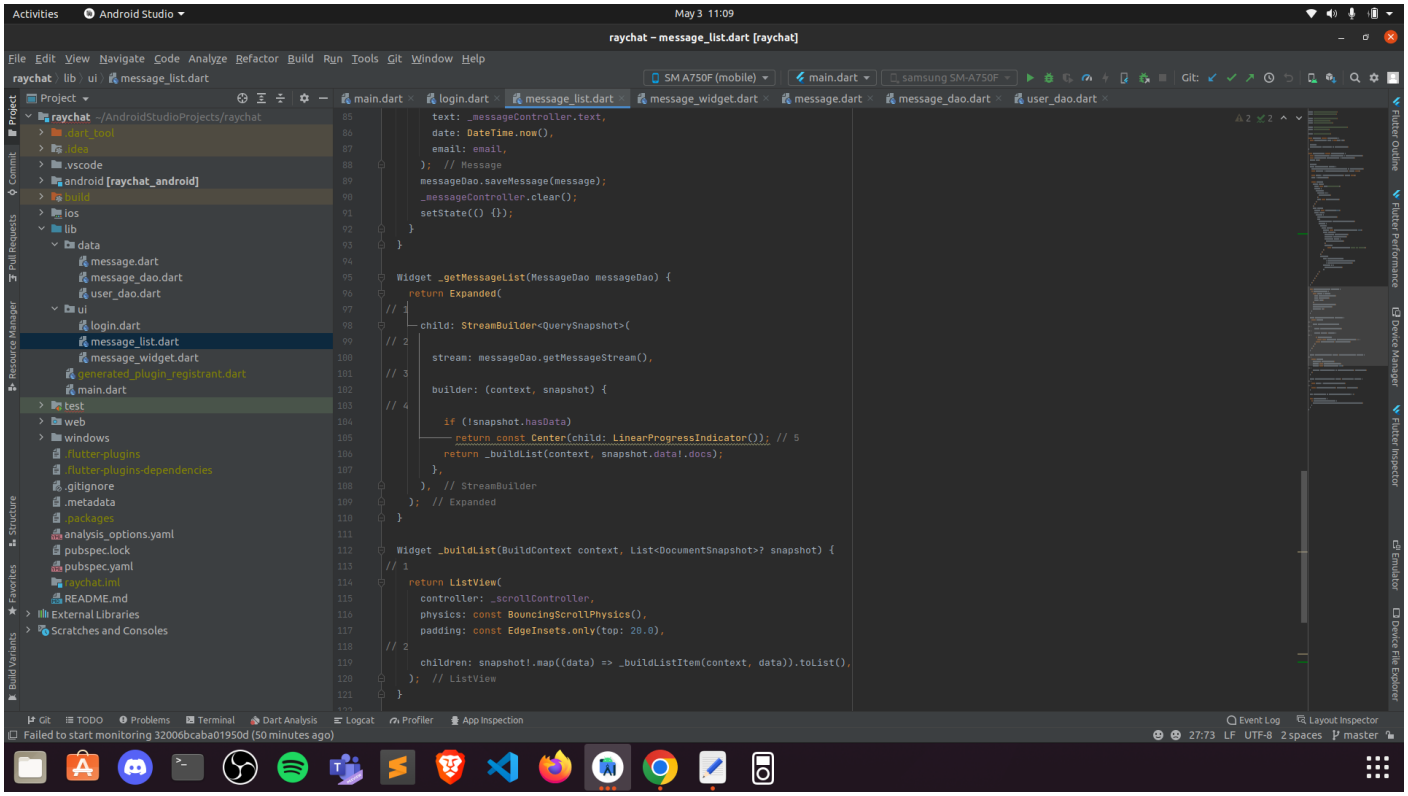


Fig. 110

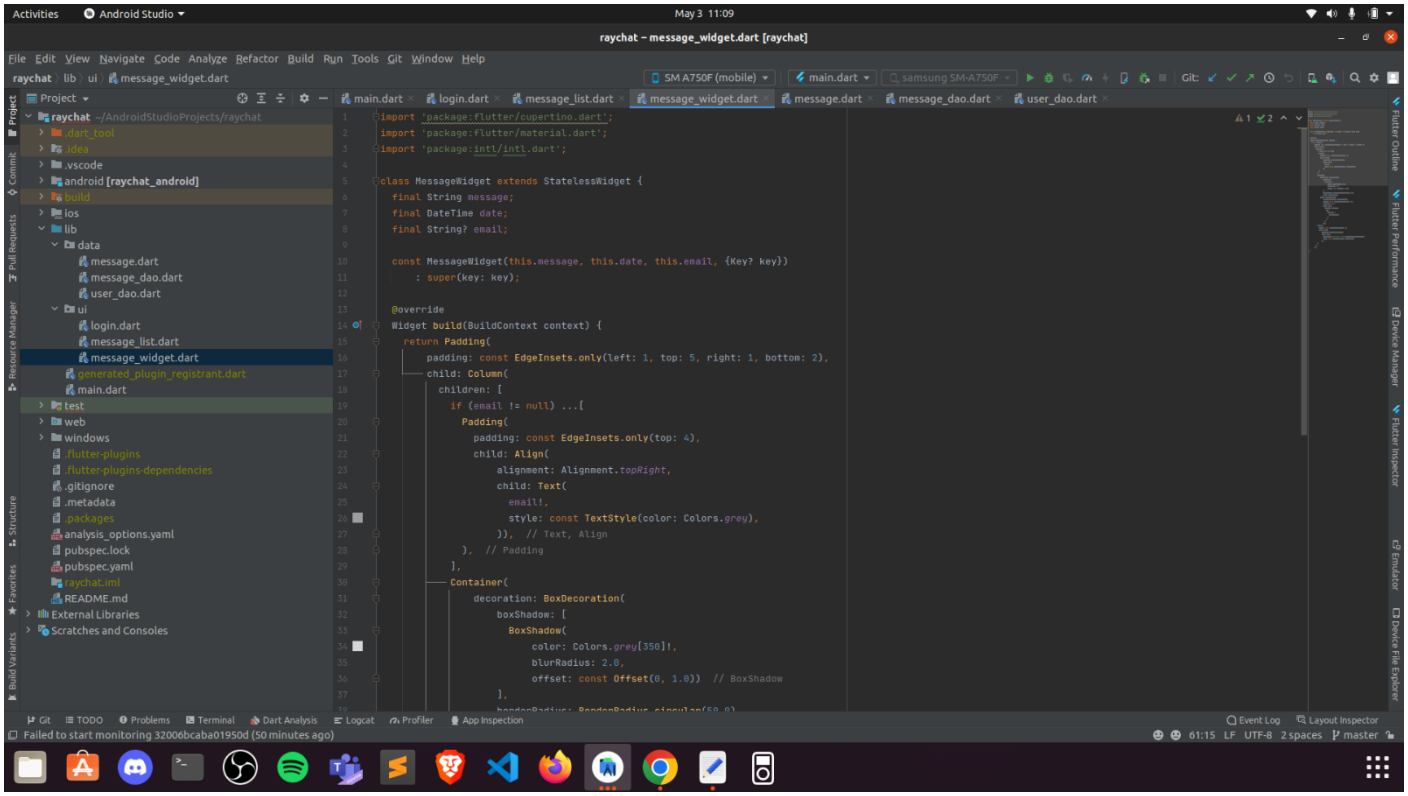


Fig. 111

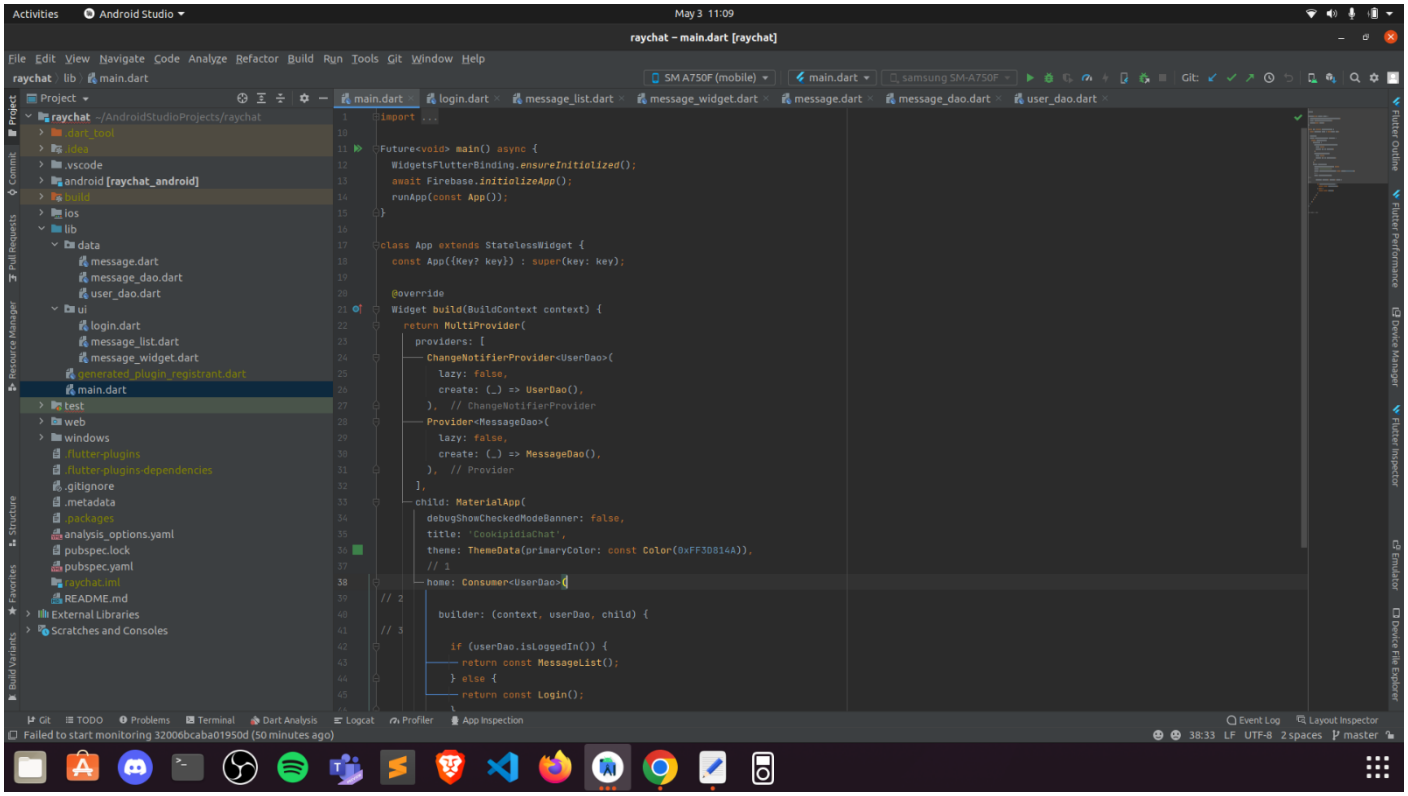


Fig. 112

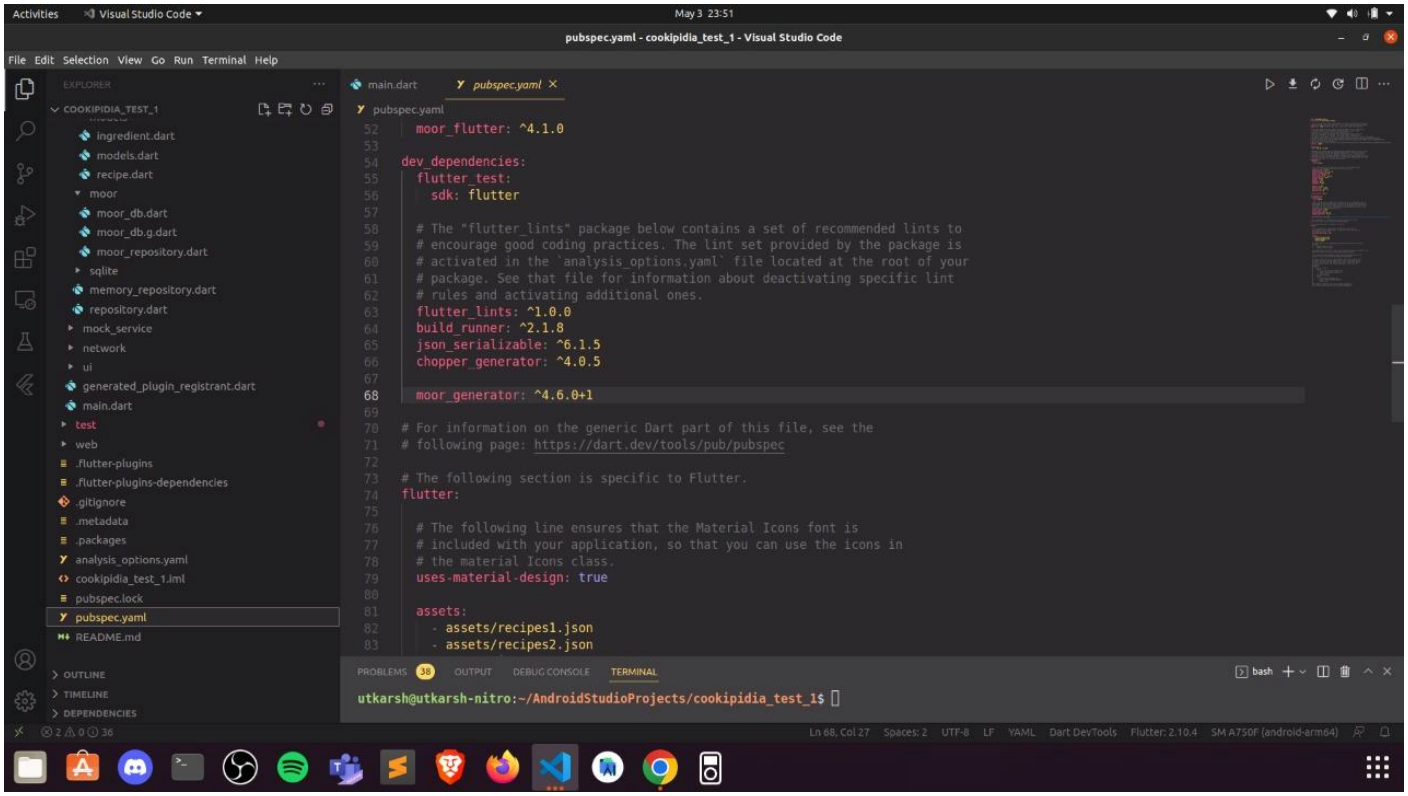


Fig. 113

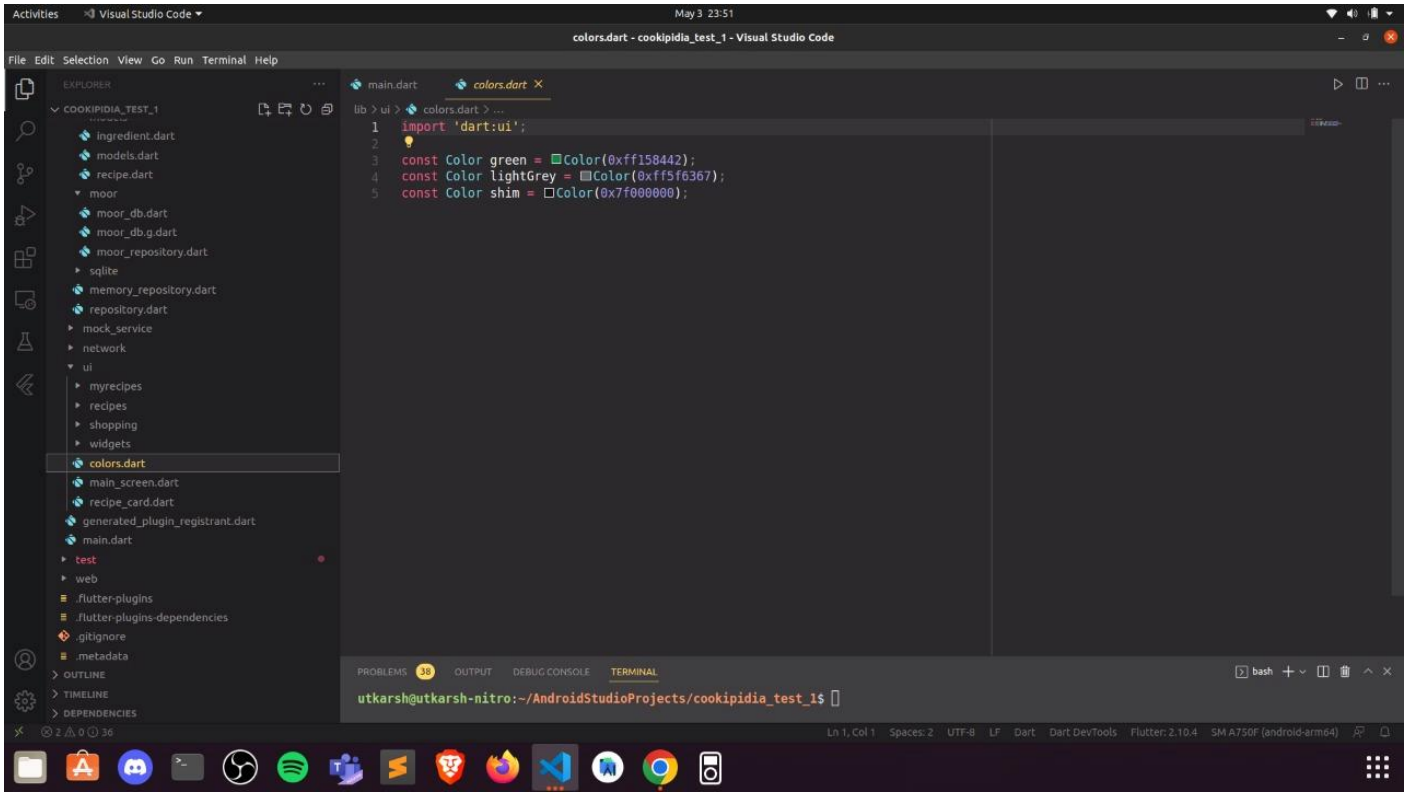


Fig. 114

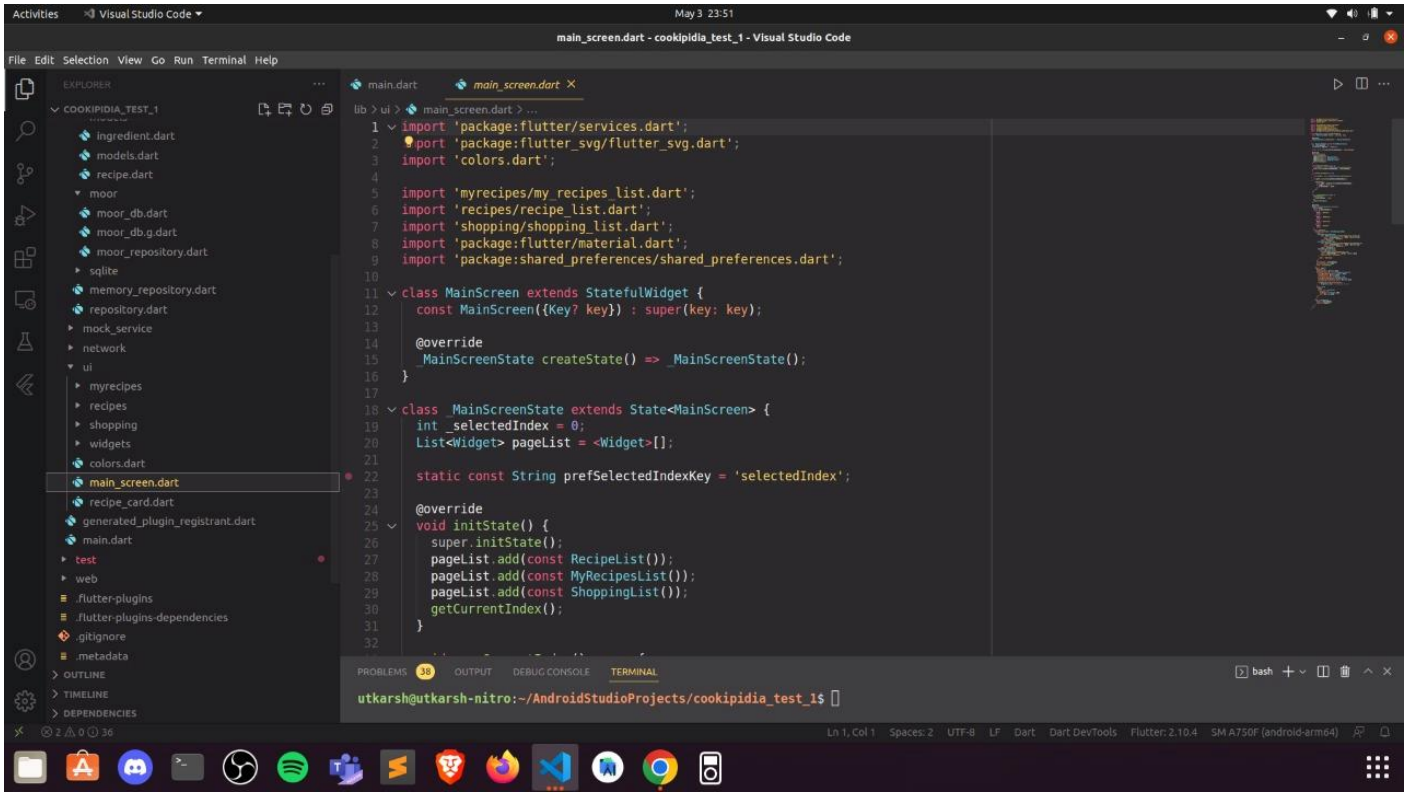


Fig. 115

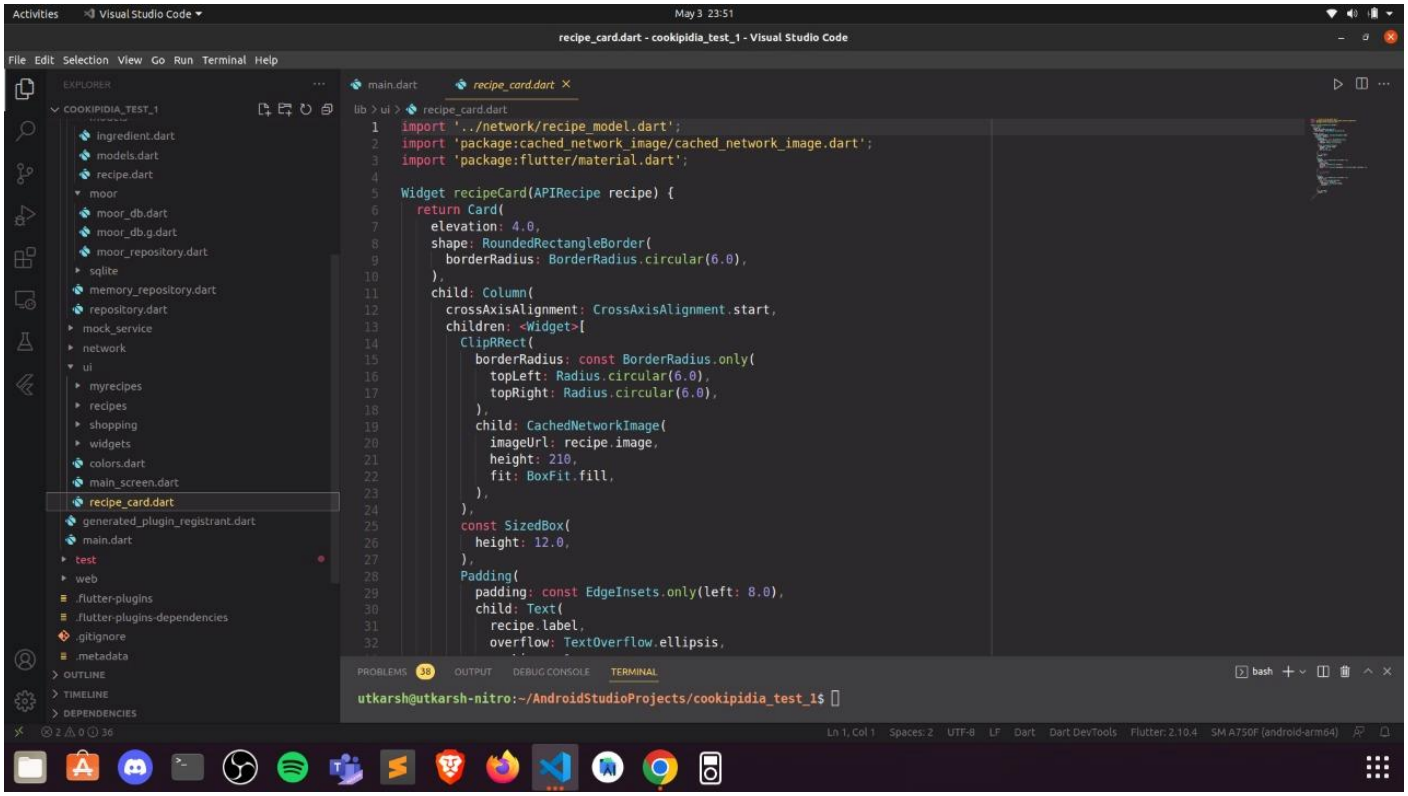


Fig. 116

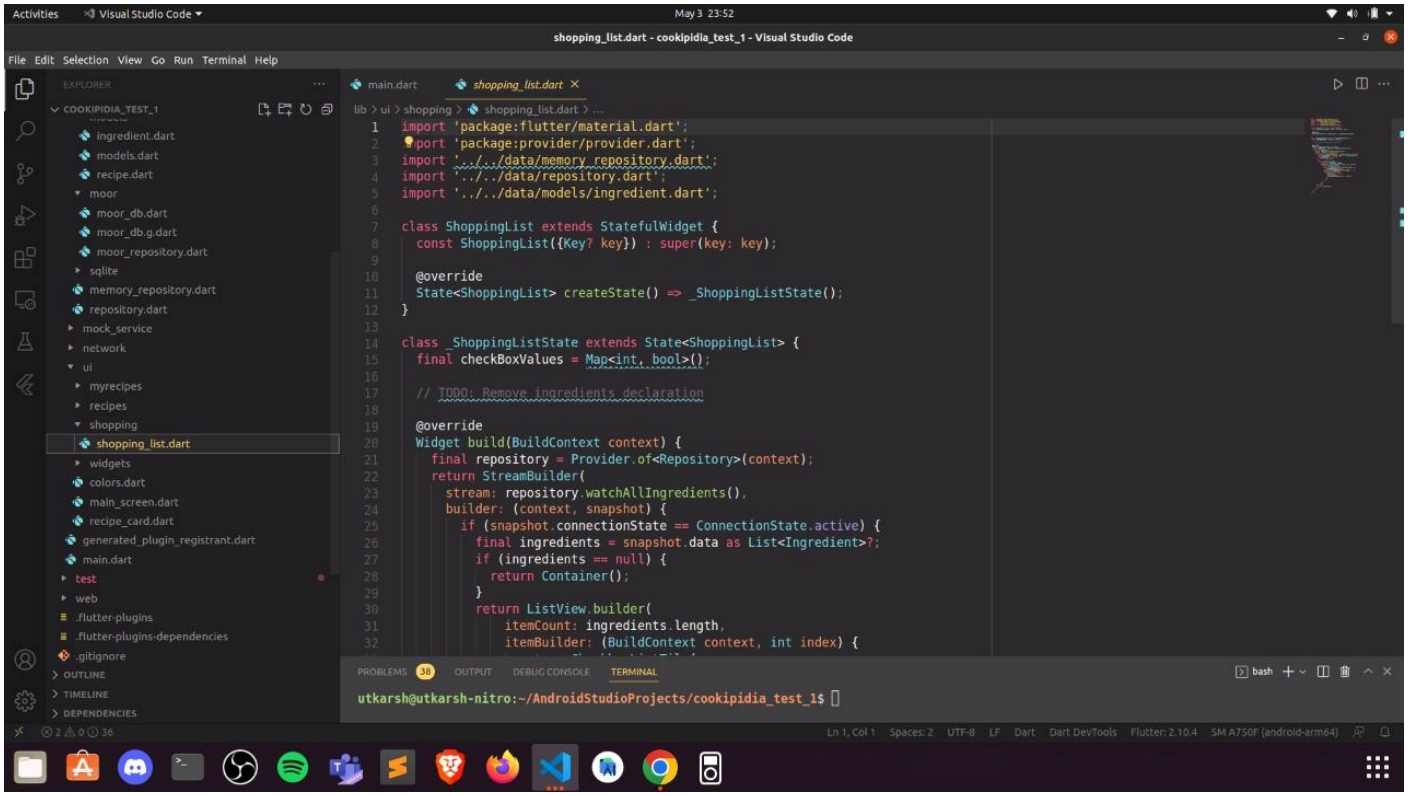


Fig. 117

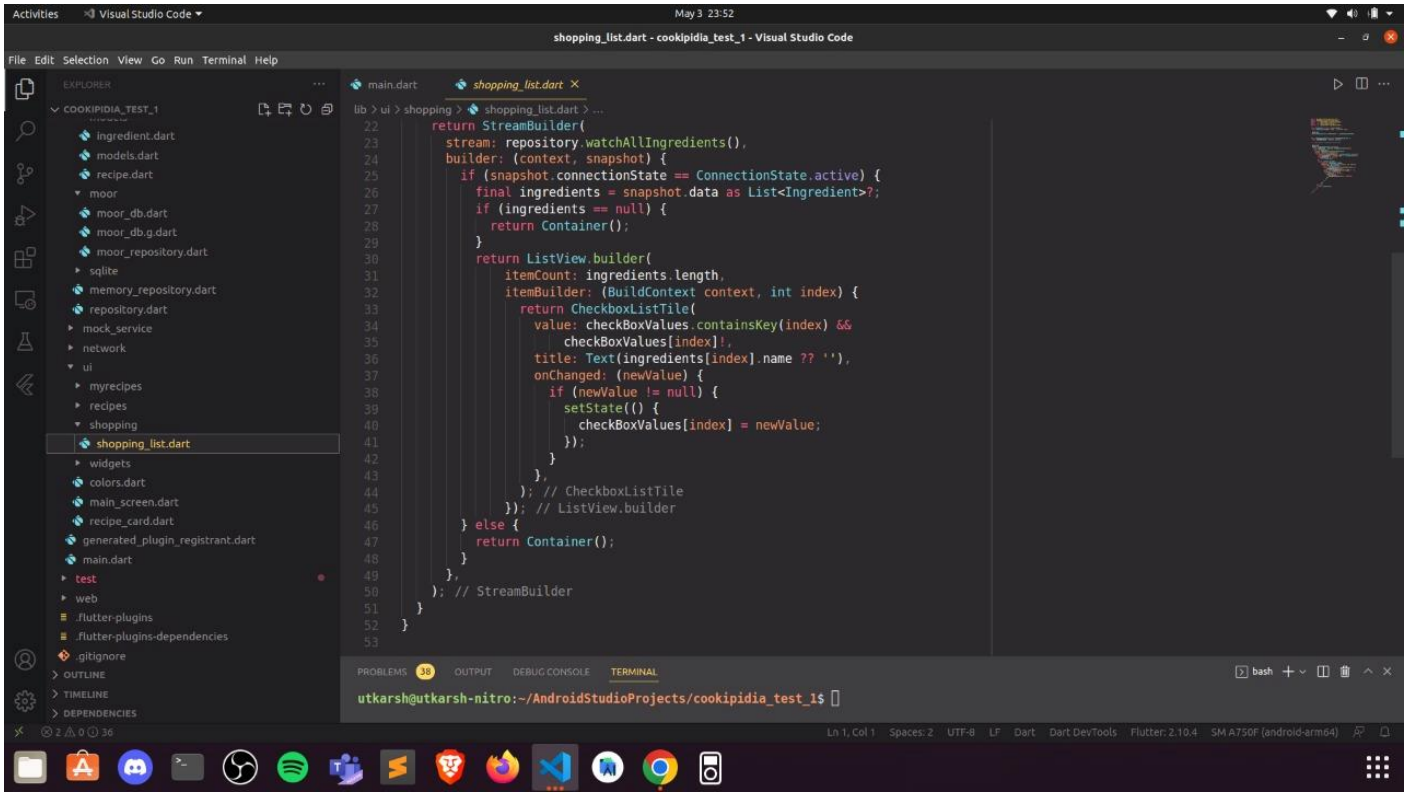


Fig. 118

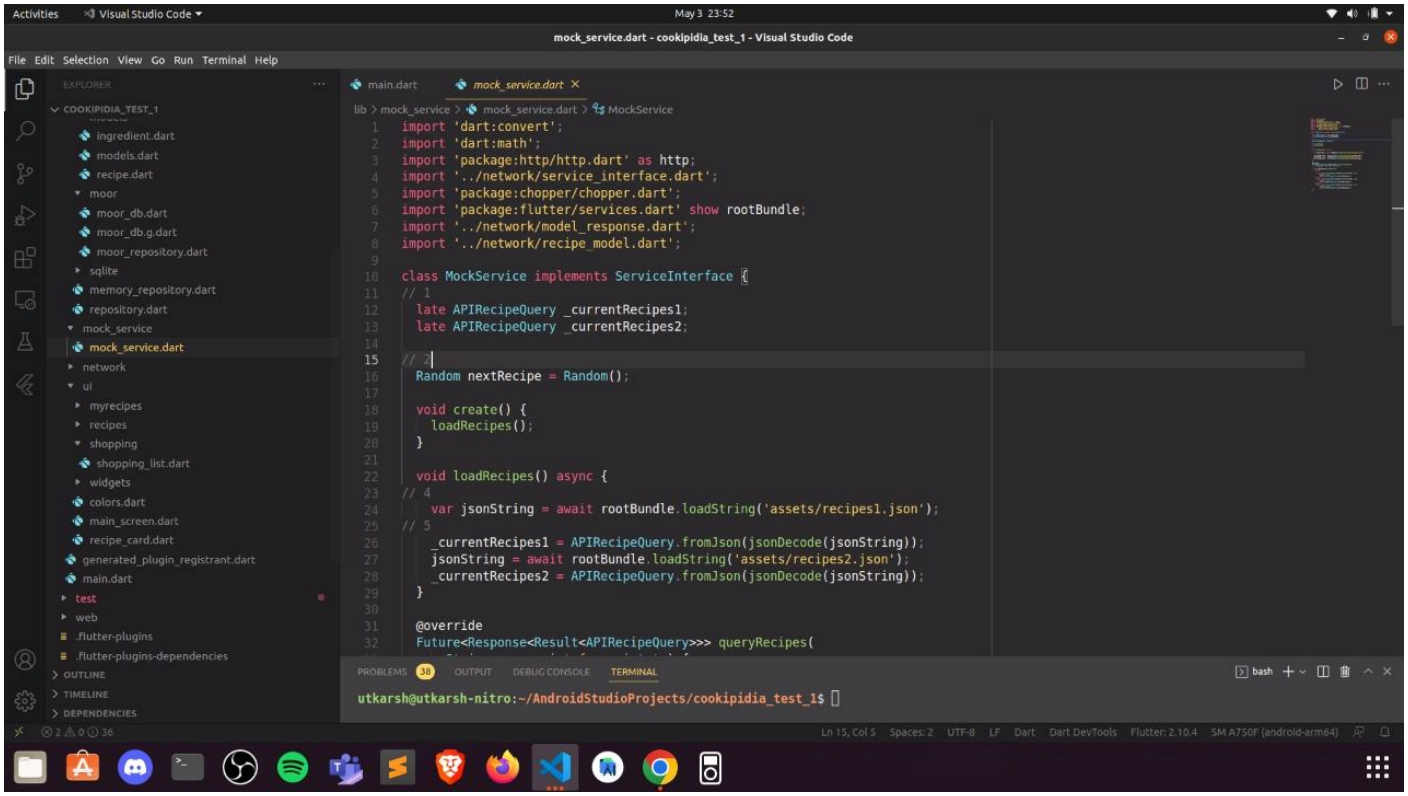


Fig. 119

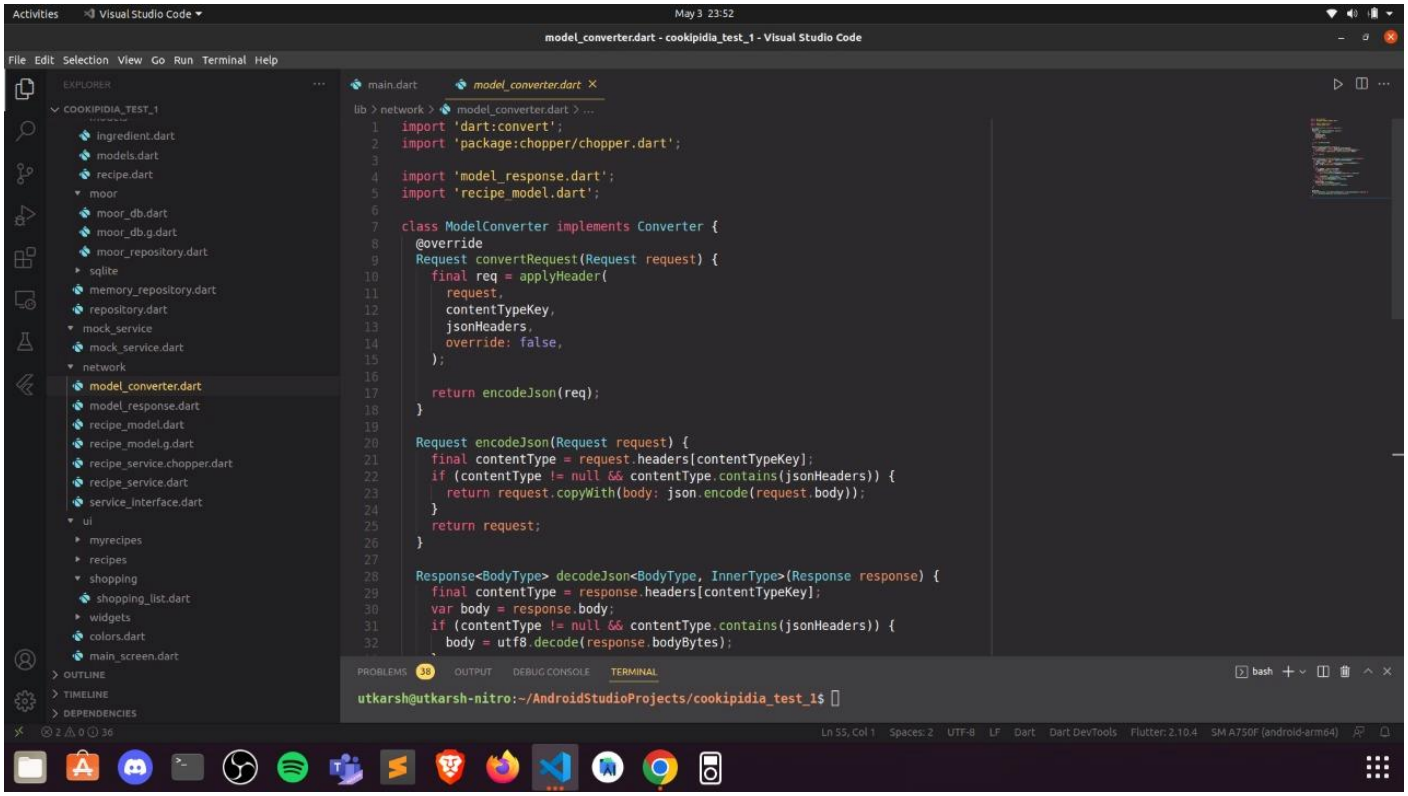


Fig. 120

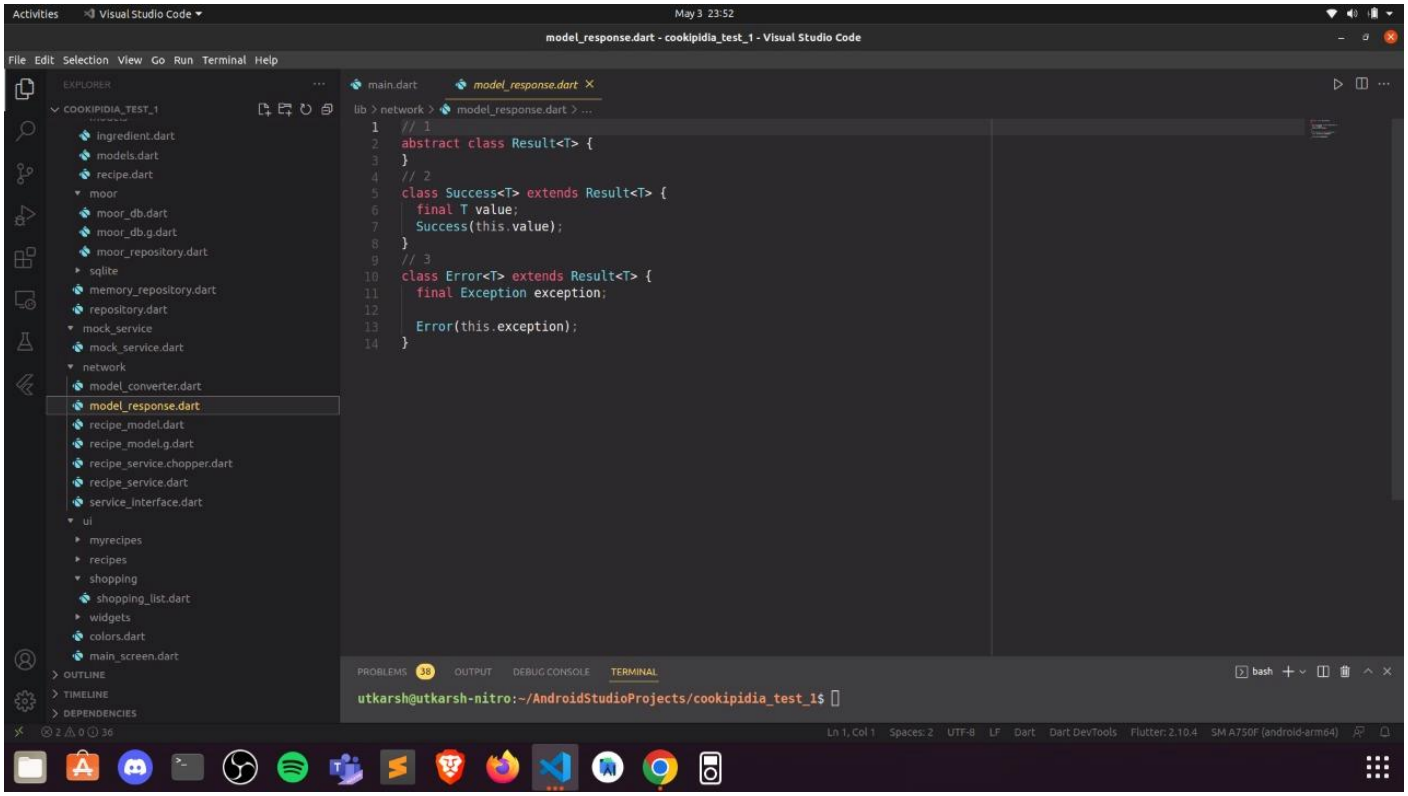


Fig. 121

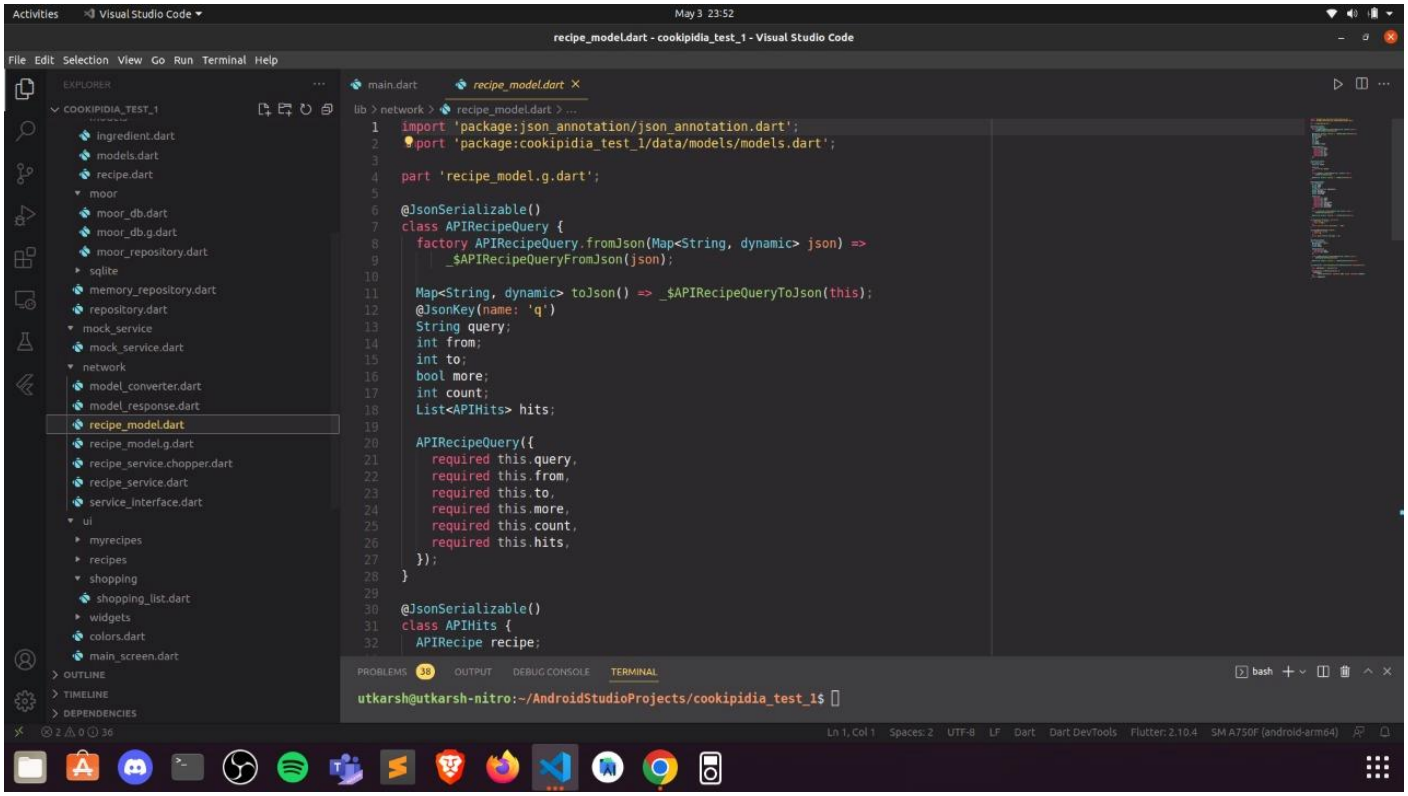


Fig. 122

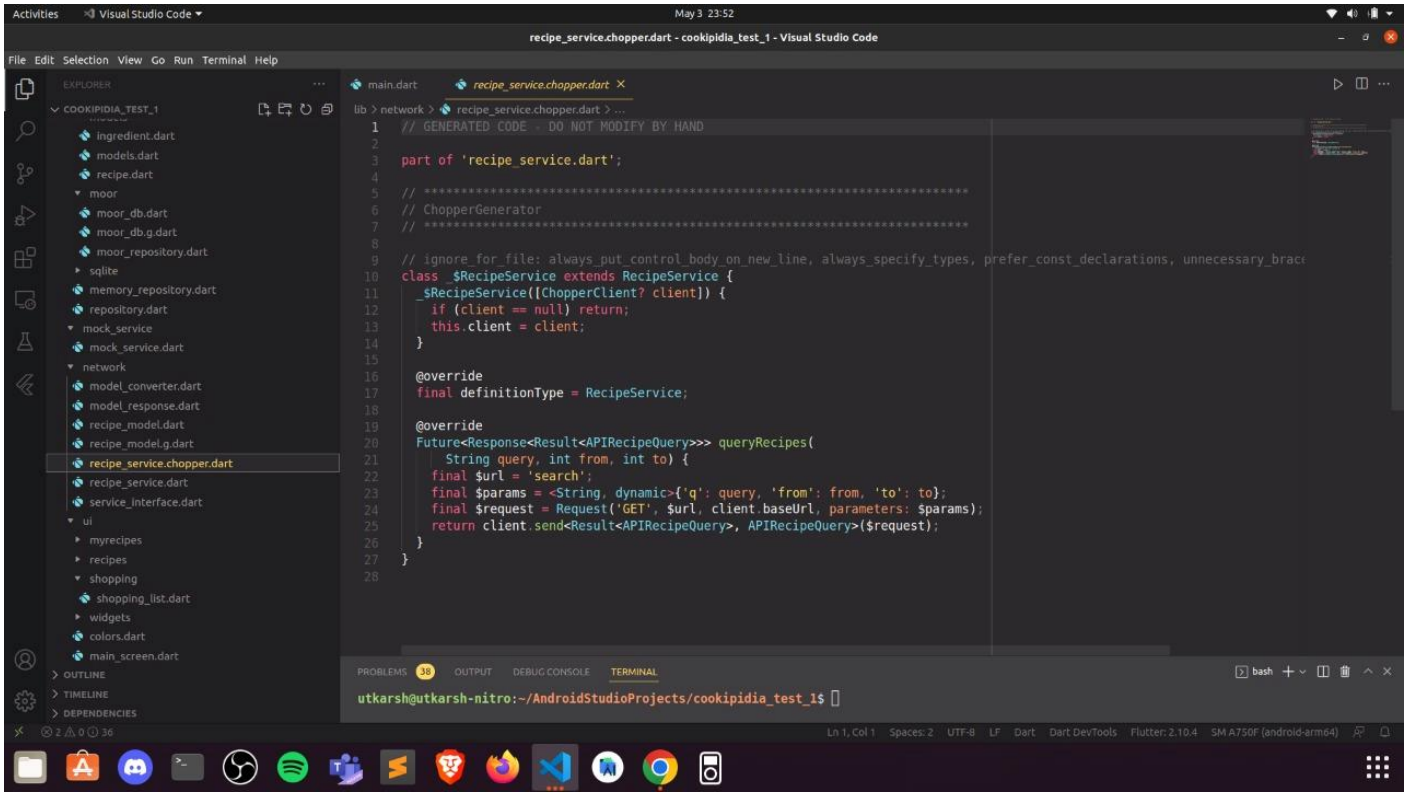


Fig. 123

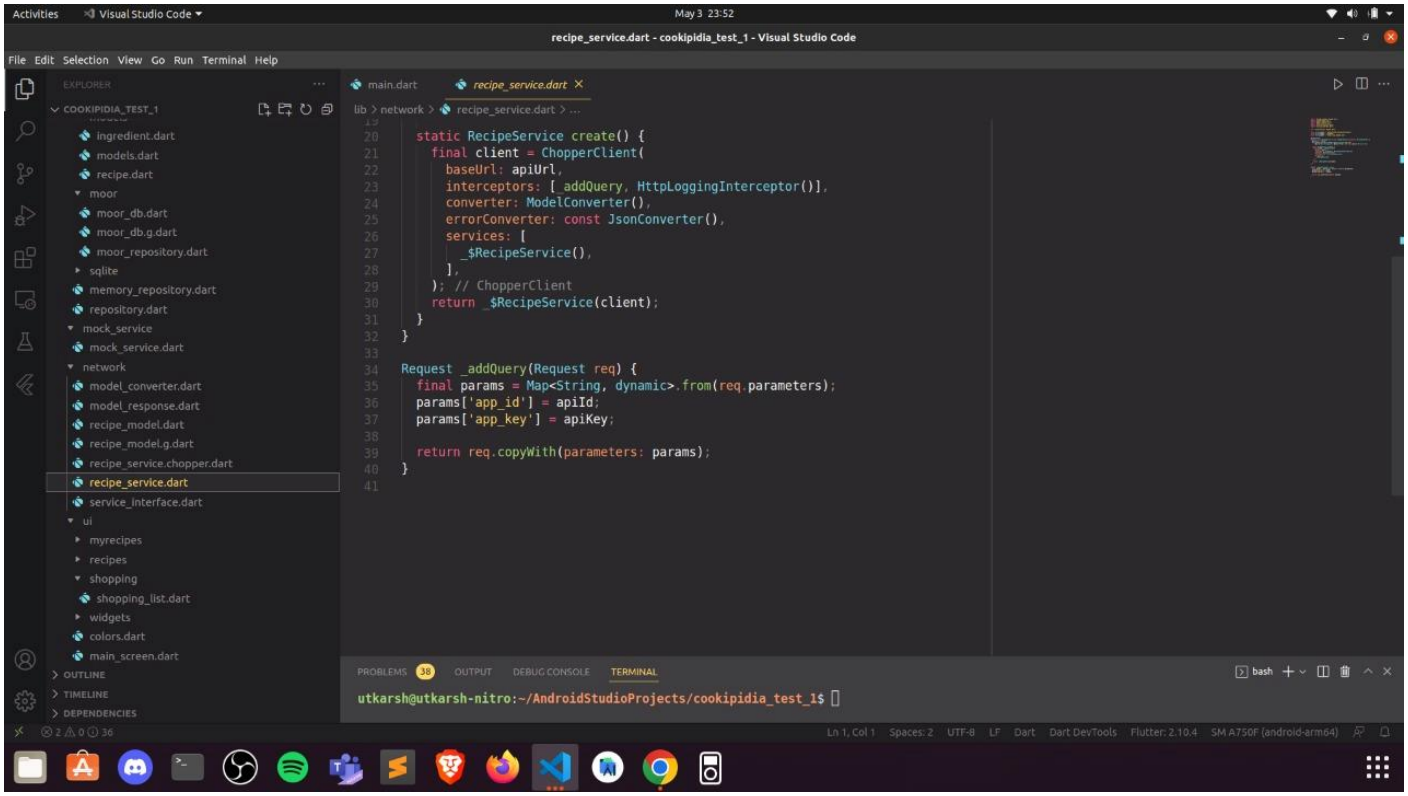


Fig. 124

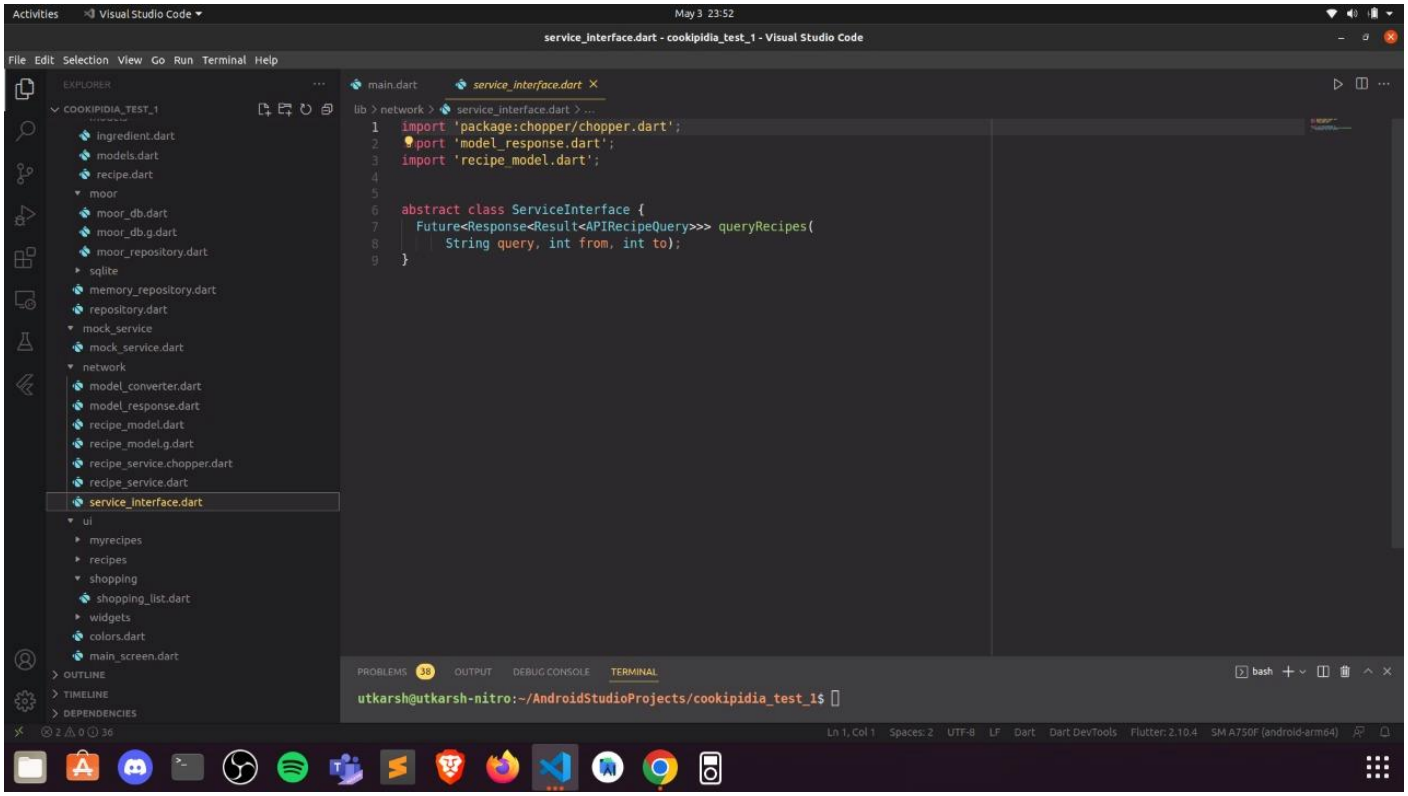


Fig. 125

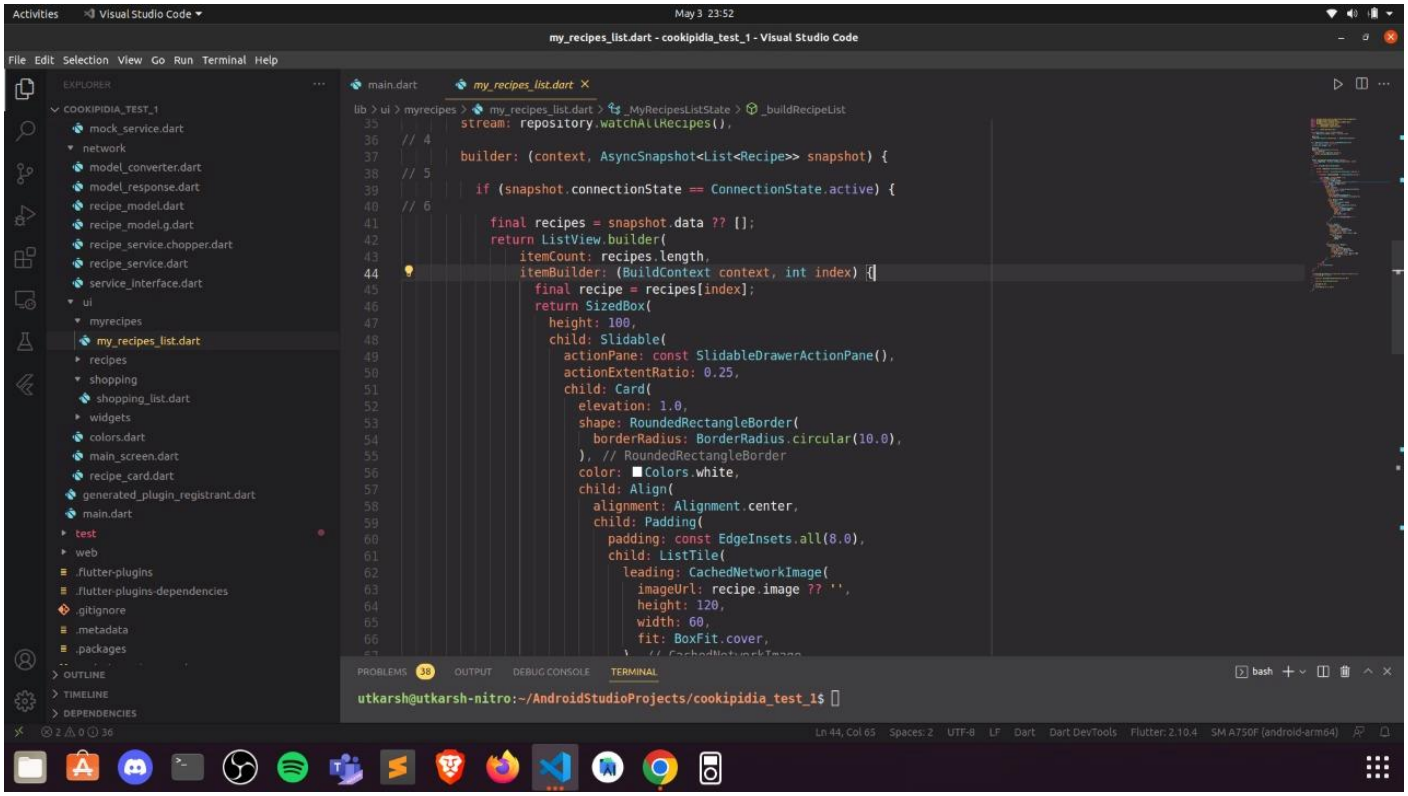


Fig. 126

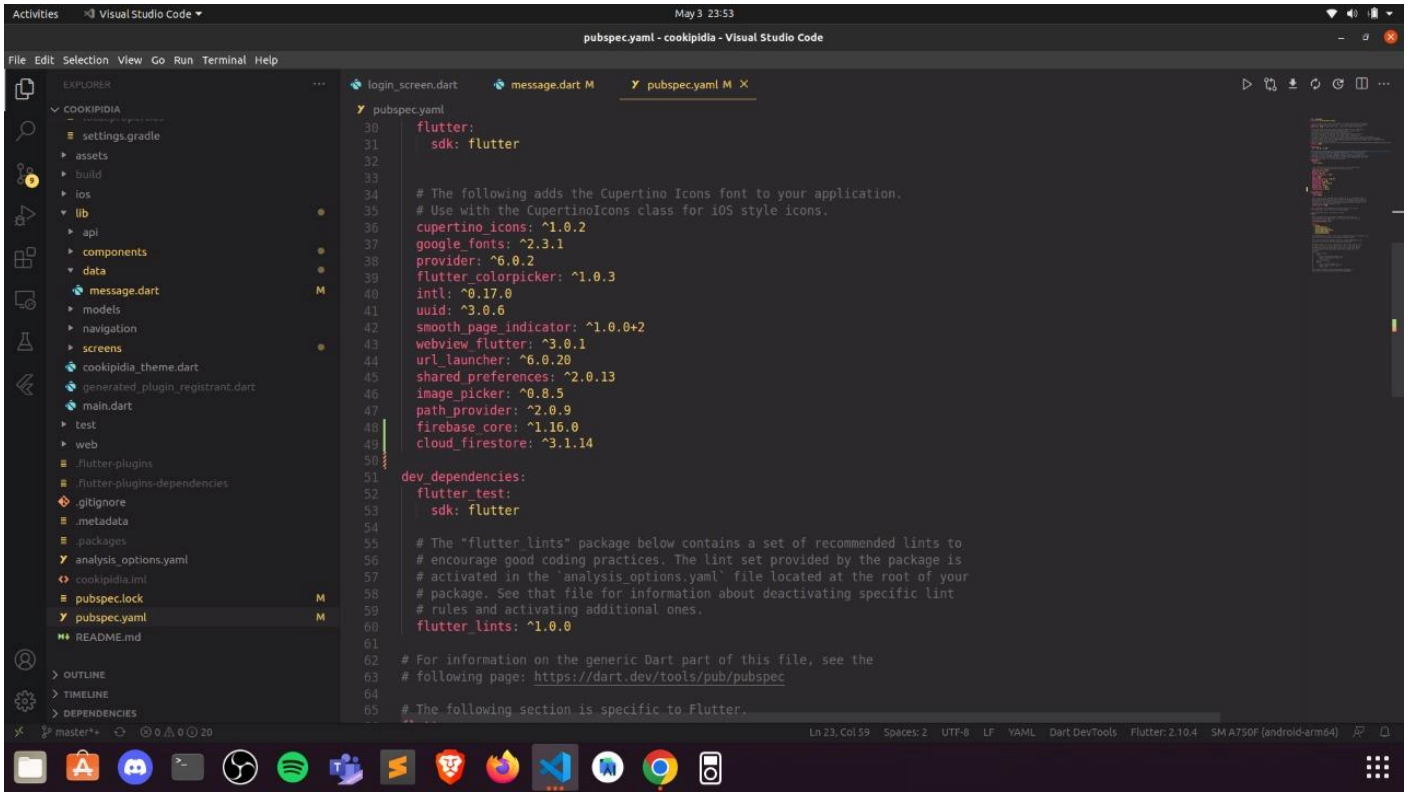


Fig. 127

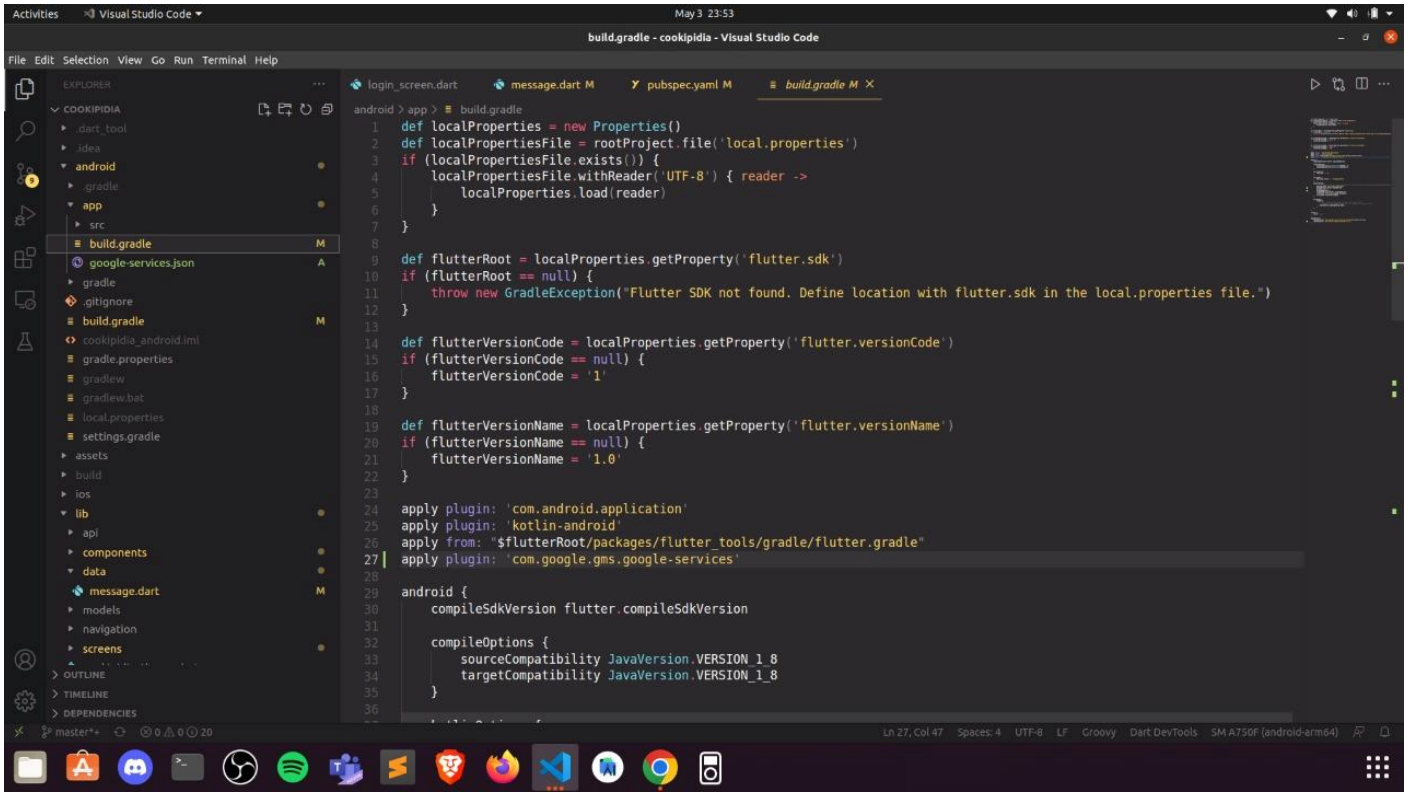


Fig. 128

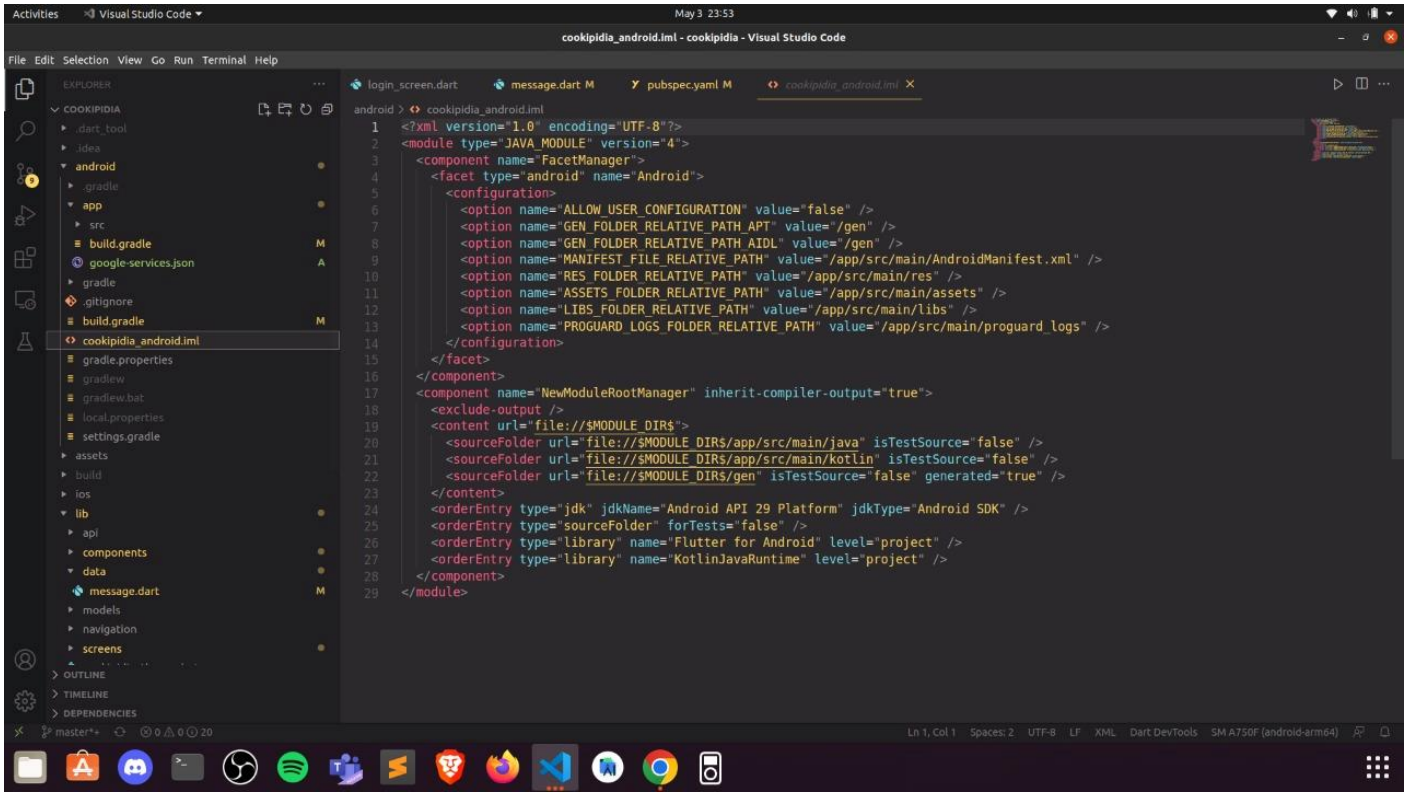


Fig. 129

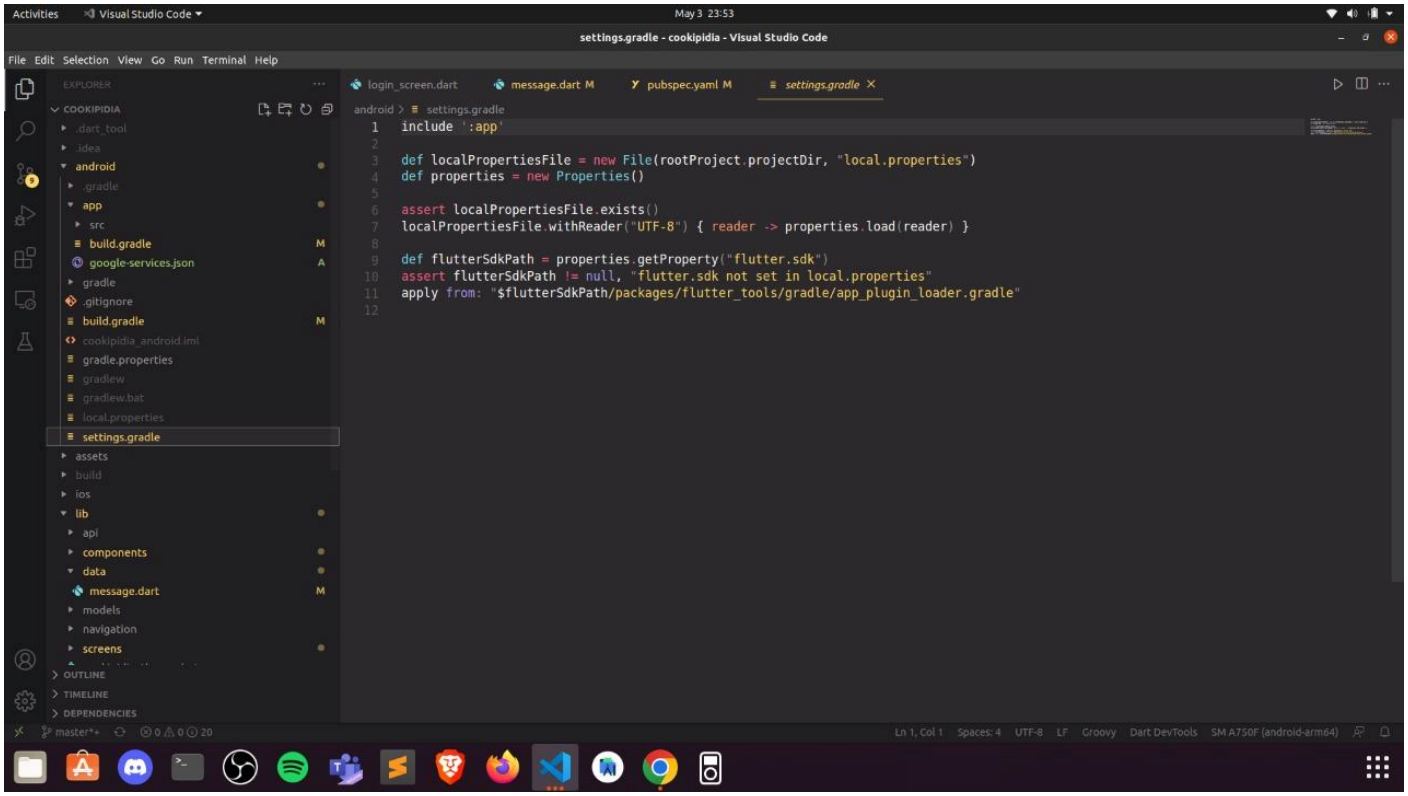


Fig. 130

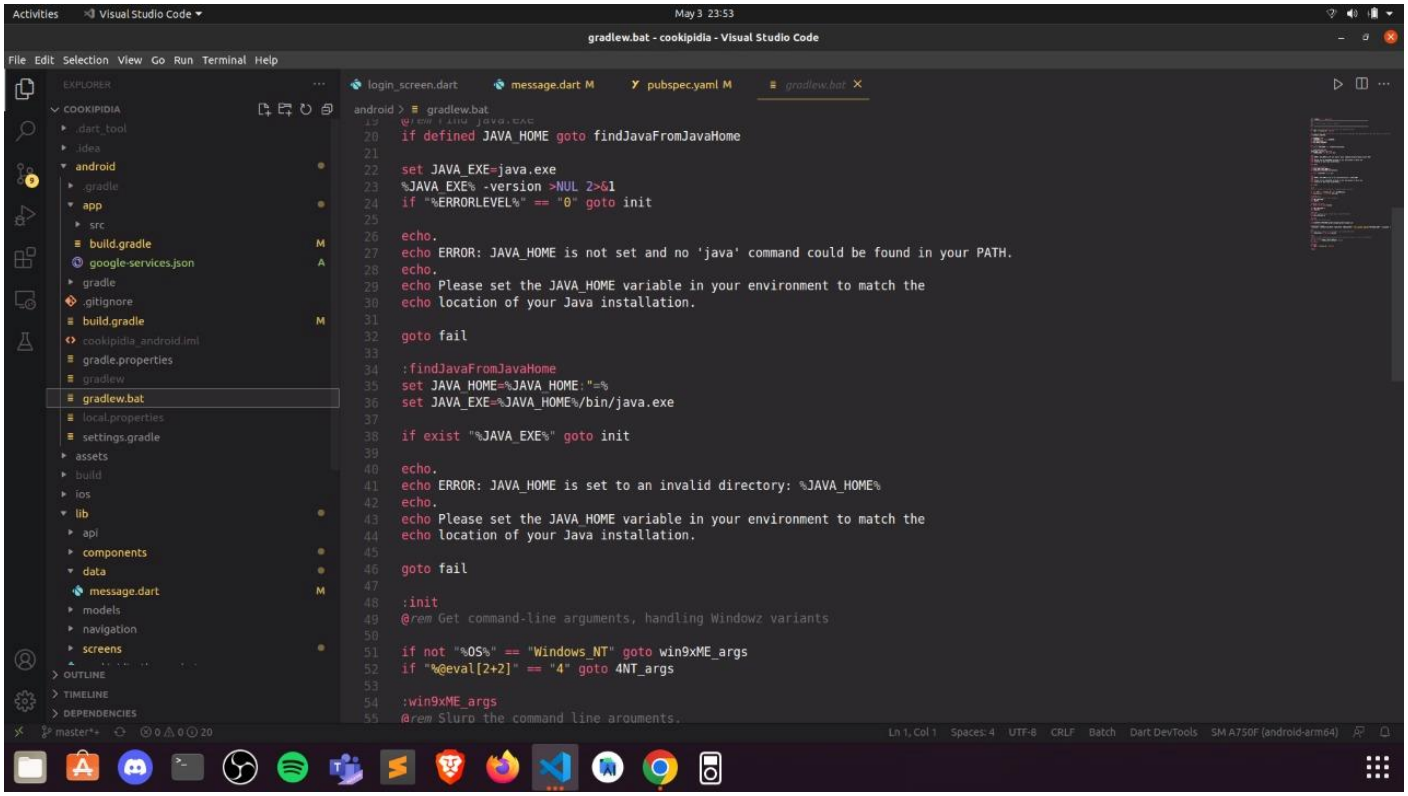


Fig. 131

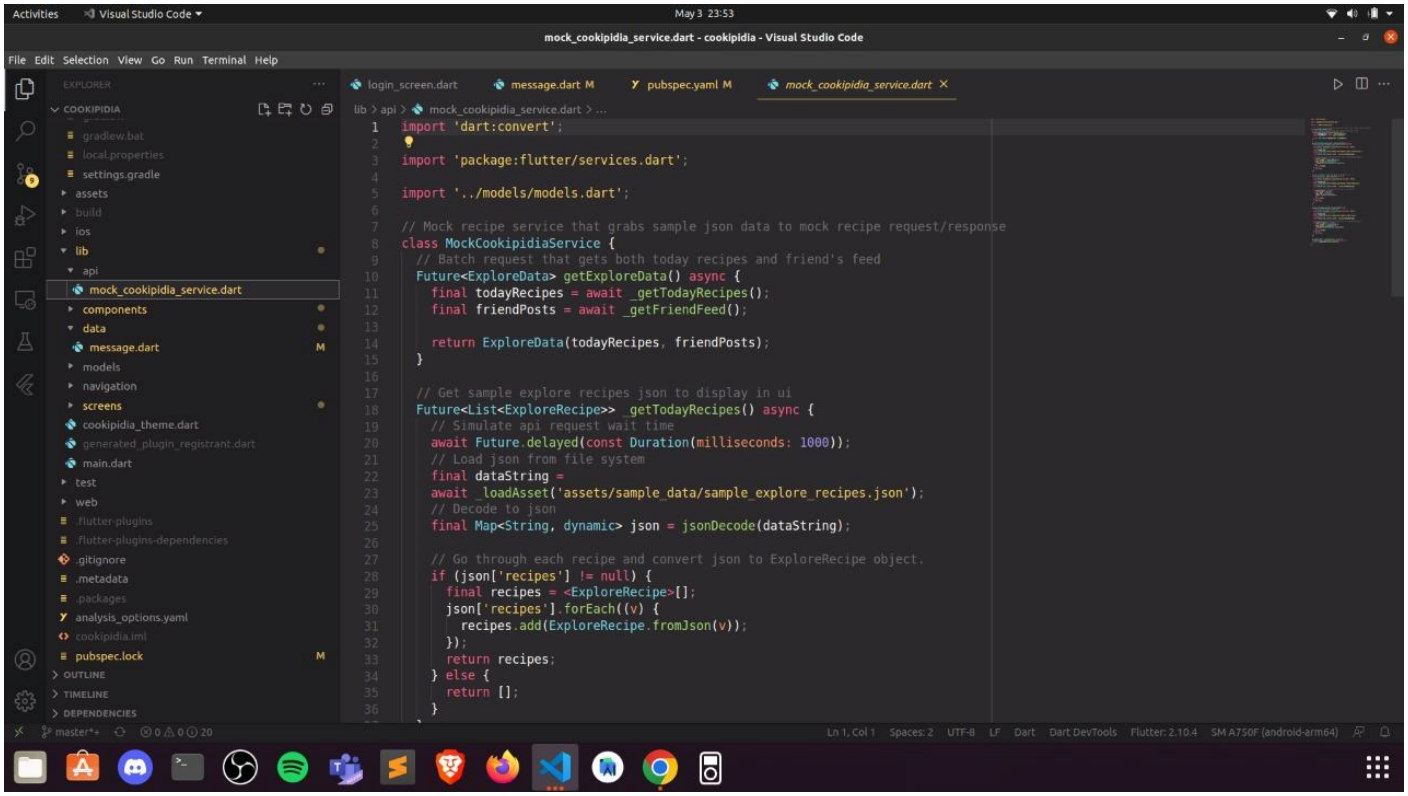


Fig. 132

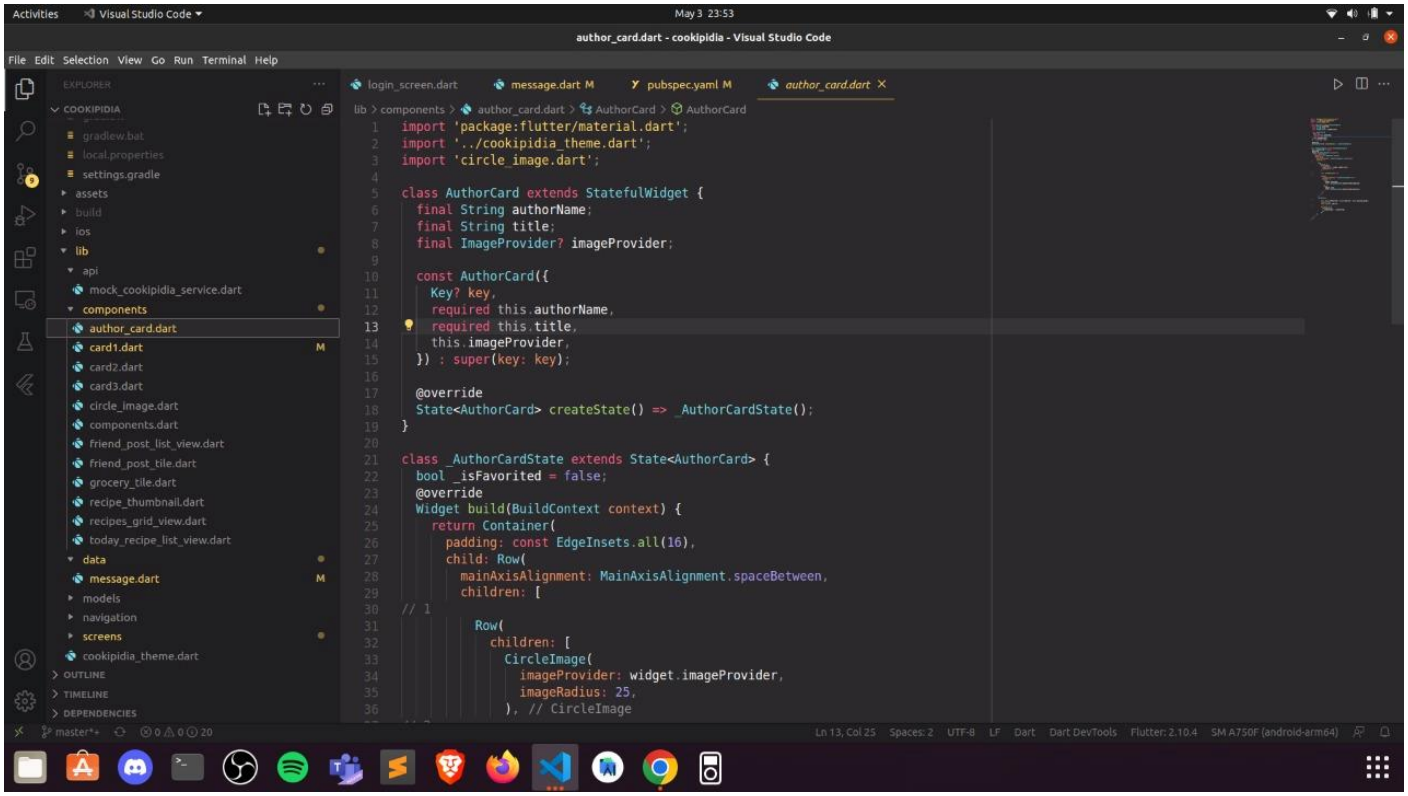


Fig. 133

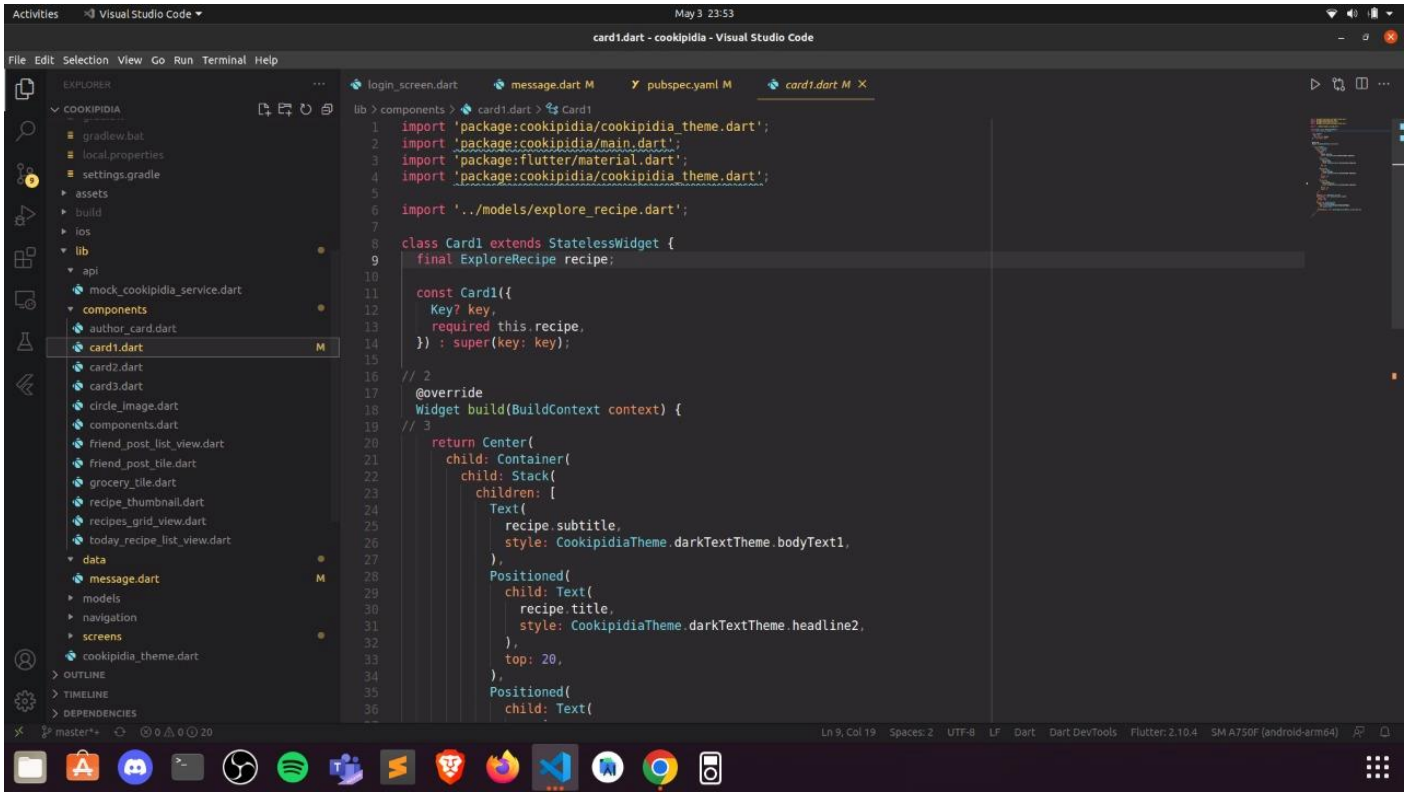


Fig. 134

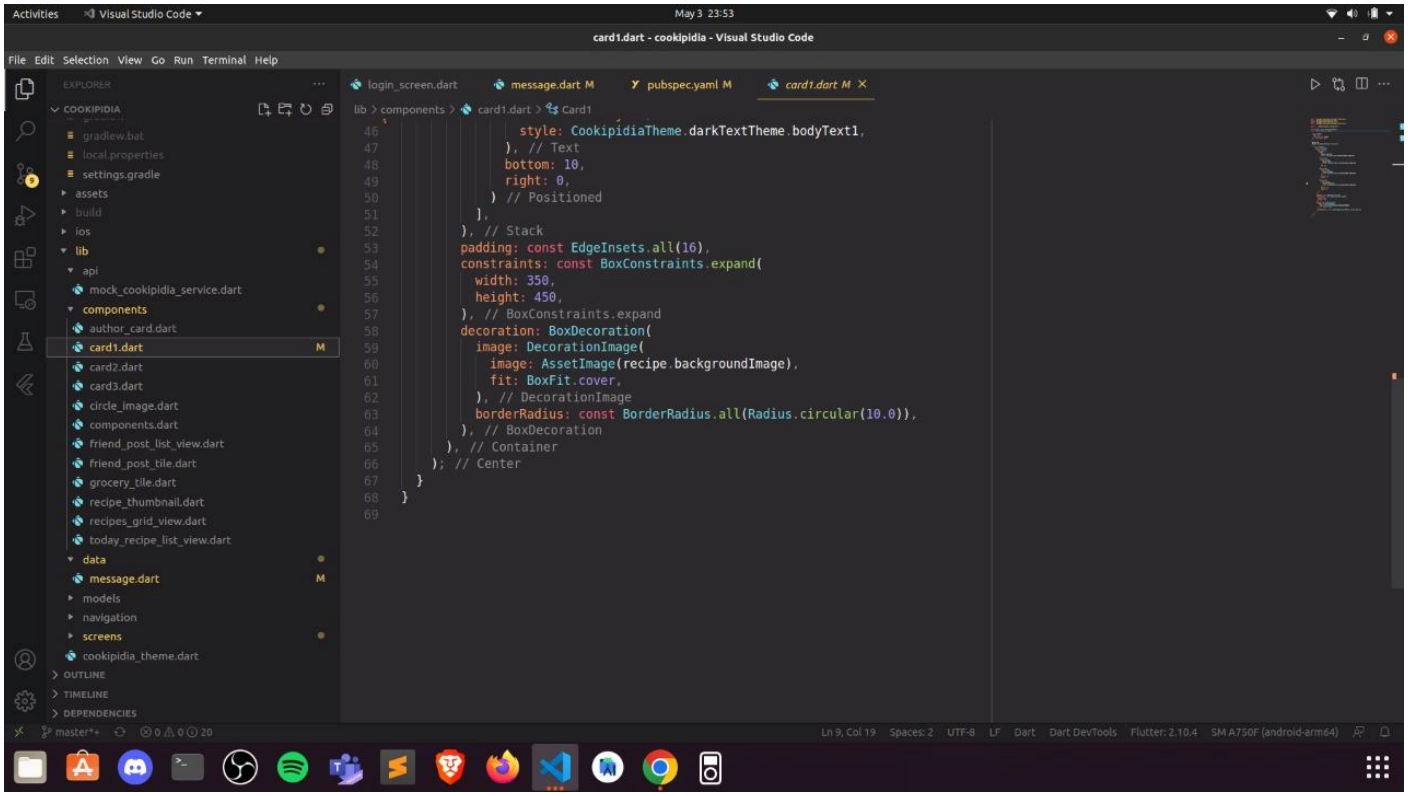


Fig. 135

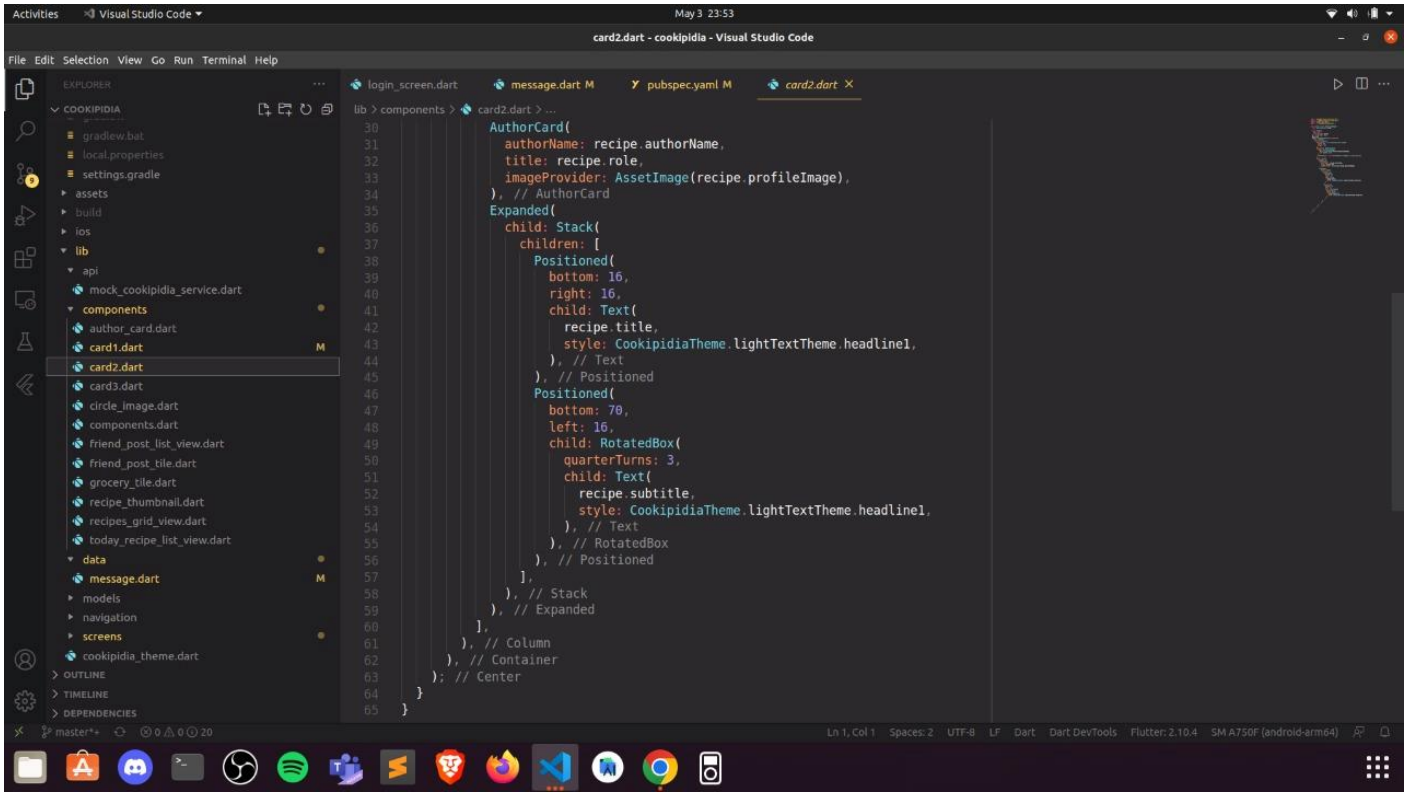


Fig. 136

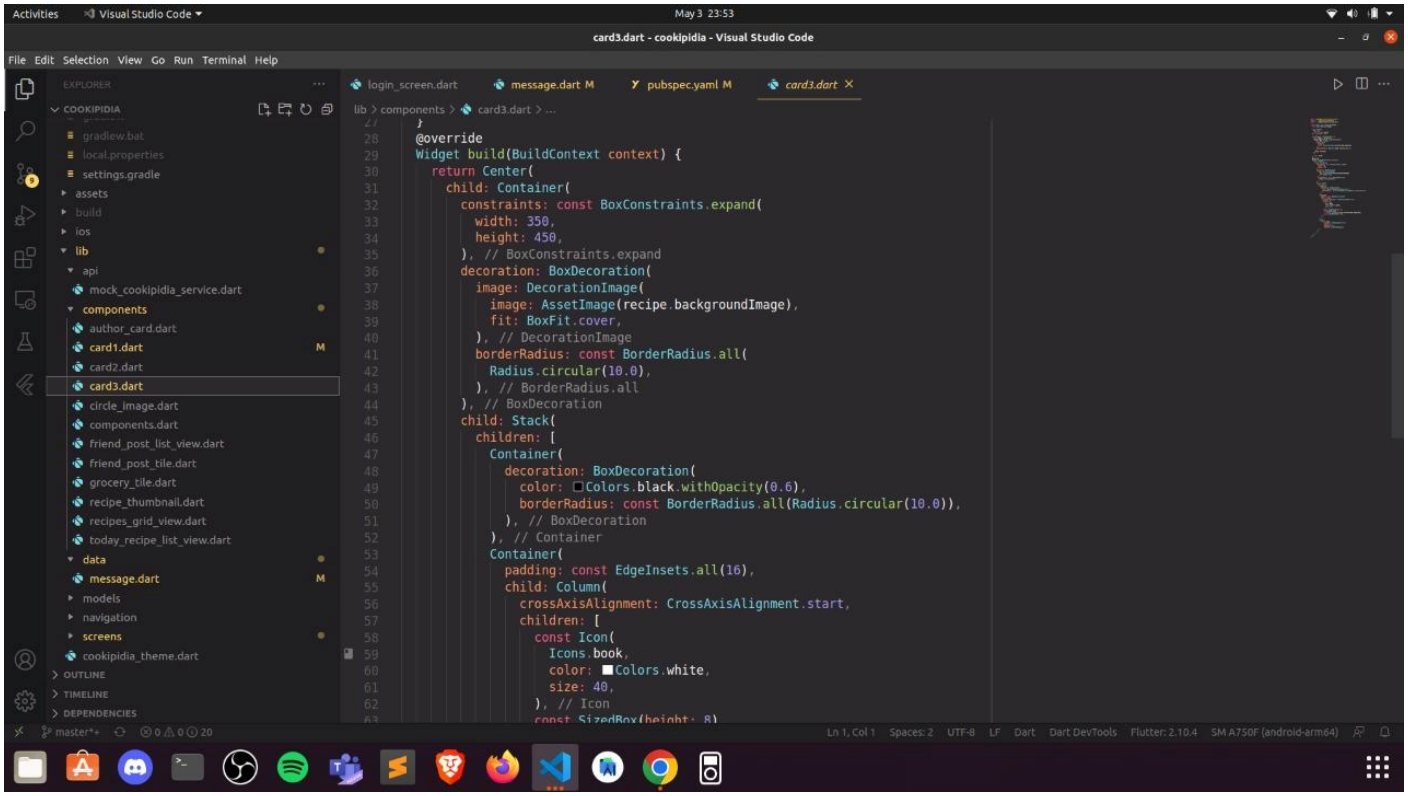


Fig. 137

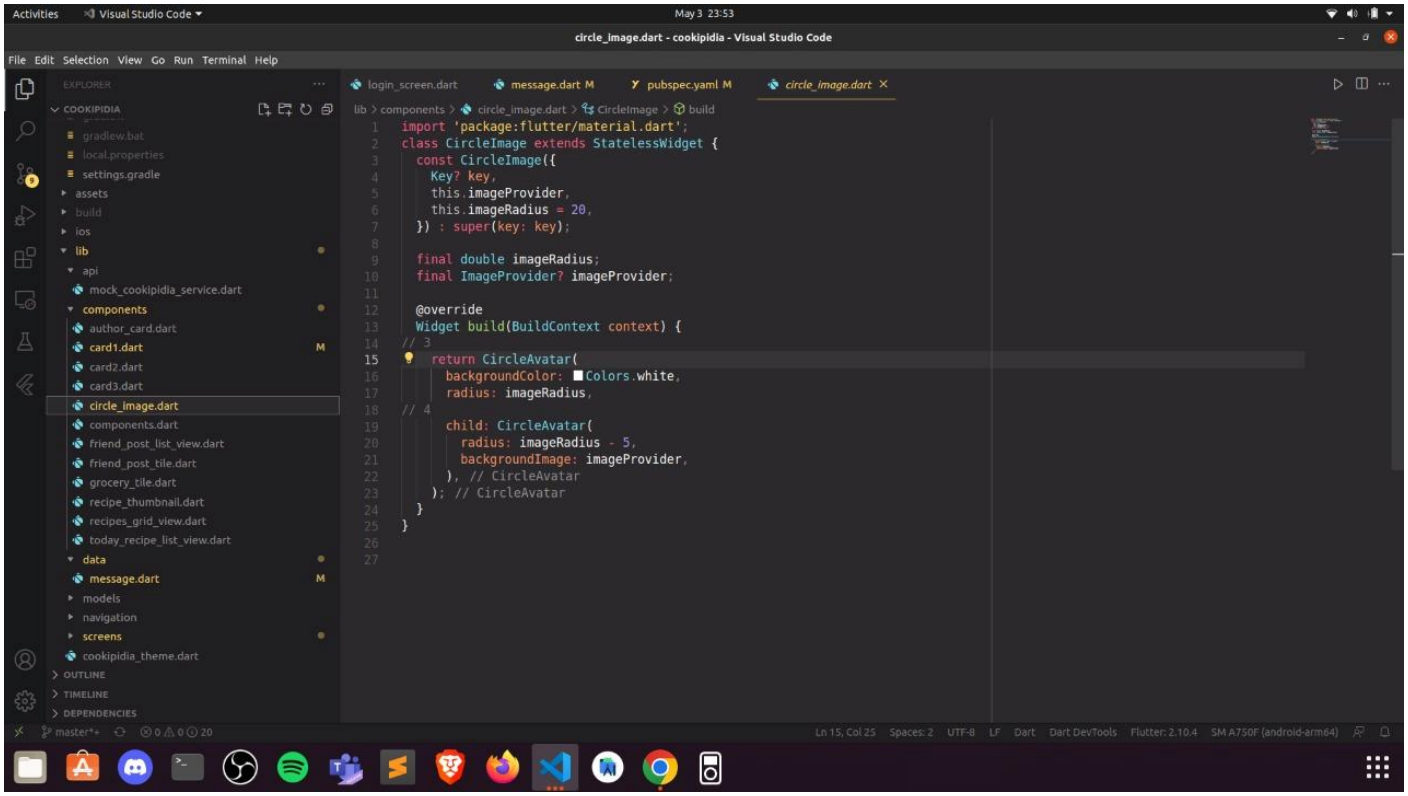


Fig. 138

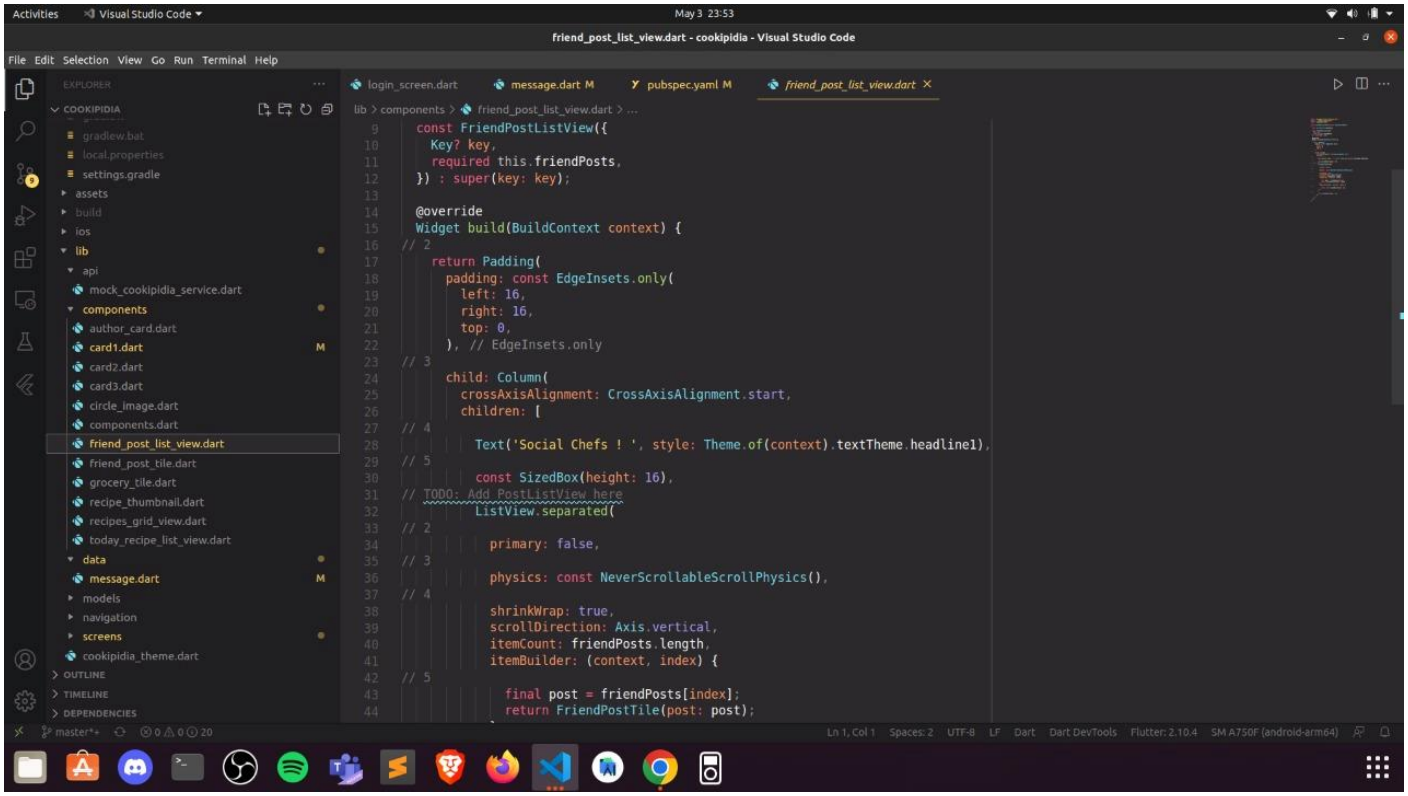


Fig. 139

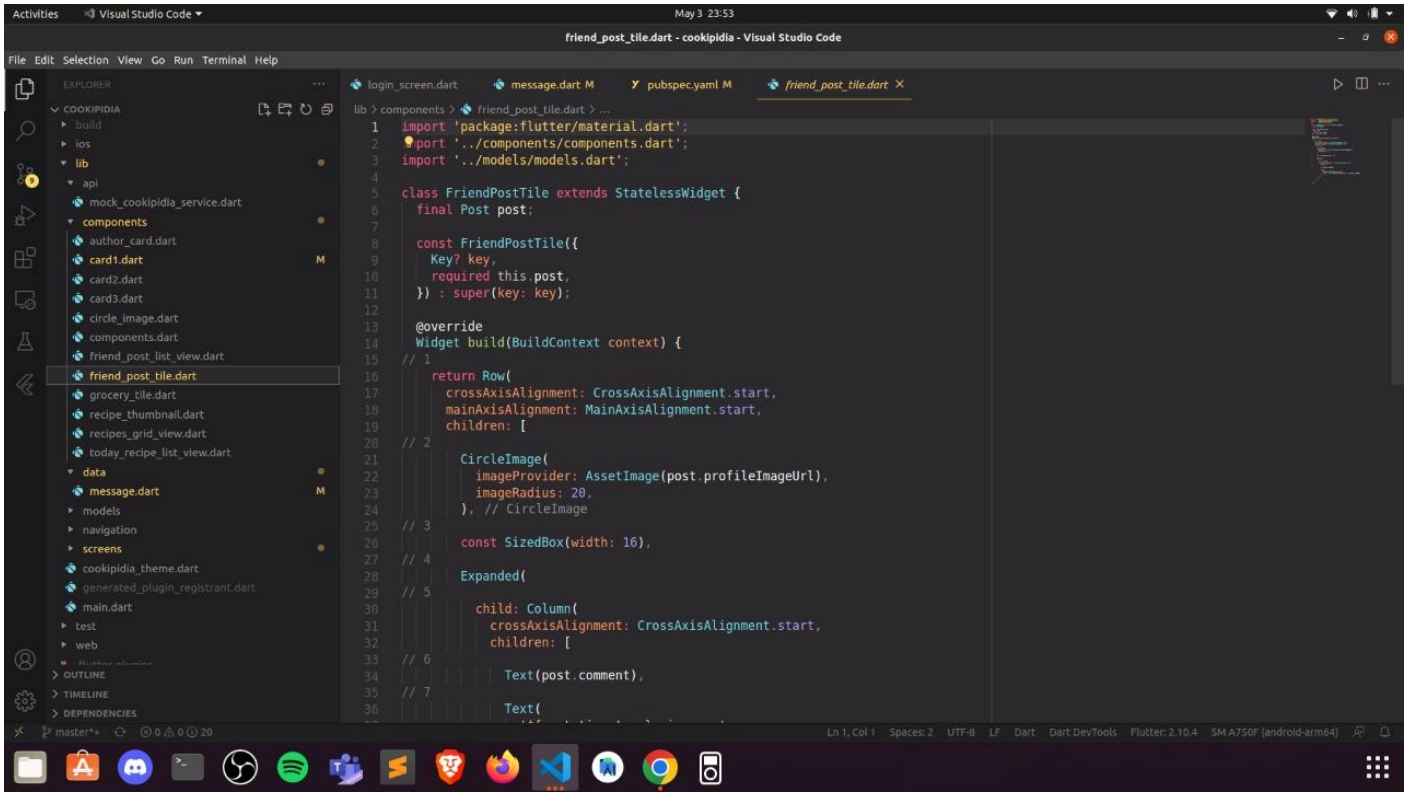


Fig. 140

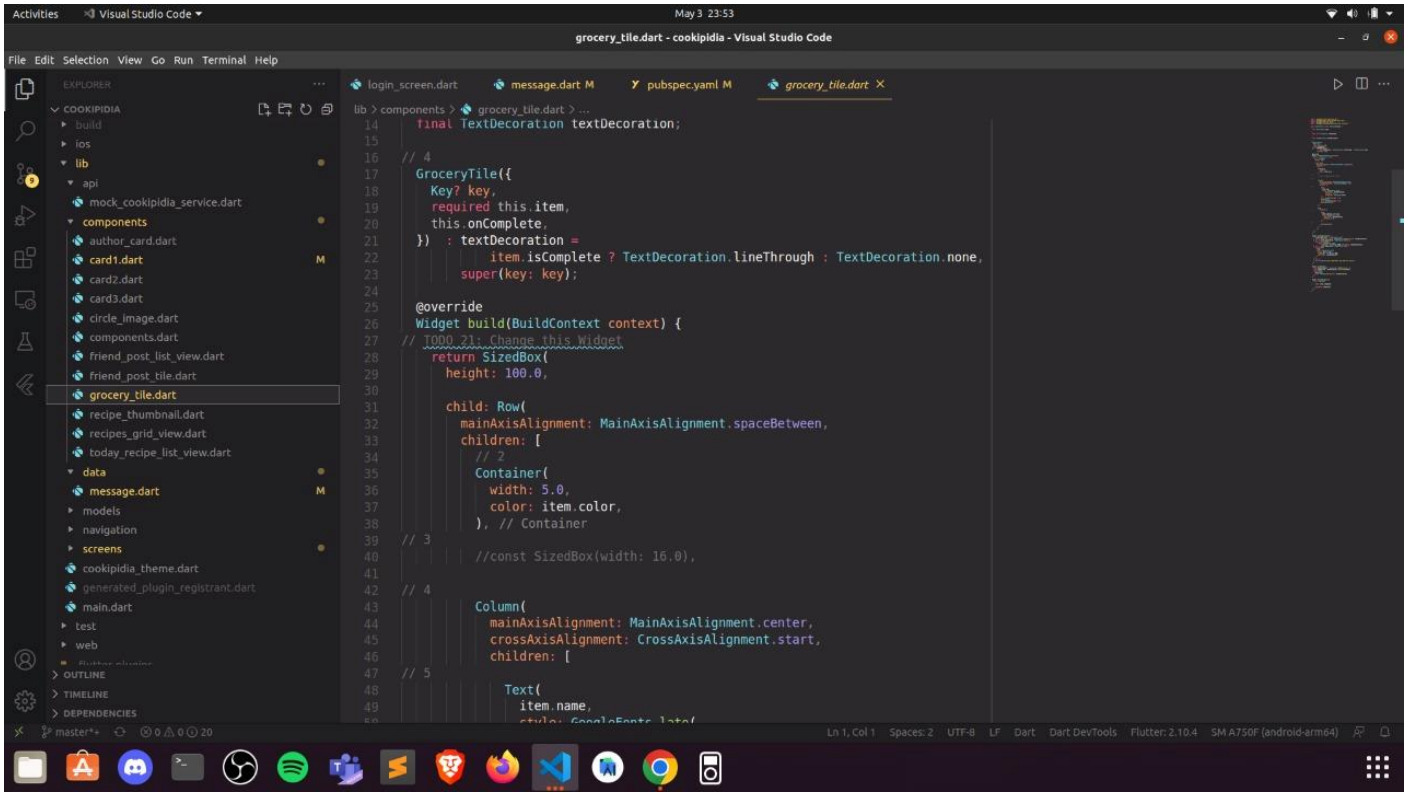


Fig. 141

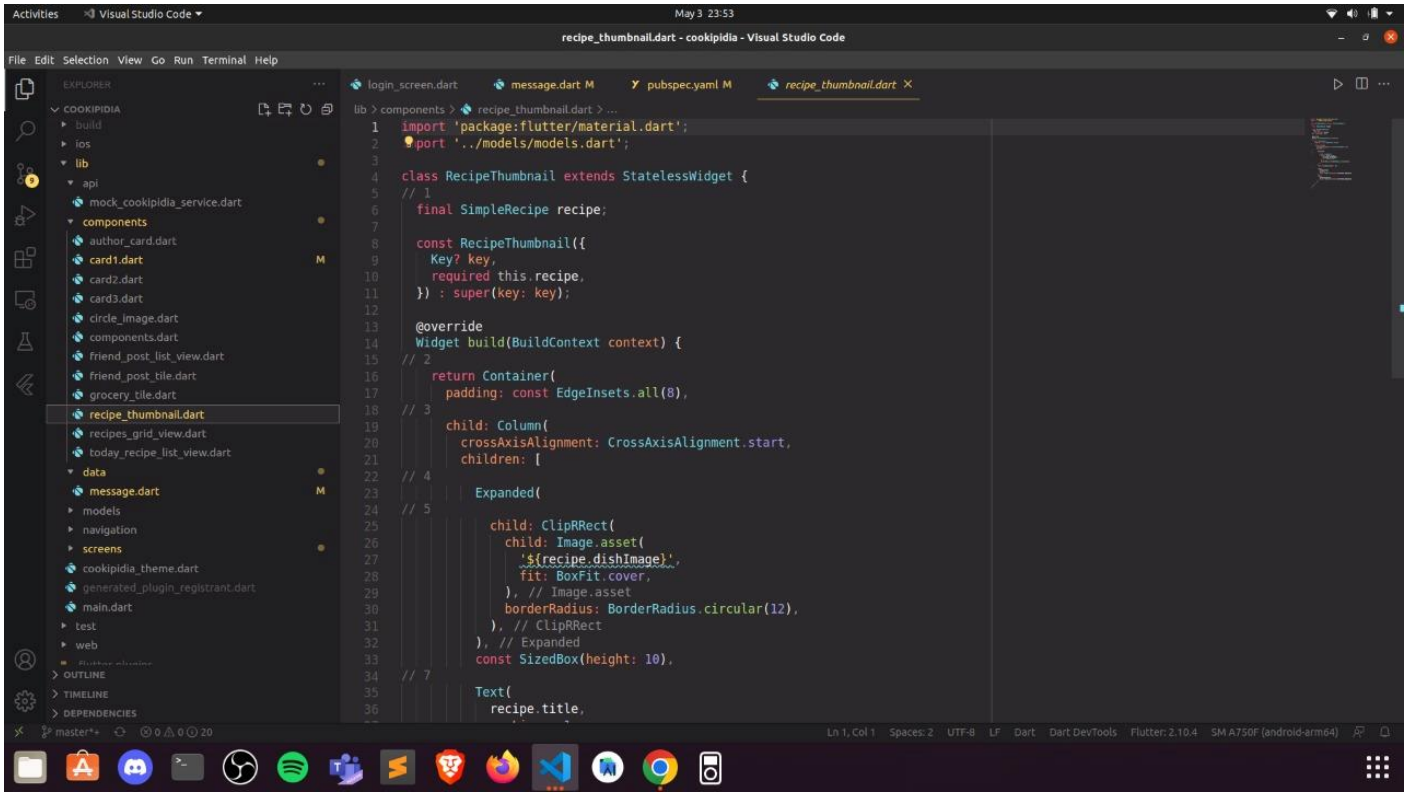


Fig. 142

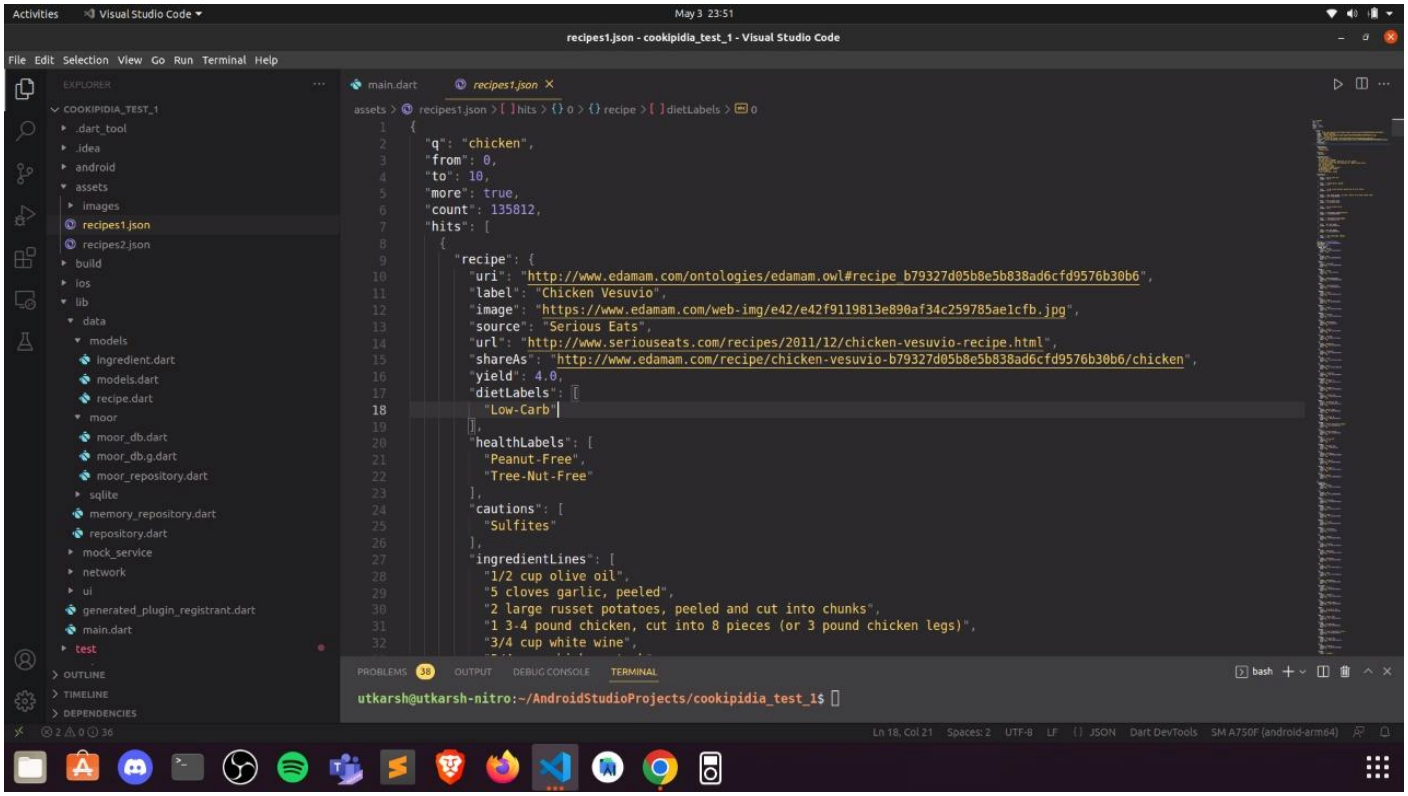


Fig. 143

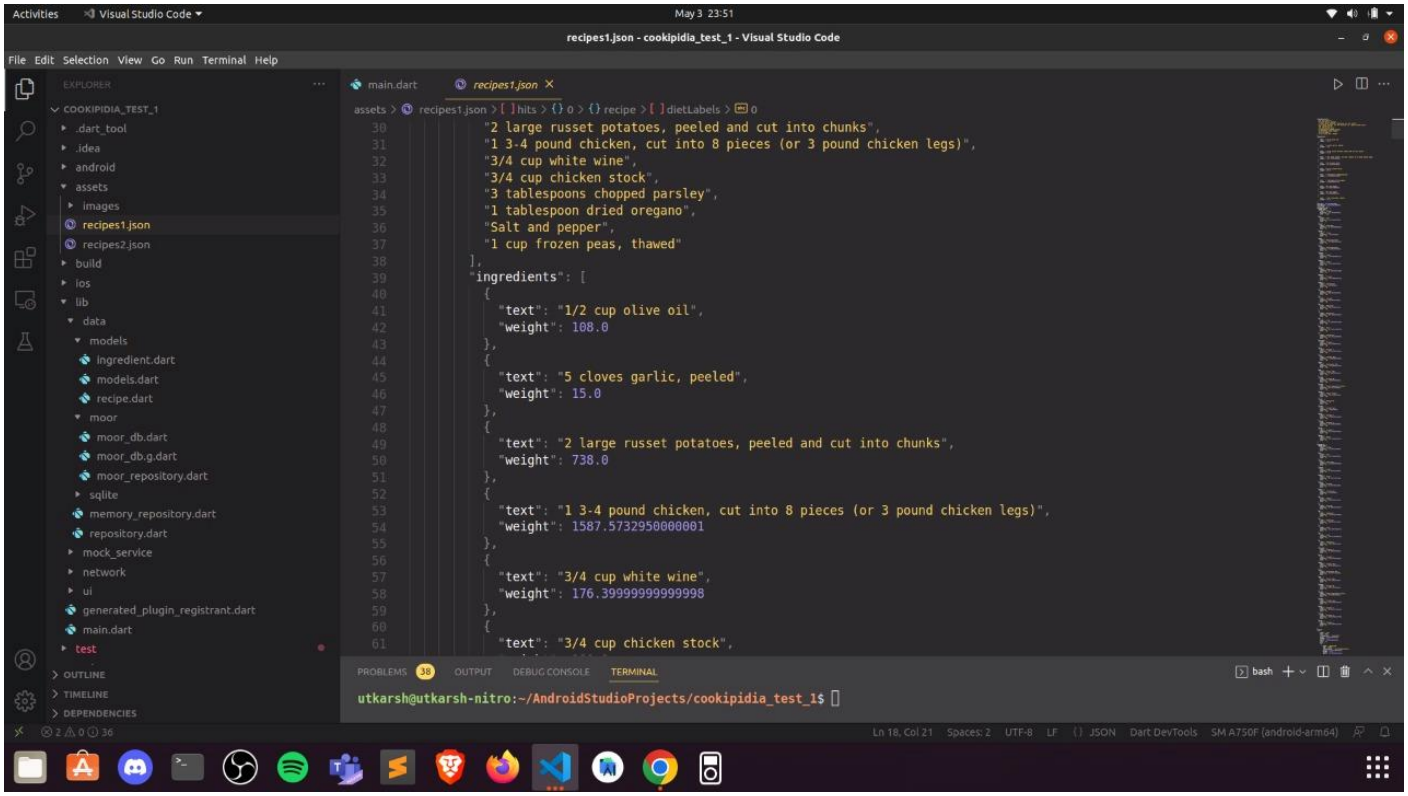


Fig. 144

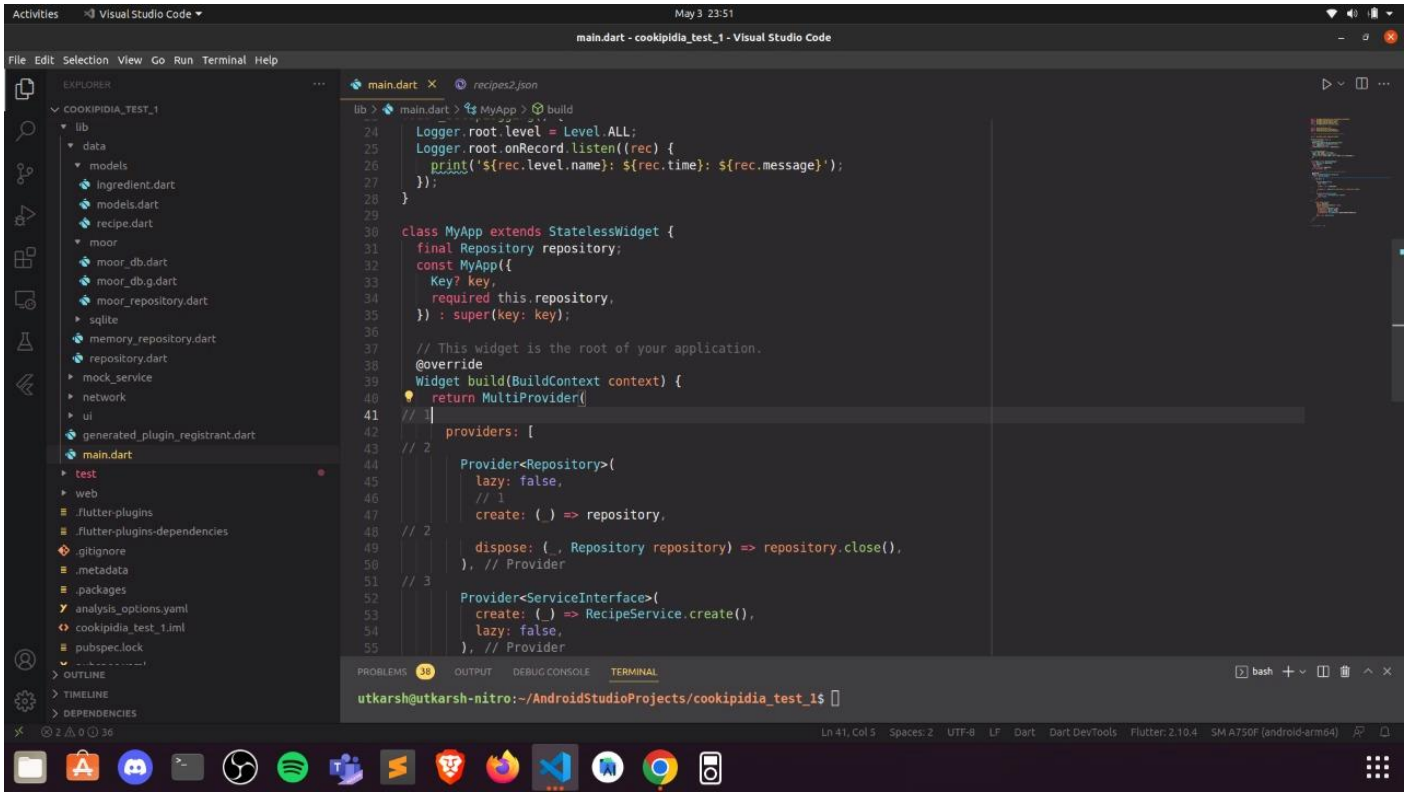


Fig. 145

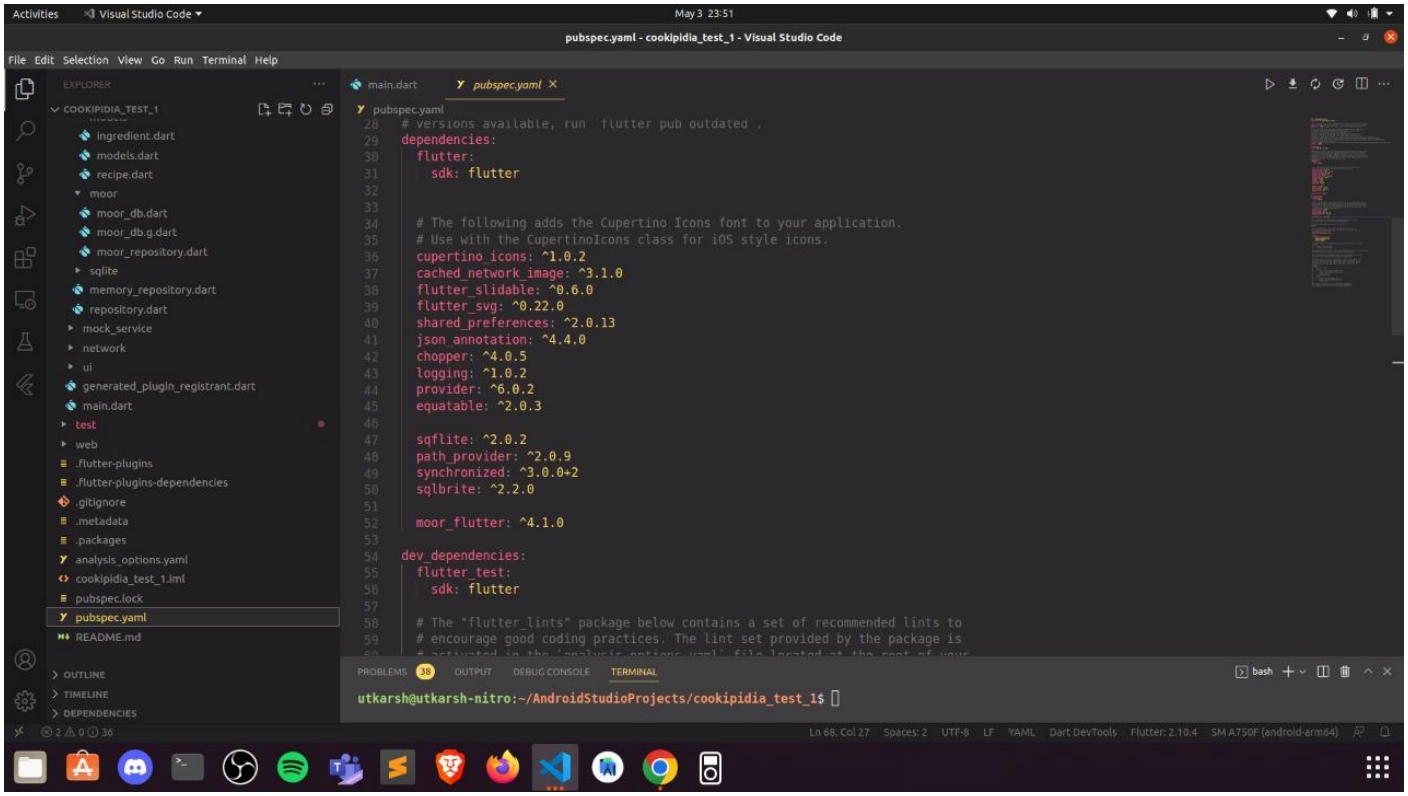


Fig. 146

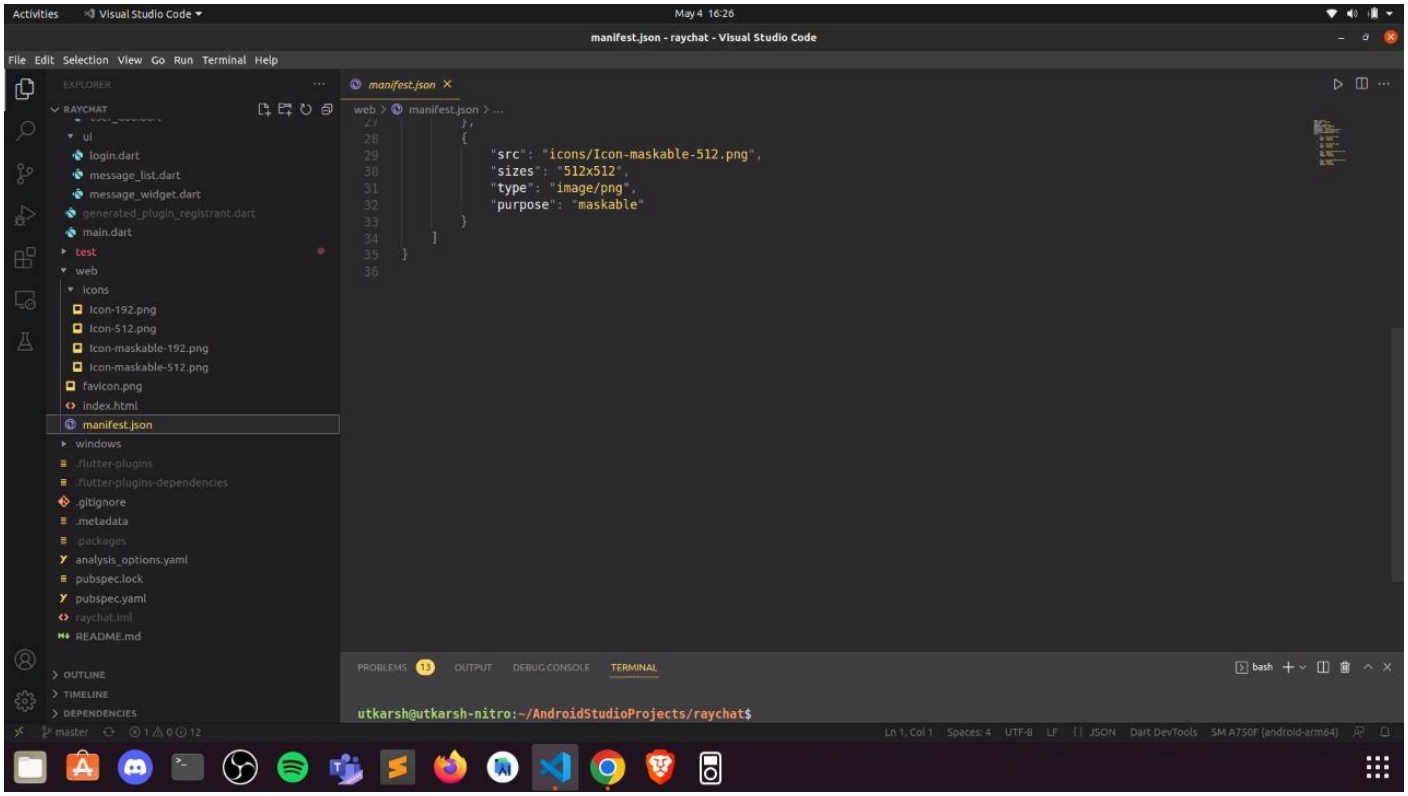


Fig. 147

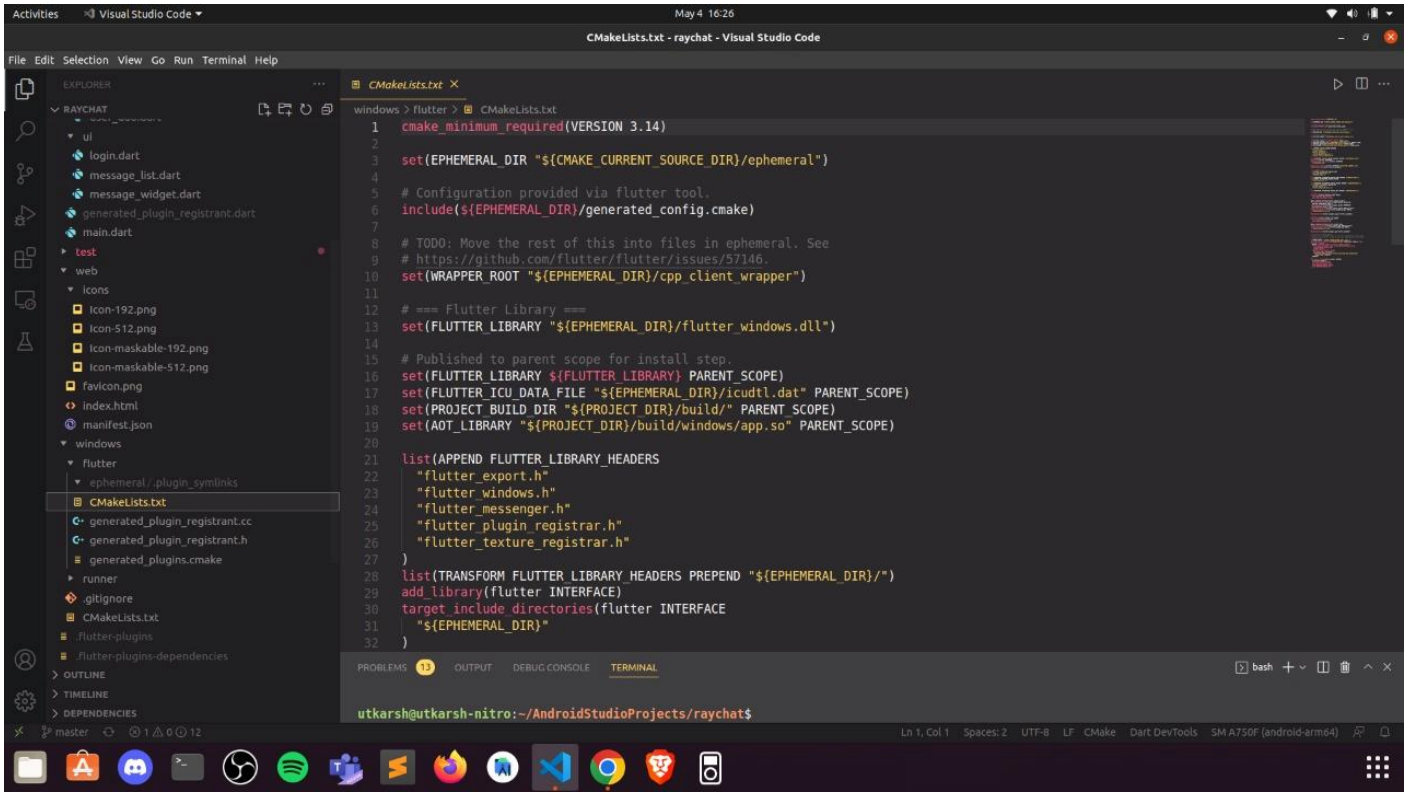


Fig. 148

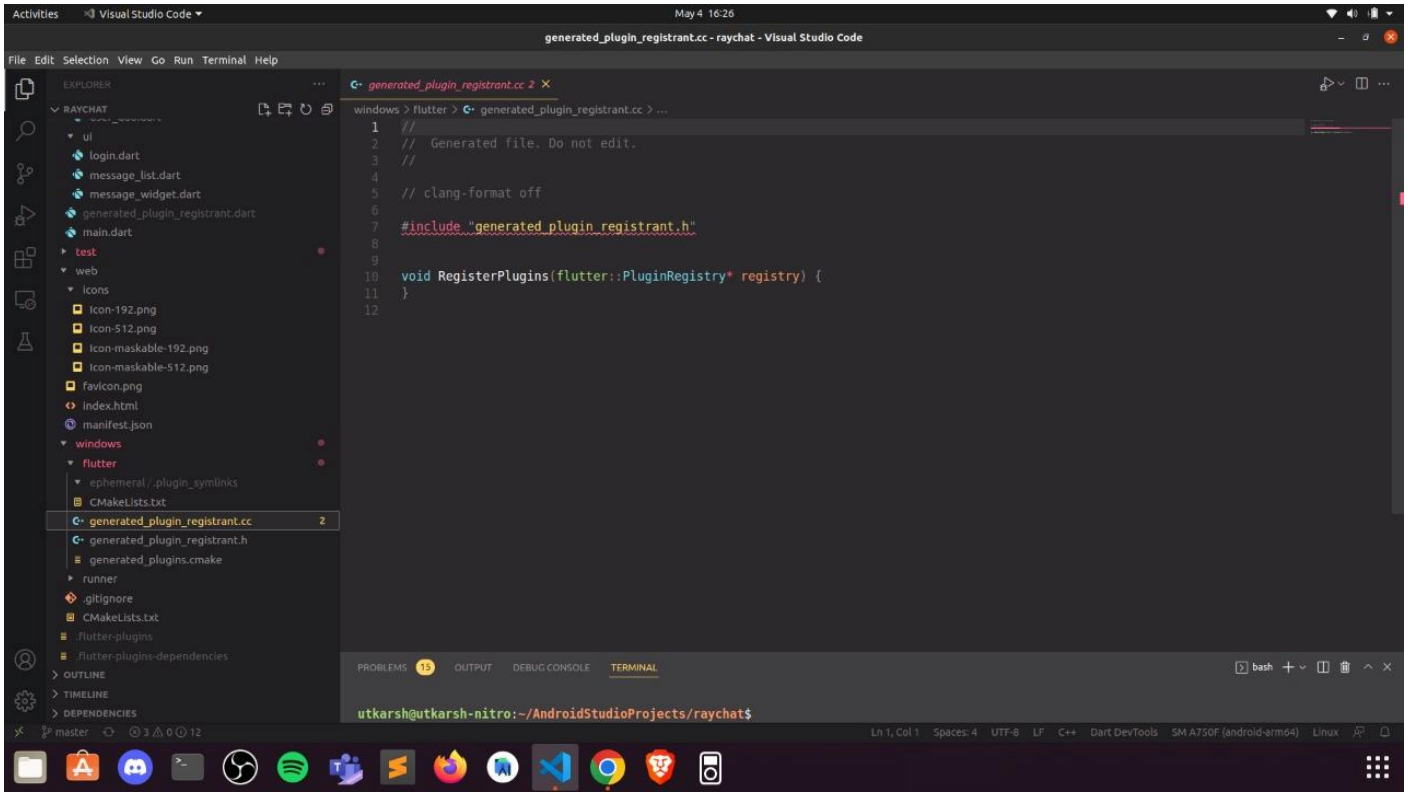


Fig. 149

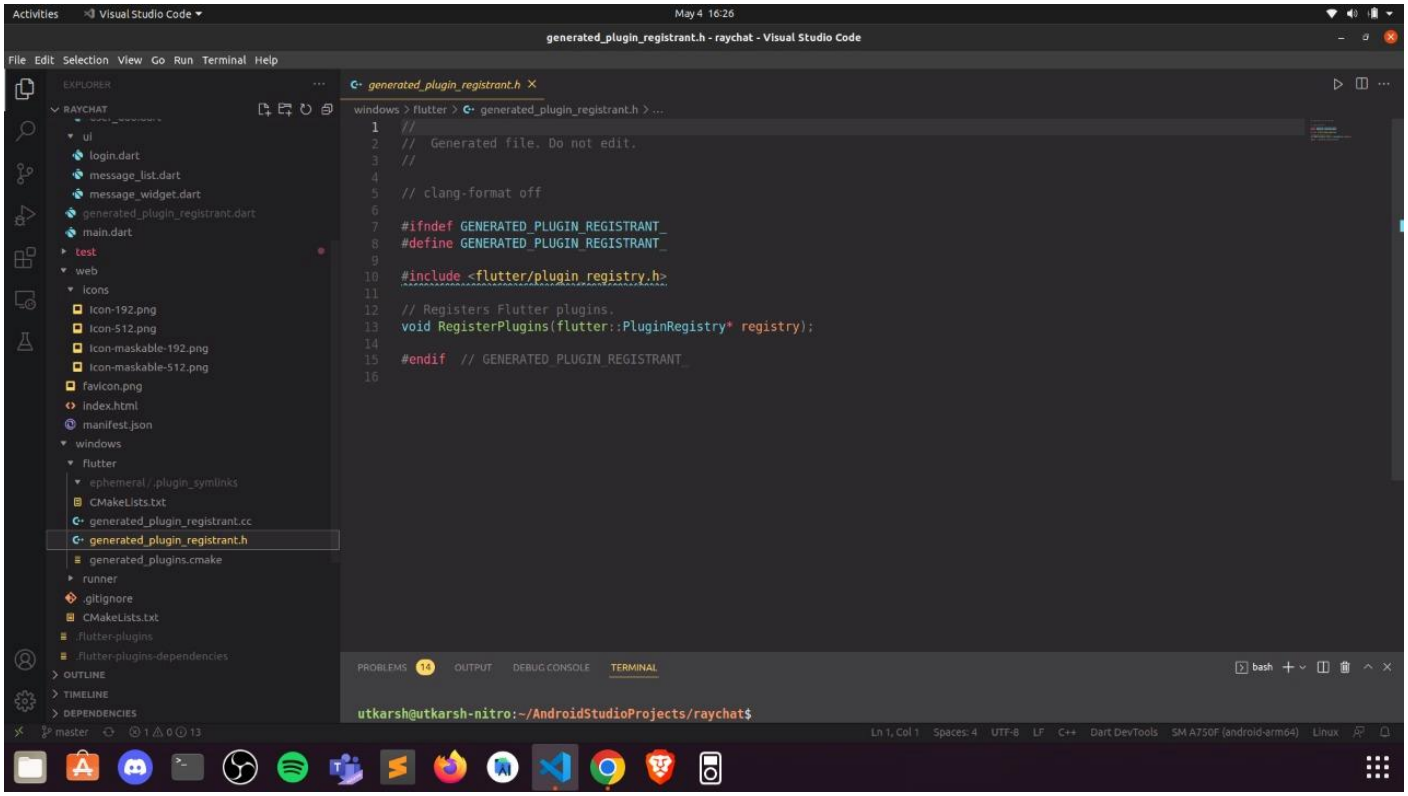


Fig. 150

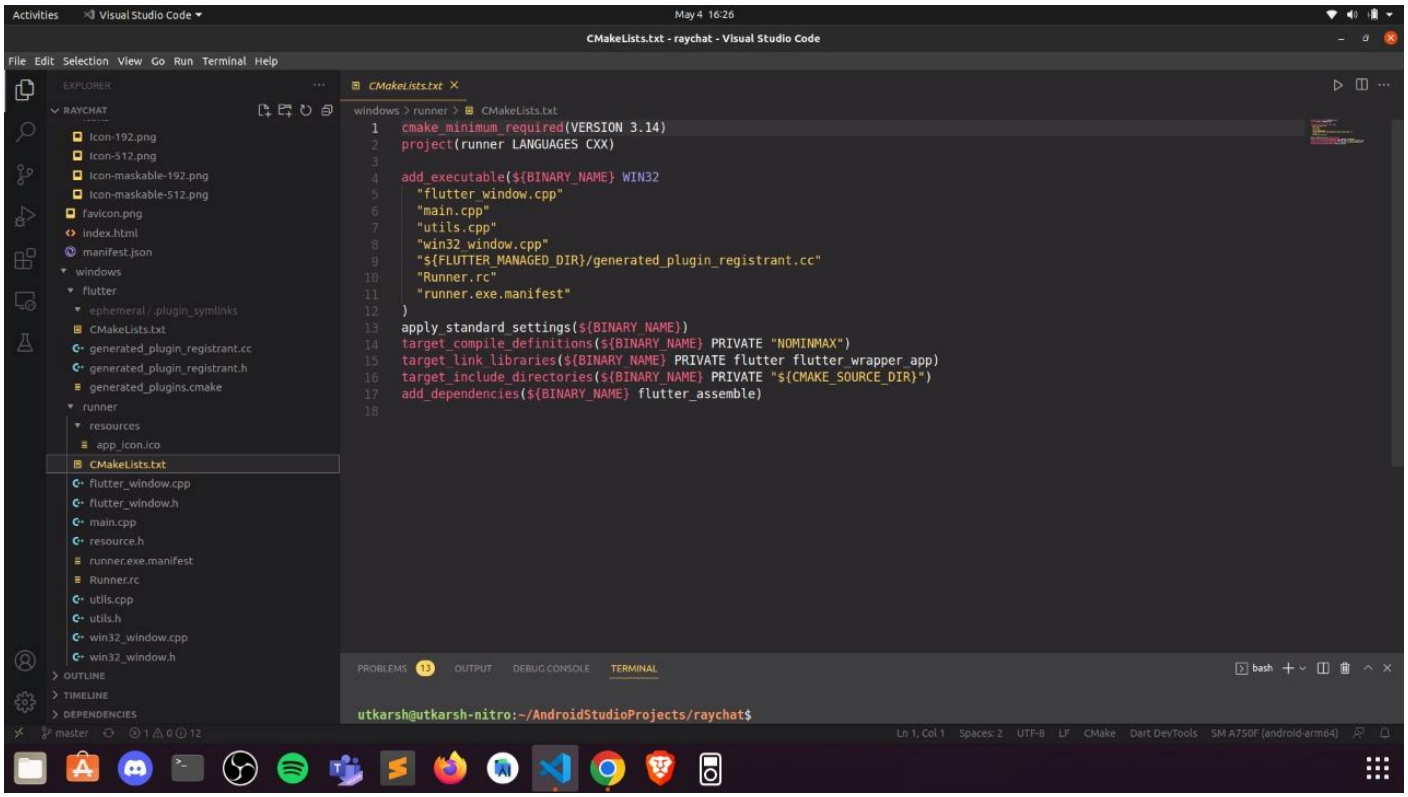


Fig. 151

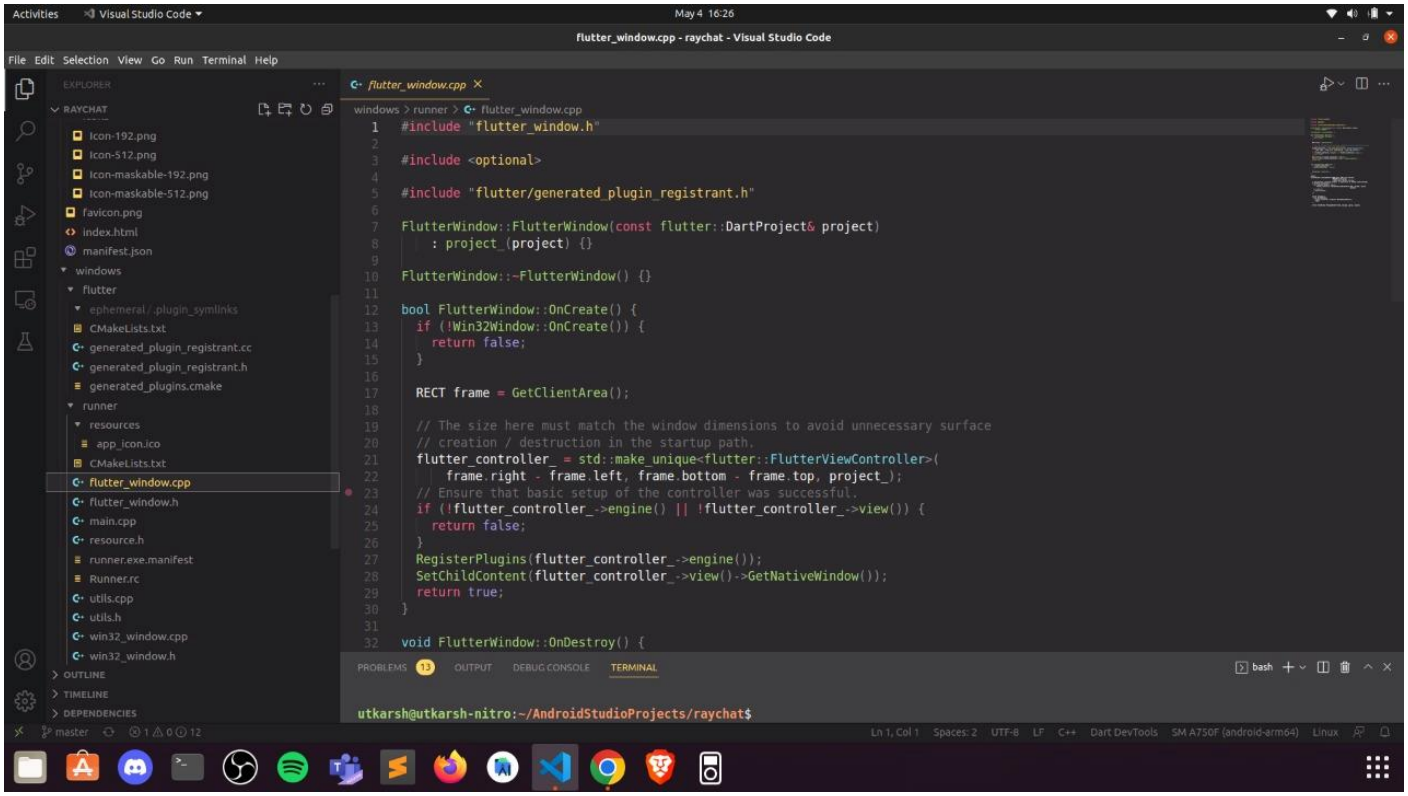


Fig. 152

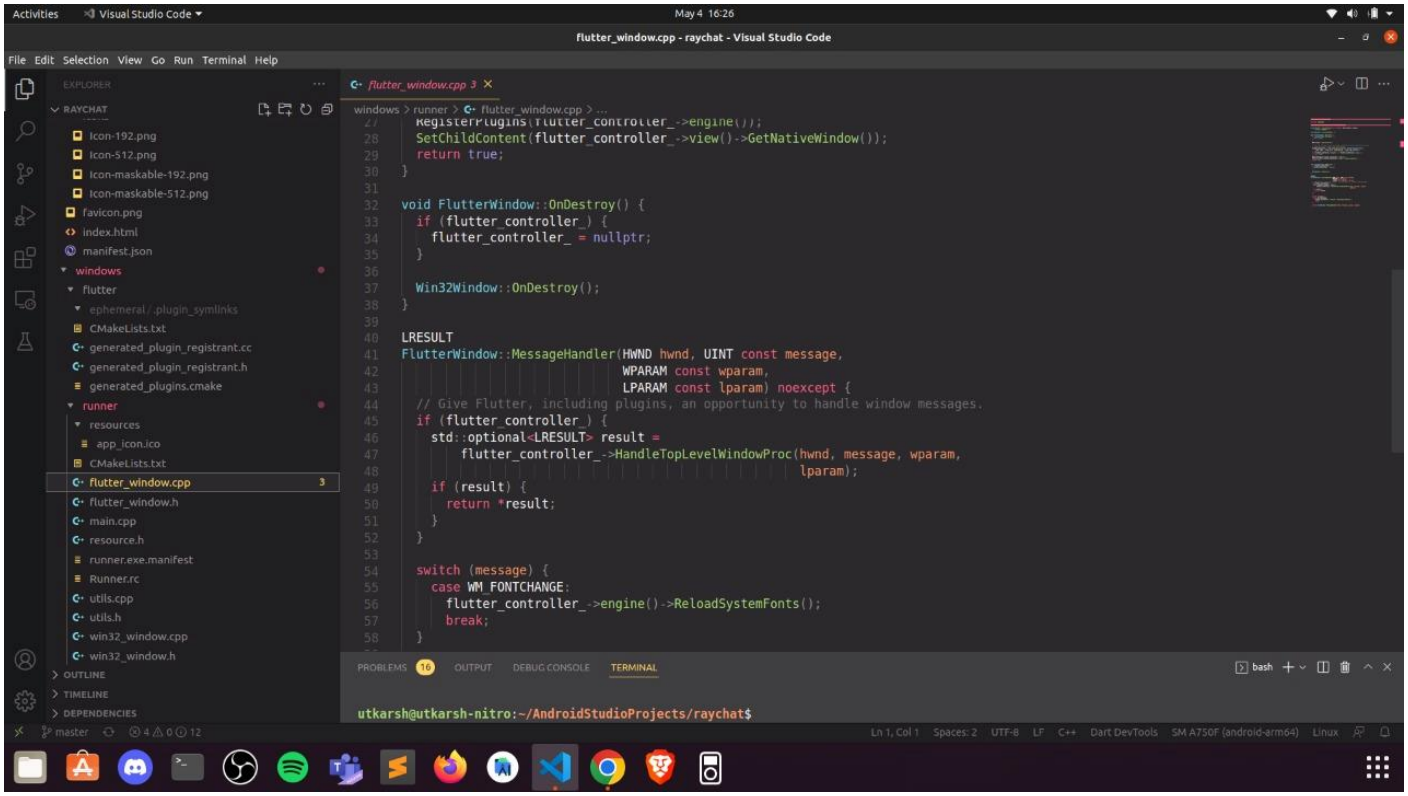


Fig. 153

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. Colum 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

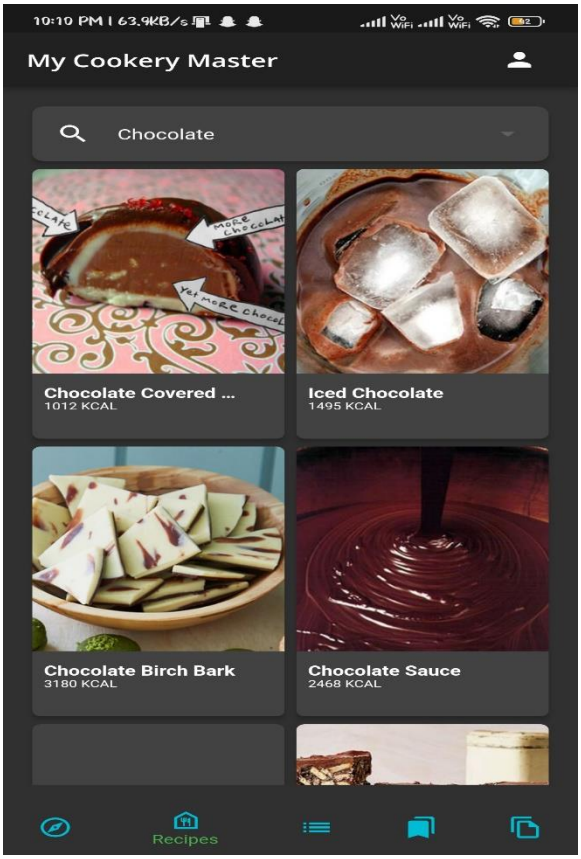
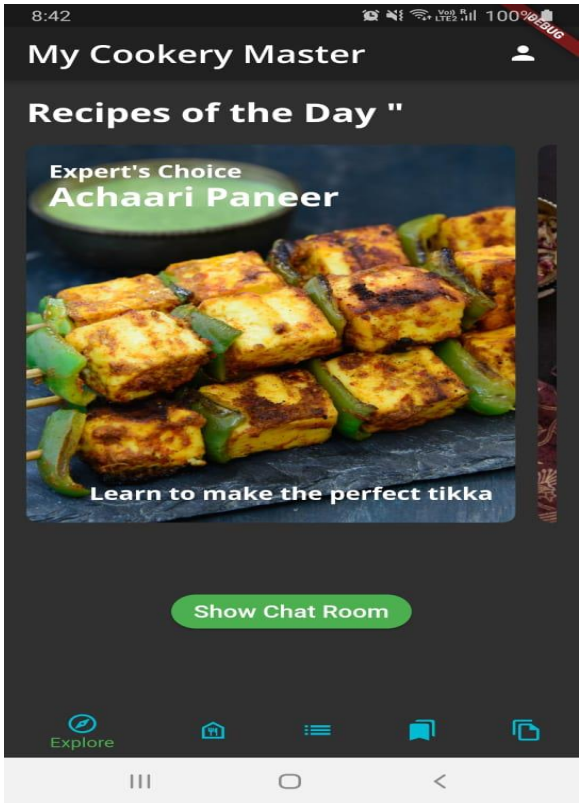


Fig. 154 & 155: Recipe Page (My Cookery Master)



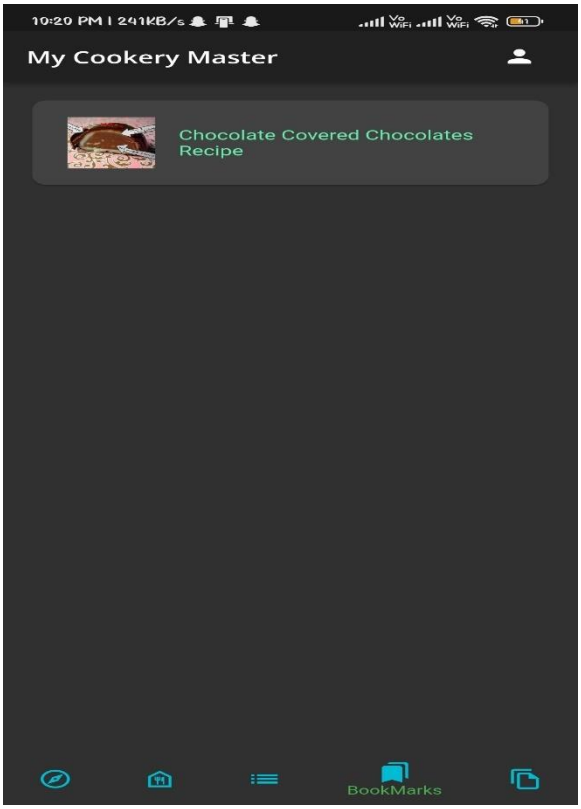


Fig. 156: Explore Page

Fig. 157: Bookmarks Page

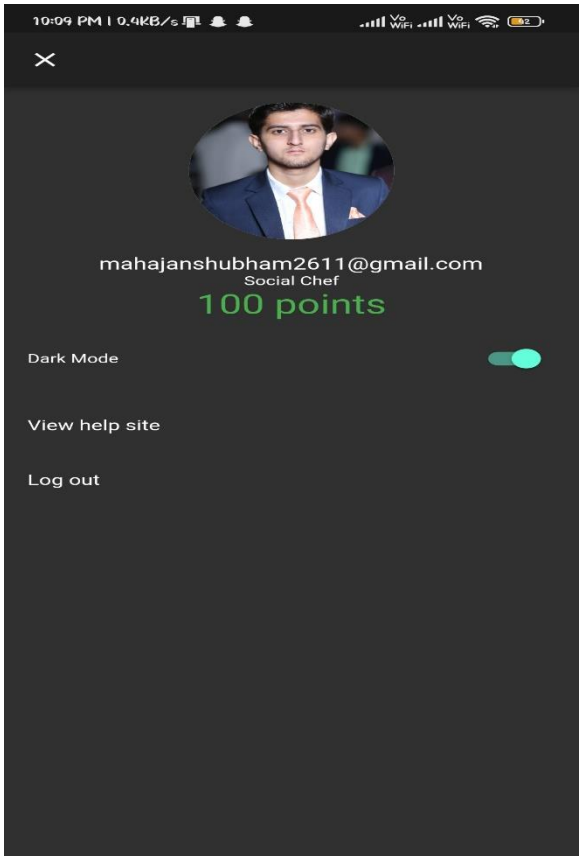


Fig. 158: Profile Page

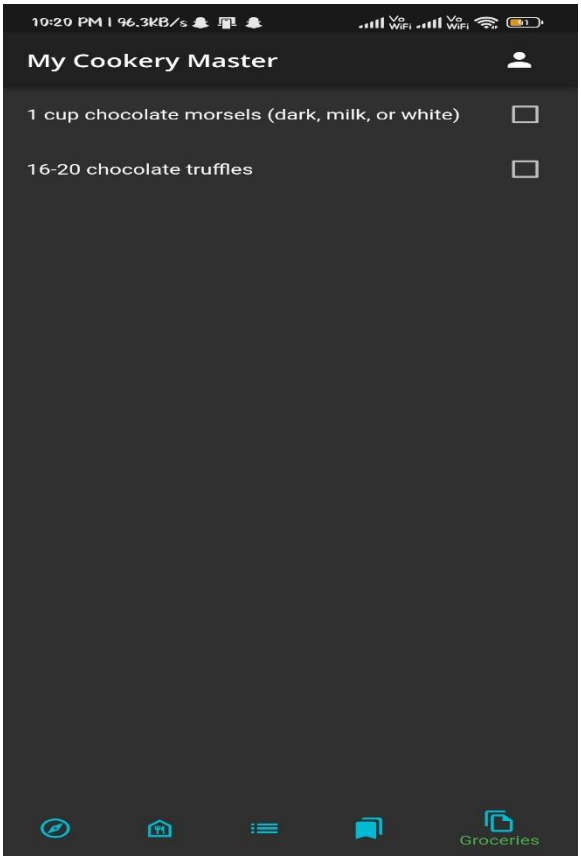


Fig. 159: Grocery Page

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 Yargeau. 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, 2018from 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.jpgaccessed 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