INDY-9 RED — MyFoodScan App

Software Design Document

CS 4850, Section 01/02, Spring 2024

Feb 13, 2024

Ibrahima	Jedae
Bri	Victoria

Table of Contents

1. Introduction

1.1. Document Outline

- Document Description
    - Introduction
    - System Overview
- Design Considerations
    - Assumptions and Dependencies
    - General Constraints
    - Goals and Guidelines
    - Development Methods
- Architectural Strategies
    - Choice Product
    - Reuse of Existing Software Components
    - Future Plans
    - User Interface Paradigms
    - Error Detection and Recovery
    - External Database Management
    - Concurrency and Synchronization
- System Architecture
    - Subsystem Architecture
- Policies and Tactics
    - Choice of Product
    - Engineering Trade-offs
    - Coding Guidelines and Conventions
    - Testing the Software
    - Maintaining the Software
- Detailed System Design
    - Frontend Module
        - Home Page
        - Scanner
        - User Profile
        - History
        - Product Details
    - Backend Module
        - Firebase Authentication
        - Firebase Firestore
        - Firebase Cloud Functions
    - External API Integration
        - OpenFoodFacts API
    - Detailed Subsystem Design
- Glossary

1.2. Document Description

1.2.1. Introduction

The detail design phase plays a major role in the software engineering process. One important role of the design engineering activities is the Software Design Document (SDD). A SDD is an overall written description of a software product's design, detailing its general architecture. The purpose of this SDD is to highlight the architecture and system design of our mobile application, doing so in such a manner that allows the software development process to progress with a clear indication of how the software should be built.

Overall, the goal of this document is to describe the "how" of the development process, making sure that the possibilities of the product and how it can be achieved is clear to all. Thus, the intended audience of this SDD include any overseers who are interested in reading an explanation of the design process of the application and how it is planned to be built. A prerequisite for this document is the software requirements specification (SRS), which provides background of the requirements for the application. The contents of this document will also play a major role in the development of the final report document.

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 a database containing nutrition information, providing the user with the necessary knowledge about their products. Moreover, the application will conveniently store the user's previous scans to provide a concise user archive and prevent redundancy.

For the development of the app, a frontend framework with comprehensive and cross-platform capabilities will be chosen. Additionally, UX/UI prototyping with a user-friendly design will be conducted. In terms of backend development, a backend cloud computing platform will be used, particularly for its smooth compatibility with the frontend framework. Lastly, the implementation of Machine Learning (ML) algorithms is crucial for the image recognition and recommendation component of the application, ensuring accurate product scanning and analysis.

1.2.2. System Overview

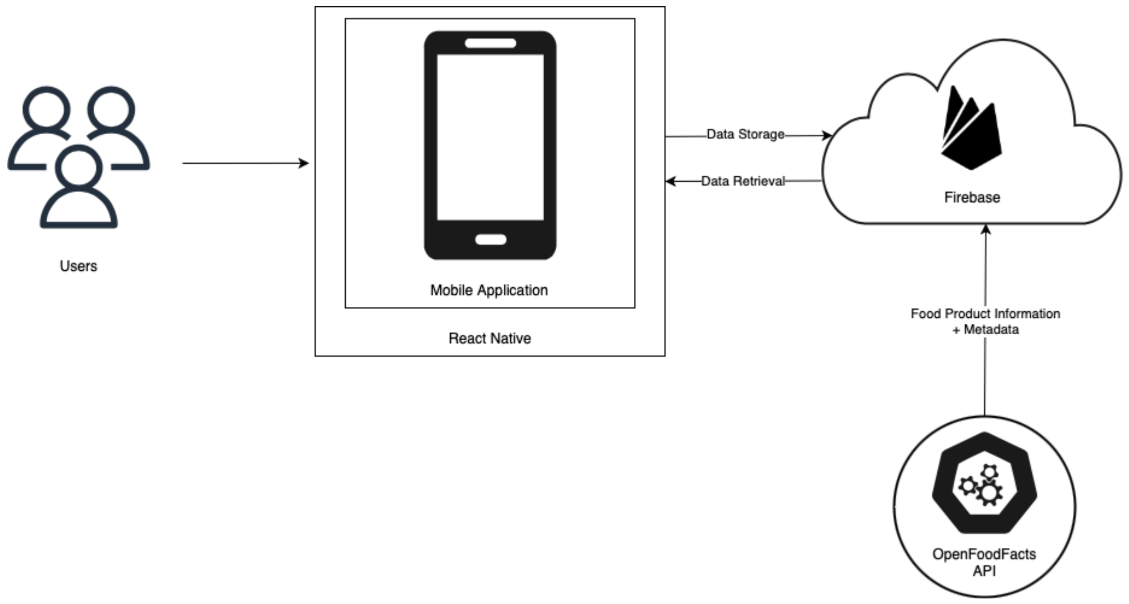

Figure 1 Diagram of Application Overview

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.

2. Design Considerations

2.1. Assumptions and Dependencies

There exist 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.

2.2. General Constraints

Physical, operational, and deployment environments that should all be considered. In regard 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.

2.3. 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.

2.4. 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. Architectural Strategies

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.

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.

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.

User Interface Paradigms

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.

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.

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.

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.

4. 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.

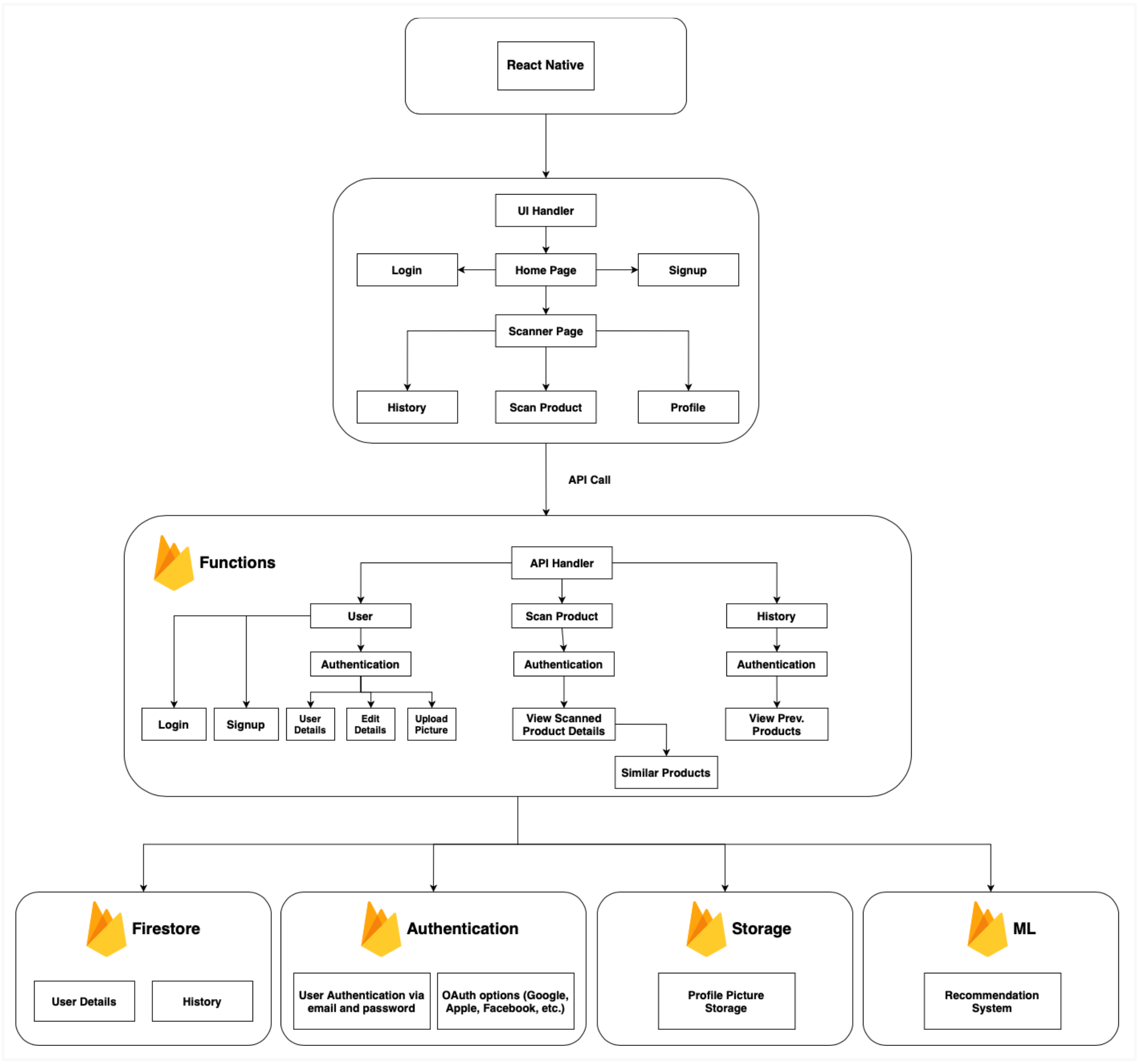

Figure 2 High-Level Architecture Diagram

4.1. 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 component needs 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.

5. Policies and Tactics

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.

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.

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.

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.

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.

6. Detailed System Design

6.1. Frontend Module

6.1.1. 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.

6.1.2. 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.

6.1.3. 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.

6.1.4. History

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.

6.1.5. 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.

6.2. Backend Module

6.2.1. 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.

6.2.2. 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.

6.2.3. 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”).

6.3. External API Integration

6.3.1. 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.

6.4. 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.

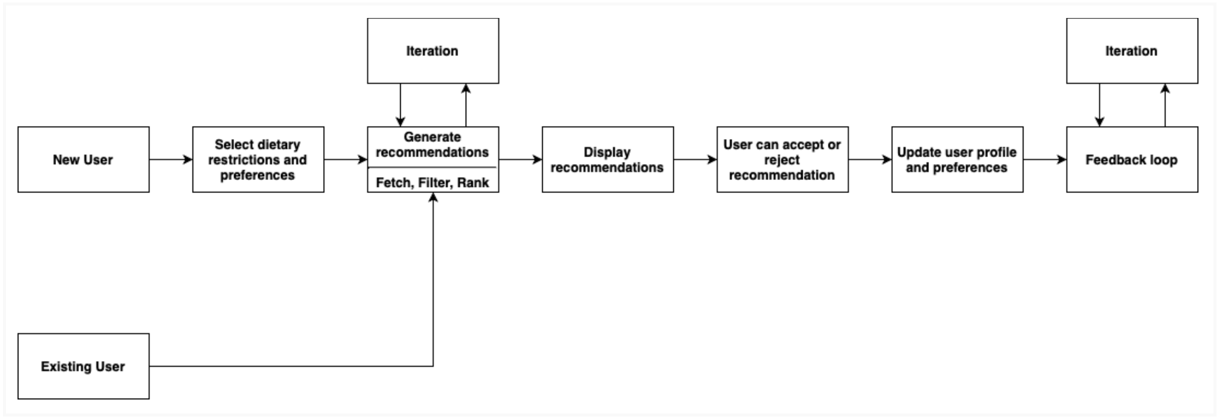
Iteration
Iteration
New User
Select dietary restrictions and preferences
Generate recommendations
Fetch, Filter, Rank
Display recommendations
User can accept or reject recommendation
Update user profile and preferences
Feedback loop
Existing User

Figure 3 Overview of Recommendation System

7. 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