INDY-9 RED — MyFoodScan

Draft of Final Report

CS 4850, Section 01/02, Spring 2024

Sharon Perry

Apr 27, 2024

Team Members-

Ibrahima Gueye

Jedae Lisbon

Bri Noel

Victoria Kuswita

Links-

Github: https://github.com/Indy009/myfoodscan_app.git

Website: https://indy009.github.io/

Number of Lines of Code in Project: approximately 3,000

Number of Components in Project: 18 files

Table of Contents

1.0 Introduction

1.1 Abstract

MyFoodScan is a mobile app that enables users to scan barcodes of various food and drink items to ensure they are in compliance with their specific dietary needs. Individuals utilize this application to make a customized profile based on diet, such as vegan, vegetarian, dairy-free, allergies, etc. MyFoodScan promotes compliance with dietary limitations. The application was developed with React Native, Expo Go, Google Firebase, using the React Native Camera for barcode scanning. The OpenFoodFacts API database is used for product information. The goal of this application is to enhance awareness and safety for various dietary needs and monitor dietary restrictions.

1.2 Project Goals

The application will feature user profiles where individuals can specify their dietary restrictions, allowing for a customized experience. The goal of the application is to provide users with an opportunity to scan their purchased products to see if they align with their dietary needs. Moreover, the application encourages users to find more products that comply with their dietary restrictions. To enhance engagement, the app will also provide educational content about different dietary practices and health tips. By offering a user-friendly and informative platform, this project aims to empower individuals with dietary restrictions, making their shopping experiences safer, more convenient, and aligning with their health and beliefs.

1.3 Definitions and Acronyms

The following definitions and acronyms are provided in this SRS for proper document interpretation.

Definitions
Vegan Diet: A diet that avoids consuming any food sourced from animals or use products derived from animals
Vegetarian Diet: A diet that avoids consuming any meat, but still consists of other products derived from animals
Dairy-free Diet: A diet that avoids consuming animal milk or any products made from milk
Gluten-free Diet: A diet that avoids consuming gluten, a protein that is commonly found in grains
Shellfish: Seafood that includes shrimp, crab, lobster, clams, scallops, crayfish, oysters, and mussels
Acronyms
API: Application Programming Interface - allows two applications to communicate with each other
UX: User Experience

UI: User Interface

1.4 Assumptions
The assumptions may include technical and user assumptions. Technical assumptions can include the availability of particular hardware components, compatibility with specific databases and operating systems, and the existence of required network connectivity. It is assumed that the user has previous experience with similar technology and familiarity with functions of similar mobile applications. User assumptions also include that the user has access to an iOS or Android device.

2.0 Design Constraints

2.1 Environment
There are three environments that should be considered: physical, operational, and deployment. Regarding the physical environment, the software should comply with the latest version of iOS and android. In terms of the operational environment, the system should operate best in a high-speed internet environment with the lowest requirement being 256 kbps to run. For the deployment environment, the application needs to be deployable on Google Firebase as a cloud infrastructure to be utilized for database management.

2.2 User Characteristics
The application is intended for users approximately 18 years of age and older, specifically with a demand to discover if their purchased products comply with a specific dietary need. Users can be categorized into various groups, such as users who personally have dietary restrictions, users who know someone with dietary restrictions, users who are curious about specific dietary restrictions, etc.

2.3 System
The software must integrate with the OpenFoodFacts API for product consumption services. The data from OpenFoodFacts will be used in the Google Firebase database along with additional consumer data. The implementation of a RESTful API is necessary to extract the data from the Google Firebase database.

2.4 Assumptions and Dependencies
There exists a few assumptions, such as technical and user assumptions. Technical assumptions consist of the availability of specific hardware components, compatibility with particular databases and operating systems, and the presence of required network connectivity. User assumptions involve the user having prior experience with similar technology and familiarity with functions of similar mobile applications. It is also assumed that the user has access to an iOS or Android device.

2.5 General Constraints

Physical, operational, and deployment environments that should all be considered. In regards to the physical environment, the software should be in compliance with the latest version of iOS and android. For the operational environment, the system should operate best in a high-speed internet environment that has the lowest requirement of 256 kbps to run. Furthermore, in terms of the deployment environment constraints, the application must be deployable on Google Firebase as a cloud infrastructure to be used for data storage and retrieval from OpenFoodFacts API.

The application's target audience is users approximately 18 years of age and older, particularly those with a demand to seek information about whether their purchased products comply with a specific dietary need(s) or not. In terms of categorization, distinct groups of users include those who personally have dietary restrictions, users who know others that have dietary restrictions, users who are seeking more knowledge regarding specific dietary restrictions, etc.

Regarding system constraints, the software needs to integrate with the OpenFoodFacts API for product consumption services. The data from OpenFoodFacts will be utilized in the Google Firebase database, in addition to further consumer data.

2.6 Goals and Guidelines

One main goal of the design of the application includes quick information retrieval from OpenFoodFacts API. Other aims pertaining to the speed are low latency for users of the application and maintaining scalability with the addition of multiple users on the application at one time. Priorities also include creating an application that is both user-friendly and intuitive to use. Moreover, guidelines for the development of the application include creating a product that works, looks, and feels like an existing product. Lastly, another objective is that the application has a strong demand among potential users and the intended audience.

2.7 Development Methods

The methods used for the software design process will be the Waterfall and Prototyping method. The waterfall method is a software development method of completing tasks in sequential phases. The phases consist of requirements, design, implementation, verification, and maintenance. This method makes the development process easy to execute since it establishes clear requirements and goals. In the prototyping method, the goal is to create a usable and working prototype that can be tested before the actual development begins. With this method, we can identify any potential issues early on and receive valuable feedback to produce a high-quality application.

3.0 Functional Requirements

3.1 Phase 1

3.1.1 Login and Register

After opening the app, the user is prompted to register for security concerns. It will be required to sign up using their email and password or through OAuth from different providers such as Google, Facebook, Apple, etc.

3.1.1 Questionnaire

After the login/register page, the user will be prompted to complete a questionnaire about which dietary restrictions apply to them. This information will be automatically updated to the user's profile, which they will be able to modify later within the user profile page.

3.1.2 Display Home Page with Scan

When signed into the app, the user is presented with the home page. The primary section of the home page will be the barcode scanner that can be utilized by the user for scanning a food item. This scanner will be seen in the middle of the page and will take up most of the space. The navigation bar at the bottom of the screen will contain a history button, a scan button, and a user profile button. Since the scanner is the home page and the home page is the first page displayed, the user will be on the scan option of the navigation bar.

3.1.3 Barcode Scanning

Access to the user's camera will be required. A pop-up will show up when the app requests access to the users' camera. Users should be able to use the camera function to scan barcodes of different products to identify additional information.

3.1.4 Navigate from Home Page to Data Pop-up After Scanning

After scanning the item, the user will see a pop-up that displays the information regarding the product. This page will display the scanned product's title, nutrition information, a message about if the product complies with the user's dietary restrictions based on their profile, the product image from the database, and recommendations for similar products that comply with the user's restrictions in cases where the current product does not comply. Furthermore, the recommended similar products will be displayed at the bottom of the pop-up.

3.1.5 Navigation Bar

The navigation bar previously mentioned will be displayed at the bottom of the application. The home page will automatically begin on the scan page since this page is the home page. The other two options on the navigation are the history button and the user profile. The history button will be on the bottom left, and the user profile will be on the bottom right.

3.1.6 Navigate from Home Page to User Profile Page

When users select the button for the user profile on the navigation bar, they will be prompted to a page that allows users to view and edit their information. Users will have a profile page with their personalized preferences and changes can be made anytime. This page will include their name and other personal information and a restrictions section. This section will have buttons with various dietary restrictions that are highlighted according to which dietary restrictions apply to the user from the answers of the questionnaire. On this page, users will be able to see these dietary restrictions

highlighted and can modify them, if needed. This page will also display a section for access to the camera, where users modify this access.

3.1.7 Navigate from Home Page to Previous History Page

When users select the button on the navigation bar for the previous history, they will be prompted to a page that users can utilize to view the past previous items. This page will display the last ten previously scanned items for user convenience.

3.1.8 Machine Learning for Recommendation of Similar Items

Machine learning algorithms will be used to provide users with recommendations to similar products in cases where the current product does not comply with the selected dietary restrictions of the user.

3.2 Phase 2

3.2.1 Machine Learning for Compliance Checking

Machine learning will be utilized for ingredient analysis and compliance checking to classify scans more efficiently.

3.2.2 Additional Menu Options

There will be a favorites section where users can save scans of different products and an education section providing tips and information on different dietary needs. The favorites section will allow further convenience for users to have their most common or liked items in one easy-to-access selection. Moreover, the education section will advance one of the goals of the application, which is to educate users on various dietary restrictions.

3.2.2 Notification System

Notifications will be sent to the user periodically to encourage/remind them to return to the application for further scanning of products and to provide additional educational fun facts to increase user involvement.

4.0 Non-Functional Requirements

4.1 Performance

The system should be able to run within 5 seconds after opening the app. If multiple users are using the app concurrently, the system should still be able to run efficiently without slowing down or crashing. Processes such as loading and retrieving data should occur as quickly as possible.

4.2 Security

The app will have a secure authentication system that requires users to make a username and password to create their profile. Each user that creates a profile is limited to view the information

that they put into the app. The users are not allowed to access or edit other profiles. In addition, the app does not share any data with other users.

4.3 Usability
The application will be user-friendly as it will display an intuitive interface that appeals to our target audience. It will be fully responsive for both iOS and android devices and should deliver a smooth user experience where the user encounters little to no delays or errors.

4.4 Reliability
The system should be constantly available as long as the user has network connectivity. The user should be able to consistently open the app without facing any errors. When prompted to create a user profile, the app should save the information to be available to view and edit. Each option presented to the user should be constantly available and running properly when selected. When the user allows the app to access the camera, the barcode scanner should be functioning properly and reading the barcodes on selected food items.

4.5 Scalability
As more users join the app and make their own profiles, the system should be able to handle the increasing amount of data without decreasing the overall performance. The system should allow updates to include new features without causing any problems to the users or reducing the app's efficiency.

5.0 External Interface Requirements

5.1 User Interface Requirements

5.1.1 Loading Screen
This is the first interface that the user will see when using MyFoodScan for the first time.

5.1.2 Login Screen
On this screen, there will be the name of the application, the option to either login with the email and password or login by providers like Google, Facebook, Apple, etc.

5.1.3 Register Screen
In this screen, MyFoodScan will enable users to register. After entering the necessary information, the user is registered to use the application.

5.1.4 Questionnaire Interface

On this screen, MyFoodScan will allow the user to choose from many dietary restrictions provided by the application that apply to them. This information will be stored and automatically updated to the user's profile.

5.1.5 Main Interface

The interface will contain three screens and one pop-up overlay. All screens/overlay will have a consistent layout.

5.1.6 Scanner Interface

On this screen, the application shows a scanner overlay, providing visual cues within the product barcode that should be placed for scanning.

5.1.7 Product Detail Pop-Up Interface

After scanning a product, the user will have a screen pop-up showing the product image, name, and contains allergens section. Underneath, the screen contains a list of dietary restrictions that comply with the user's preferences. In terms of the messages about if the product complies with the user's dietary restrictions, the user's dietary restrictions that do not comply, if any, will be prioritized to the top of the list of dietary restrictions listed, and they will be listed with an "x" next to them. Next, the user's dietary restrictions that do comply, if any, will be followed, but the font will be grayed out since the information has less of a priority compared to the user finding out that the product does not comply with a certain restriction. These will be listed with a check next to them. Below that will be a section where similar product recommendations that'll be provided to the users.

5.1.8 View History

The screen will present a list of the last ten previously scanned items. This screen will display these products and their images in a two-columns list.

5.1.9 Show/Edit Profile

In this screen, a user can see their name, information, profile photo, and settings to change information or profile photo. Additionally, users will be able to change their dietary restrictions and access the camera.

5.2 Software Interface Requirements

For database services, the application will use the latest version of Google Firebase. The application will also run the Android version above or equal to Red Velvet Cake (version 11) or iOS version 8 or higher. To access the device's camera for scanning and interpreting barcodes, the application will use the React Native library '*react-native-camera*'.

5.3 Communication Interface Requirements

Users will complete the necessary forms to register or login into the MyFoodScan application. These details i.e. email, and password are encrypted before they are passed on to the

database. Email communication is required if the users forget their password. HTTP protocol is pertinent with mobile devices in communicating with Google Firebase.

6.0 System Overview

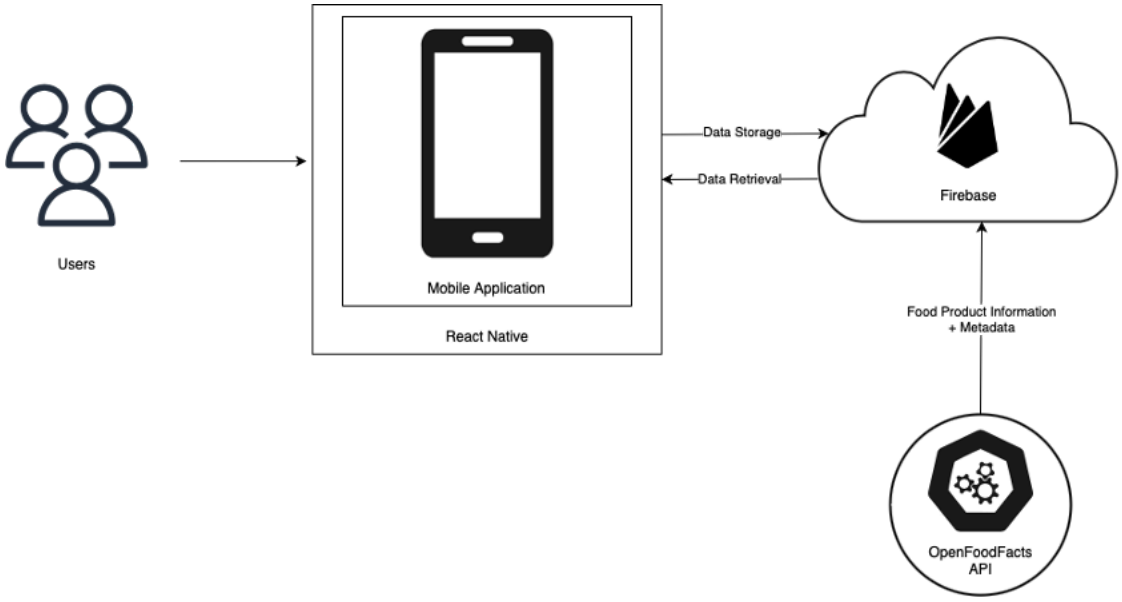

Figure 1 Diagram of Application Overview

Figure 1 above showcases the architectural structure chosen for developing MyFoodScan. The backend will be responsible for requesting and receiving information from OpenFoodFacts API that the application will be dependent upon. Furthermore, the frontend development will involve using React Native as the frontend framework, as it has many comprehensive and cross-platform capabilities. For the prototype, Canva and Figma will be used. In terms of backend development, Google Firebase will be used for data storage and retrieval from OpenFoodFacts API.

7.0 Architectural Strategies

7.1 Choice of Product

After considering several options, we decided to utilize React Native for the frontend development of MyFoodScan. React Native's cross platform capability satisfies both users who

may be using other operating systems and provides tools such as libraries that will help support the development process. The libraries that will be used to develop MyFoodScan are React Native Camera, React Native Firebase, React Native Scanner, and TailwindCSS. React Native Camera will provide the app with camera access and will be cooperating with the scanner library to scan barcodes. TailwindCSS will be used to help with the frontend development. React Native was also chosen for its main programming language JavaScript due to its functionality and ease of use. As for the backend, our strategy is to use a Google firebase to securely store user scans and personal information. In addition, Firebase was also chosen for its real-time database, query speeds, and compatibility with React Native.

7.2 Reuse of Existing Software Components

Some software components will be reused to implement various parts/features of the system. In regard to design, many components of the software might be reused, such as the software to create a button on one page being reused to create a button on a new page, for example.

7.3 Future Plans

There are a few future plans for extending and enhancing the software, such as providing additional menu options and a notification system. For the additional menu options, we will extend the software to include two more pages, allowing the navigation bar to include five options compared to only three. These two new pages will be a favorites page and an education page. The favorites page will be a section where users can save scans of various products, allowing further convenience for users to have their most common or liked items in one easy-to-access selection. The education page will provide a section where users can be educated on various dietary restrictions, advancing one of the goals of the application. Moreover, for the notification system, we will enhance the software to include notifications for users who agree to receive them. These notifications will consist of friendly reminders to users to return to the application for increased user involvement and engagement, fun facts about various dietary restrictions to further educate the users, and more.

7.4 User Interface Paradigm

Each page will have several icons the user can interact with through touch gestures to promote user engagement. The navigation bar provides the user with three options. One icon takes the user to the scanner, another will take the user to an archive of their previous scans, and the last icon takes the user to their user profile. Users can edit their own user profile to update their preferences by selecting different buttons and typing information. In the history page, users can select the image of their previously scanned item and view information like the name of the product and the ingredient list.

7.5 Error Detection and Recovery

Errors are inspected in relation to bugs, parts of source code that create undesirable or unintended results. Jest will be used as the main testing software, as it is commonly used to test JavaScript projects. Software testing methodologies include performance testing and end-to-end tests. Performance testing examines the reliability, speed, and responsiveness of the source code. For end-to-end tests, we will act as the user and test the functionality of the application to ensure users can navigate the functions as planned. For example, for error detection, we will act as the user creating an account, scanning a product’s barcode, seeing if the information regarding if the product aligns with the desired dietary needs is accurate, etc.

For recovery, after completing end-to-end testing and errors are detected, adjustments in the source code will be made and end-to-end testing will occur again until acting as the user and navigating all of the functions of the application is finally conducted as planned.

7.6 External Database Management

Firebase Cloud Functions will be used for the external database management of the OpenFoodFacts API. Cloud Functions is a serverless framework for developing event-driven applications. Cloud Functions will be used to retrieve the data of the product's information from OpenFoodFacts.

7.7 Concurrency and Synchronization

Google Firebase provides concurrency and synchronization for our users with Realtime Database and Firestore. Both databases use a NoSQL cloud database, which allows data to be synced and updated at all times. This tool is intended for the user profiles, as we will securely store each user's information in an organized collection. Each user will have access to view and edit their information at any time, even if they have no internet access. If a user wants to edit their information, Firestore offers data synchronization to provide the user with real-time updates. Firestore also allows multiple users who create a profile to independently access their own information concurrently.

8.0 System Architecture

MyFoodScan is intended to handle user interaction, data retrieval, data management, authentication, and security. The user interface component governs user input, output, and navigation. The data retrieval component collects and processes food product data from additional sources, whereas the data management component maintains and manages user profiles, preferences and historical data. The authentication and security component provides safe access to the application and user data.

In the application, collaboration among components is key: React Native app interacts with Firebase Authentication for user registration and sign-in via Frontend and Backend interaction. Firebase Firestone is the source from which user profiles and preferences are saved and retrieved. Firebase Cloud Functions receives barcode data from the app and uses it to communicate with the OpenFoodFacts API. External API and Backend Interaction pertains to Firebase Cloud. By requesting product data from OpenFoodFacts via API calls and sending the processed results back to the frontend, Cloud Functions serve as a secure middleman.

According to the Model-View-Controller (MVC) design, the decomposition that was selected makes it possible to distinguish clearly between client-side and server-side activities. Because

of this division, it is possible to design and update the user interface separately from the backend functionality. Furthermore, it enables the backend to be scaled separately to meet dynamic loads, which is crucial for the app's prospective expansion.

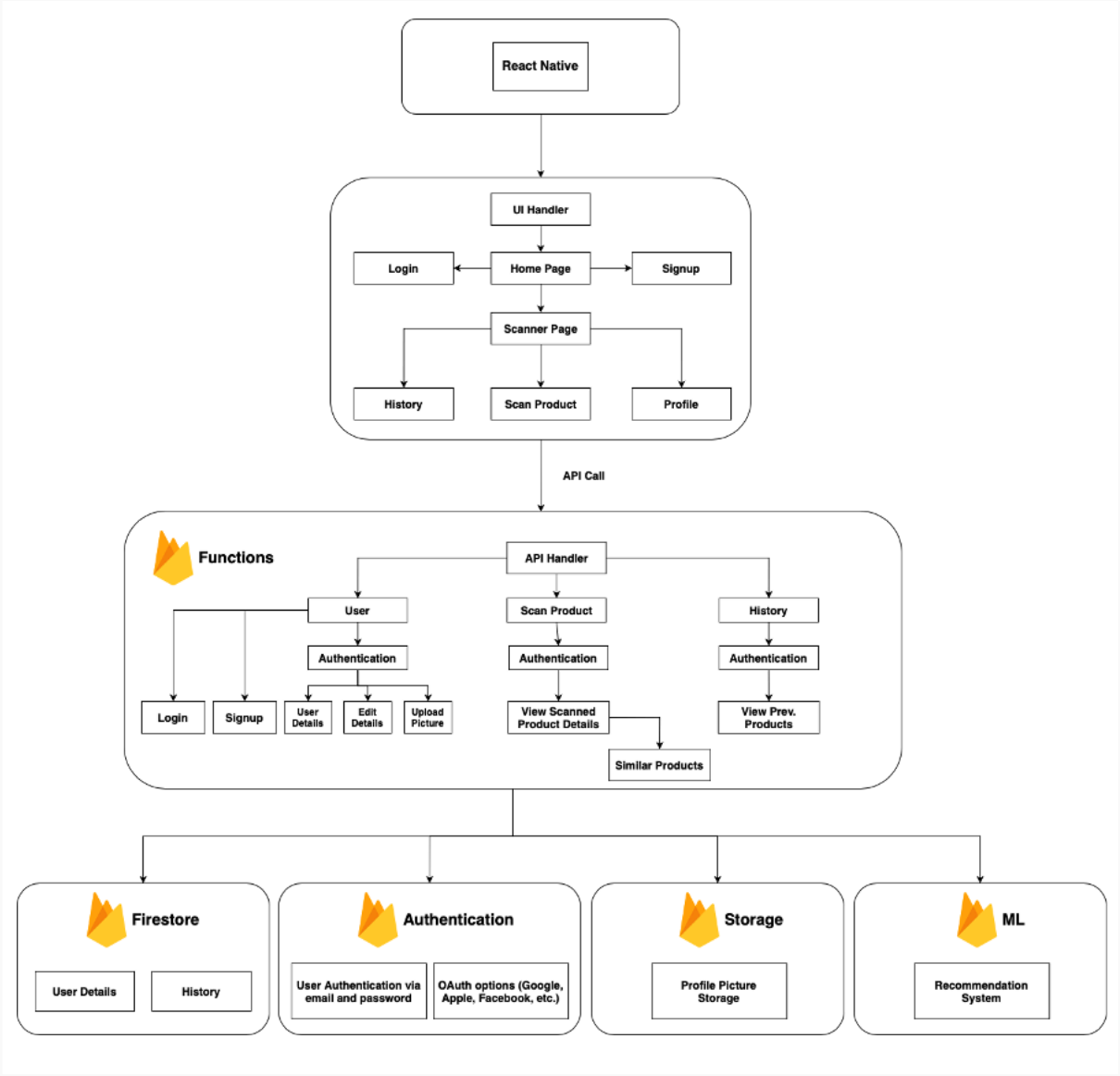

Figure 2 High-Level Architecture Diagram

8.1 Subsystem Architecture

One of the most important components of the application is the ML Recommendation System. Its main duties include learning, pattern recognition, and data analysis. To comprehend preferences and dietary constraints, it examines user behavior and product data. The pattern of the dataset can be used to forecast consumer preferences. The subsystem adapts product

recommendations to each user's specific dietary needs and keeps learning from user interactions and makes accurate recommendations over time.

The ML subsystem communicates with the rest of the application. It trains its models using product information from the OpenFoodFacts API and user profiles from Firebase Firestone. It works with React Native frontend to seamlessly present recommendations to the user. Depending on the complexity, the ML recommendation system can be contained within Firebase Cloud Functions or an alternative ML server.

As a subsystem, ML components need to be flexible and scalable for the app architecture. Recommendation as a separate subsystem is the best approach because ML tasks are resource-intensive, having allocation be done to ensure efficient performance.

9.0 Analysis

9.1 Choice of Product

We had two options to consider for the front-end development of our application. The first option was to use Flutter, an open source framework created by Google that allows developers to easily create a user interface. Flutter supports cross-platform app development and supports various platforms, such as iOS and Android. The main programming language used is C and Dart, which is a language that is optimized to support building UIs. The second option that was considered is React Native, an open source framework created by Meta Platforms. Similar to Flutter, React Native supports cross-platform app development and allows developers to create applications for iOS and Android simultaneously. It also uses various programming languages, but the main language that will be used is JavaScript. After some consideration, we decided to use React Native because of its simplicity and familiarity. There are several libraries included in React Native that can help support the development of the application, such as React Native Camera, React Native Firebase, React Native Scanner, and TailwindCSS. In addition to using libraries, React Native also supports Firebase, the database that the application will use.

9.2 Engineering Trade-offs

Performance vs Functionality: The goal is to create an application that contains various usable features without decreasing the overall performance of the application. The users should be able to access and use all features on the application without encountering any problems or delays. To balance both features, the application will prioritize the essential functions the user needs, such as the barcode scanner. Images can be compressed to reduce the size and improve the load time. The user interface will be simple and will not have any extra content that may affect the overall performance.

Security vs Usability: Each user that downloads MyFoodScan will be prompted to create a user profile for a more enhanced and interactive experience. The application will clearly communicate what permissions it will need from the user, such as access to the camera. These permissions can also be revoked by the user at any time. Any errors a user may encounter will

include clear error messages to help them understand what happened.

Customization vs Standardization: MyFoodScan will give the users customization options to provide a more personalized user experience. However, it is also important to keep the interface simple and intuitive to prevent any confusion. When users set up their own profile, they are given the option to select their dietary restrictions and update their preferences, including their notification settings and camera access permission. This personalization tailors the application towards the user, but the general functionality of the application should be the same for each user.

9.3 Coding Guidelines and Conventions

When using React Native, it is important to follow the standard coding guidelines to ensure readability. consistency, and maintainability. This includes using the proper naming conventions and organizing all files in folders. Comment documentation in code can help with readability and allows everyone to understand the purpose of each piece of code. Some commenting conventions that can be implemented are placing comments on separate lines and only using it when necessary to keep the code visually clean and organized. Since the main platforms for this application are iOS and Android, it is also important to follow the guidelines and best practices for both platforms. The guidelines that will be followed for iOS devices can be found in the Human Interface Guidelines by Apple Developers. The guidelines for Android can be found in Material Design for Android by Android Developers.

10.0 System Design

10.1 Frontend Module

10.1.1 Home Page

The home page of the application consists of sign in and sign up options. One of the displayed options will be to sign up/in manually. This option allows users to manually set a username and password which is securely stored in the firebase database after authentication. The second option asks users if they want to sign in with a Social Login. This feature allows users to access new applications or websites using their existing login information from social networking services (Google, Yahoo, etc.) thus simplifying the registration and authentication.

10.1.2 Scanner

Upon passing the home sign-in page and account setup (if first time user), the MyFoodScan application will directly then go to the scanning page. The scanner will be centered on the navigation menu and centered on the user profile and history. Scans will operate by importing the barcode scanner library and linking the native code. A permissions configuration will also be implemented to request camera access from users. Once an item is scanned, the app then makes a call to the OpenFoodFacts API for database matching.

10.1.3 User Profile

The user profile page displays comprehensive personal information provided by users. Name, date of birth, email, phone number, dietary preferences, and a logout option will be displayed here. If users wish to update their email or phone number, they will undergo a verification process to ensure secure authentication of the new information.

10.1.4 History Page

The history page displays a list of the past ten previous items that the user scanned. This list will be displayed in a 2-column grid layout. Each product is showcased with an image and title of the product. In the corner of each product's display box, either a check will be shown if the product does align with their current dietary needs or an "x" will be shown if the product does not. This format will allow for the convenience of users to quickly access the previous items they have scanned and also to see if the product does or does not fit into their personalized diet.

10.1.5 Product Details

After the successful scanning of a product, the app transitions to the product details page, where users are presented with comprehensive information sourced from the OpenFoodFacts database. The page will display an image of the product and below that a note of ingredient information. After the image and ingredients, there is a box that displays if the scanned food follows dietary restrictions. "x" if it is not and a checkmark if it is compliant. Below that will contain the recommendations, showing users similar products that are compliant or other options if they are not compliant.

10.2 Backend Module

10.2.1 Firebase Authentication

This service will handle both manual sign-ups/sign-ins and social logins. For manual authentication, it stores and manages user credentials (username and password). For social logins, it integrates with various social networking services like Google and Yahoo, enabling users to sign in with their existing accounts.

Manual Authentication Process: When a user chooses to sign up/in manually, the backend will validate against the Firebase Authentication system. After successful authentication, the user is granted access to the application

Social Login Process: The backend will utilize OAuth protocols to authenticate users via their social media accounts ensuring a secure exchange of user data between social platforms and the application.

10.2.2 Firebase Firestore

The NoSQL database will store and manage user profiles with their personal information, and the history of scanned products. Firestore offers real-time data across user devices, ensuring that users have access to their data anytime.

User profile management: Firestore will be used to create, update, and delete user profile information. When a user updates their email or phone number, the verification process will be triggered to authenticate the new information before updating.
History Management: The application will store the history of the last ten scanned items in Firestore. Each entry will include the product image, title, and dietary alignment status (check or “x”), allowing users to view previously scanned products.

10.2.3 Firebase Cloud Functions

These functions will serve as backend logic for the application, handling operations that are triggered by app events or HTTP requests. The functions interact with the OpenFoodFacts API to fetch details after scanning.

Product Lookup: When users scan a product, a Firebase cloud function will be triggered to make a call to the OpenFoodFacts API, retrieve the product details, and return the information to the frontend.

Compliance Check: Another set of cloud functions will analyze the product details against the user’s dietary preferences to determine compliance (represented by a check or “x”).

10.3 External API Integration

10.3.1 OpenFoodFacts API

The integration process involves making HTTP requests to the OpenFoodFacts API, parsing the return data, and then utilizing this data to enhance the user experience by providing detailed product information and dietary compliance insights. Upon receiving a response from the OpenFoodFacts API, the backend function parses the JSON data to extract the relevant product details.

10.4 Detailed Subsystem Design

The recommendation system is an important content-based subsystem that tailors food product recommendations to individual dietary choice and limits. During initialization, the system distinguishes between new and returning users. New users must complete a registration process in which they provide their dietary preferences and restrictions,
allowing the system to develop a personalized user profile. Existing users just log in, and their existing profiles are used to guide the recommendation process.

The algorithm, which filters and ranks food items from the database depending on the user’s dietary profile. This algorithm evaluates nutritional facts and ingredients, ensuring that they meet the user’s dietary requirements. It also learns from prior encounters in order to improve its accuracy over time. When users interact with the recommendations-via acceptance or rejection, the system collects the feedback and dynamically updates user profiles.

The feedback loop is critical to the system's iterative design, allowing the recommendation engine to continuously enhance the customization of the system's suggestions. Every user interaction provides a chance for learning, ensuring that recommendations are tailored to user's changing preferences and constraints. The content-based strategy was purposefully designed to satisfy specific dietary demands while stressing user privacy and tailored health concerns.

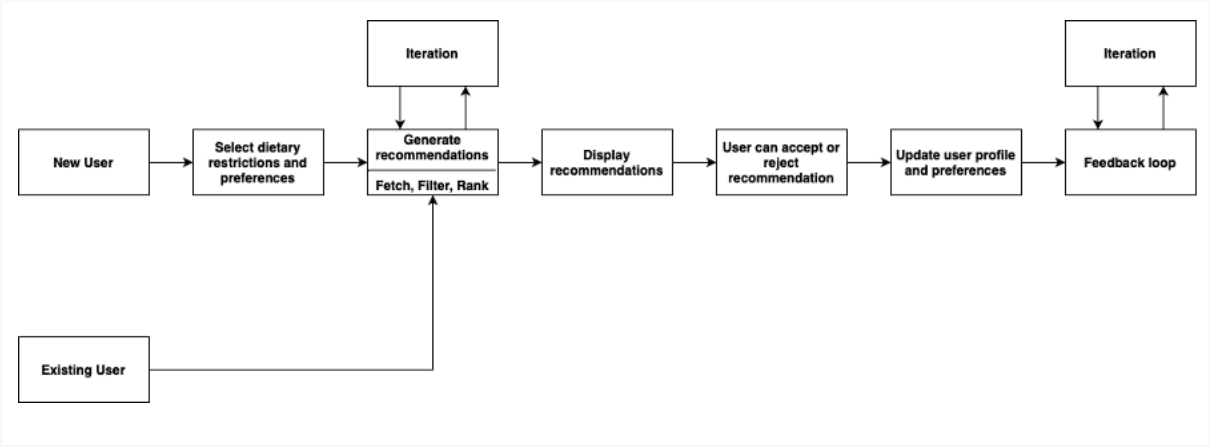

Figure 3 Overview of Recommendation System

11.0 Prototype Design

11.1 Design Process

Our design process consisted of planning, low-fidelity prototyping, and high fidelity prototyping. This section aims to show the development of our app from the initial idea to the final design, as well as discussing our thoughts behind each decision.

11.1.1 Planning

The goal of this project is to create a user-friendly application that lets users scan any food item while they are shopping. We wanted to focus on simplicity and functionality while making the application visually appealing. This stage consisted of discussing what each page will contain and how the contents will be organized.

11.1.2 Low-Fidelity Prototyping

For this stage, we began by sketching a story-board of how each page is connected to each other, followed by rough sketches of the page layouts. Afterwards, we utilized Figma to create the low-fidelity prototype. We began creating each page in order of how the app flows, starting with the loading page, the sign-up page, and the home page. While creating the prototype, we mainly focused on creating a clean and simple layout that a user can understand without

needing any instructions. We had several discussions regarding different components for each page, including the shapes of the buttons and text bars and what we wanted the layout to look like. Overall, we kept the layout straightforward and familiar to what many users have seen before with other applications, like keeping the login and sign up page in the same format many other applications use.

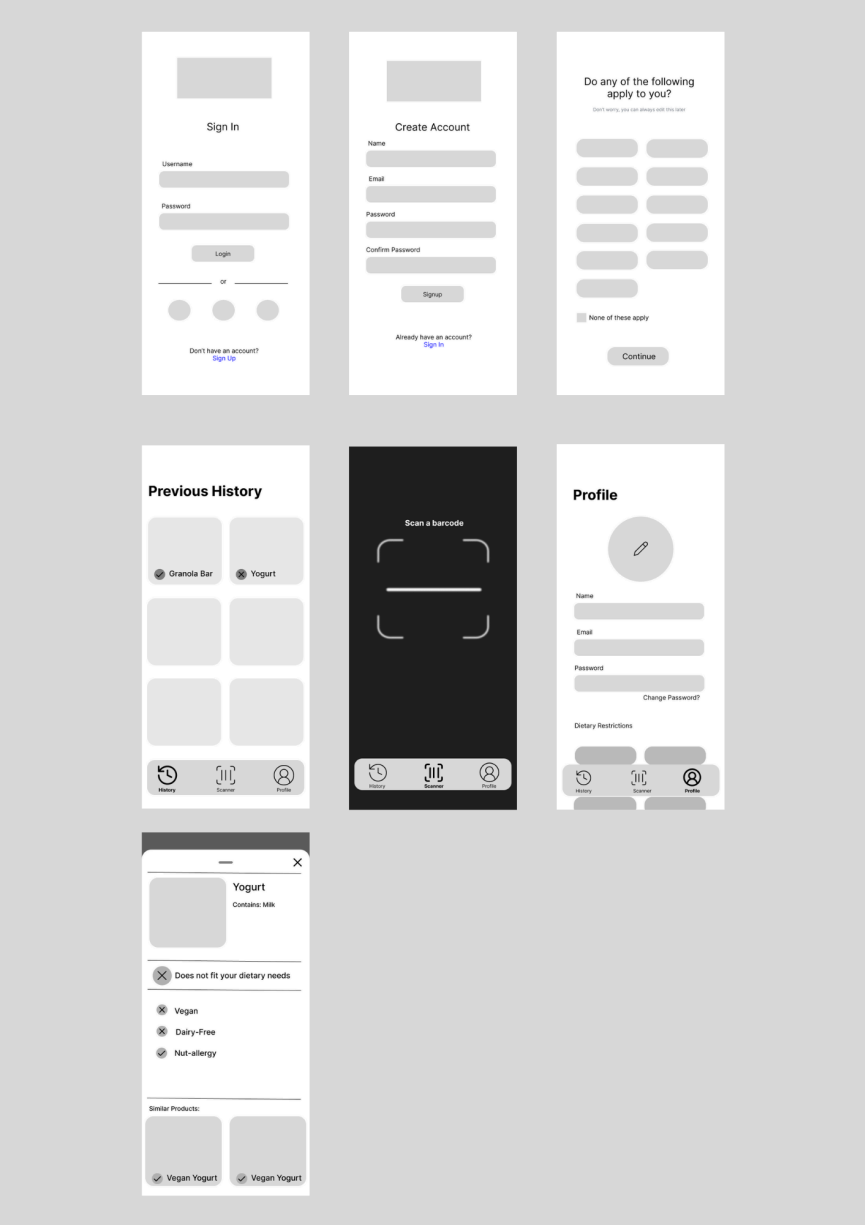

Figure 4 Low-Fidelity Prototype

11.1.3 High-Fidelity Prototyping

Figure 5 showcases the transition between each page of the application. Starting from the Loading Page, users are prompted to the Login Page where returning users can login to their account. If the user is a new user, they can click the “Sign Up” button, and they will be prompted to the Create Account Page where they can make a new account. If the user wishes to return to the Login Page from the Create Account Page (ie. if they already have an account), they can return to the Login Page, as well, by clicking the “Log In” button. Once a new user creates an account, they are prompted to the Questionnaire Page, where they are asked to select all of the listed dietary restrictions that apply to them. Once the user clicks the “Continue” button, they will be prompted to the Home Page (Scanner Page). Similarly, the returning users who login will be prompted to the Home Page (Scanner Page), as well. Once users are prompted to the Home Page (Scanner Page), they will be able to see the Navigation Bar at the bottom. They can use this Navigation Bar to transition between all three pages: Home Page (Scanner Page), History Page, and Profile Page. On the Home Page (Scanner Page), users can scan any item’s barcode. Once the scanner processes the barcode, the Product Details Pop-Up will be displayed on the screen, where users can learn more about the product. When users are done reading the information about the product, they can either slide down on the pop-up from the top or simply click the “X” button. Since the user was previously on the Home Page (Scanner Page), they will then be returned to the Home Page (Scanner Page) after doing so. Additionally, if a user transitions to the History Page using the Navigation Bar, they will be able to see their previously scanned items. If they click on one of these items, the Product Details Pop-Up will be displayed on the screen again, in which they can exit this pop-up by either sliding down on the pop-up from the top or simply clicking the “X” button, just the same as before. However, since they entered the pop-up from the History Page, they will be redirected back to the History Page.

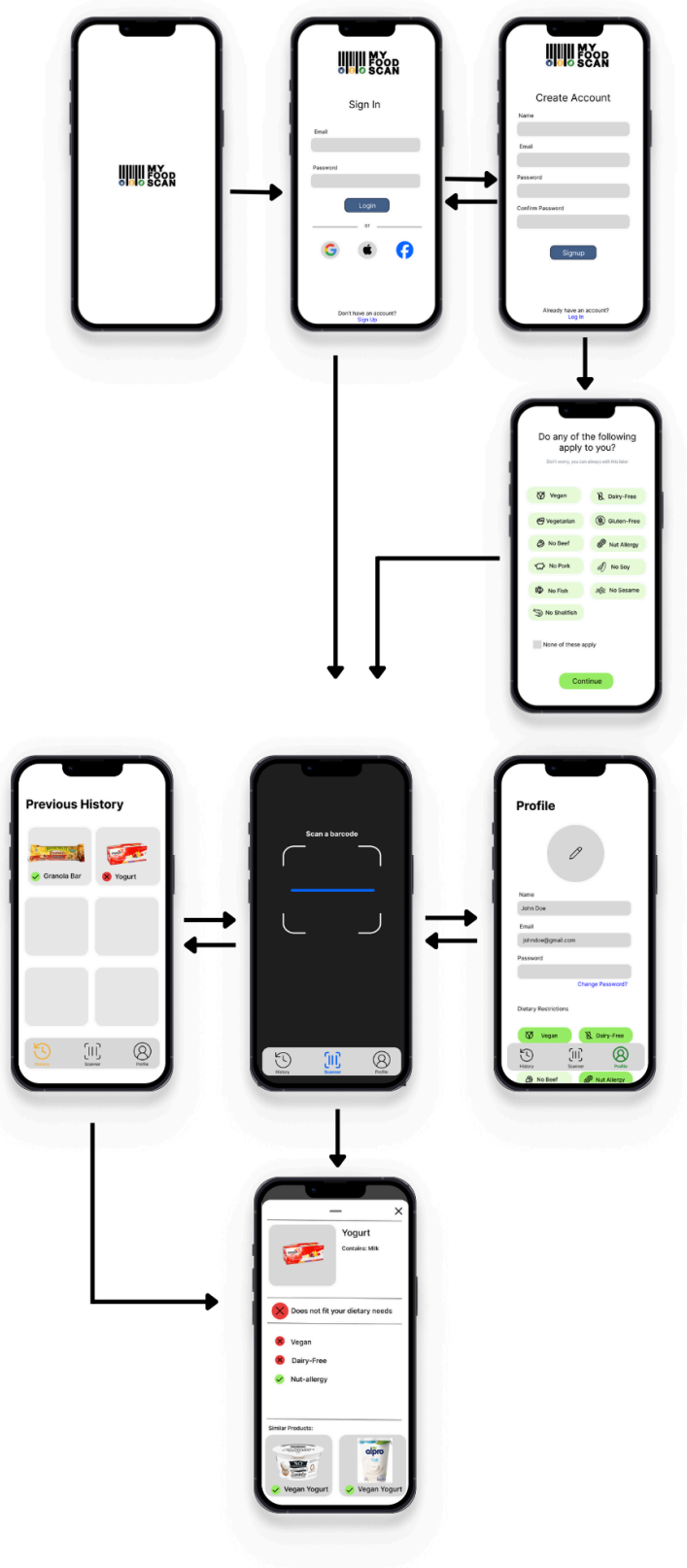
MY FOOD SCAN
Sign In
Email
Password
Login
Don't have an account?
Sign Up
Create Account
Name
Email
Password
Confirm Password
Signup
Already have an account?
Log In
Do any of the following apply to you?
Don't worry, you can always edit this later.
Vegan
Dairy-Free
Vegetarian
Gluten-Free
No Beef
Nut Allergy
No Pork
No Soy
No Fish
No Sesame
No Shellfish
None of these apply
Continue
Previous History
Granola Bar
Yogurt
Scan a barcode
Profile
John Doe
johndoe@gmail.com
Change Password?
Dietary Restrictions
Yogurt
Contains: Milk
Does not fit your dietary needs
Vegan
Dairy-Free
Nut-allergy
Similar Products:
Vegan Yogurt
Vegan Yogurt

Figure 5 High Fidelity Prototype (Flow Diagram)

12.0 Development

12.1 Major Components of Technology:

For the development stage of the mobile app, React Native was used as the JavaScript-based open-source UI software framework of the app. The various pre-built components and libraries offered from React Native were also implemented in the development process. Expo Go, an open-source sandbox, was also used in the development of the app to experiment with React Native. Expo Go was used in the coding stage to act as a simulation to preview the app in real-time during the development process. For the following pages, both React Native and Expo Go were implemented.

12.1.1 Register Page/ Login Page

The development of the register and login pages involved the use of Firebase Firestore and Firebase Authentication. Firebase Firestore was used to store the user's information, such as their first name, last name, and email. Firebase Authentication was used for a secure authentication system. This includes enabling authentication using passwords and providers such as Google, Apple, and Facebook.

12.1.2 Questionnaire Page

The questionnaire page was developed with the purpose of asking the user to select which of the following dietary restrictions apply to them, along with the option to also select a button indicating none of the options apply. This option was set as a conditional to empty if the user decides to select the button that none of the restrictions apply.

12.1.3 Scanner/Home Page

React Native Camera is a library within React Native that allows the application to access a mobile device's camera. The camera will refer to the OpenFoodFacts API whenever the barcode of a product has been scanned to extract the necessary information. This information will then be organized and transferred to another page for the user to view on screen.

12.1.4 Product Details Pop-Up

The details pop-up is what appears after the scanner has been utilized and referred to the OpenFoodFacts API. The pop-up will detail all relative information from the API in order of image URL, ingredients, pass/fail, detailed checklist of restrictions relative to the user, and a recommendation of " similar products ".

12.1.5 History Page

Firestore database is utilized to store each user's previously scanned product scans. Each scan will be saved as a document that includes the product's name, image URL, scan timestamp, and compatibility with the user's dietary choices, as obtained via the OpenFoodFacts API. Last Recently Used (LRU) cache is applied to keep only the 10 most recent scans in the database. This entails developing a Firebase Cloud Function that is activated when new documents are added to the Firestore history collection. Using React Native components like FlatList, these scans are displayed dynamically, showing the product image, name, and an icon that indicates dietary compatibility (a check mark or x mark). This setup ensures the interface is user-friendly and informative, automatically updating to reflect the latest scans and their compatibility with dietary restrictions.

12.1.6 Profile Page

Firestore and Firebase Authentication store the information that is displayed on the profile page. Firestore saves the user's first and last name and their preferences. Firebase Authentication sets and stores the user's email and the password. Using the method in Firebase Authentication, passwords can be reset by sending the user a password reset email. For the implementation of allowing the user to upload a profile picture, the Expo Image-Picker library was used due to its compatibility with Expo Go.

12.2 Issues Encountered:

During the development phase, a few issues pertaining to each page were encountered. The following list of pages explains these issues.

12.2.1 Questionnaire Page

The main issue with the questionnaire page is once the user selects their dietary restrictions, this information is not automatically updated to the user's profile as intended.

12.2.2 Scanner/Home Page

The first issue with the Scanner page was the barcode scanning function which was depreciated. We resorted to just using the React and Expo camera and developed a function that looks at and reads barcodes.

12.2.3 Product Details Pop-Up

The main issue with the pop up is placing key information (photos and ingredients). Other crucial issues with the pop up are checking compliance with the profile preferences and the recommendation system.

12.2.4 History Page

An issue in the history page is the asynchronous nature of data updates and rendering in React Native, along with the real-time data flow from Firestore. The LRU process is

causing a delay or inconsistency in showing the updated list of scanned products immediately after new scans are added or old scans are discarded. This issue shows as a user seeing outdated information or seeing a momentary flicker while the list updates, leading to bad user experience.

12.2.5 Profile Page

One issue from the profile page was the automatic update of the user's select dietary restrictions from the questionnaire page to the profile page. In other words, even though the user can also change and select their dietary restrictions on the profile page similar to the questionnaire, their responses from the questionnaire were not saved. We have currently been unable to resolve this issue. Another issue encountered on the profile page was the implementation of the user adding a profile picture. Due to the limitations of using the React Native Image-Picker library with Expo Go for live testing, issues pertaining to grabbing a user's image from their device were encountered. Switching from React Native Image-Picker library to Expo Image-Picker library enabled the success for the implementation of the user adding a profile image. This is because the Expo Image-Picker library is more compatible with Expo Go than the React Native Image-Picker library. Hence, we were able to resolve this issue that was encountered.

13.0 Testing

13.1 Testing the Software

Before launching the application, it is important to ensure that the application is able to function properly. There are several testing methods and tools that will be used to test the application. A tool that will help determine the performance of the application is Jest, a framework that tests JavaScript projects. This will help determine the overall performance of the application. A testing method that will be used is functional testing, which tests the functions in the application to ensure that they perform the tasks that were previously specified. Afterwards, performance testing can be executed to examine the overall performance of the application such as scalability, speed, and reliability.

13.2 Maintaining the Software

The corrective software maintenance method will be followed to maintain the software of the application. When an error occurs in the software, the problem must be addressed and solved as quickly as possible. This can also be enforced by taking in feedback from users who download the application. Whenever the user encounters an error, we can take their feedback to fix the errors. The feedback can also be used to further improve the application by taking in suggestions from the users. However, the current goal is to track any bugs or issues and solve them as quickly as possible before the users encounter them. The maintenance method is subject to change in the future as the application improves overtime.

13.3 Functional Testing:

Functional testing involves testing the mobile app's processes against each of the previously listed functional requirements. The goal of functional testing is to observe whether each of the functional specifications were successful or not. The following is a list of the functional requirements that were tested.

- Login and Register
- Questionnaire
- Display Home Page with Scan
- Barcode Scanning
- Navigate from Home Page to Data Pop-up After Scanning
- Navigation Bar
- Navigate from Home Page to User Profile Page
- Profile Page
- Navigate from Home Page to Previous History Page
- History Page
- Machine Learning for Recommendation of Similar Items

Functional Requirement	Pass/Fail
Login and Register	Pass
Questionnaire	Pass
Display Home Page with Scan	Pass
Barcode Scanning	Pass
Navigate from Home to Pop-Up	Pass
Navigation Bar	Pass
Navigate from Home to Profile	Pass
Profile Page	Fail
Navigate from Home to History	Pass
History Page	Pass
Recommendation of Similar Products	Fail

13.4 Non-Functional Testing

Non-functional testing pertains to testing of the app's processes against each of the previously mentioned non-functional requirements. These non-functional specifications are requirements that contribute to the user's experience. The following lists the non-functional requirements, which were tested to see if each passed or failed the testing phase.

- Performance
- Security
- Usability
- Reliability
- Scalability

Non-Functional Requirement	Pass/Fail
Performance	Pass
Security	Pass
Usability	Pass
Reliability	Pass
Scalability	Pass

13.5 UI/UX Testing:

The UI/UX testing involves testing the application to ensure that the overall user interface is functional. The goal of this test is to confirm that each component on a page works and is organized properly. The following chart demonstrates the different pages checked to determine if the components pass or fail.

UI/UX Requirement	Pass/Fail
Login and Register	Pass
Questionnaire	Pass
Home Page with Scan	Pass
Pop-Up	Pass
Navigation Bar	Pass
Profile Page	Pass
History Page	Pass

14.0 Version Control

We created and will maintain and utilize a Github account for the Sr. Project; where all code and document versions are submitted.

Github Link: https://github.com/Indy009

15.0 Conclusion

In conclusion, due to the increasing demand for personalized dietary management, this project involved creating an advanced food scanning application, catering to common personalized diets such as vegan, vegetarian, dairy-free, and more.

The application features user profiles for individuals to create their own personalized diet account. When users want to know if certain items are in compliance with their diet, they can use the application's scanner to scan the barcode of the item, serving queries to the OpenFoodFacts API. Machine Learning was also utilized to recommend to the user similar products of the items scanned that are in compliance with their dietary restrictions. Lastly, users can also view the history page to see their previously scanned items and their profile page to update their customizable dietary information.

React Native and Expo Go were chosen for the front-end development of the app due to their comprehensive and cross-platform benefits. The React Native Camera was utilized to connect to the OpenFoodFacts API for product scanning. Additionally, Canva and Figma were used for UX/UI prototyping, allowing for a user-friendly design. Google Firebase was used for the backend development due to its smooth compatibility with React Native. Google Firestore, Authentication, Cloud storage, Cloud functions, and Machine Learning purposes were all accomplished by using Google Firebase. Lastly, the implementation of Machine Learning

algorithms played a vital role in the image recognition and analysis aspect of the application, ensuring accurate product scanning and analysis.

16.0 Glossary

OpenFoodFacts: a database containing information about food products, such as the products' ingredients, allergens, nutrition information, etc.
Google Firebase: a backend cloud computing service that allows for mobile app development
Model-View-Controller (MVC) pattern: a software design pattern that arranges an application's logic three interconnected layers

Appendix A

Project Overview:

In response to the growing need for personalized dietary management, this project proposes the development of an advanced food/snack scanning application. The application will utilize cutting-edge image recognition and AI technology to analyze food products and identify their compliance with various dietary restrictions such as kosher, Halal, vegan, vegetarian, and any allergen-specific needs.

Additionally, the application will feature user profiles where individuals can specify their dietary restrictions, allowing for a customized experience. To enhance engagement, the app will also use a recommendation algorithm to provide users with similar products that follow their dietary needs. The process will involve scanning the barcodes of the user's purchased food items, which will serve queries to the OpenFoodFacts API, providing the user with all the necessary information. Moreover, the application will conveniently store the user's previous scans to provide a concise user archive and prevent redundancy.

For the front-end development of the app, React Native is chosen for its comprehensive and cross-platform capabilities. Additionally, Canva and Figma are going to be utilized for UX/UI prototyping, ensuring a user-friendly design. In terms of backend development, Node.js will be used, particularly for its smooth compatibility with React Native. Lastly, the implementation of Machine Learning algorithms is crucial for the image recognition and analysis component of the application, ensuring accurate product scanning and analysis.

Features:

Phase 1:

- Barcode Scanning: Simplifying the process of identifying products.
- User Profile: Personalizing based on individual dietary needs.
- Saved Items: History of previously scanned items.
- Recommendation for Similar Items: Items that are in compliance with the user's specific dietary needs.

Phase 2:

- Machine Learning: Utilizing image recognition for ingredient analysis and compliance checking.

Final Deliverables:

1. Final Report
2. Software files
3. Website
4. Final Video Presentation

Milestone Events (Prototypes, Draft Reports, Code Reviews, etc.):

#1 - By Feb 4th

- SRS and SDD

#2 – By March 17th

- Develop prototype

#3 – By March 31st

- Prototype presentations
- Peer reviews

#4 – By April 7th

- Review Draft of Final Report

#5 – By April 21st

- Product presentations

Meeting Schedule Date/Time

F2F Meeting on Monday around 11am – 2pm

Thursdays and Fridays on MS Teams at approximately 2pm – 4pm.

Collaboration and Communication Plan

Communication – Teams, Cellphones (Call/Text)

Collaboration- Teams, In-person

Version Control- GitHub

Project Schedule and Task Planning

Project Name: INDY-9 RED
Report Date: 1/30/24

					Milestone #1				Milestone #2				Milestone #S3				C-Day	
Phase	Tasks	Complete%	Current Status Memo	Assigned To	01/23	01/26	01/29	01/30	02/04	03/17	03/24	03/31	04/03	04/07	04/14	04/21	04/25	
Requirements	Project Selection	100%	Completed	All	3													
	Meet with SP	0%	Will meet	All		1												
	Project Plan	100%	Completed	All				5										
	Pre-design Prep	0%	Will meet	All			9											
Project design	SRS	0%	Not Completed	Bri, Victoria					18									
	SDD	0%	Not Completed	Bri, Victoria					22									
	Develop working prototype	0%	Not Completed	Ibra, Jedae						15								
	Test prototype	0%	Not Completed	All							5							
	Present prototype	0%	Not Completed	All								7						
Development	Review prototype design	0%	Not Completed	Ibra, Jedae									10					
	Rework requirements	0%	Not Completed	Bri, Victoria										8				
	Document updated design	0%	Not Completed	Bri, Victoria										3				
	Test product	0%	Not Completed	Ibra, Jedae											6			
Final report	Presentation preparation	0%	Not Completed	All											10			
	Poster preparation	0%	Not Completed	All											5			
	Final report submission to D2L and project owner	0%	Not Completed	All												15		
																	5	
			Total work hours	**147**	**3**	**1**	**9**	**5**	**40**	**15**	**5**	**7**	**10**	**11**	**21**	**15**	**5**	**0**

* formally define how you will develop this project including source code management

Legend
Planned
Delayed
Number Work: man hours

Version Control Plan

We created and will maintain and utilize a Github account for the Sr. Project; where all code and document versions are submitted.

Github Link: https://github.com/Indy009

Appendix B

Training:

STATEMENT OF COMPLETION

By signing below, I, Ibrahima Gueye, acknowledge that I completed the React Native tutorial.

Signed by:

_____ Ibrahima Gueye _____ _____4/26/2024_____

Team Member 1 Date

STATEMENT OF COMPLETION

By signing below, I, Jedae Lisbon, acknowledge that I completed the React Native tutorial.

Signed by:

_____ Jedae Lisbon _____ _____4/26/2024_____

Team Member 2 Date

STATEMENT OF COMPLETION

By signing below, I, Brianna Noel, acknowledge that I completed the React Native tutorial.

Signed by:

_____ Brianna Noel _____ _____4/26/2024_____

Team Member 3 Date

STATEMENT OF COMPLETION

By signing below, I, Victoria Kuswita, acknowledge that I completed the React Native tutorial.

Signed by:

_____ Victoria Kuswita_____ _____4/26/2024_____

Team Member 4 Date