Mobile Computing

CSCI5708

Project Report

BeFIT

Track. Eat. Achieve

Team Members:

Manish Jadhav - B00969328

Manish Jadhav B00969328

Abstract

BeFit is a smart meal planning and calorie tracking app designed to help users achieve their health and fitness goals [1]. It features Firebase authentication, a calorie calculator, personalized diet plans, and a barcode scanner for food using the device camera. Users can track daily intake through dynamic pie and bar charts, scan and log food items using the OpenFoodFactory API, and manage meals with intuitive swipe gestures. The app supports a wide range of dietary preferences and restrictions, adjusting plans based on user goals such as weight loss, gain, or maintenance. With real-time progress tracking, Firebase integration, and a user-friendly interface, BeFit simplifies and personalizes nutrition management for both beginners and fitness enthusiasts alike.

Completion report

All the requirements mentioned for the BeFit application have been successfully completed. Below is a breakdown of how each module and functionality was achieved:

Authentication Module

The Authentication Module is complete. I have implemented a combination of Firebase Authentication and local session storage to manage user login and signup [3]. Users can securely sign up or log in using their email and password. Once authenticated, their session is saved locally using AsyncStorage, allowing automatic login for returning users until they explicitly log out.

API Integration

I have successfully integrated two external APIs:

1. **Spoonacular API** – This API is used to generate personalized diet plans based on user input parameters such as calorie goals and dietary preferences (e.g., vegetarian, eggetarian, vegan) [4].
2. **OpenFoodFactory API** – This API powers the food scanning feature. When a user scans a product using their camera, the app fetches detailed nutritional information including calories and macronutrients from this API.

Data Storage – Simple and Complex Data

BeFit uses both local and cloud-based storage for efficient data handling:

- **Local Storage (AsyncStorage)** is used to save simple data like the username after login, allowing users to skip the login screen on future app launches.
- **Firebase Firestore** is used for storing complex data such as user preferences, calorie goals, selected diet plans, daily meals, and graph data for nutrition tracking. This ensures persistent and synchronized data across sessions and devices.

Native Features

BeFit utilizes the device's **native camera** to enable the food barcode scanning feature [2]. This native integration enhances usability by allowing users to quickly scan items and retrieve nutritional data, making food logging effortless.

Native Maps Integration

Currently, there is no requirement for map integration in BeFit. However, to enhance user interaction and experience, I have implemented **native animations and gestures**. For example:

- A **cookie crumbs animation** is used during app launch to replace the traditional loading screen.
- Custom gestures are implemented for meal tracking—users can swipe right to mark a meal as eaten and left to remove it.
- **Charts and visualizations** show real-time data for daily calorie and macronutrient goals.

Final App

The BeFit application has been thoroughly tested on both Android and iOS devices to ensure consistent performance, smooth UI interactions, and compatibility across platforms. All core features are fully functional and provide a seamless user experience.

Challenges

1. Barcode Scanner Integration

One of the most challenging tasks was integrating the barcode scanner using React Native's official documentation. The implementation consistently failed with unknown errors. After extensive debugging, online searches, and even consulting ChatGPT, the issue persisted.

Root Cause & Fix:
Eventually, I found a forum post identifying a **spelling mistake in the React Native documentation** related to importing the barcode scanner module. Once the correct import statement was used, the scanner worked as expected.

2. Homepage Rendering Twice

Another tricky issue occurred where the homepage rendered twice after login.

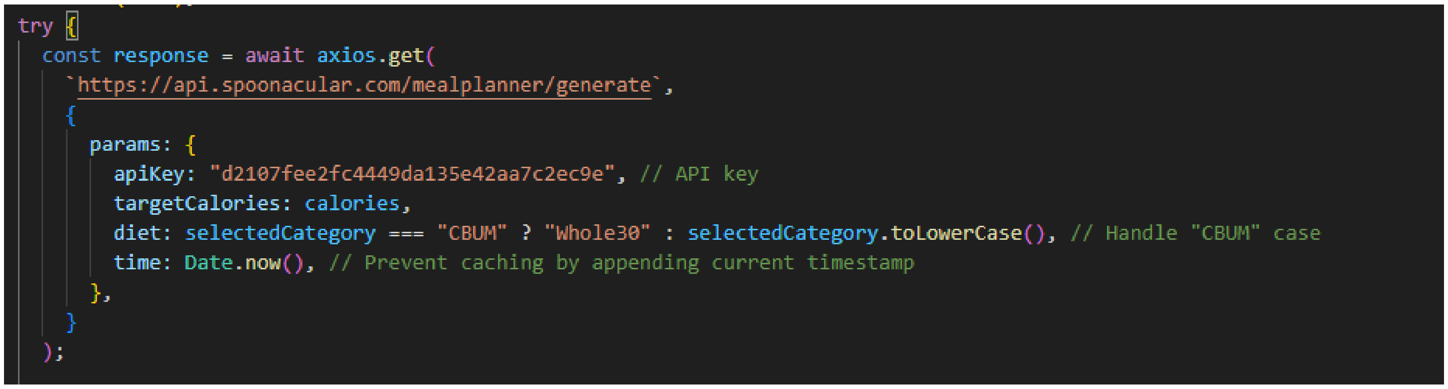
Root Cause:
This was caused by the onAuthStateChanged function from Firebase triggering navigation multiple times.

Solution:
I introduced a hasNavigatedRef flag to track whether navigation had already occurred. The issue was fully resolved by setting hasNavigatedRef.current = true **before** triggering navigation, ensuring the homepage rendered only once

3. Repeated Meals in Meal Plan (Bonus Lab Feedback)

During a bonus lab demo, a TA pointed out that the meal plan sometimes showed **repeated meals**. Upon investigation, I discovered that **Spoonacular occasionally returns cached responses**, leading to duplicate meal entries.

Solution:
To fix this, I appended a timestamp to the API request using:



```
try {
  const response = await axios.get(
    `https://api.spoonacular.com/mealplanner/generate`,
    {
      params: {
        apiKey: "d2107fee2fc4449da135e42aa7c2ec9e", // API key
        targetCalories: calories,
        diet: selectedCategory === "CBUM" ? "Whole30" : selectedCategory.toLowerCase(), // Handle "CBUM" case
        time: Date.now(), // Prevent caching by appending current timestamp
      },
    }
  );
```

Figure 1: Solving Problem

Additionally, I used a **Map data structure** to store only **unique meals**, ensuring each meal plan is distinct and not repeated.

4. Daily Calorie Reset at 12 AM

Though not a major challenge, I implemented a useful feature in Firebase that resets the user's **current calorie count to 0 at 12 AM** daily. This ensures accurate tracking and gives users a fresh start each day without manual input.

<u>Reflection</u>

Coming from a backend development background, working on **BeFit** was both a rewarding and challenging experience. I'm not typically a frontend-focused developer, so designing an intuitive and visually appealing user interface pushed me outside of my comfort zone. Since the app needed to feel smooth and user-friendly, I spent considerable time researching existing fitness apps to understand common patterns and best practices. This helped guide my design decisions and improve the overall UX.

The UI design process was iterative. My first version didn't meet my own expectations, so I revised it multiple times —eventually finalizing the structure and color schemes after feedback from my TA and professor. While I was confident in implementing backend logic and Firebase integration, the frontend structure and layout took significant effort and learning.

One of the biggest technical learnings for me was working with **React Native's camera modules** for the first time. Implementing the food scanning feature introduced me to native device integrations, especially barcode scanning using react-native-camera, which deepened my understanding of how React Native interacts with native APIs.

Overall, this project helped me grow as a **full-stack mobile developer**, giving me hands-on experience across both frontend UI/UX and backend logic, and helping me build a complete, user-focused application from start to finish.

Images

Here's the difference between what I initially imagined my design would look like in the mockups, and what the result turned out to be.

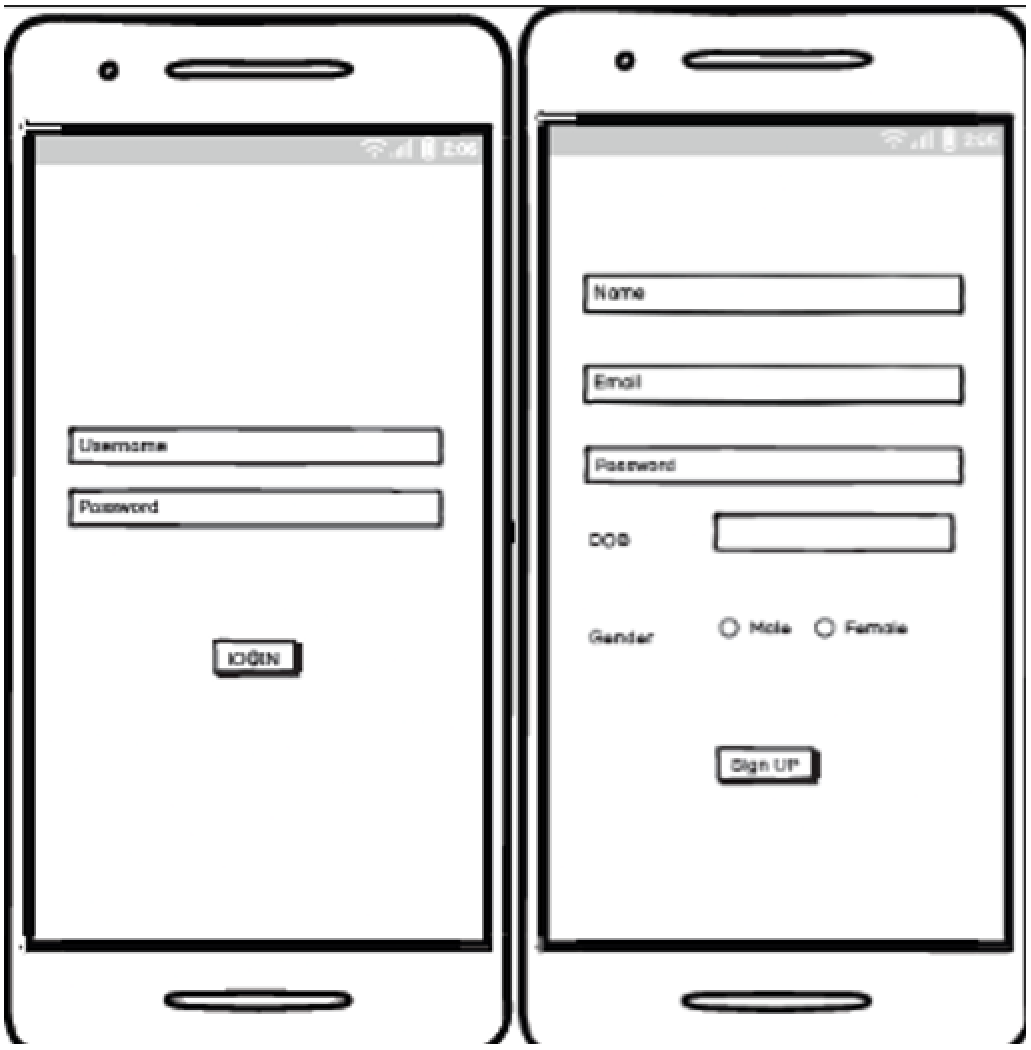


Figure 2: Mock Ups Login and Signup Page

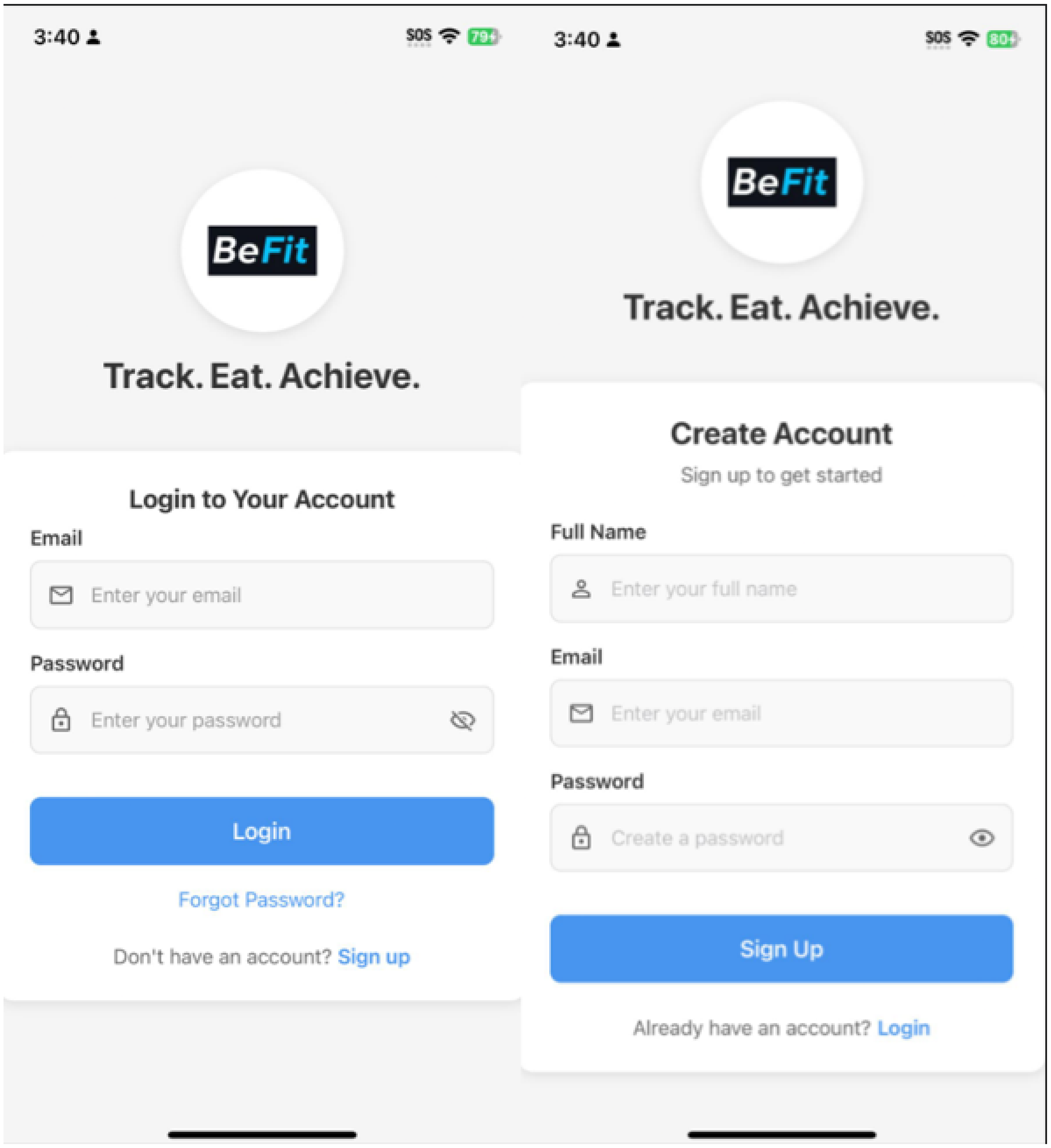


Figure 3: Application Login and Signup Page

There isn't much difference between the mockups and the actual login and signup pages. However, I removed the Date of Birth and Gender fields from the sign-up page to reduce friction for users when accessing the application. I also added a logo, a tagline, and included a "Forgot Password" option in the login page.

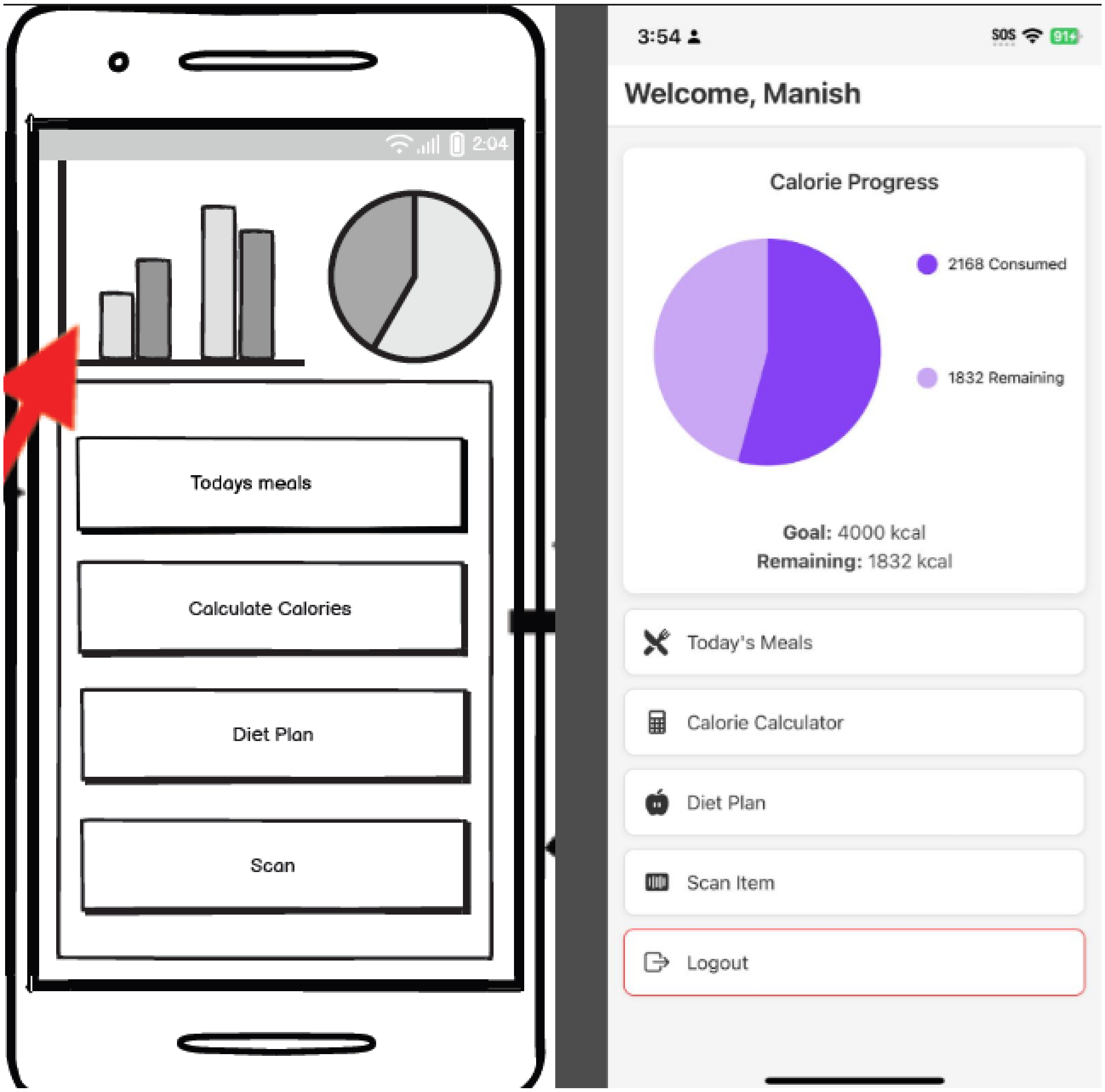
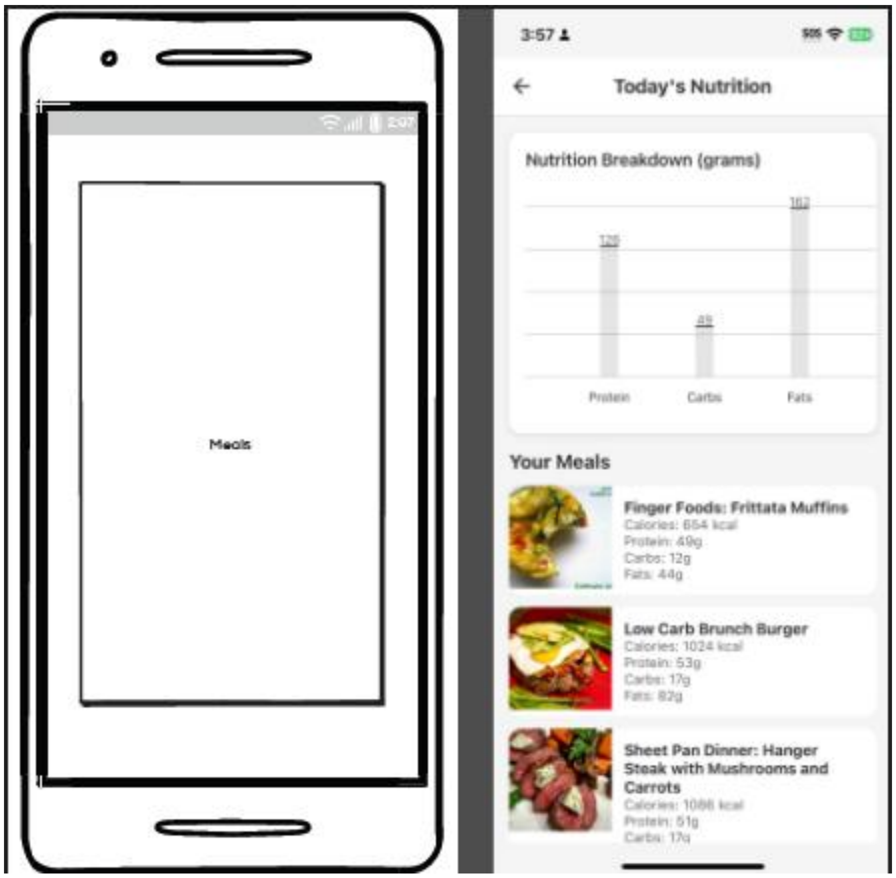


Figure 4: Mock Ups and Actual Home Page

As you can see, there aren't many changes on the home page. However, I've implemented the logout functionality and moved one of the graphs to a different page, as having both graphs on the same page made it feel too cluttered and visually unappealing.



Finger Foods: Frittata Muffins
Calories: 654 kcal
Protein: 49g
Carbs: 12g
Fats: 44g

Low Carb Brunch Burger
Calories: 1024 kcal
Protein: 53g
Carbs: 17g
Fats: 82g

Sheet Pan Dinner: Hanger Steak with Mushrooms and Carrots
Calories: 1086 kcal
Protein: 51g
Carbs: 17g

Figure 5: Mock Ups and Actual Today's meal Page

Initially, I was unsure about what to include on the Today's Meal page. However, as you can see, I've added the graph that was previously on the home page along with the meal details. I also implemented several animations and gesture-based interactions on this page. This graph is dynamic means if you eat some item in the list(swipe right) the graph values automatically gets changed.

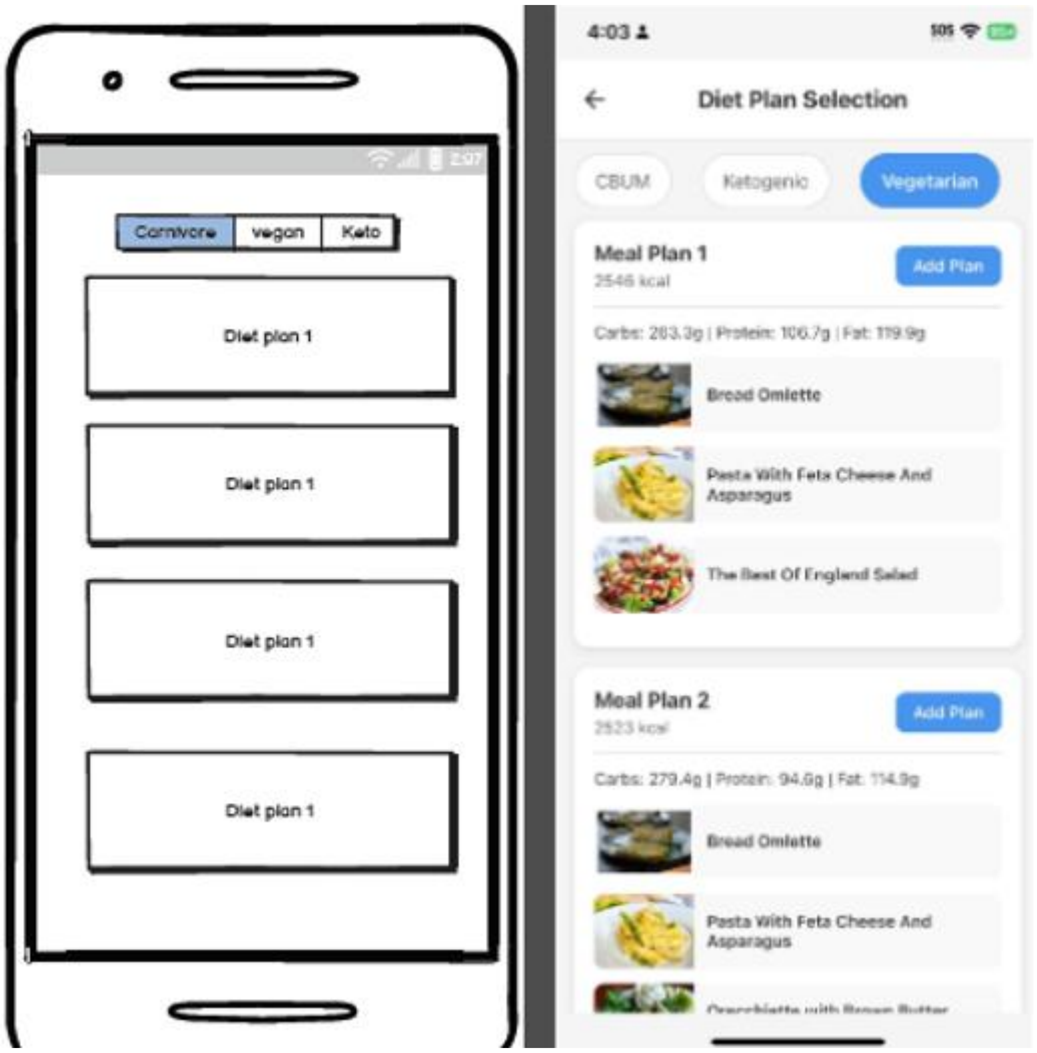
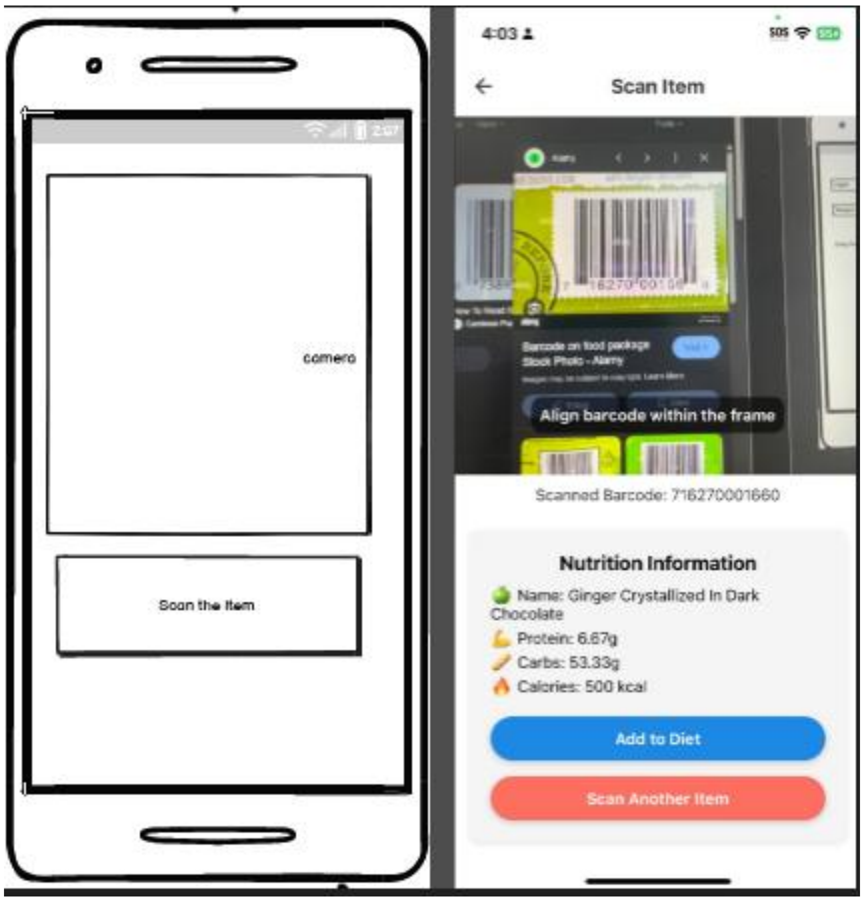


Figure 6: Mock Ups and Actual Diet Plan

I followed an approach very similar to the mockup. The only addition is a feature where, whenever you tap on a recipe, it opens a dedicated page with information on how to prepare it.



Scanned Barcode: 716270001660

Nutrition Information

Name: Ginger Crystallized In Dark Chocolate

Protein: 6.67g

Carbs: 53.33g

Calories: 500 kcal

Figure 7: Mock Ups and Actual Scan camera page

On the scanning page, I removed the button from the actual design for better usability. Now, users simply need to hold the camera in front of a barcode, and the information is displayed automatically.

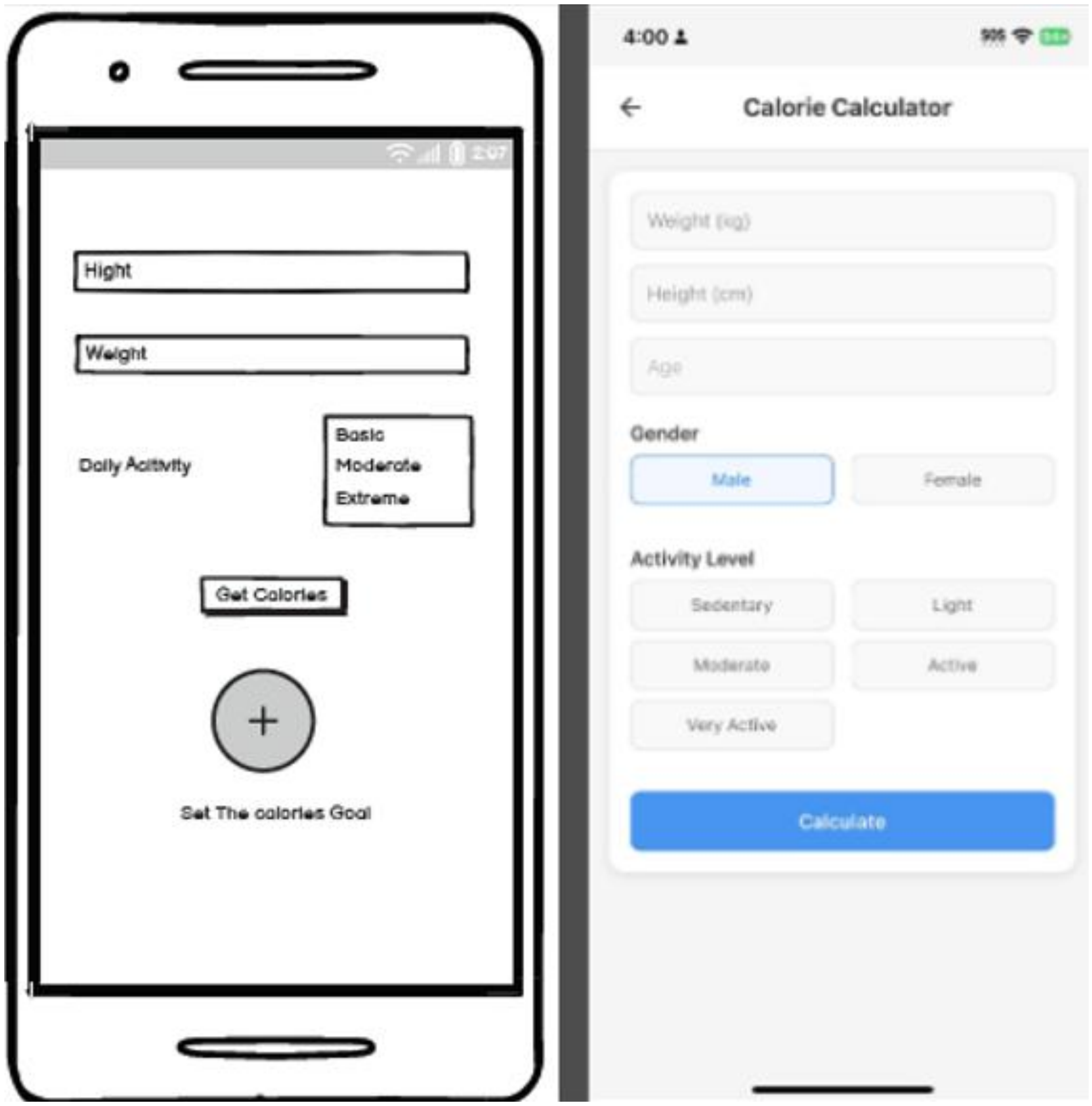


Figure 8: Mock Ups and Actual calorie calculator page

This page follows the same approach as the mockups.

References

[1] Y. Zhang, C. C. Y. Poon and P. Bonato, "A Framework for Smart Nutrition Monitoring and Food Intake Estimation in mHealth Applications," *IEEE Reviews in Biomedical Engineering*, vol. 12, pp. 202–214, 2019. doi: 10.1109/RBME.2019.2930074.

[2] J. Lee, J. Song, and M. Kim, "Design and Implementation of a Mobile Diet Management Application Using Barcode Recognition," in *Proc. 2015 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, 2015, pp. 36–37. doi: 10.1109/ICCE.2015.7066346.

[3] B. Persson and M. Persson, "Integration of Firebase Services into Mobile Applications: Authentication and Cloud Firestore," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 15, no. 17, pp. 108–120, 2021. doi: 10.3991/ijim.v15i17.23825.

[4] A. R. Mohammad and M. Abdelrahman, "Food Recommendation System Using Machine Learning and Nutritional Analysis," in *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, Khartoum, Sudan, 2020, pp. 1–6. doi: 10.1109/ICCCEEE49695.2020.9276705.