Iteration 2

Group E

Iteration Roadmap

- 0.1: Initial calorie tracking functionality
- 0.2: Database connection, UI refinements, backend code improvements, barcode scanning
- 0.3: Sleep tracking, feature refinements
- 0.4: Weight tracking, home screen with graphs and data overview

Goals for Iteration 2

- Able to query database
- Input user information

Iteration 2: Database

Xampp → **SQL** → **SQLITE**

Initially, we started out using Xampp. We realized MySQL and PHP is for one server we would have to manage, where as SQLite is a database on the users phone.

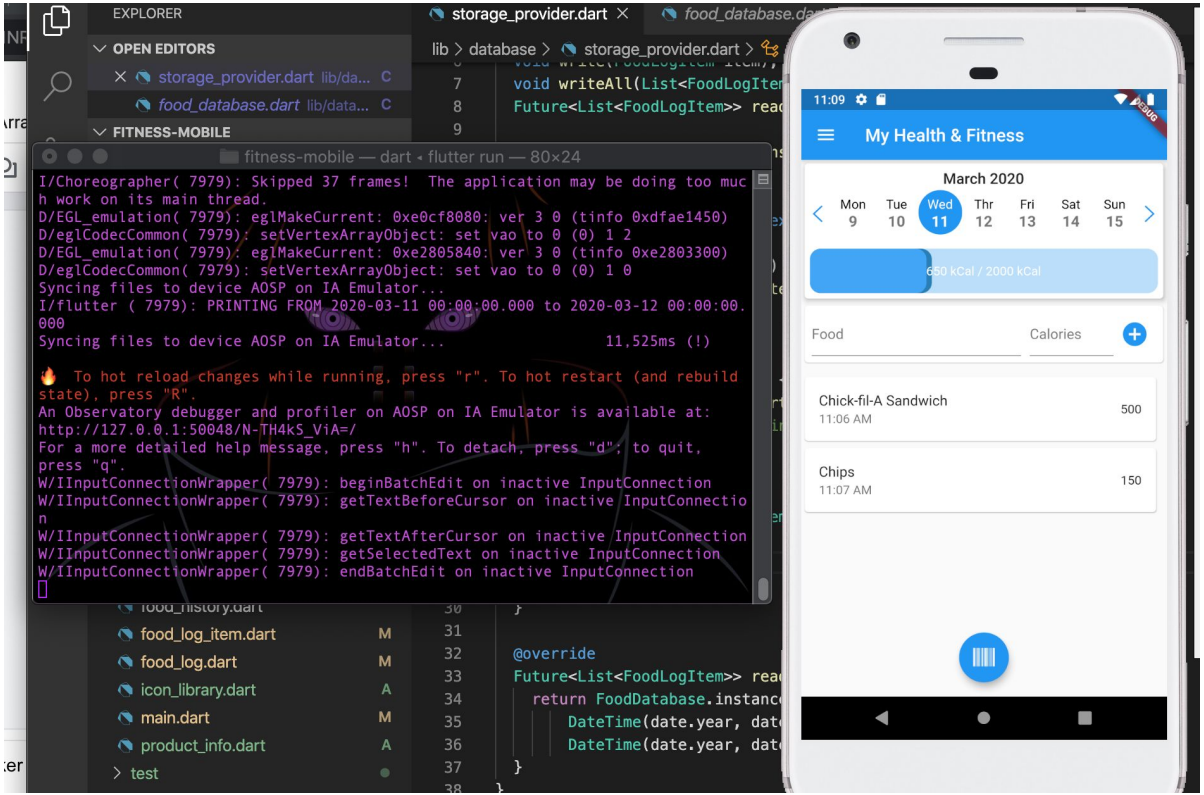
Iteration 2: Database Helper Functions

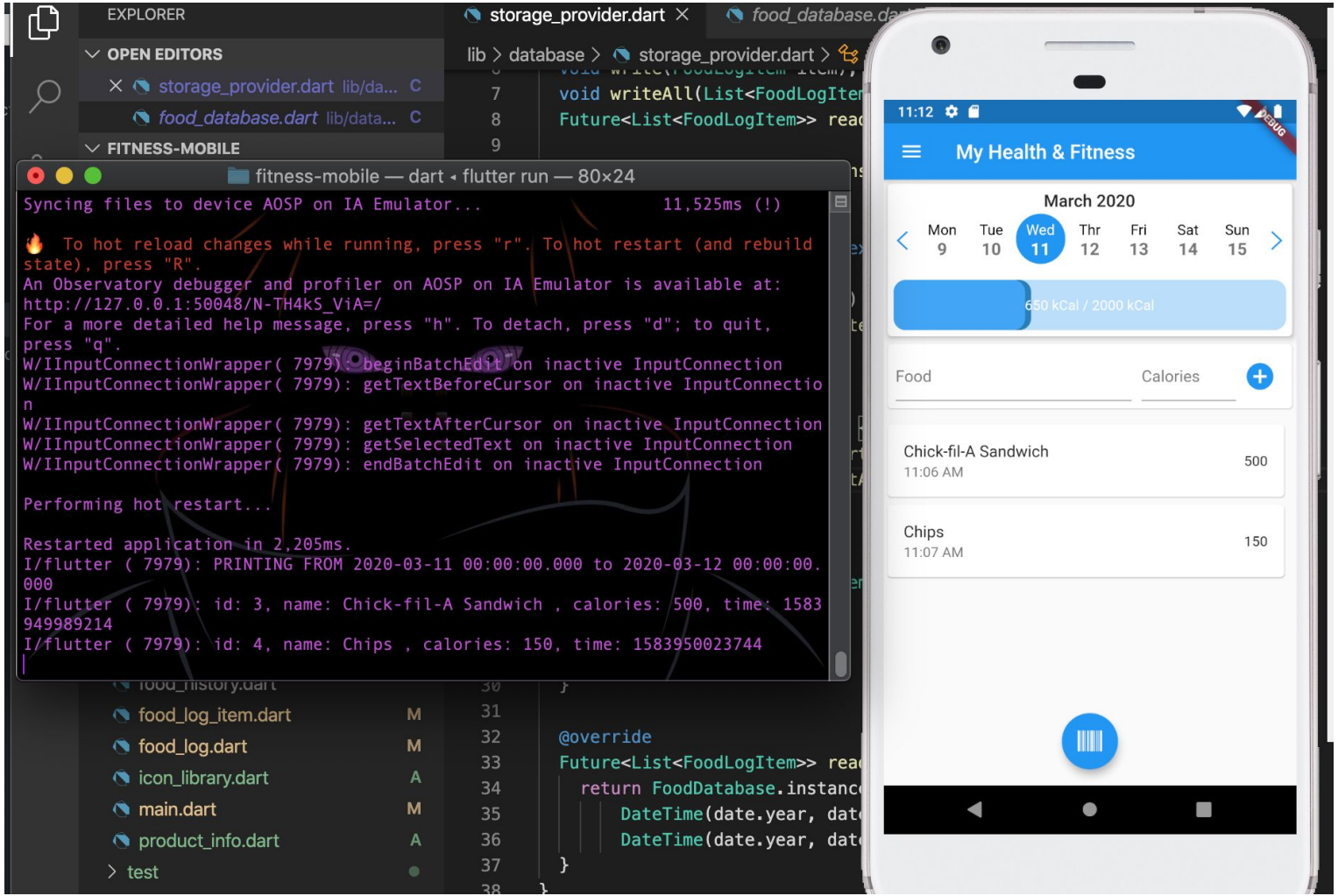
```dart
import 'dart:io';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import 'package:path_provider/path_provider.dart';
import '../food_log_item.dart';

class FoodDatabase {
  // DB info
  static final _databaseName = "dbfood.db";
  static final _databaseVersion = 1;
  static final tableName = 'food';

  // DB columns
  static final columnId = '_id';
  static final columnName = 'name';
  static final columnCalories = 'calorie';
  static final columnTime = 'time';

  // Singleton ~~ only one instance
  FoodDatabase._privateConstructor();
  static final FoodDatabase instance = FoodDatabase._privateConstructor();

  // Database reference
  static Database _database;
  Future<Database> get database async {
    if (_database != null) return _database;
    // initialize database if it is not created yet
    _database = await _initDatabase();
    return _database;
  }
```

```dart
  // Opens/Creates database
  _initDatabase() async {
    Directory documentsDirectory = await getApplicationDocumentsDirectory();
    String path = join(documentsDirectory.path, _databaseName);
    return await openDatabase(path,
        version: _databaseVersion, onCreate: _onCreate);
  }

  // SQL code to create the database table_name
  Future _onCreate(Database db, int version) async {
    await db.execute('''
          CREATE TABLE $tableName (
            $columnId INTEGER PRIMARY KEY,
            $columnName TEXT NOT NULL,
            $columnCalories INTEGER NOT NULL,
            $columnTime INTEGER NOT NULL
          )
          ''');
  }

  // Helper methods

  // Inserts a row in the database where each key in the Map is a column name
  // and the value is the column value. The return value is the id of the
  // inserted row.
  Future<int> insert(FoodLogItem food) async {
    Database db = await instance.database;
    Map<String, dynamic> row = {
      columnName: '${food.name}',
      columnCalories: '${food.calories}',
      columnTime: '${food.time.millisecondsSinceEpoch}'
```

```dart
// All of the rows are returned as a list of maps, where each map is
// a key-value list of columns.
Future<void> printAllRows() async {
  Database db = await instance.database;
  var table = await db.query(tableName);
  table.forEach((row) {
    print(
        'id: ${row["$columnId"]}, name: ${row["$columnName"]}, calories: ${row["$columnCalorie
  });
}

Future<void> printRow(var row) async {
  print(
      'id: ${row["$columnId"]}, name: ${row["$columnName"]}, calories: ${row["$columnCalories'
}

Future<List<FoodLogItem>> queryBetweenDates(
    DateTime leftDate, DateTime rightDate) async {
  Database db = await instance.database;
  var leftEpoch = leftDate.millisecondsSinceEpoch;
  var rightEpoch = rightDate.millisecondsSinceEpoch;
  var rows = await db.rawQuery(
      'SELECT * FROM $tableName WHERE $columnTime BETWEEN $leftEpoch AND $rightEpoch');
  List<FoodLogItem> items = new List<FoodLogItem>();
  print('PRINTING FROM $leftDate to $rightDate');
  rows.forEach((row) {
    items.add(FoodLogItem(row[columnName], row[columnCalories],
        DateTime.fromMillisecondsSinceEpoch(row[columnTime]))); // FoodLogItem
    print(
        'id: ${row["$columnId"]}, name: ${row["$columnName"]}, calories: ${row["$columnCalorie
  });
  return items;
```

```dart
import 'package:csuf_fitness/food_log_item.dart';
import 'food_database.dart';

abstract class StorageProvider {
  void delete(FoodLogItem item);
  void write(FoodLogItem item);
  void writeAll(List<FoodLogItem> items);
  Future<List<FoodLogItem>> read(DateTime date);

  static StorageProvider get instance => DatabaseStorageProvider();
}

class DatabaseStorageProvider extends StorageProvider {
  @override
  void delete(FoodLogItem item) async {
    FoodDatabase.instance.deleteByTimestamp(item.time);
  }

  @override
  void write(FoodLogItem item) {
    FoodDatabase.instance.insert(item);
    FoodDatabase.instance.printAllRows();
  }

  @override
  void writeAll(List<FoodLogItem> items) {
    items.forEach((item) {
      write(item);
    });
  }

  @override
  Future<List<FoodLogItem>> read(DateTime date) async {
    return FoodDatabase.instance.queryBetweenDates(
        DateTime(date.year, date.month, date.day),
```

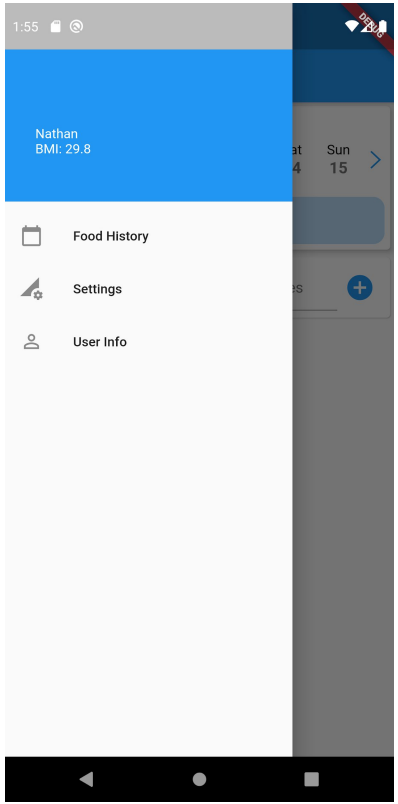

EXPLORER
storage_provider.dart
food_database.dart
OPEN EDITORS
storage_provider.dart lib/da... C
food_database.dart lib/data... C
FITNESS-MOBILE
lib > database > storage_provider.dart
7 void writeAll(List<FoodLogIte
8 Future<List<FoodLogItem>> rea
9
fitness-mobile — dart ◂ flutter run — 80×24
I/Choreographer(7979): Skipped 37 frames! The application may be doing too muc
h work on its main thread.
D/EGL_emulation(7979): eglMakeCurrent: 0xe0cf8080: ver 3 0 (tinfo 0xdfae1450)
D/eglCodecCommon(7979): setVertexArrayObject: set vao to 0 (0) 1 2
D/EGL_emulation(7979): eglMakeCurrent: 0xe2805840: ver 3 0 (tinfo 0xe2803300)
D/eglCodecCommon(7979): setVertexArrayObject: set vao to 0 (0) 1 0
Syncing files to device AOSP on IA Emulator...
I/flutter (7979): PRINTING FROM 2020-03-11 00:00:00.000 to 2020-03-12 00:00:00.
000
Syncing files to device AOSP on IA Emulator... 11,525ms (!)
To hot reload changes while running, press "r". To hot restart (and rebuild
state), press "R".
An Observatory debugger and profiler on AOSP on IA Emulator is available at:
http://127.0.0.1:50048/N-TH4kS_ViA=/
For a more detailed help message, press "h". To detach, press "d"; to quit,
press "q".
W/IInputConnectionWrapper(7979): beginBatchEdit on inactive InputConnection
W/IInputConnectionWrapper(7979): getTextBeforeCursor on inactive InputConnectio
n
W/IInputConnectionWrapper(7979): getTextAfterCursor on inactive InputConnection
W/IInputConnectionWrapper(7979): getSelectedText on inactive InputConnection
W/IInputConnectionWrapper(7979): endBatchEdit on inactive InputConnection
food_log_item.dart M
food_log.dart M
icon_library.dart A
main.dart M
product_info.dart A
test
30 }
31
32 @override
33 Future<List<FoodLogItem>> rea
34 return FoodDatabase.instanc
35 DateTime(date.year, dat
36 DateTime(date.year, dat
37 }
38 }
11:09
My Health & Fitness
DEBUG
March 2020
Mon 9
Tue 10
Wed 11
Thr 12
Fri 13
Sat 14
Sun 15
650 kCal / 2000 kCal
Food
Calories
Chick-fil-A Sandwich
11:06 AM
500
Chips
11:07 AM
150

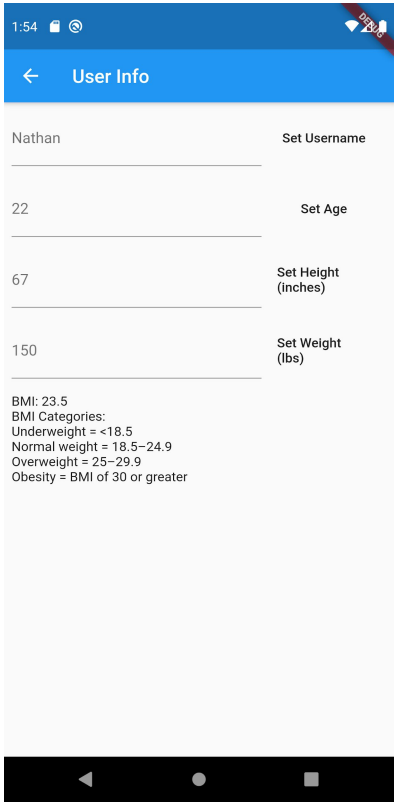

EXPLORER
OPEN EDITORS
storage_provider.dart lib/da... C
food_database.dart lib/data... C
FITNESS-MOBILE
storage_provider.dart
food_database.da
lib > database > storage_provider.dart
7 void writeAll(List<FoodLogIter
8 Future<List<FoodLogItem>> read
9
fitness-mobile — dart • flutter run — 80×24
Syncing files to device AOSP on IA Emulator... 11,525ms (!)
To hot reload changes while running, press "r". To hot restart (and rebuild state), press "R".
An Observatory debugger and profiler on AOSP on IA Emulator is available at: http://127.0.0.1:50048/N-TH4kS_ViA=/
For a more detailed help message, press "h". To detach, press "d"; to quit, press "q".
W/IInputConnectionWrapper(7979): beginBatchEdit on inactive InputConnection
W/IInputConnectionWrapper(7979): getTextBeforeCursor on inactive InputConnection
W/IInputConnectionWrapper(7979): getTextAfterCursor on inactive InputConnection
W/IInputConnectionWrapper(7979): getSelectedText on inactive InputConnection
W/IInputConnectionWrapper(7979): endBatchEdit on inactive InputConnection
Performing hot restart...
Restarted application in 2,205ms.
I/flutter (7979): PRINTING FROM 2020-03-11 00:00:00.000 to 2020-03-12 00:00:00.000
I/flutter (7979): id: 3, name: Chick-fil-A Sandwich , calories: 500, time: 1583949989214
I/flutter (7979): id: 4, name: Chips , calories: 150, time: 1583950023744
food_log_item.dart M
food_log.dart M
icon_library.dart A
main.dart M
product_info.dart A
test
30 }
31
32 @override
33 Future<List<FoodLogItem>> read
34 return FoodDatabase.instanc
35 DateTime(date.year, dat
36 DateTime(date.year, dat
37 }
38 }
11:12
My Health & Fitness
DEBUG
March 2020
Mon 9
Tue 10
Wed 11
Thr 12
Fri 13
Sat 14
Sun 15
650 kCal / 2000 kCal
Food
Calories
Chick-fil-A Sandwich
11:06 AM
500
Chips
11:07 AM
150

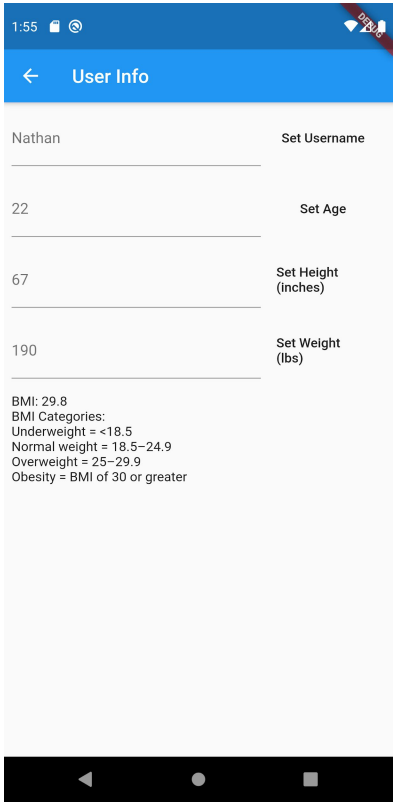
Demo: User Info

- Input user information: Name, Age, Height, Weight
- Calculates user’s Body Mass Index (BMI)
- Displays BMI Categories

Demo: User Info (Android Emulator)







Goals For Next Iteration: User Info

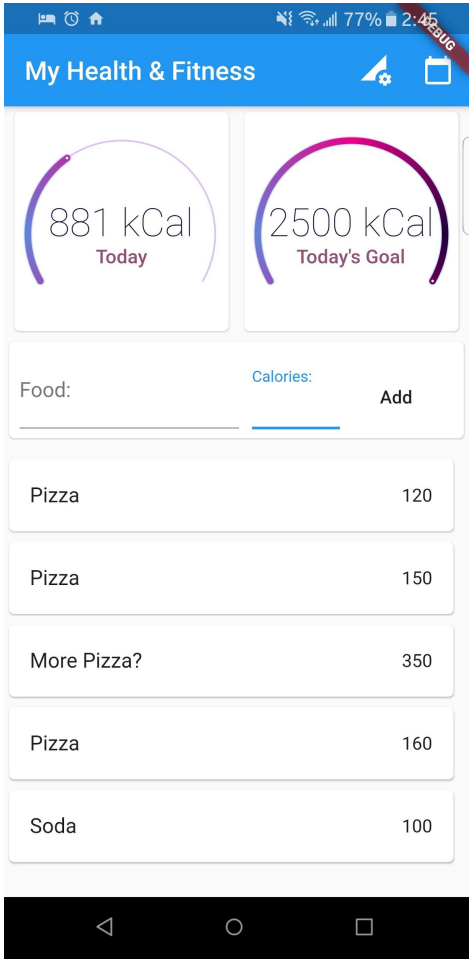
- Database for user login with email
- Track weight/BMI history, similar to the food history
- Weight Goal
- Integrate Metric scale
- Sleep Tracker
- Add a Distance Traveled widget
    - Floors scaled
    - Calories burned
    - Time spent traveling

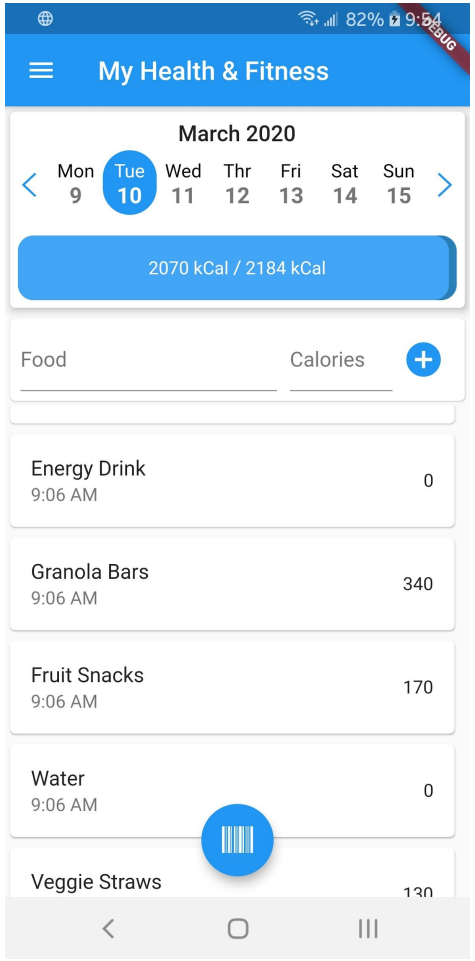
Iteration 2 Improvements

1. Backend code
    a. Full connectivity of database and user interface
    b. Near total refactor of all state handling code to improve maintainability and extensibility
2. Frontend code
    a. Major main UI refresh to improve look and feel of app
3. New features
    a. Barcode scanning
    b. Dark mode
    c. New settings page

UI Refresh

- Added a week view header that allows viewing other days
- Card and drop shadows around the header
- Floating action button for barcode scanning
- Animated bar display of current calorie values





New Features

1. Barcode Scanning
    a. Automatically adds items to the list, including the name and calorie count if available
    b. Pulls data from USDA Food Data Central API
    c. More data sources (Open Food Facts API) coming soon
2. Dark mode
3. Settings Page