**Project Implementation:**

**Diet Mobile Application - MunchHealthy**

Dhruv Rana

CSIT335\_01SP24 – Human Computer Interaction

April 18, 2024

Dr. Hongbo Zhou

**Sections Table of Contents Page**

[**Major Implemented Features in Digital Prototype 2**](#_hyx0gimhf9jm)

[**Implementation of Features 2**](#_9qgpxve7d7)

[**1. Create Account 2**](#_3x2nsil85omt)

[**2. User login 3**](#_bw3hsav0hruf)

[**3. Calorie Tracker 3**](#_l240ef4z5yie)

[**4. Food Tracking 4**](#_b9yznjqgq0gh)

[**5. View Meal Plans 5**](#_6c6jmg1m4xdn)

[**6. Profile Page 6**](#_va36cdwz3a1a)

# Major Implemented Features in Digital Prototype

1. Create Account
2. User Login
3. Calorie Counter
4. Food Tracker
5. View Meal Plans
6. Profile Page

# Implementation of Features

## 1. Create Account

**Possible Solutions:**

* You can implement a custom account creation system using server-side scripting languages like PHP, Python, or Node.js along with a database.
* You can also utilize a third-party authentication and management service like OAuth or Firebase Authentication.

**Limitations:**

* Custom and self developed implementation may require more development effort and maintenance, as well a database management system.
* Third-party services cost more than self developed systems, and may have limitations on customization, integration, and settings.

**Proposed Solution:**

* I would create a user registration form where new users can sign up for an account using HTML, PHP, and JavaScript. The form will collect necessary information such as username, email address, password, and any other relevant details, and store it in a database like MySQL on phpMyAdmin. By implementing server-side validation to ensure that the provided information is valid and meets the strong password policy. When the user enters their password, the password will be hashed and securely stored using hashing algorithms like bcrypt, and store the password in the database. During the account creation process, ask the user for personal information which can be used to make their profile later on.

## 2. User login

**Possible Solutions:**

* You can implement session-based authentication using server-side technologies like PHP or Django.
* You can use token-based authentication with JSON Web Tokens for both client and server sides.
* Utilize a third-party authentication and login like with Google Sign-In.

**Limitations:**

* Session-based authentication may be susceptible to session fixation attacks if not implemented securely.
* JWT-based authentication requires additional handling of tokens on both the client and server sides.

**Proposed Solution:**

* I would create a form where the user can enter their username and password to login using HTML, PHP, CSS, and JavaScript. When attempting to login, the server will retrieve the user’s data from the MySQL database and check the user’s credential and authenticate them. By implementing a token-based authentication using JSON Web Tokens with Node.js, a JWT is securely on the client side and verifies the token on the server side when logging in to ensure that it is the right user. For each login, set a token expiration time to improve the security, and the expiration time will be refreshed based on the user’s activity on the app, so the user does not have to login every time they open the app.

## 3. Calorie Tracker

**Possible Solutions:**

* Retrieve calorie information from a centralized food database based on user-entered food items by importing the database and having the system search through the database for tracking purposes.
* Develop a calorie calculation algorithm based on standard nutritional data using JavaScript.

**Limitations:**

* Database retrieval relies heavily on the availability and accuracy of food data in the database so it may not cover all food items or food variations.
* A calculation algorithm requires accurate nutritional data and possibly a nutritionist expert.

**Proposed Solution:**

* I would implement a mixed approach combining database lookup for common food items and a calculation algorithm for custom or user-defined foods based on the nutrition values of the items used to make the food. By using JavaScript to implement the USDA FoodData Central API, it makes it easier to look up food data and implement a calorie calculation algorithm based on already provided nutritional data.

## 4. Food Tracking

**Possible Solutions:**

* Allow users to search for food items in a database using a search bar feature or manually enter nutritional information using a form in HTML, PHP, and JavaScript.
* Integrate and use a barcode scanner to scan food packaging and retrieve nutritional data from a database or API.

**Limitations:**

* Manual entry relies on the user knowing their food data and can lead to inaccurate or incomplete data. Users may also find manual entry tedious, especially for complex and harder meals.
* Barcode scanning requires access to a food database and barcode scanning technology, and users may have issues with barcode recognition for certain products.

**Proposed Solution:**

* I would combine manual entry and barcode scanning features to allow for faster, flexible and accurate results. The USDA FoodData Central API provides nutritional information for various foods so you can implement the library using JavaScript, and so when the user searches for a food item using the search bar feature, the server fetches the information from the API. For the barcode scanning, use Scandit, an SDK. An pre-built SDK makes it easier to turn a phone camera into a barcode scanner, so a user always has a barcode scanner on them.

## 5. View Meal Plans

**Possible Solutions:**

* Allow users to create custom meal plans by selecting food items from MySQL database and then entering their own recipes. This approach offers flexibility but requires manual effort from users.
* Provide pre-designed meal plans tailored to specific dietary preferences or goals such as keto and vegan by creating a MySQL database table for each meal plan, and then retrieving the data from the tables. This approach offers convenience but may not suit individual preferences.

**Limitations:**

* Custom meal plans rely heavily on user knowledge of nutrition and meal planning, as well as user data retrieval and storage for their recipes, which takes up many resources.
* Predefined meal plans need to be hard coded into the database prior, and cannot be updated frequently.

**Proposed Solution:**

* I would use predefined meal plans as it reduces the need for additional forms and retrieval of data from the user. The predefined meal plans are stored in the MySQL database, and the front end is made using HTML, CSS, and PHP, and when the user requests to view a meal plan, the data is retrieved and presented to the user in the front end view.

## 

## 6. Profile Page

**Possible Solutions:**

* Collect user information during account creation and store the user profile data locally on the device using built-in storage mechanisms such as SharedPreferences for Android or UserDefaults for iOS, and then present the data on a screen using HTML, CSS, PHP, and JavaScript.
* Use a server-side database to store user profile data on a remote server using a database like MySQL and then present the data on a screen using HTML, CSS, PHP, and JavaScript.

**Limitations:**

* When using local storage on devices, data is not synced across devices and can be lost if the app is uninstalled.
* Server side database storage for profile data requires internet connection and server maintenance.

**Proposed Solution:**

* I would collect the user information during account creation and store the profile data in a MySQL database as it can be stored for longer periods of time, under my control. Then using JavaScript and PHP for retrieving user specific data using user JSON Web Tokens, and display the data to the user on a profile using HTML, and CSS. In case the user wants to make updates to their profile, an updates feature will be implemented, which is another form where the user can input new information. This information is then saved into the database. The profile page will also have a sign out button, which when tapped will expire the user’s token, signing them out of the app.