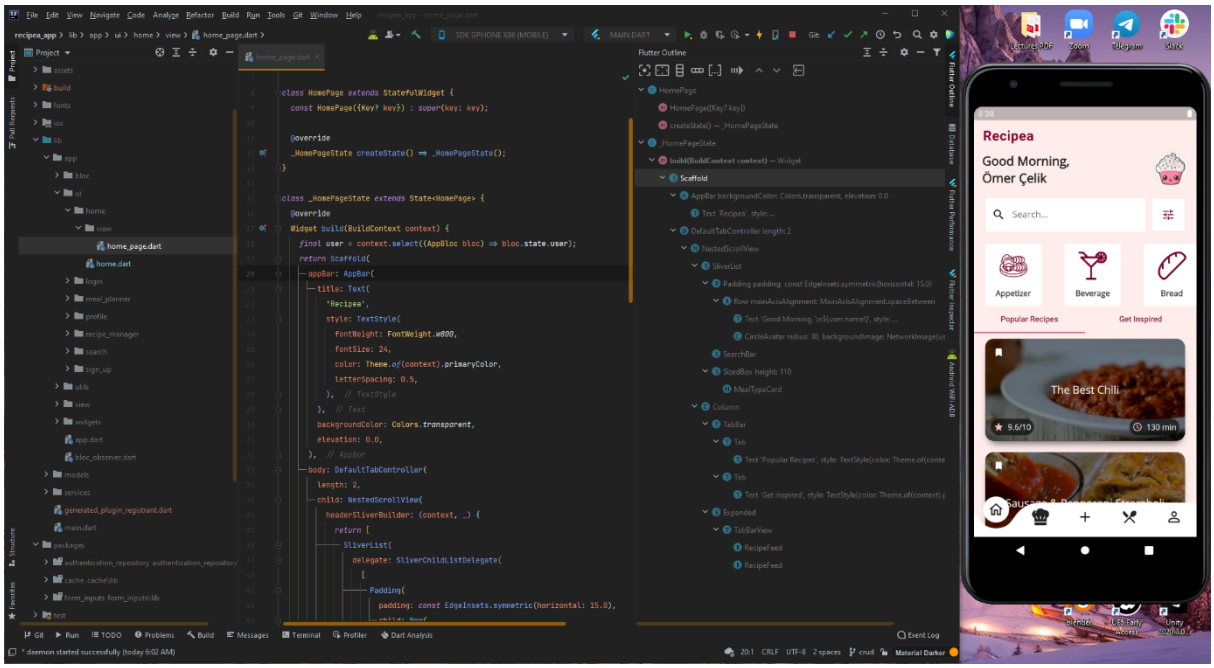
Table of Contents

Summary

During my internship, I learned the Dart language and Flutter, which an open-source UI software development kit created by Google. I also picked up knowledge about front-end development, REST API, and Google technologies like Firebase and Firestorm. It was hard to learn, but the many hours of work made it easier to understand the concepts encouraged me to solve my problems by own. I concentrated my time doing research. I want to learn about front-end jobs. It was tough to work out alone. But I guess no one will go further with a team unless they know the concepts and have adequate knowledge to talk about them to teammates. It was an original experience. I thought am working remote for a company. I always wondered to be a front-end developer. So, I developed cross-platform cookbook application that supports CRUD operations called “Recipea” and it’s looking well.



Introduction

Purpose of The System

In real life, making a dish can be quite though. Cook will have burnouts. If they did not know what to prefer for dinner, they will browse some recipes all over the internet. Sources are different so they cannot decide which one they are going to made. Of course, there is a lot of mobile applications to make this decision process easier.

But most of them have paid plans or just like a social media app more than a cookbook. Social media apps are different, people can share everything even recipes thought. So that makes the recipe apps (but more like a social media app) useless. After implementation of our system, they will have a more accurate cookbook application they want.

Objectives and Success Criteria of The Project

System aims to increase the satisfaction be eliminate the ambiguity of meal planning. Meanwhile usage of provided tools helping people throughout the cook, carefully prepared modules can quickly and effectively handle the actions that users normally must perform with more than one program. The convenience provided in this way increases the efficiency of the people, decreases the preparation time, and becomes an important criterion.

Current System

Since the digital transformation started for cookbooks, there has been a lot of different solutions created. Most of them delivered what they aimed, and every institute started to use the one that fits their case. In real life usage, some people find confusing or hard to use all of them while trying to just do a basic task. This behavior caused people to spend more time, effort, and money while they trying to solve easy problems. We believe this problem can be solved a cookbook app that can connect the platforms with portable modules.

Proposed System

Overview

In our case, everyone can cook becomes suitable for our solution. Those in this cycle will quickly and effectively access solutions that will begin to simplify their lives before they even realize it, with many simple operations, including meal planning. The system calculates the meal plan from the data of the cook. This system will enable cooks to search recipes or create they own. If they want, they can save the recipe to find them easily later. With this modern structure, the cooking will be easier.

Functional Requirements

User Login

Name	FR1 – User Login
Summary	Every user will be welcomed with same login page. After user enters the given credentials, modules will be shown.
Rationale	Allow users to login into application with their privileges using correct credentials.
Requirements	The system must be responsive and informative, and the credentials must be correct.

User Sign Up

Name	FR2 – User sign Up
Summary	Every new user must create an account. After user enters them credentials, reroutes user to login page.
Rationale	Allow users to create account to use application with their privileges.
Requirements	Credentials must be valid inputs.

Create Recipe

Name	FR3 – Create Recipe
Summary	User can create recipes.
Rationale	Allow users to create new recipe if they cannot find what they want or just want to add own recipes. Operation can be done from dashboard. By clicking the add button on flexible app bar.
Requirements	Recipe attributes must complete.

Read Recipe

Name	FR4 – Read Recipe
Summary	User can list recipes that they added to system. Or provided by services
Rationale	Allow users to list recipes. Or they can create own list such as saved recipes. List can be found on dashboard and profile page. On home page recipes provided by services.
Requirements	Recipes must legit and exist.

Update Recipe

Name	FR5 – Update Recipe
Summary	User can edit recipes in case they miss something or made a typo.
Rationale	Allow users to edit recipe info. Operation can be found on dashboard. Users need to select the recipe and then they can edit the form fields.
Requirements	Recipes must exist.

Delete Recipe

Name	FR6 – Delete Recipe
Summary	User can delete recipes.
Rationale	Allow users to delete recipe that they don't want to see any more. Operation can be found on dashboard page. User simply selects the recipe. And click the trash button on app bar.
Requirements	Recipes must exist first.

Search Recipe

Name	FR7 – Search Recipe
Summary	User can search recipes with tags or without.
Rationale	Allow users to search recipe to find what they want. Detailed search can be used by simply selecting the filter chips.
Requirements	Search query not required unless tags are not empty. Such as diet tag equal vegan.

Save Recipe

Name	FR8 – Save Recipe
Summary	User can save recipe to find them easily later. It will be appeared on profile page.
Rationale	Allow users to save recipe provided by services. Operation can be done by just simply choosing a recipe to view details. And clicking the button on bottom draggable scrollable sheet.
Requirements	Recipes must exist.

Meal Planner

Name	FR9 – Meal Planner
Summary	User can generate daily meal plan.
Rationale	Allow users to generate daily mail plan by diet and given calories. Result will be shown as breakfast, lunch, and diner with total nutrients. User can look recipe details and save them if they want too. Operation can be reached from bottom app bar.
Requirements	Diet and calorie intake must be provided by the user.

Nonfunctional Requirements

Usability

Name	NFR1 – Usability
Summary	System should have user-friendly GUI and will designed according to user experience.

Reliability

Name	NFR2 – Reliability
Summary	If one of the users in the system has crash, this should not affect other users.

Performance

Name	NFR3 – Performance
Summary	This system must support intensive workloads in acceptable response time no scalability issues.

Supportability

Name	NFR4 – Supportability
Summary	System must be configurable, adaptable, testable, and compatible with the web, android and iOS.

Implementation

Name	NFR5 – Implementation
Summary	The code must be clean, scalable, performant, documented and maintainable.

Legal

Name	NFR8 – Legal
Summary	The data must store on client's server, thus user privacy liable as their system’s security.

System Models

Scenarios

User Login

Name:	User Login
Actors:	User
Flow of Events:	• User opens the application. • User enters the credentials. • User clicks login button.
Entry Condition:	Opening app for the first time.
Exit Condition:	Clicking login button.
Quality Requirements:	Credentials must be correct.

User Sign Up

Name:	User Sign Up
Actors:	User
Flow of Events:	• User opens the application. • User enter create profile page. • User clicks the sign-up button.
Entry Condition:	Opening app for the first time.
Exit Condition:	Clicking the sign-up button.
Quality Requirements:	System must check undesirable usage of text.

Create Recipe

Name:	Create Recipe
Actors:	User
Flow of Events:	• User logins. • User clicks add button from bottom app bar to go to the dashboard. • User clicks add button on app bar. • User fills the text field with recipe information • User clicks the save button which located end of the page.
Entry Condition:	User must be login and navigate.
Exit Condition:	Clicks the save button.
Quality Requirements:	Fields cannot be empty.

Read Recipe

Name:	Read Recipe
Actors:	User
Flow of Events:	• User logins. • User clicks add button from bottom app bar to go to the dashboard.
Entry Condition:	User must be login and navigate.
Exit Condition:	User must close the page.
Quality Requirements:	Recipes must exist.

Update Recipe

Name:	Update Recipe
Actors:	User
Flow of Events:	• User logins. • User clicks add button from bottom app bar to go to the dashboard. • User selects the recipe by clicking on it • User changes the text fields • User clicks the edit button which located end of the page.
Entry Condition:	User must be login and navigate.
Exit Condition:	Clicks the edit button.
Quality Requirements:	Recipe must exist and text fields cannot be empty.

Delete Recipe

Name:	Delete Recipe
Actors:	User
Flow of Events:	• User logins. • User clicks add button from bottom app bar to go to the dashboard. • User selects the recipe by clicking on it • User clicks the trash icon on app bar.
Entry Condition:	User must be login and navigate.
Exit Condition:	Clicks the trash icon.
Quality Requirements:	Recipe must exit. After deleted, recipe list will be updated itself.

Search Recipe

Name:	Search Recipe
Actors:	User
Flow of Events:	• User logins. • User types what they want to search into search bar on home page if wanted meal type card or click tune icon to detailed search. Or user can navigate to search by cuisine page as simply clicking by the chef icon on bottom app bar then clicking the cuisine card. • User clicks the search icon. • User rerouted to search screen. • User can repeat the process from this screen. • User can make detailed search by clicking tune icon on right hand side of the search bar.
Entry Condition:	User must be login and search
Exit Condition:	Clicks search button.
Quality Requirements:	Recipe must exit.

Save Recipe

Name:	Save Recipe
Actors:	User
Flow of Events:	• User logins. • User clicks on the recipe from home page or simply clicks the bookmark icon located in the card by upside left corner. • User clicks the bookmark icon on recipe details screen that recently rerouted.
Entry Condition:	User must be login and navigate.
Exit Condition:	Clicks bookmark icon.
Quality Requirements:	Recipe must exit.

Meal Planner

Name:	Meal Planner
Actors:	User
Flow of Events:	• User logins. • User clicks the spoon icon on the bottom app bar. • User selects diet • User slides the bar until desired calorie limit • User clicks the search button
Entry Condition:	User must be login and navigate.
Exit Condition:	Clicks search button.
Quality Requirements:	Calorie intake cannot be empty.

Diagrams

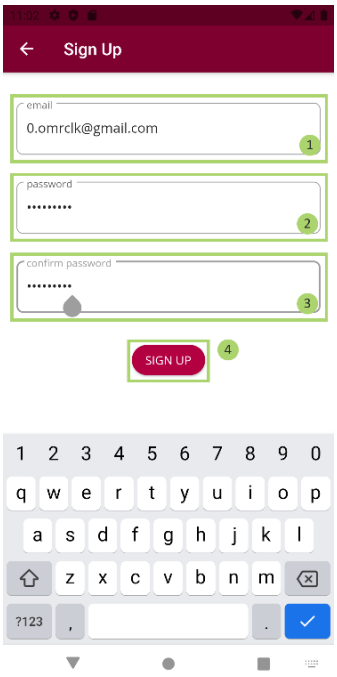
All Firebase Realtime Database data is stored as JSON objects. You can think of the database as a cloud hosted JSON tree. Unlike a SQL database, there are no tables or records. When you add data to the JSON tree, it becomes a node in the existing JSON structure with an associated key. Here is how the JSON tree looks like of this project. You can find the code that handles CRUD operation in appendix.

User Manual

Getting Started

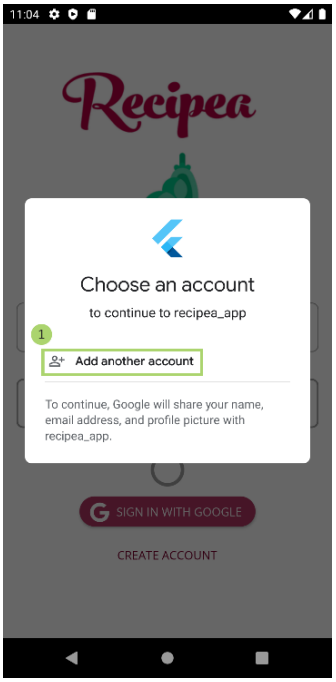
1. Fill the fields with your credential
2. Press the button to authenticate
3. You can use you Google account.
4. If you are new be sure to create account.

New Account



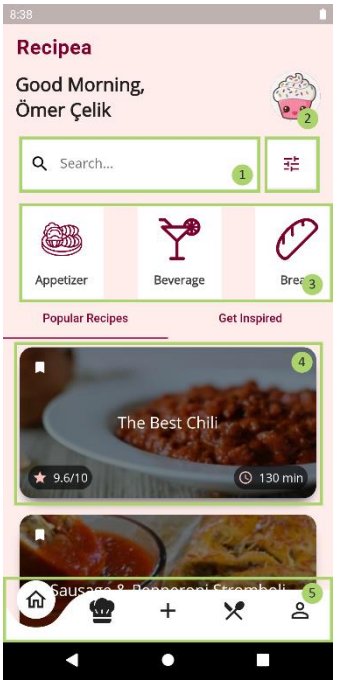
1. Type in your mail account.
2. Create a password long than 6 characters with numbers, letters, and special characters.
3. Re-type your password to confirm.
4. Click sign up to finish process.

Google Account



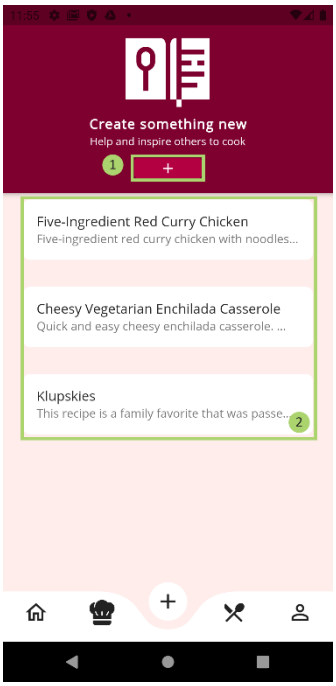
1. To add a google account follow the standard procedure.

Welcome



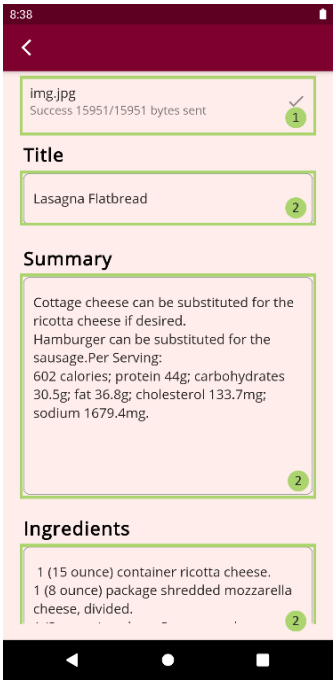
1. This bar allows you to search a recipe.
2. If you want to search with filters checkout the tuning options.
3. Faster interaction to search certain type of meal.
4. Tap on these cards to see detailed information about the recipe.
5. This bar allows you to navigate through pages.

New Recipe

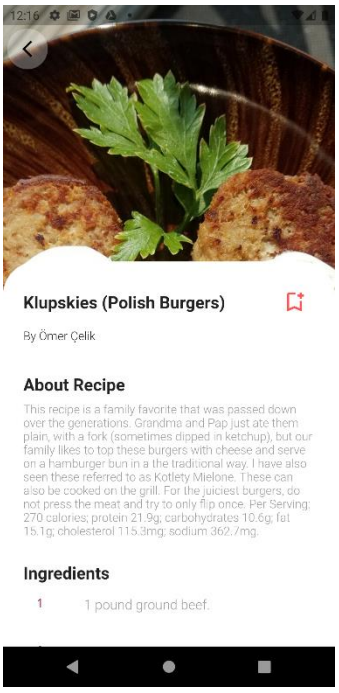


1. To add new recipe, press this button.
2. Your recently added recipes shown here.

3.Once you done be sure to save.
4.After you saved you can click and go back to dashboard.

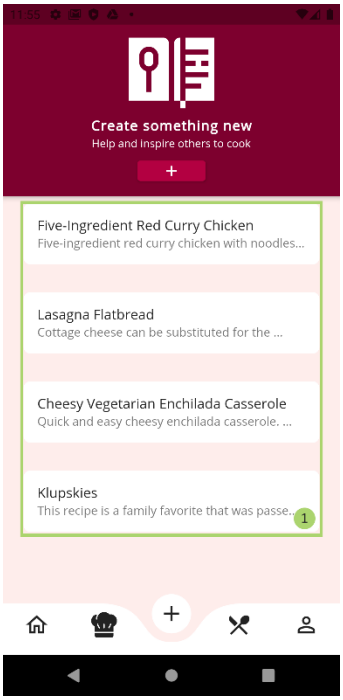


1. To add an image, check out this part.
2. You can start writing your recipe. Be sure to put "." after every instruction step and ingredient.



Here is how your recipe looks like on your profile.

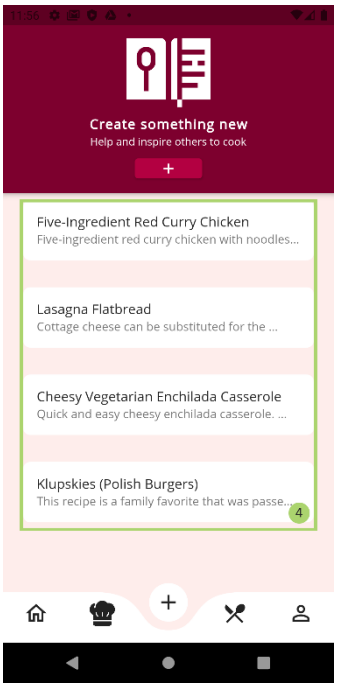
Edit Recipe



1. Choose the recipe you want to edit from the dashboard.

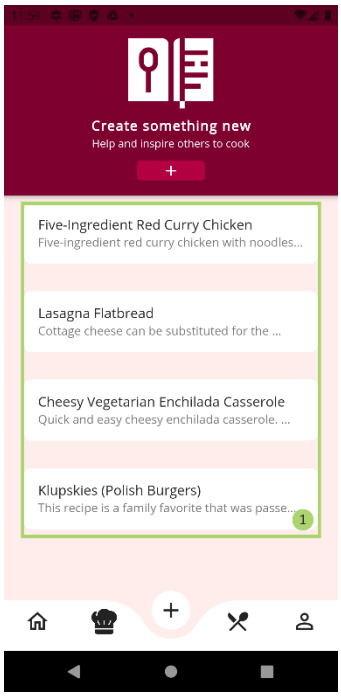
3. Be sure to click edit when you done.

2. Edit the fields that you want. As you can see Klupskies changed into Klupskies (Polish Burgers).

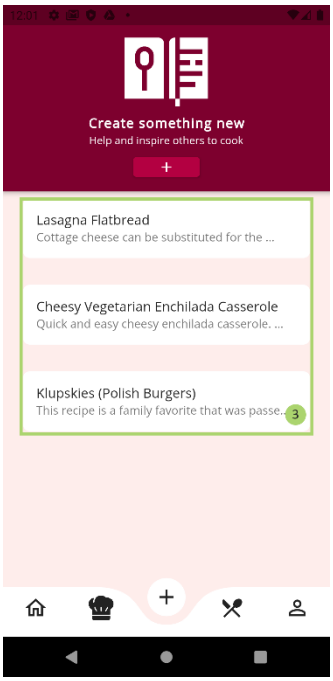


4. When you come back to the dashboard you can see the change.

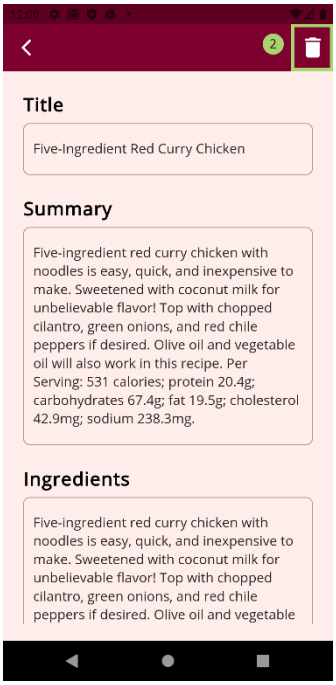
Delete Recipe



1. Select recipe that you want to delete from your cookbook.

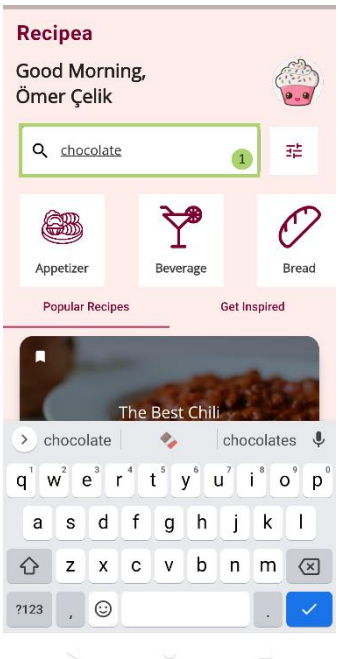


2. When you come back to the dashboard you can see the recipe is not listed anymore.

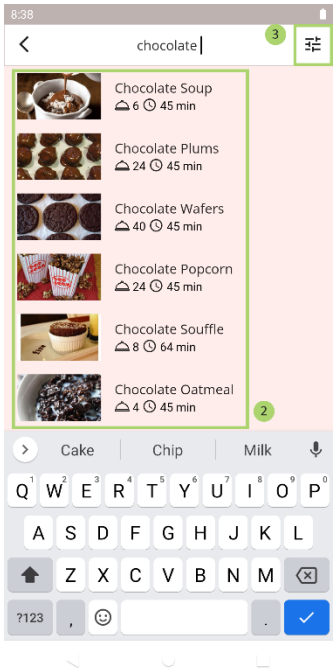


1. Click the trash icon on app bar. That the recipe you want to delete will be gone.

Search

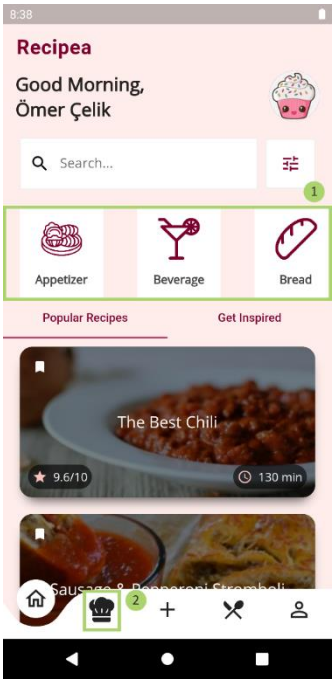


1. To search recipe, tap on the search bar located in home page. And start type in the phrase that you are looking for.

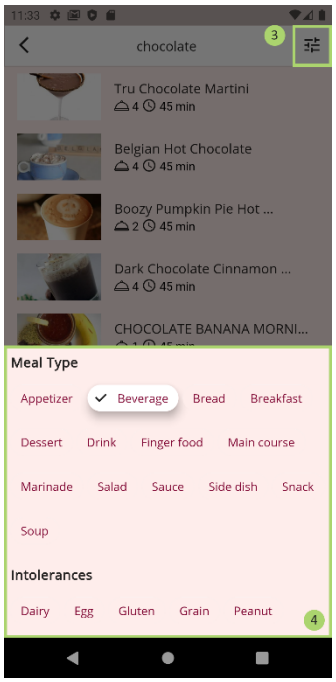


2. Once you hit the search icon on previous step you will redirected to this page. And you can check out your search result in here.
3. To use filtering option on your search tap on tuning icon.

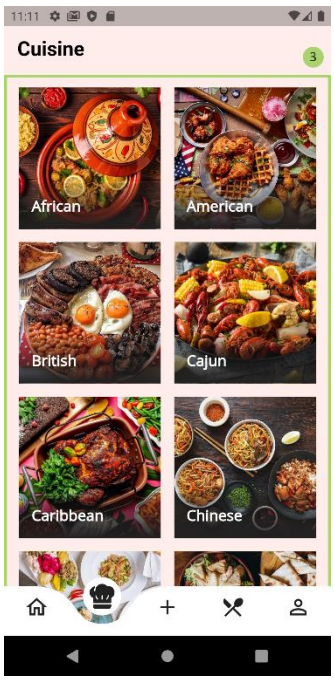
Fast Search



1. To fasten you research speed you can use the meal type cards on home page.
2. Or if you want to search as cuisine you can navigate through the bottom app bar simply taping by the chef hat icon.

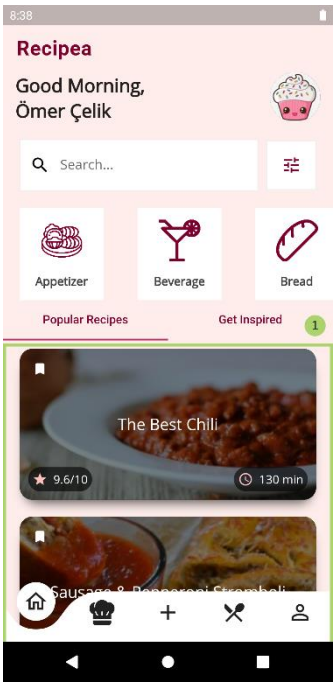


4. For filtering options there is Meal Type, Intolerances, Diet, and cuisine are available. As you can see beverages that contain chocolate is listed on the search.

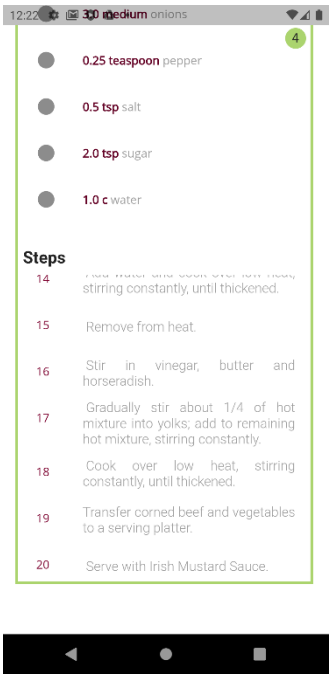


3. Once you get this page a simple tap to cuisine card will be initiate the search sequence for you. And you are going to redirected to search page.

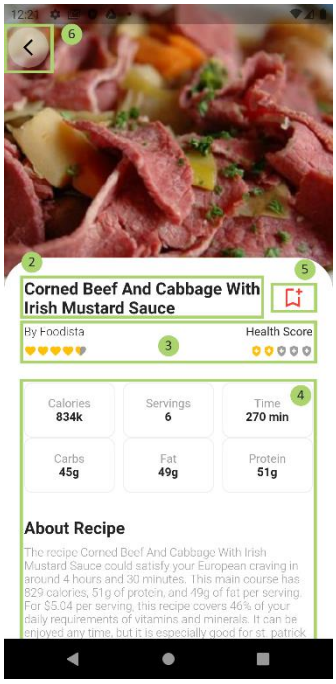
Details



1. To look recipe details, you must tap on recipe on home page, on dashboard, when you made a search, created a meal plan, or on your profile. You can access recipe details on every page you navigate.

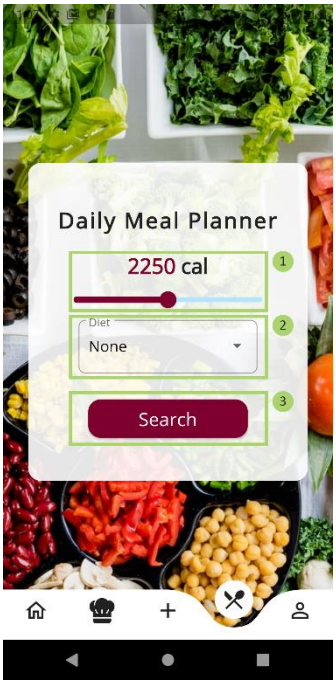


5. If you want to save recipe tap on the bookmark icon.
6. When you done with details page you can simply tap and go back to previous page.

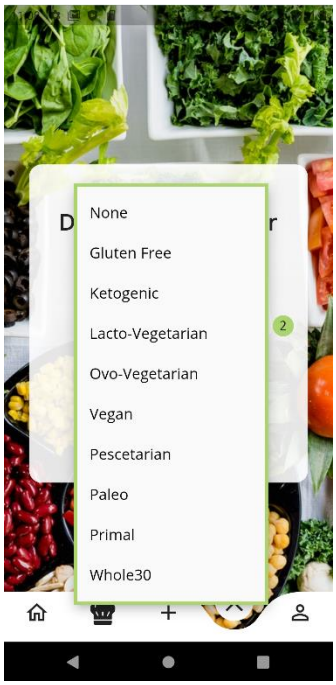


2. Once you get into details page you can see recipe title.
3. Rating and health score represented below the title.
4. In this part you can see nutrition information about the recipe. And must know details are written. This part is scrollable. Ingredients and instructions are mentioned end of the page.

Meal Planner

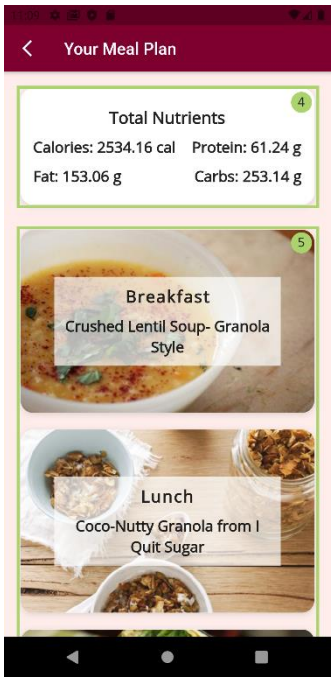


1. Once you use bottom app bar to navigate this page you must select a target calorie. The plan will be generated based on this calorie.
2. Diet can be none but there is a drop-down menu that you can select your diet among to options.
3. Be sure to tap on search button when you are done.

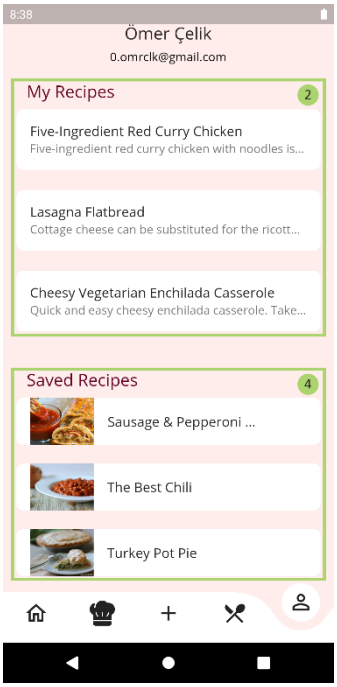


Profile

1. Here are your profile details as the user you logged in.
2. You can check your recipes on you profile page too. Those tiles are interactable so you can find your recipe details whenever you want to see.
3. There is a log out button on app bar it is not necessary but if you want, feel free to use.

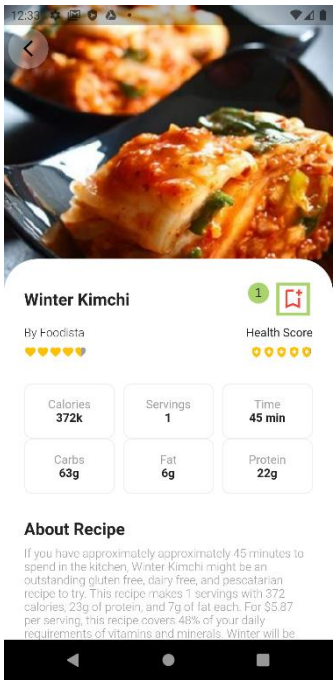


4. When you generate meal plan here is your statics of nutrients for the total meal plan.
5. Below that part you got breakfast lunch and dinner. You can tap and look further information about each recipe as mentioned on previous topic.

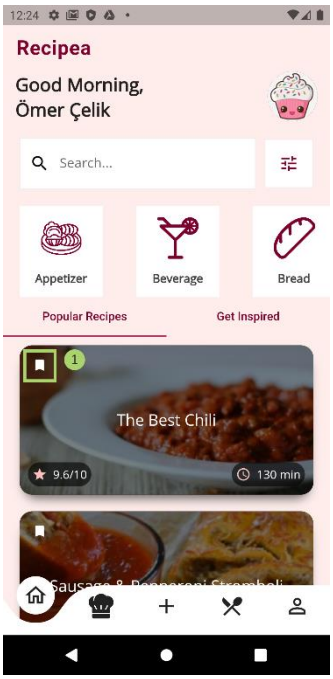


4. These are the recipes that you saved. Further information about saving will be covered on next topic.

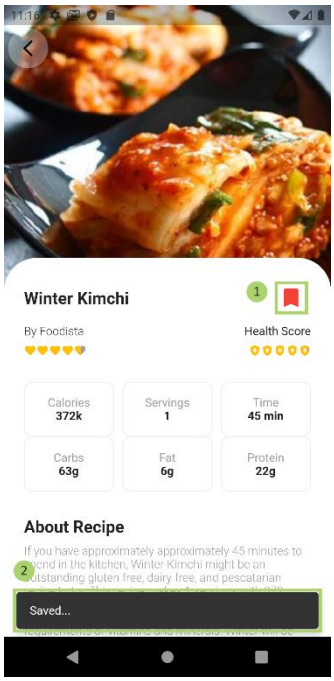
Save Recipes



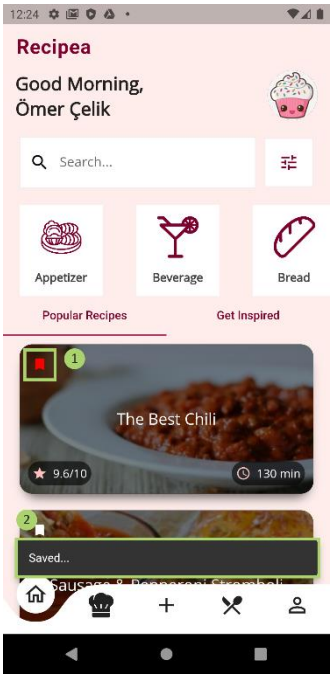
1. To save a recipe there are several ways. One of them is you must navigate to recipe details that you want to save. Once you navigate this page there is a bookmark icon. A simple tap will save the recipe.



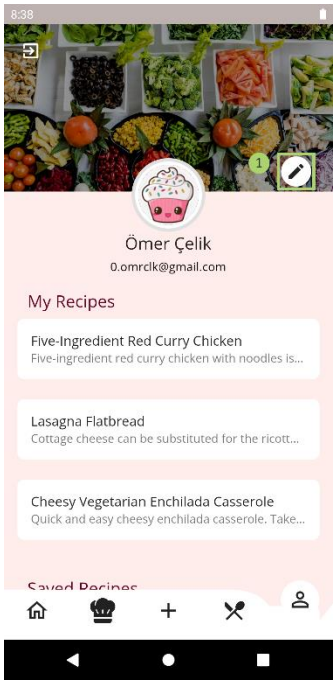
1. The other way to save recipe is you must navigate through to home page and there is a bookmark located in the upper left side of cards. A simple tap to this icon will save the recipe. And again, you can see the saved recipes on your profile page.



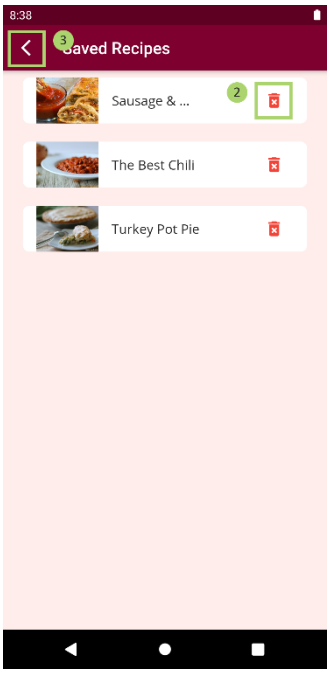
2. Once you save the recipe the button will turn into red and there will be a snack bar notification that tells you the recipe is saved.



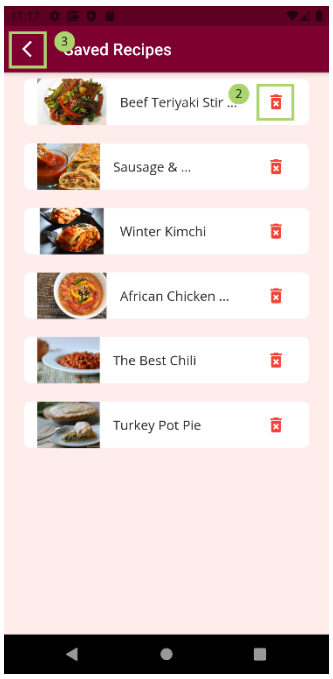
2. When you tapped on the button, the icon will turn into red, and recipe will be saved. You can check your saved recipes on your profile page.



1. To manage you saved recipe you must navigate through the profile page and tap on the pen button. This will redirect to you a new page.



3. When you are done with removing you can tap and go to previous page.



2. Once you come to this page you can see the recipes that we add on previous steps. To delete them tap on to trash icon it will remove the recipe from your list.

Appendix

CRUD Operation Full Code

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

final FirebaseFirestore _firestore = FirebaseFirestore.instance;
final CollectionReference _mainCollection = _firestore.collection('cookbook');

class Database {
  static String? userUid;

  static Future<void> addItem({
    required String title,
    required String summary,
    required String ingredient,
    required String instruction,
  }) async {
    DocumentReference documentReferencer =
        _mainCollection.doc(userUid).collection('recipes').doc();

    Map<String, dynamic> data = <String, dynamic>{
      "title": title,
      "summary": summary,
      "ingredients": ingredient,
      "instructions": instruction,
    };

    await documentReferencer
        .set(data)
        .whenComplete(() => print("Recipe added to the database"))
        .catchError((e) => print(e));
  }

  static Future<void> updateItem({
    required String title,
    required String summary,
    required String ingredient,
    required String instruction,
    required String docId,
  }) async {
    DocumentReference documentReferencer =
        _mainCollection.doc(userUid).collection('recipes').doc(docId);

    Map<String, dynamic> data = <String, dynamic>{
      "title": title,
      "summary": summary,
      "ingredients": ingredient,
      "instructions": instruction,
    };

    await documentReferencer
        .update(data)
        .whenComplete(() => print("Recipe updated in the database"))
        .catchError((e) => print(e));
  }

  static Stream<QuerySnapshot> readItems() {
    CollectionReference notesItemCollection =
        _mainCollection.doc(userUid).collection('recipes');

    return notesItemCollection.snapshots();
  }

  static Future<void> deleteItem({
    required String docId,
  }) async {
    DocumentReference documentReferencer =
```

```
        _mainCollection.doc(userUid).collection('recipes').doc(docId);

    await documentReferencer
        .delete()
        .whenComplete(() => print('Recipe deleted from the database'))
        .catchError((e) => print(e));
  }

  static Future<void> savedItem({
    required String id,
    required String title,
    required String imageUrl,
  }) async {
    DocumentReference documentReferencer =
    _mainCollection.doc(userUid).collection('saved_recipes').doc();

    Map<String, dynamic> data = <String, dynamic>{
      "savedID": id,
      "title": title,
      "imageUrl": imageUrl,
    };

    await documentReferencer
        .set(data)
        .whenComplete(() => print("Recipe saved"))
        .catchError((e) => print(e));
  }

  static Stream<QuerySnapshot> readSavedItem() {
    CollectionReference notesItemCollection =
    _mainCollection.doc(userUid).collection('saved_recipes');

    return notesItemCollection.snapshots();
  }

  static Future<void> deleteSavedItem({
    required String docId,
  }) async {
    DocumentReference documentReferencer =
    _mainCollection.doc(userUid).collection('saved_recipes').doc(docId);

    await documentReferencer
        .delete()
        .whenComplete(() => print('Recipe deleted from the database'))
        .catchError((e) => print(e));
  }
}
```