



Mobile Application

SOFE 4640U

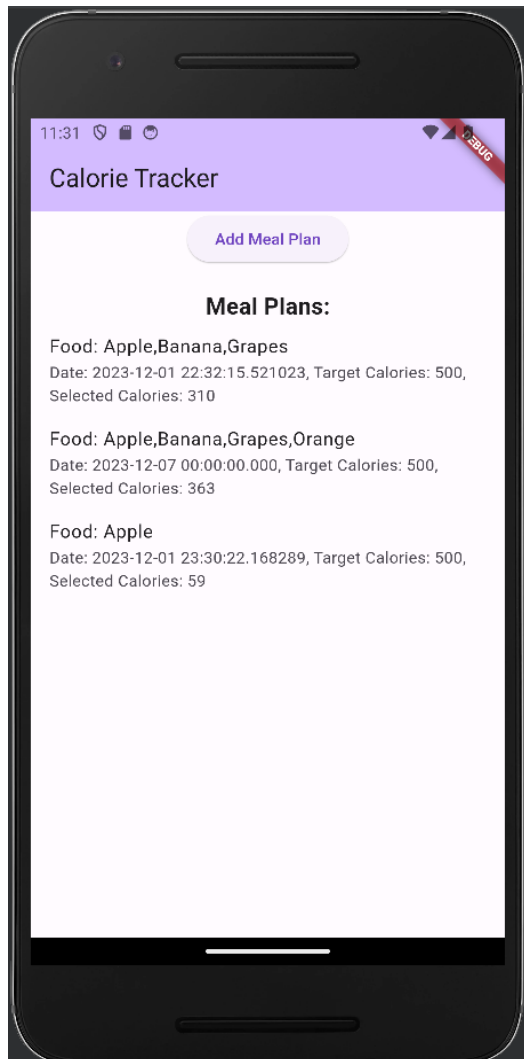
Assignment #2

Tanuj Patel

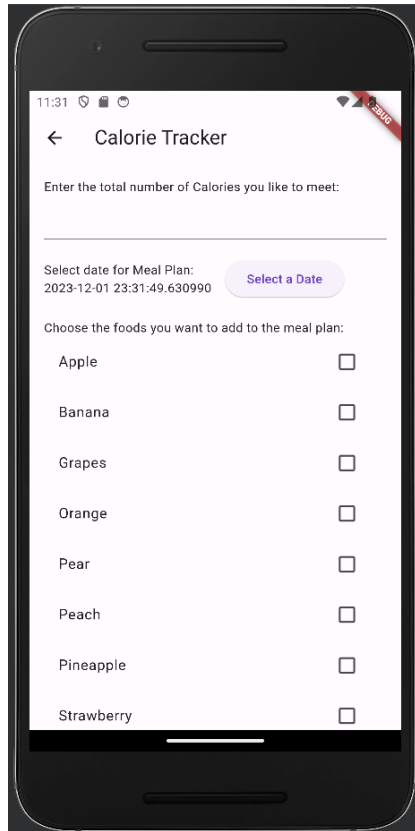
100749957

## Functionality:

Home Screen:



## Meal Plan Screen:



11:31

← Calorie Tracker

Enter the total number of Calories you like to meet:

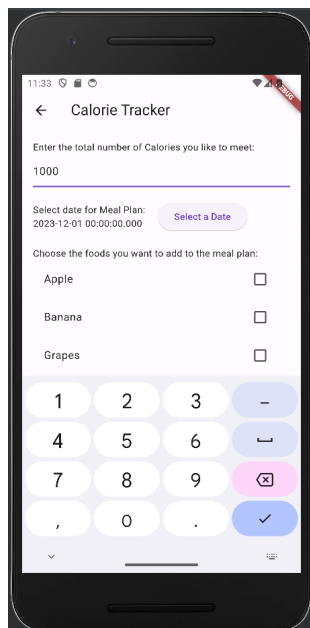
---

Select date for Meal Plan:  
2023-12-01 23:31:49.630990 [Select a Date](#)

Choose the foods you want to add to the meal plan:

Apple	<input type="checkbox"/>
Banana	<input type="checkbox"/>
Grapes	<input type="checkbox"/>
Orange	<input type="checkbox"/>
Pear	<input type="checkbox"/>
Peach	<input type="checkbox"/>
Pineapple	<input type="checkbox"/>
Strawberry	<input type="checkbox"/>

## Inputting calories:



11:33

← Calorie Tracker

Enter the total number of Calories you like to meet:

1000

---

Select date for Meal Plan:  
2023-12-01 00:00:00.000 [Select a Date](#)

Choose the foods you want to add to the meal plan:

Apple	<input type="checkbox"/>
Banana	<input type="checkbox"/>
Grapes	<input type="checkbox"/>

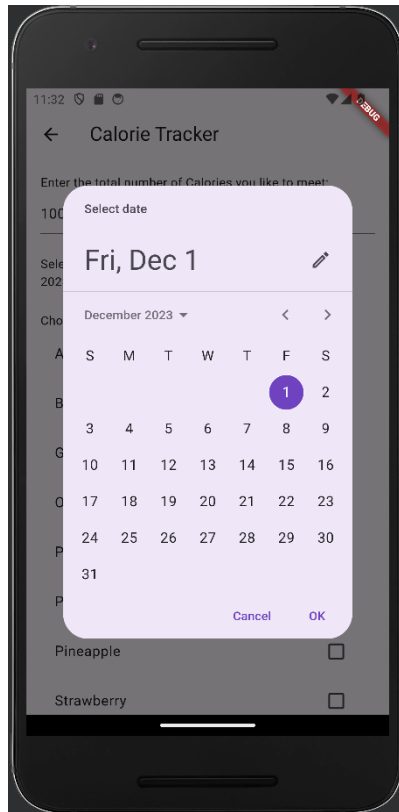
123-

456↵

789⌫

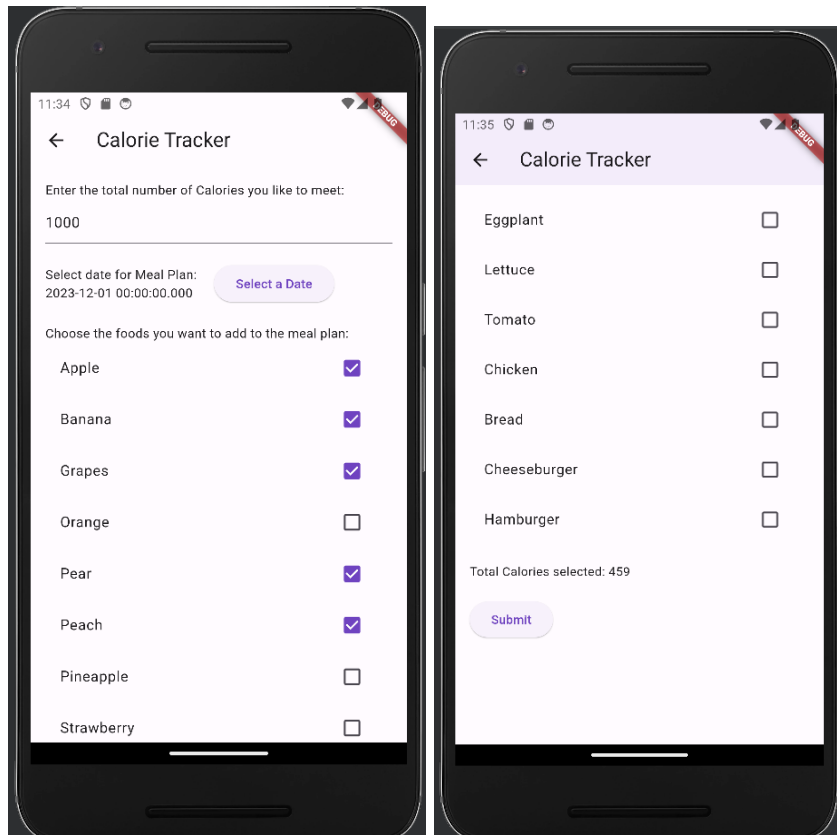
,0.✓

Choosing a date:



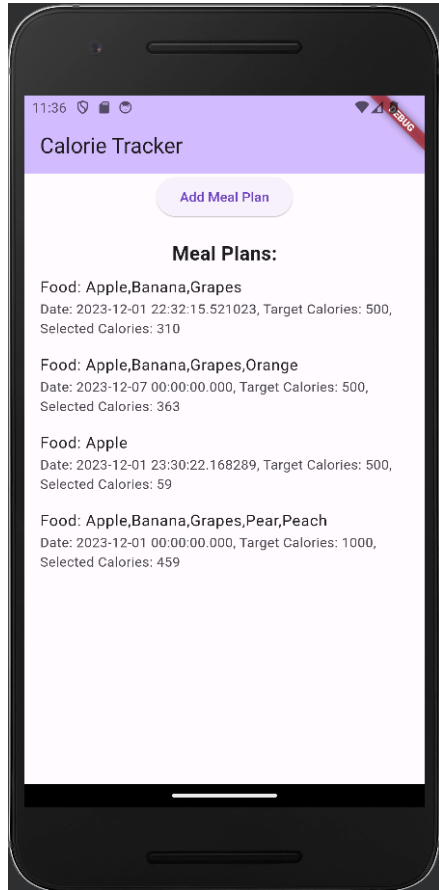
When choosing a date, the app will automatically display the current date, but user can then choose any date they want from 2023-2100.

Selecting food and showing list of calories:



For the list, i had an issue where the list will keep printing duplicates each time, so when you do run the code you will have to keep scrolling down and you will eventually get to the bottom of the screen as shown on the right picture. To scroll you will have to click and scroll along the very right hand side of the screen and not from the middle.

Updated home screen:



We can see that the new meal plan has been updated with the food that was selected, the date it was created for, the target calories the user inputted and what the total calories of the meal plan is.

## Code:

Dbhelper - database to store the foods and calories:

```

import 'dart:async';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';

class dbHelper {
  static final dbHelper _instance = dbHelper._();
  static Database? _database;

  dbHelper._();
  factory dbHelper() => _instance;

  Future<Database> get database async{
    if(_database != null) return _database!;

    _database=await initDatabase();
    return _database! ;
  }

  Future<Database> initDatabase() async{
    String path = join(await getDatabasesPath(), 'calories_db.db');
    return openDatabase(path,version: 1,onCreate: _createDatabase);
  }
}

```

```

Future<void> _createDatabase(Database db, int version) async{
  await db.execute('''
    CREATE TABLE food_calories(
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      food_name TEXT,
      calories INTEGER
    )
  ''');
}

Future<void> insertEntry(String food, int calories) async{
  final Database db = await database;
  await db.insert(
    'food_calories',
    {'food_name':food, 'calories':calories},
    conflictAlgorithm: ConflictAlgorithm.replace,
  );
}

```

```

Future<List<Map<String, dynamic>>> getEntry() async{
    final Database db = await database;
    return await db.query('food_calories');
}

Future<List<Map<String, dynamic>>> getAllFoods() async {
    final Database db = await database;
    return await db.query('food_calories');
}
}

```

The first step we did was create a dart file for the database helper that will store the foods and calories. We first created a table by defining the table names and the names of the columns for what we will be storing(food, calories). We then created methods to methods to insert the foods and calories from the main functions into the database and a get method to get the foods and calories when called.

Main dart file:

```

import 'package:flutter/material.dart';
import 'package:mobile_assignment3/dbhelper.dart';
import 'package:mobile_assignment3/mealplan_dbhelper.dart';

void main() {
    runApp( MyApp());
}

class MyApp extends StatelessWidget {
    //const MyApp({super.key});

    // This widget is the root of your application.
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            title: 'Calories Tracker',
            theme: ThemeData(

                colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
                useMaterial3: true,
            ), // ThemeData
            home: MyHomePage(),
        ); // MaterialApp
    }
}

```



```

class MyHomePage extends StatefulWidget {
  //const MyHomePage({super.key, required this.title});
  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {

  final dbHelper helper = dbHelper();
  final mealplan_dbhelper mealHelper = mealplan_dbhelper();

  List<Map<String, dynamic>> foodEntries = [];
  List<Map<String, dynamic>> mealPlans = [];

  @override
  void initState(){
    super.initState();
    _initializeDb();
  }

  Future<void> _initializeDb() async{

```

```

Future<void> _initializeDb() async{

  List<String> foodName = ['Apple','Banana','Grapes','Orange','Pear','Peach','Pineapple','Strawberry','Watermelon','Asparagus',
    'Broccoli','Carrots','Cucumber','Eggplant','Lettuce','Tomato','Chicken', 'Bread','Cheeseburger','Hamburger'];
  List<int> calories = [59,151,100,53,82,67,82,53,50,27,45,50,17,35,5,22,136,75,285,250];

  for(int i=0; i<foodName.length;i++){
    await helper.insertEntry(foodName[i], calories[i]);
  }

}

Future<void> _initializeData() async{
  foodEntries = await helper.getAllFoods();
  mealPlans = await mealHelper.getMealPlans();
  setState({});
}

```

```

@override
Widget build(BuildContext context) {

  return Scaffold(
    appBar: AppBar(
      // TRY THIS: Try changing the color here to a specific color (to
      // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
      // change color while the other colors stay the same.
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      // Here we take the value from the MyHomePage object that was created by
      // the App.build method, and use it to set our appbar title.
      title: Text("Calorie Tracker"),
    ), // AppBar
    body:

    Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Center(
          child: ElevatedButton(
            onPressed: ()async{
              await Navigator.push(context, MaterialPageRoute(builder: (context)=> mealPlanScreen(helper: helper),));
            },
          ),
        ),
      ],
    ),
  );
}

```

```

    SizedBox(height: 20),
    Text('Meal Plans:', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
    Expanded(
      child: FutureBuilder<List<Map<String, dynamic>>>(
        future: mealHelper.getMealPlans(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return CircularProgressIndicator();
          } else if (snapshot.hasError) {
            return Text('Error: ${snapshot.error}');
          } else {
            List<Map<String, dynamic>> mealPlans = snapshot.data!;
            return ListView.builder(
              itemCount: mealPlans.length,
              itemBuilder: (context, index) {
                var mealPlan = mealPlans[index];
                return ListTile(
                  title: Text('Food: ${mealPlan['food_name']}'),
                  subtitle: Text('Date: ${mealPlan['date']}', '
                    'Target Calories: ${mealPlan['target_calories']}', '
                    'Selected Calories: ${mealPlan['selected_calories']}' ), // Text
                ); // ListTile
              },
            );
          }
        },
      ),
    ),
  );
}

```

```
class mealPlanScreen extends StatefulWidget{
  final dbHelper helper;
  const mealPlanScreen({Key? key, required this.helper}) : super(key: key);

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

class _mealPlanScreenState extends State<mealPlanScreen> {
  late int totalCal;
  late DateTime dateSelected;
  List<String> selectedFoodItems =[];
  int totalCaloriesSelected =0;

  @override
  void initState(){
    super.initState();
    totalCal=0;
    dateSelected = DateTime.now();
  }
}
```

```

Future<void> _submitMealPlan() async{
  await mealplan_dbhelper().insertMealPlan(
    foodName: selectedFoodItems.join(', '),
    date: dateSelected.toLocal().toString(),
    targetCalories: totalCal,
    selectedCalories: totalCaloriesSelected,
  );

  Navigator.pop(context);
}

Future<void> _selectDate(BuildContext context) async{
  final DateTime? picked = await showDatePicker(
    context: context,
    initialDate: dateSelected,
    firstDate: DateTime(2023),
    lastDate: DateTime(2100),
  );
  if(picked != null && picked!=dateSelected){
    setState(() {
      dateSelected=picked;
    });
  }
}

```

```
@override
Widget build(BuildContext context){
  return Scaffold(
    appBar: AppBar(
      title: Text('Calorie Tracker'),
    ), // AppBar
    body: SingleChildScrollView(
      child: ConstrainedBox(
        constraints: BoxConstraints(
          minHeight: MediaQuery.of(context).size.height,
        ), // BoxConstraints

        child: Padding(
          padding: const EdgeInsets.all(16.0),

          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text('Enter the total number of Calories you like to meet:'),
              TextField(
                keyboardType: TextInputType.number,
                onChanged: (value){
                  setState(() {
```

```

), // TextField
— SizedBox(height: 20),
— Row(
  children: [
    — Text('Select date for Meal Plan: \n$dateSelected'),
    — SizedBox(width: 15),
    — ElevatedButton(
      onPressed: () => _selectDate(context),
      — child: Text('Select a Date'),
    ), // ElevatedButton
  ],
), // Row
— SizedBox(height: 20),
— Text('Choose the foods you want to add to the meal plan:'),
— FutureBuilder<List<Map<String, dynamic>>>(
  future: widget.helper.getEntry(),
  builder: (context, snapshot){
    — if(snapshot.connectionState == ConnectionState.waiting){
    — return CircularProgressIndicator();
    }else if(snapshot.hasError){
    — return Text('Error: ${snapshot.error}');
  }
)

```

```

} else if (snapshot.hasError) {
  return Text('Error: ${snapshot.error}');
} else {
  List<Map<String, dynamic>> listFood = snapshot.data!;
  return ListView.builder(
    shrinkWrap: true,
    itemCount: listFood.length,
    itemBuilder: (context, index) {
      var entry = listFood[index];
      return CheckboxListTile(
        title: Text(entry['food_name']),
        value: selectedFoodItems.contains(entry['food_name']),
        onChanged: (bool? value) {
          setState(() {
            if (value!) {
              selectedFoodItems.add(entry['food_name']);
              totalCaloriesSelected +=
                (entry['calories'] as int);
            } else {
              selectedFoodItems.remove(entry['food_name']);
              totalCaloriesSelected -=
                (entry['calories'] as int);
            }
          });
        }
      );
    }
  );
}

```

In the main dart file, we first initialized the database for the dbHelper that we created to store the foods and calories. I then created 2 arrays one for the foods and one for the calories that acted as a parallel array. I then went through the arrays using a for loop and entered the food and calories into the array using the index i from the for loop. In the main screen I then created a button that the user can click that will navigate them to the new screen where they can make their meal plan. In the new screen I made a text input field that only takes numbers to get the user to input the target calories and then make a button that when pressed will open a calendar for the user to input the date. TO display all the foods I created a checkbox list that the user can check multiple boxes for the food that they want to select and while they do this at the bottom i made a text view that prints the updated calories based on what food items the user selects. This text view will keep being updated when the user clicks on unclick an food item. I then created a submit button that will first vcalidate and ensure that the total calories selected does not exceed the target calories the user inputted, and if it does exceed a error message is printed and wont let the user move forward to the next page, but if the total calories does not exceed the target then it will save the foods selected, the date, the target calories and total calores to a new database and then display this in the main page by creating a listview that will call this database and get the values to print the meal plan.

mealPlan database helper file:

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class mealplan_dbhelper {
  static final mealplan_dbhelper _instance = mealplan_dbhelper._();
  static Database? _database;

  mealplan_dbhelper._();
  factory mealplan_dbhelper() => _instance;

  Future<Database> get database async{
    if(_database != null) return _database!;

    _database = await initDatabase();
    return _database!;
  }
  Future<Database> initDatabase() async{
    String path = join(await getDatabasesPath(), 'mealplan_db.db');
    return openDatabase(path, version: 1, onCreate: _createDb);
  }
}
```



```
Future<void> _createDb(Database db, int version) async{
    await db.execute('''
        CREATE TABLE mealplan(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            food_name TEXT,
            date TEXT,
            target_calories INTEGER,
            selected_calories INTEGER
        )
    ''');
}

Future<void> insertMealPlan({
    required String foodName,
    required String date,
    required int targetCalories,
    required int selectedCalories,
}) async{
    final Database db= await database;
    await db.insert('mealplan',
        {
            'food_name': foodName,
```

```

})) async{
    final Database db= await database;
    await db.insert('mealplan',
        {
            'food_name': foodName,
            'date' : date,
            'target_calories' : targetCalories,
            'selected_calories' : selectedCalories,
        },
        conflictAlgorithm: ConflictAlgorithm.replace,
    );
}

Future<List<Map<String, dynamic>>> getMealPlans() async{
    final Database db = await database;
    return await db.query('mealplan');
}
}

```

Next I then created another database helper file but this time to create a new database to store the meal plan. I first created a table with the names needed, and then created a method to insert the values received from the meal plan page upon clicking submit and insert the values into the database it self

NOTE: I had errors with trying to get the meal plan to open onto a new screen when clicked, so I was not able to implement this functionality, and I was also not able to implement the search query due to the fact I took too much time trying to solve the problem with clicking the list view.