Mobile Application: Assignment 3

SOFE 4640U

Name	Student ID
Jean-Paul Saliba	100741759

A lab report submitted in fulfillment of the lab component of the Faculty of Software Engineering.

SOFE 4640U: Mobile Application Development (Fall 2023) Professor Akramul Azim

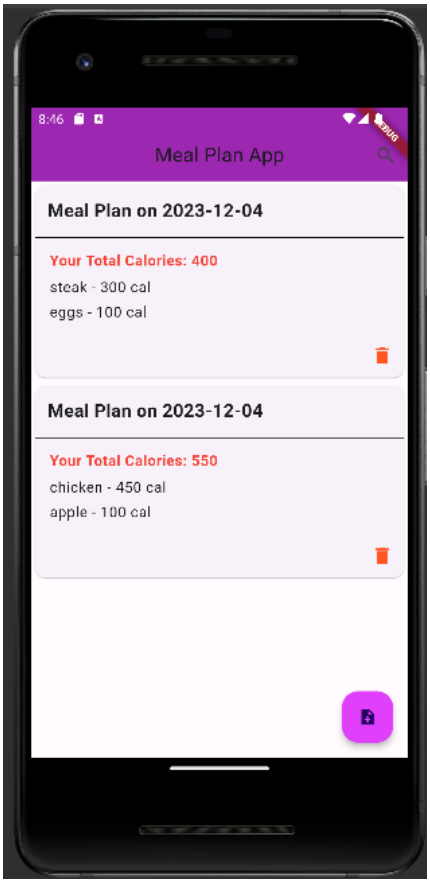
Objective

To practice Mobile application development utilzing Flutter and Dart to develop a food calorie tracker app. The main aim of the food calorie tracking app is to allow users to add meal plans for different days in the week and assist users with their diet by keeping track of all the calories they consume daily.

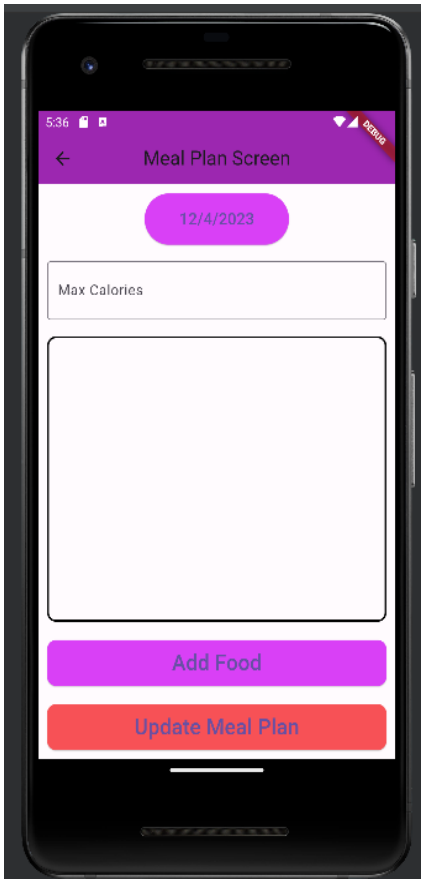
Github: https://github.com/Jp-s24/Mobile-Assignment-3

Screenshots

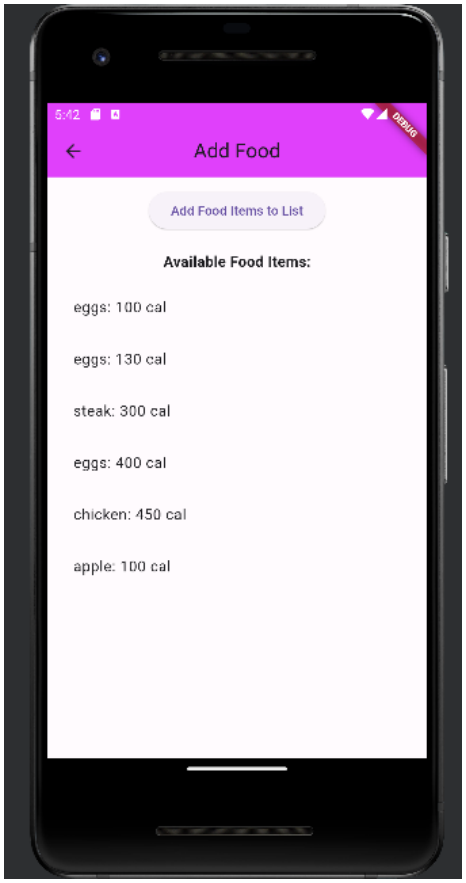
Home Page



Edit Meal Plan page



Add Meal Plan Page



Introduction

The mobile application was developed to provide users with an interactive and easy UI to use to users to keep track of their caloric intake. This app was built using Flutter and Dart which provides simple maintenance allowing the developer to apply all concepts covered in course lectures.

How Tasks Were Accomplished

A. Layouts

Homepage:

The primary purpose of the homepage is to offer users an engaging and interactive experience, allowing them to effortlessly access their current meal plan and view its associated date. Users can conveniently filter meal plans or individual meals using the interactive search bar, which queries through all existing meal plans.

Within the body, a Column is placed to deliver a seamless and linear design. The search bar, represented by a text box, precedes the ListView, where the meal plan is presented within card widgets. The layout is completed with a floating action button that, upon pressing, swiftly directs users to the meal plan screen.

Edit Meal Plan Page:

The Edit Meal Plan page is structured similarly to the home screen, encapsulating its components within a framework. Positioned at the top is the app bar, featuring a text section for entering the maximum calorie count. The chosen food items are organized using a ListView widget employing ListTile. A delete button allows users to effortlessly remove items at their discretion. The date functionality seamlessly integrates an elevated button invoking the select date function, which leverages the showDatePicker widget. Additionally, an "add food" button streamlines navigation to the add food page, broadening users' choices for selecting items.

Add Meal Plan Page:

Commencing with the app bar, an elevated button stands ready for the effortless addition of new food items. Upon activation, this button initiates an alert dialogue, facilitating user input and efficient storage of food items on the list. Subsequently, a thoughtfully organized ListView and ListTile format showcases an array of available foods for meal plans. The selection of an item from this list seamlessly integrates it into the meal plan, promptly guiding the user back to the previous screen.

B. Database Implementation

The database helper class was the key class that helped ensure all 20 preferred food items and calorie items were stored correctly in the database. The database contains several methods that were utilized to ensure the calorie tracker app was fully functional and interactive with all of the

app's components. I have methods such as buildFoodTable(), buildMealPlanTable(), addFood() and deleteFood().

Key Functions In the Database:

addFood() → This function allows the users to click on two input fields. The first input field is the food item and the second one is the number of calories the food item consists of. This function handles adding new items to the database and the user's meal plan. The functions implementation is shown below.

```
static Future<int> addFood(Food food) async {
 final db = await _getDB();
 return await db.insert("Food", food.toJson());
}
```

deleteMealPlan() → This function allows the user to delete any food or meal plans in the database. As shown on the home screen the trash icon on click will delete the entire meal plan that the user has created. This function associates the meal plan based on its ID to delete it as demonstrated below.

```
static Future<void> deleteMealPlan(int id) async {
 final db = await _getDB();
 await db.delete(
   'MealPlan',
   where: 'id = ?',
   whereArgs: [id],
 );
}
```

saveMealPlan() → This function saves the user's meal plan into the database and ensures the data is stored effectively. The function saves the meal plan using dynamic mapping and saves the three key fields date, item and totalCalories.

updateMealPlan() → This method updates the user's current meal plans on the home screen on click of the meal plans container to perform any edits. The user's data is repopulated into the database and the user is redirected to the edit meal plan page where they can make any changes to their meal plan.

queryData() → This function handles the search query on the homepage where the user can search for meal plans based on the name of the food in their existing meal plans, as well as the date the meal plan was created. The query method uses a dynamic list to query through the user's current data. The search icon is shown on the homepage and is located on the top right corner of the page.