# BSCS FINAL PROJECT

# SugarSage – AI Companion For Diabetics


Advisor: **Dr. Imran Arshad Choudhary**

Presented by:
**Group ID: F23CS014**


| Student Reg# | Student Name |
|---|---|
| L1F20BSCS0204 | Khizar Iqbal |
| L1F20BSCS0207 | Ahmed Ali |
| L1F20BSCS0466 | Hamail Shahbaz |
| L1F20BSCS0467 | Safoora Masood |

**Faculty of Information Technology**

**University of Central Punjab**

# SugarSage – AI Companion for Diabetics

By

Khizar Iqbal
Ahmed Ali
Hamail Shahbaz
Safoora Masood

Thesis/Project submitted to

Faculty of Information Technology,

University of Central Punjab,

Lahore, Pakistan.

in partial fulfillment of the requirements for the degree of

**BACHELOR OF SCIENCE**
**IN**
**COMPUTER SCIENCE**

| | |
|---|---|
| Project Advisor | Manager Projects |

# Abstract

This project introduces SugarSage, a mobile application powered by AI, specifically designed to address the complex dietary management needs of diabetic patients in Pakistan. The problem of diabetes is significant in Pakistan, with a staggering 31% of the population affected. SugarSage aims to fill the gap in available resources for diabetic individuals by offering personalized dietary recommendations that consider individual food preferences, sugar levels, energy requirements, and locally available food options.

The application utilizes machine learning techniques to analyze various factors influencing dietary choices and suggests optimal diet plans. Additionally, it allows users to record calories count, track walking and sleeping patterns, and maintain a calorie count, thus providing a comprehensive solution for diabetes management. The project's scope includes not only dietary recommendations but also the development of a user-friendly interface and functionalities that allow for effective monitoring of diabetes-related activities.

The outcomes of this project are expected to empower diabetic patients in Pakistan to take control of their diabetes management process, thereby improving their overall well-being and quality of life. The project also aims to provide valuable insights into the application of machine learning algorithms in a real-life healthcare scenario, along with experience in mobile application development and backend management.

# Dedication

This project is driven by a steadfast commitment to revolutionizing diabetic care and management. Our dedication to enhancing healthcare through AI-powered solutions is embodied in SugarSage, an innovative platform designed to offer personalized dietary recommendations and comprehensive diabetes management. By utilizing advanced machine learning techniques, we aim to empower diabetic patients to take control of their health and improve their quality of life.

Our mission is to establish a secure, efficient, and user-friendly platform that sets a new standard for personalized healthcare. We acknowledge the critical need for tailored dietary plans and effective diabetes management tools in Pakistan, where diabetes affects 31% of the population. By harnessing the power of AI and machine learning, we strive to transform the management and delivery of diabetic care.

With an unwavering commitment to integrity, transparency, and efficiency, we pledge to uphold the principles of ethical AI usage and patient-centered care. Through SugarSage, we provide a reliable solution that enhances the well-being of individuals, freeing them from the limitations of traditional diabetes management methods. Each feature of SugarSage serves as a testament to the transformative potential of technology in healthcare.

We dedicate ourselves to creating a future where diabetic care is personalized, effective, and accessible to all. By empowering individuals and healthcare providers alike, SugarSage stands as a symbol of progress, breaking the barriers of conventional diabetes management. We envision a world where personalized healthcare is embraced without skepticism, supported by cutting-edge technology.

This dedication reflects our collective passion for change and our commitment to creating a healthier future for diabetic patients. It underscores our ongoing efforts to innovate and improve healthcare, enabling individuals to confidently pursue their health goals without obstacles.

# Acknowledgements

The completion of this final year project for our university has been a journey marked by dedication, learning, and collaboration. We extend our heartfelt gratitude to all individuals and entities whose support and contributions have been instrumental to the success of this project.

Firstly, we are deeply grateful to our esteemed faculty advisor, Dr. Imran Arshad Choudhry, whose expert guidance, insightful feedback, and encouragement have been invaluable throughout this project. His exceptional insights have inspired us to strive for excellence and have provided us with the direction needed to refine our ideas.

We also extend our appreciation to our dedicated team members—Khizar Iqbal, Ahmed Ali, Hamail Shahbaz, and Safoora Masood. Their unwavering commitment and willingness to tackle challenges created an environment of creativity and productivity. Each member's unique skills and contributions were crucial in bringing this project to fruition.

We acknowledge with gratitude the academic and technology research community, whose significant advancements in AI and machine learning have been a source of inspiration and guidance. Their contributions paved the way for the innovation at the heart of our project.

Our thanks also go to the open-source communities, whose spirit of collaboration and shared knowledge greatly enriched our project's development. Their resources and collective wisdom enabled us to build on existing knowledge and make significant progress.

Lastly, we offer our heartfelt thanks to our families and friends for their unwavering support, understanding, and encouragement throughout our academic journey. Their belief in our potential and their sacrifices have been a constant source of strength and motivation.

As we celebrate the successful completion of this final year project, we express our gratitude to the university and the broader community of supporters who have significantly shaped this scholarly journey. We look forward to applying the lessons learned and the skills acquired through this project, with a steadfast commitment to the pursuit of knowledge. We are excited to see the positive impact of SugarSage on diabetic care and beyond.

# List of Figures

# List of Tables

# Table of Contents

# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| | | | |
| | | | |

# Chapter 1. Introduction

Managing the dietary needs of individuals with diabetes has always been a tricky and complex task, and it becomes even more challenging for diabetic patients in Pakistan who lack readily available resources to help them cope with their food-related issues. The food they consume plays a crucial role in maintaining their sugar levels and managing the severity of their diabetes, making it essential to have an effective dietary plan.

While doctors can recommend food options, their pre-made generic eating plans may not consider individual preferences and health-related information nor consider all the possible effective and appealing diets for each individual. Moreover, doctors may not always be available to monitor and track patients' diabetes-related activities, leaving patients with limited support and guidance. With Pakistan having the world's highest diabetes ratio of 31%, it makes a dire need to develop a system that can manage the dietary requirements of diabetics, including keeping track of their everyday routines such as walking, sleeping and calorie count.

SugarSage is an AI-based mobile application that uses machine learning techniques to suggest optimal diet plans for people with diabetes while considering their food preferences, sugar levels, and energy requirements. The app takes a systematic approach to process all the possibilities and gives the most effective option for individual users.

In addition to providing personalized diet plans, SugarSage also allows users to record their calorie count, track their walking and sleeping, and keep a calorie track. These features enable users to monitor their diabetes-related activities and make informed decisions about their health. The app's user-friendly interface and functionalities make managing dietary needs cost-efficient and convenient for diabetic patients in Pakistan.

SugarSage empowers diabetic patients in Pakistan to take control of their diabetes management process and overcome the challenges associated with managing their dietary needs. The app's comprehensive features and functionalities provide users with a personalized and effective solution to manage their diabetes, improving their overall well-being and quality of life

## 1.1 Product (Problem Statement)

In Pakistan, 31% of the population is affected by diabetes, a chronic condition with severe repercussions when poorly managed. Poorly managed diabetes can lead to vision loss and limb amputations, according to the World Health Organization. Diet is one of the key factors in effective diabetes management. A well-balanced diet can significantly control blood sugar levels, reduce dependency on medication, and mitigate the risk of complications. However, medications remain the more commonly utilized approach to manage the condition. These medications can introduce additional health issues, such as hormonal imbalances and gastrointestinal disturbances.

Unfortunately, the currently available diet plans for diabetics lack personalization. Also it is impossible for a doctor to provide 24/7 monitoring. To fill this gap, we introduce SugarSage, an AI-driven mobile application that provides personalized dietary management solutions to diabetics in Pakistan. With SugarSage, users can access around-the-clock monitoring and receive locally relevant dietary advice tailored to their individual preferences and health requirements. The app integrates state-of-the-art machine learning algorithms to analyze dietary choices and suggest suitable meal plans. It also allows users to track their physical activities and maintain a healthy lifestyle. SugarSage is a complete diabetic care system that offers a user-friendly interface, making it a valuable resource for

managing diabetes effectively and conveniently.

## 1.2 Background

Pakistan has the highest ratio of diabetics in the world. Despite being the most in need of awareness and facilities to help people manage diabetes efficiently, there is a lack of both. Living with diabetes requires managing it through a combination of medication and diet. While medication is important, it's equally important to keep an eye on what a diabetic person eats. Their diet can either contribute to the development of diabetes or help reduce it. In some cases, a balanced diet can even control diabetes more efficiently than medication because medications can cause complications such as stomach problems, kidney issues, hormonal imbalances, and more.

Although diabetes dietitians can provide diet plans for diabetics, they often lack enough customization based on the patient's taste preferences. While if asked, they may make some adjustments to a rigid plan, it may not be as effective as needed. However, if a system could access a large database of foods, consider the complexities of what is good and what is not for a particular diabetic person, and consider the patient's preferences, it could offer a tailored diet plan. Such a system would be a step forward in helping patients manage their diabetes with ease.

## 1.3 Objective(s)/Aim(s)/Target(s)

- The main objective is to provide a better solution to Pakistani diabetes patients for managing their diet.
- Patients can easily manage diabetes with an all-in-one app for tracking sugar levels, calories and activities.
- Providing patients with 24/7 accessibility for easy management of diabetes.

## 1.4 Scope

The main scope of this project lies in dietary recommendations for diabetics based on these factors:

1. Their sugar levels
2. Their food preferences
3. Their energy requirements
4. Locally available food options

The secondary scope lies in that the app will be able to track sugar (user input dependent), detect and count daily steps, detect, and measure sleep time. The app will provide users with access to blogs and news about diabetes and diet.

This system does not aim to cure but control diabetes through diet. The system cannot assist people with severe diabetes complications that can no longer be controlled merely through diet. The system does not provide an alternative to a real-life diabetes doctor who can provide professional medical consultation.

It must be noted that any misuse of the system or failure to use it as intended falls outside the scope of this project. The system is not responsible for forcing users to comply or recover from misuse, as these are not part of the project's objectives.

## 1.5 Business Goals

The proposed system will be able to provide round-the-clock services to its users for diabetes management. One of the business goals include introducing a subscription model for premium features, including personalized support, and advanced tracking capabilities. That being said, implementing sustainable monetization strategies while ensuring accessibility for all income groups. Another business goal is establishing SugarSage as a sole and leading diabetes management app in Pakistan. We aim to continuously grow the user base by providing exceptional services and user experience. We also aim to prioritize user satisfaction through continuous improvement based on feedback and evolving needs.

## 1.6 Challenges (P1)

- Acquiring datasets for diabetes patients
- Developing a comprehensive food database
- Collecting local foods and their nutrients
- Learning and making Mobile Application
- Learning and implementing Data Science and Machine Learning
- Learning and implementing efficient APIs
- Designing an easy and interactive UI

## 1.7 Learning Outcomes ()

- To better understand how machine learning algorithms work and how they can be implemented in a real-life scenario.
- To get hands-on experience of developing a mobile application from scratch.
- To gain practical experience in building and managing backends and databases for real-life scenarios.
- To understand how the APIs are built for effective communication between front-end and back-end.
- To understand the intricacies of designing a user interface that is user-friendly.

## 1.8 Nature of End Product

The end product of our project will be a mobile app written in React Native, called “SugarSage” with a ML Algorithm in the backend. The backend will be made in NodeJS, and the ML Algorithm will be written in Python connected to NodeJS through APIs. The app will feature a user-friendly interface.

## 1.9 Related Work/ Literature Survey/ Literature Review

1. **Papers:**

   **A. Intelligent Nutrition Diet Recommender System for Diabetic's Patients [1]**

   **Objective:**

   This paper aimed to create an intelligent nutrition diet recommender system specifically tailored for diabetic patients. The focus was on offering individualized meal plans, considering various factors such as medical requirements, personal constraints, and socio-cultural differences.

**Findings:**

The research concluded that their expert agent system was effective in suggesting individualized food plans for diabetic patients. They employed fuzzy logic to enhance the accuracy of recommendations and incorporated adaptive learning techniques for ongoing improvement.

**B. Pakistan's Recommendations for Optimal Management of diabetes from Primary to Tertiary care level [2]**

**Overview:**

The document discusses the updated National Clinical Practice Guidelines for Type-2 diabetes management in Pakistan. It focuses on developing affordable and accessible treatment, establishing a referral system from primary to tertiary care, and forming multidisciplinary teams for comprehensive diabetes management. The guidelines include diagnostic criteria, referral criteria, lifestyle and pharmacological management strategies, and address complications like diabetic foot, obesity, hypertension, and dyslipidemia. These comprehensive guidelines aim to improve diabetes care across Pakistan.

**C. Pakistan and diabetes-A country on the edge [3]**

**Overview:**

The paper "Pakistan and diabetes-A country on the edge" focuses on the increasing burden of diabetes in Pakistan. It attributes this rise to lifestyle changes, including sedentary habits and poor diet, leading to obesity. The paper emphasizes the need for a cost-effective, population-based approach for diabetes screening and management, highlighting the results of the National Diabetes Survey of Pakistan.

**D. Current management strategies to target the increasing incidence of diabetes within Pakistan [4]**

**Overview:**

"Current management strategies to target the increasing incidence of diabetes within Pakistan" discusses the strategies for managing the growing diabetes incidence in Pakistan. It stresses the importance of developing multidisciplinary teams and standardized protocols for healthcare professionals in diabetes care.

## 1.10 Document Conventions

1. **Section Numbering:**
   - The document uses a hierarchical numbering system for sections, sub-sections, and sub-subsections (e.g., 1., 1.1, 1.1.1).

2. **Font Styles:**
   - Italics: Italics are used for emphasis, titles of books, and when introducing new terms or concepts for the first time.
   - Bold: Bold text is used for headings, subheadings, and to highlight important points.

3. **Lists:**
   - Bullet Points: Bullet points are used for listing items within a section.
   - Numbered Lists: Numbered lists are used when presenting a sequence of steps or items.

4. **Tables:**
   Tables are employed for organizing and presenting structured data, such as use case descriptions and functional requirements.

5. **Quotes:**
   Block quotes are used for excerpts from external sources, such as references, to clearly distinguish them from the main text.

6. **Hyperlinks:** Hyperlinks are used for references, citations, and external resources to provide easy access to additional information.

7. **Acronyms and Abbreviations:** Acronyms and abbreviations are spelled out upon first use, followed by the abbreviation in parentheses. The abbreviation is then used consistently throughout the document.

# Chapter 2. Overall Description

## 2.1 Product Features

The "SugarSage - AI Companion for Diabetics" project aims to address the challenges of diabetes management in Pakistan considering user's local food preferences. It offers the following key features:

Product Features:

1. **Personalized Diet Planning:** The system includes an AI algorithm capable of generating personalized diet plans for diabetics, taking into account individual health metrics, medication regimens, and activity levels.

2. **Local Food Database:** The system integrates a comprehensive food database that considers local Pakistani dietary data, ensuring that dietary recommendations align with cultural norms, making it more feasible for users to follow.

3. **Mobile Application:** The end product is a user-friendly mobile application that serves as a digital dietitian for diabetics. Users can access and adhere to their personalized dietary recommendations through this intuitive platform.

4. **User Experience:** The project focuses on designing a user-friendly interface that caters to a diverse user base, from tech-savvy individuals to those less familiar with mobile applications, ensuring ease of use and accessibility.

5. **Activity Tracking:** Integration of manual entry of physical activity can help users monitor their exercise routines (Daily steps, sleep schedule). The app can adjust meal plans based on activity levels to help maintain blood sugar levels.

6. **Health Reports:** Users can generate health reports summarizing their progress and adherence to the dietary recommendations. These reports can be shared with healthcare providers for better-informed care.

## 2.2 User Classes and Characteristics

1. **Diabetic Individuals:**
   Diabetic individuals are the primary users of the product. They vary in terms of age, gender, and cultural background. They may have different levels of technical expertise, ranging from tech-savvy individuals to those less familiar with mobile applications.

   Usage:
   - Diabetic individuals use the mobile application to access personalized diet plans and manage their diabetes diet.
   - They input their health metrics, medication information, and activity levels.
   - They follow the recommended meal plans, track their progress, and receive guidance and feedback.
   - They may log their meals, monitor their physical activity, and record blood glucose levels using the app.

2. **Technical Support Team:**
   Technical support specialists with expertise in the product.
   Usage:

- Users who encounter technical issues may contact the technical support team for assistance.
- The support team helps troubleshoot and resolve technical problems.

## 2.3 Design and Implementation Constraints

- **Hardware Limitations:**
  Mobile devices may have limited processing power, memory, and storage capacity. The application must be designed to operate efficiently within these constraints to ensure optimal performance.
- **Compatibility with Mobile Operating Systems:**
  The app must be compatible with various versions of Android and iOS. Ensuring compatibility and optimizing performance across different devices and OS versions can be a constraint.
- **Local Food Database Integration:**
  Incorporating local dietary data may be challenging due to the availability and accuracy of such data. Efforts to obtain and maintain this data can be a constraint.
- **AI and ML Model Development:**
  Developing and fine-tuning AI and ML algorithms to generate personalized diet plans may require access to large and diverse datasets, which could be a constraint if such data is limited or hard to acquire.

## 2.4 Assumptions and Dependencies

**Assumptions:**

1. **Data Availability:** It is assumed that relevant dietary and nutritional data, including local food databases, will be available or obtainable for integration into the application.

2. **User Adoption:** The project assumes that diabetic individuals in Pakistan will adopt and use the mobile application effectively for managing their condition.

**Dependencies:**

1. **Mobile Operating Systems:** The application is dependent on the stability and compatibility of Android and iOS operating systems, as updates or changes can impact the app's performance.

2. **AI and ML Models:** The development of AI and ML algorithms for personalized diet planning depends on access to suitable datasets and resources for model training and fine- tuning.

3. **User Acceptance:** The project's success is dependent on user acceptance and engagement. Ensuring that the app aligns with user needs and preferences is critical.

# Chapter 3. System Requirements

## 3.1 Functional Requirements

### 3.1.1 Patient Use Cases

#### 3.1.1.1 Name of Use-Case 1: Signup

| | |
|---|---|
| **Identifier** | UC-1 |
| **Purpose** | User can create a new account. |
| **Priority** | High |
| **Pre-conditions** | 1. User should not have an account in the system.<br>2. Internet connectivity required. |
| **Post- conditions** | **Failure guarantees:**<br>1. Registration failed due to invalid input.<br>2. User already exists.<br><br>**Success guarantees:**<br>1. User successfully registered. |

| **Typical Course of Action** | | |
|---|---|---|
| **S#** | **Actor Action** | **System Response** |
| **1** | The user clicks on the "Sign up" option on the Login screen of the application. | The system will redirect to the signup form. |
| **2** | The user enters email, password, name, and medical information (activity levels, blood sugar levels etc.) on the registration form and clicks the Signup button. | The system will authenticate the user credentials and create the user account and redirect to the Login Page of the application. |

| **Alternate Course of Action** | | |
|---|---|---|
| **S#** | **Actor Action** | **System Response** |

| | | |
|---|---|---|
| **1** | The user enters email, password, medical information, etc., and clicks the Signup button. | The system detects that the user credentials are invalid and displays an error message.<br>**Example:**<br>1. "Your Password doesn't contain any special characters."<br>2. "Your BMI input is invalid."<br>3. "Email doesn't match the |

**Table 1: UC-1**

#### 3.1.1.2 Name of Use-Case 2: Login

| **Identifier** | UC-2 | |
|---|---|---|
| **Purpose** | User must be to access their account. | |
| **Priority** | High | |
| **Pre-conditions** | 1. User should be registered in the system.<br>2. Internet connectivity required. | |
| **Post- conditions** | **Failure guarantees:**<br>1. The user account does not exist in the system.<br>**Success guarantee:**<br>1. The user successfully logged in. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | User clicks on "Login" option on the Signup Page or accesses SugarSage web or mobile app. | The system will redirect you to the login form. |
| **2** | The user enters the email and password on the login form and clicks the Login button. | The system will verify user credentials and redirect to the home screen of the application. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user enters the email and password on the login form and clicks the Login button. | The system detects that the user doesn't exist in the database and displays the error message<br>**"User doesn't exist."** |

**Table 2: UC-2**

### 3.1.1.3 Name of Use-Case 3: Get Meal Plan

| | | |
|---|---|---|
| **Identifier** | UC-3 | |
| **Purpose** | To generate personalized meal plans, delete meal plans, and share meal plans. | |
| **Priority** | High | |
| **Pre-conditions** | 1. User should be logged in on the system<br>2. User medical information should be known.<br>3. Internet connectivity required. | |
| **Post- conditions** | System will generate a meal plan according to user preferences | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user navigates to the "Meal Plan" Page and clicks on the Generate Meal Plan button. | The system will ask to enter their current blood sugar level. |
| **2** | User enters their current blood sugar level within the range. | The system will generate a list of top food recommendations based on user preferences and health data. |
| **3** | User can select food items, their mealtimes, and portion sizes. | The system updates and displays the selected meal plan on the "View Meal Plans" Page. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | User enters their current blood sugar level above or below the range. | The system will display an error asking the user to enter correct value. |

**Table 3: UC-3**

### 3.1.1.4 Name of Use-Case 4: User Profile

| **Identifier** | UC-4 |
|---|---|
| **Purpose** | Allow users to view and edit their personal information, as well as check their achievements and give feedback. |
| **Priority** | Medium |
| **Pre-conditions** | 1. User should be logged in on the system.<br>2. Internet connectivity required. |
| **Post- conditions** | User profile is successfully viewed and edited. |

| **Typical Course of Action** | | |
|---|---|---|
| **S#** | **Actor Action** | **System Response** |
| **1** | The user clicks on the "User Profile" icon on the application. | The system will display personal information and a few achievements of the user and the option to log out. |
| **2** | | The system also shows options to edit personal information and view more achievements. |
| **3** | The user clicks on the edit icon on the Personal Information. | The system will allow the user to edit their personal information. |
| **4** | The user edits their personal information and saves it. | The system will verify and update the edited information on the database. |
| | | The system now displays the newly updated personal information. |
| **5** | The user clicks on "View More Achievements". | The system will redirect to another page and display the user's achievement in detail. |
| **6** | The user clicks on the "Give Feedback" option | The system will redirect to the feedback form page. |
| **7** | The user writes the feedback and click the submit button. | The system will store the feedback in the database and redirect to the User Profile tab. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user edits their personal information and saves it. | The system detects that the edited information is incorrect and displays an error message. |

**Table 4: UC-4**

### 3.1.1.5 Name of Use-Case 5: Health Profile

| | | |
|---|---|---|
| **Identifier** | UC-5 | |
| **Purpose** | Users can update their health status. | |
| **Priority** | Medium | |
| **Pre-conditions** | 1. User should be logged in on the system<br>2. Internet connectivity required.<br>3. User gives permissions to access device sensors and locations etc. | |
| **Post- conditions** | **Minimal guarantees:**<br>1. Invalid input values for HbA1c score, and blood sugar levels.<br>**Success guarantee:**<br>1. Successfully updated HbA1c score, and blood sugar levels.<br>2. Successfully viewed health information. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user clicks on “Health Profile” icon on the application. | The system will display health information like sleep tracking, physical activity (steps), HbA1c score, and blood sugar. |
| **2** | | The system shows option view sleep patterns, physical activity track, and blood sugar in detail. |
| **3** | | The system also shows the option to update HbA1c score and blood sugar etc. |
| **4** | The user clicks on update icon and enters updates values for their HbA1c score, and blood sugar etc. | The system verifies the values and updates them in the database. |
| **5** | | The system now displays the newly updated information. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |

| | | |
|---|---|---|
| **1** | The user clicks on update icon and enters updates values for their HbA1c score, and blood sugar etc. | The system detects the incorrect values for HbA1c score, blood sugar and displays an error message. |

**Table 5: UC-5**

#### 3.1.1.6 Name of Use-Case 6: Logout

| **Identifier** | UC-6 | |
|---|---|---|
| **Purpose** | Users can log out of the system | |
| **Priority** | Low | |
| **Pre-conditions** | 1. User should be logged in on the system<br>2. Internet connectivity required. | |
| **Post- conditions** | User has successfully logged out. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user clicks on "User Profile" icon on the application. | The system will display other information along with the option to logout. |
| **2** | The user clicks on the logout option. | The system displays the dialogue box with message" Are you sure you want to log out from SugarSage?" |
| **3** | The user clicks on "Yes" button. | The system ends the session and redirects to the startup screen. |
| | | Go to UC-1 or UC-2 |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | N/A | N/A |

**Table 6: UC-6**

### 3.1.2 Admin Use Cases

#### 3.1.2.1 Name of Use-Case 7: Admin Login

| **Identifier** | UC-7 | |
|---|---|---|
| **Purpose** | Admin will be able to access their account. | |
| **Priority** | High | |
| **Pre-conditions** | 1. Admin should exist in the system.<br>2. Internet connectivity required. | |
| **Post- conditions** | **Minimal guarantees:**<br>The admin account does not exist in the system.<br>**Success guarantee:**<br>The admin successfully logged in. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin enters the email and password on the login form and clicks the "Continue" button. | The system will check whether the user exists in the system and upon successful authentication, redirect to the Dashboard Page. |
| **2** | The admin can reset their password if they forget by clicking the "Forgot your password" button. | The system will redirect user to the authentication page, asking them to enter the OTP code sent to their contact number. |
| **3** | The admin enters the OTP code on the authentication page. | The system verifies the code and upon success, displays a form allowing the user to reset their password. |
| **4** | The admin enters their new password and clicks the "Confirm" button. | The system will redirect the user to the login form. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin enters the email and password on the login form and clicks the Login button. | The system detects that the admin does not exist in the database and displays the error message<br>**"Admin doesn't exist."** |
| **2** | The admin enters the OTP code on the authentication page. | The system verifies the code and if the input is incorrect, it will display **"Incorrect OTP entered".** |

**Table 7: UC-7**

### 3.1.2.1 Name of Use-Case 8: Admin Dashboard

| **Identifier** | UC-8 | |
|---|---|---|
| **Purpose** | Admin will be able to view analytics related to the application. | |
| **Priority** | Medium | |
| **Pre-conditions** | 1. Admin should be logged into the system.<br>2. Internet connectivity required. | |
| **Post- conditions** | Display analytical graphs and charts according to the application usage. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin clicks on “Dashboard” on the sidebar. | The system will redirect to the dashboard page and display analytics regarding user traffic, type of diabetic patients etc. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | N/A | N/A |

**Table 8: UC-8**

### 3.1.2.2 Name of Use-Case 9: User Management

| **Identifier** | UC-9 | |
|---|---|---|
| **Purpose** | Admin will be able to view, update, and delete user’s accounts. | |
| **Priority** | High | |
| **Pre-conditions** | 1. Admin should be logged into the system.<br>2. Internet connectivity required. | |
| **Post- conditions** | 1. User deleted successfully.<br>2. User details updated successfully.<br>3. Updated list displayed successfully. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin clicks on “Users” button on the sidebar. | The system will redirect to a page displaying all the user’s profile details along with the option to update and delete users. |
| **2** | The admin clicks on the update icon under the “Action” column. | The system will display a form containing all the data of a user |

| | | |
|---|---|---|
| | | and allows the permission to change them except the email address. |
| **3** | The admin updates the user account details and clicks on the "Update" button. | The system will verify the changes and update the corresponding fields in the database. |
| **4** | The admin clicks on the delete icon under the "Action" column. | The system will display a dialogue box with the message "Are you sure you want to delete this user?" |
| **5** | The admin clicks on "Delete" button on the popup box. | The system will delete the user from the database and display an updated list of users. |
| **6** | The admin clicks on "Cancel" button on the popup box. | The system will redirect to the user's account page. |
| **7** | The admin clicks on the "Health Profile" button on the "Users" page. | The system will redirect to a page displaying all the user's health details along with the option to update them. |
| **8** | The admin clicks on the edit icon on the Health Profile page. | The system will display a form containing all the health details of a user and allows the permission to change them except the email address. |
| **9** | The admin updates the user health details and clicks on the "Update" button. | The system will verify the changes and update the corresponding fields in the database. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin updates the user account details and clicks on the "Update" button. | The system detects that the updated user account details are invalid and displays an error message. |
| **2** | The admin updates the user health details and clicks on the "Update" button. | The system detects that the updated user health details are invalid and displays an error message. |

**Table 9: UC-9**

### 3.1.2.3 Name of Use-Case 10: Food Management

| **Identifier** | UC-10 |
|---|---|
| **Purpose** | Admin will be able to view, add, update, or delete food items. |
| **Priority** | High |
| **Pre-conditions** | 1. Admin should be logged in the system.<br>2. Internet connectivity required. |
| **Post- conditions** | **Minimal guarantees:**<br>1. Invalid food input values.<br>**Success guarantee:**<br>1. Food Item displayed successfully.<br>2. Food Item updated successfully.<br>3. Food Item added successfully.<br>4. Food Item deleted successfully. |

| **Typical Course of Action** | | |
|---|---|---|
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin clicks on “Foods” button on the sidebar. | The system will redirect to the food page displaying all the foods along with the option to add, update, delete foods. |
| **2** | The admin clicks on “Add Food” button. | The system will display a form containing all the input fields required for a food. |
| **3** | The admin fills all the fields (protein, carbs, glycemic index etc.) on the “Add Food” form and clicks the Add button. | The system will verify the input field values and add the food to the database and display the updated food list. |
| **4** | The admin clicks on the update icon under the Action column. | The system will display a form containing all the fields of the food which can be edited. |
| **5** | The admin updates the input fields of the food and clicks the Update button. | The system will verify the field values and update the fields in the database and display the updated food list. |
| **6** | The admin clicks on the delete icon under the Action column. | The system will ask the user “Are you sure you want to delete this Food Item?”. |
| **7** | The admins click the “Delete” button on the popup screen. | The system will delete the Food Item from the database and display the updated food list. |
| **8** | The admins click the “Cancel” button on the popup screen. | The system will redirect to the food list page. |
| **Alternate Course of Action** | | |

| S# | Actor Action | System Response |
|---|---|---|
| **1** | The admin fills all the input fields on the "Add Food" form and clicks the Add button. | The system detects that the provided food's fields are invalid and display an error message<br>Example:<br>1. "Invalid Food macro-nutrients."<br>2. Invalid Food glycemic index. |
| **2** | The admin updates the fields of the food and clicks the Update button. | The system detects that the edited input fields are invalid and displays an error message. |

**Table 10: UC-10**

### 3.1.2.4 Name of Use-Case 11: Feedback Management

| **Identifier** | UC-11 |
|---|---|
| **Purpose** | Admin will be able to view and delete the feedbacks. |
| **Priority** | Low |
| **Pre-conditions** | 1. Admin should be logged into the system.<br>3. Internet connectivity required. |
| **Post- conditions** | 1. Feedbacks displayed successfully.<br>2. Feedbacks deleted successfully.<br>3. Access to the full view of the Feedback. |

| **Typical Course of Action** | | |
|---|---|---|
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin clicks on "Feedback" button on the sidebar. | The system will redirect to a page displaying a table of feedbacks. |
| **2** | The admin clicks on delete icon under the action column. | The system will display a popup box containing the message "Are you sure you want to delete this feedback?" |
| **3** | The admin clicks the "Delete" button on the popup box. | The system will delete the corresponding feedback from the database and display the updated table of feedbacks. |
| **4** | The admin clicks the "Cancel" button on the popup box. | The system will redirect to the page displaying a table of feedbacks. |

| Alternate Course of Action | | |
|---|---|---|
| **S#** | **Actor Action** | **System Response** |
| **1** | N/A | N/A |

**Table 11: UC-11**

### 3.1.2.5 Name of Use-Case 12: Blog Management

| **Identifier** | UC-12 |
|---|---|
| **Purpose** | Admin will be able to view, add, update, and delete blogs on the application. |
| **Priority** | Medium |
| **Pre-conditions** | 1. Admin should be logged into the system.<br>2. Internet connectivity required. |
| **Post- conditions** | • Blogs displayed successfully.<br>• Blog updated successfully.<br>• Blog added successfully.<br>• Blog deleted successfully. |

| Typical Course of Action | | |
|---|---|---|
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin clicks on the “Blogs” button on the sideba | The system will redirect to a page displaying a table of the all the blogs with the option to add, update and delete blogs. |

| | | |
|---|---|---|
| **2** | The admin clicks on "Add Blog" button. | The system will redirect to a form containing all the input fields required for the blog post. |
| **3** | The admin fills all the input fields on the "Add Blog" form and clicks the Upload button. | The system will add the blog on the database and also upload it on the user application. |
| **4** | The admin clicks on the update icon under the Action column. | The system will redirect to a form containing all the input fields of the Blog which can now be edited. |
| **5** | The admin updates the input fields of the blog and clicks the Update button. | The system will update the values on the database and display the updated blog. |
| **6** | The admin clicks on the delete icon under the Action column. | The system will display a dialogue box with the message "Are you sure you want to delete this Blog?". |
| **7** | The admin clicks the "Delete" button on the popup box. | The system will delete the Blog from the database and display an updated table of blogs. |
| **8** | The admin clicks the "Cancel" button on the popup box. | The system will not delete the blog and close the popup box. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **-** | - | - |

**Table 12: UC-12**

### 3.1.2.6 Name of Use-Case 13: Admin Profile Management

| | |
|---|---|
| **Identifier** | UC-13 |
| **Purpose** | Admin will be able to view and update their personal information |
| **Priority** | Low |
| **Pre-conditions** | 1. Admin should be logged into the system.<br>3. Internet connectivity required. |
| **Post- conditions** | Displays and updates admin's information. |
| **Typical Course of Action** | |
| **S#** | **Actor Action** / **System Response** |

| | | |
|---|---|---|
| **1** | The admin clicks the “Setting” button on the sidebar. | The system will redirect to a page displaying a form to update admin account details and a toggle button to switch between light and dark themes. Also, it shows an option for the admin to delete their account |
| **2** | The admin applies updates their picture, personal information and clicks the “Save” button. | The system will update the admin account details on the database and display updated values. |
| **3** | The admin flips the toggle switch. | The system will invert the theme of the application (dark to light or light to dark) |
| **4** | The admin clicks the “Delete Account” button. | The system will display a popup box containing the message “Are you sure you want to delete your account?”. |
| **5** | The admin clicks the “Delete” button on the popup box. | The system will delete the admin from the database and redirect to the login screen. |
| **6** | The admin clicks the “Cancel” button on the popup box. | The system will not delete the admin and close the popup box. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **-** | - | - |

**Table 13: UC-13**

#### 3.1.2.7 Name of Use-Case 14: Logout

| **Identifier** | UC-14 | |
|---|---|---|
| **Purpose** | Admin will be able to log out of the system. | |
| **Priority** | Low | |
| **Pre-conditions** | 1. Admin should be logged into the system.<br>2. Internet connectivity required. | |
| **Post- conditions** | Admin will be logged out of the system. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The admin clicks the "Logout" button on the sidebar. | The user will be redirected to a page asking for confirmation if they want to log out or not. |
| **2** | The admin clicks the "Logout" button. | The system will end the session and redirect to the login screen. |
| **3** | The admin clicks the "Cancel" button. | The system will redirect user to the dashboard page. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | N/A | N/A |

**Table 14: UC-14**

### 3.1.3 Patient Web Use Cases:

#### 3.1.3.1 Name of Use-Case 15: Signup

| **Identifier** | UC-15 | |
|---|---|---|
| **Purpose** | User will be able to create an account. | |
| **Priority** | High | |
| **Pre-conditions** | • User should not have an account in the system.<br>• Internet connectivity required. | |
| **Post- conditions** | **Minimal guarantees:**<br>• Registration failed due to invalid input.<br>• User already exists.<br>**Success guarantees:**<br>• User successfully registered. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |

| | | |
|---|---|---|
| **1** | The user opens the sign page of the application. | The system will display a form asking the user to enter email address and password and redirect to a page containing the form to enter personal and health details. |
| **2** | The user clicks on the "Already have an account?" button. | The system will redirect the user to login screen. |
| **2** | The user enters email and password on the signup form and click "Next" button. | The system will verify the input values and redirect to the registration form. |
| **3** | The user enters name, age, gender, and medical information (BMI, blood sugar levels etc.) on the registration form and clicks the "Continue" button. | The system will authenticate the user credentials and display the "Get Started" screens and redirect to the dashboard of the application. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **2** | The user enters email and password on the signup form and click "Next" button. | The system will detect that the input values are incorrect and displays an error message.<br>**Example:**<br>• "Your Password doesn't contain any special characters."<br>• "Email doesn't match the format." |
| **1** | The user enters name, age, gender, medical information, | The system detects that the user credentials are invalid and |
| | etc., and clicks the Signup button. | displays an error message.<br>**Example:**<br>• "Invalid Age value."<br>"Invalid BMI value." |
| **2** | User enters email, password, medical information etc. and clicks on the Signup button. | System detects that the user already exists in the system and displays the error message<br>**"User already exists."** |

**Table 15: UC-15**

### 3.1.3.2 Name of Use-Case 16: Login

| | |
|---|---|
| **Identifier** | UC-16 |
| **Purpose** | User will be able to access their account. |
| **Priority** | High |
| **Pre-conditions** | • User should be registered in the system.<br>• Internet connectivity required. |
| **Post- conditions** | **Minimal guarantees:**<br>• The user account does not exist in the system.<br>**Success guarantee:**<br>• The user successfully logged in. |

**Typical Course of Action**

| S# | Actor Action | System Response |
|---|---|---|
| **1** | The user opens the login page of the application. | The system will display a form asking the user to enter email and password. |
| **2** | The user enters the email and password on the login form and clicks the "Continue" button. | The system verifies user credentials and redirect to the dashboard of the application. |
| **3** | The user clicks on the "Don't have an account?" button. | The system will redirect the user to the signup screen. |
| **4** | The user clicks on "Forgot Password?" button. | The system will redirect to a page asking the user to enter email address. |
| **5** | The user enters the email address on the "Forgot Password?" screen and clicks the "Sent Reset Link" button. | The system will redirect the user to a "Verification" screen asking the user to enter 4-digit code sent on their corresponding email. |
| **6** | The user enters the OTP code on the "Verification" screen and clicks the "Verify" button. | The system verifies the given code and upon success, redirect to a screen asking the user to enter new password. |
| **7** | The user enters new password and clicks the "Confirm" button. | The system will check whether |

**Alternate Course of Action**

| S# | Actor Action | System Response |
|---|---|---|
| **1** | The user enters the email and password on the login form and clicks the Login button. | The system detects that the user doesn't exist in the database and displays the error message **"User doesn't exist."** |
| **2** | The user enters the OTP code on the "Verification" screen and clicks the "Verify" button. | The system detects that the given code is incorrect and displays an error message "Invalid Code". |

| | | |
|---|---|---|
| **3** | The user enters new password and clicks the “Confirm” button. | The system detects that the given password is in wrong format and displays an error message “Wrong Password Format”. |

**Table 16: UC-16**

### 3.1.3.3 Name of Use-Case 17: Dashboard

| **Identifier** | UC-17 | |
|---|---|---|
| **Purpose** | To display the health stats of the user. | |
| **Priority** | High | |
| **Pre-conditions** | • User should be logged in on the system<br>• Internet connectivity required. | |
| **Post- conditions** | System will display analytics on the user health status and activities. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user navigates to the "Dashboard" page on the sidebar. | The system will redirect to a page containing different details such as health and activity analytics. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **-** | - | - |

**Table 17: UC-17**

### 3.1.3.4 Name of Use-Case 18: Get Meal Plan

| **Identifier** | UC-18 | |
|---|---|---|
| **Purpose** | To generate personalized meal plans, delete meal plans, and share meal plans. | |
| **Priority** | High | |
| **Pre-conditions** | 1. User should be logged in on the system<br>2. User medical information should be known.<br>4. Internet connectivity required. | |
| **Post- conditions** | System will generate a meal plan according to user preferences | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user clicks the "Get Meal Plan" button on the Meal Plan icon. | The system will redirect to a page displaying buttons such as Generate Meal Plan, View Meal Plans, Edit Meal Plans etc. |
| **2** | User clicks the "Generate Meal Plan" button. | The system will redirect to a page display a form asking the user to enter their meal preferences and health details. |

| | | |
|---|---|---|
| **3** | User will enter their blood sugar levels, food preferences, food recommended by their doctor, and duration of meal plan on the meal form. | The system will use the user's food preferences and medical information to create a personalized meal plan. |
| **4** | User clicks on the "Edit Meal Plan" button and choose the particular item they want to substitute with a different one. | The system updates and displays a customized meal plan based on previous inputs and new modifications. |
| **5** | The user clicks on the "Delete Meal Plan" button. | The system deletes the previous meal plan and redirects to the Meal Plan tab. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | User will enter their blood sugar levels, food preferences, food recommended by their doctor, and duration of meal plan on the meal form. | The system detects that the provided inputs are incorrect and displays an error message. |

**Table 18: UC-18**

### 3.1.3.5 Name of Use-Case 19: Give Feedback

| | | |
|---|---|---|
| **Identifier** | UC-19 | |
| **Purpose** | Allows the user to upload their feedback. | |
| **Priority** | Low | |
| **Pre-conditions** | • User should be logged in on the system.<br>• Internet connectivity required. | |
| **Post- conditions** | Admin will be able to view the user feedback. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user clicks the "Give Feedback" button on the sidebar. | The system will redirect to a page display a form asking the user to enter subject, type, description of the feedback. |
| **2** | The user enters the feedback details and clicks the "Update" button. | The system will add the feedback on the database and display a success message. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |

| | | |
|---|---|---|
| - | - | - |

**Table 19: UC-19**

### 3.1.3.6 Name of Use-Case 20: Blogs

| **Identifier** | UC-20 | |
|---|---|---|
| **Purpose** | Allows the user to view blog posts. | |
| **Priority** | Low | |
| **Pre-conditions** | • User should be logged in on the system.<br>• Internet connectivity required. | |
| **Post- conditions** | User will be able to read the blogs and checkout their sources. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user clicks the "Blogs" button on the sidebar. | The system will redirect to a page displaying different blog posts uploaded by the Admin. |
| **2** | The user clicks on the "Copy" icon. | The system will add the source link of the original blog on the user's clipboard. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| - | - | - |

**Table 20: UC-20**

### 3.1.3.7 Name of Use-Case 21: Profile

| **Identifier** | UC-21 | |
|---|---|---|
| **Purpose** | Allows the user to update their personal information and health details. | |
| **Priority** | High | |
| **Pre-conditions** | • User should be logged in on the system.<br>• Internet connectivity required. | |
| **Post- conditions** | User's personal information and health details updated successfully. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |

| | | |
|---|---|---|
| **1** | The user clicks the “Profile” button on the sidebar. | The system will redirect to a page displaying user profile and health profile and allow the user to update them. |
| **2** | The user edits the personal information and click the “Update” button. | The system will verify and update input values and show a success message. |
| **3** | The user edits the health details and click the “Update” button. | The system will verify and update input values and show a success message. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user edits the personal information and click the “Update” button. | The system detects that the input values are incorrect and displays an error message. |
| **2** | The user edits the health details and click the “Update” button. | The system detects that the input values are incorrect and displays an error message. |

**Table 21: UC-21**

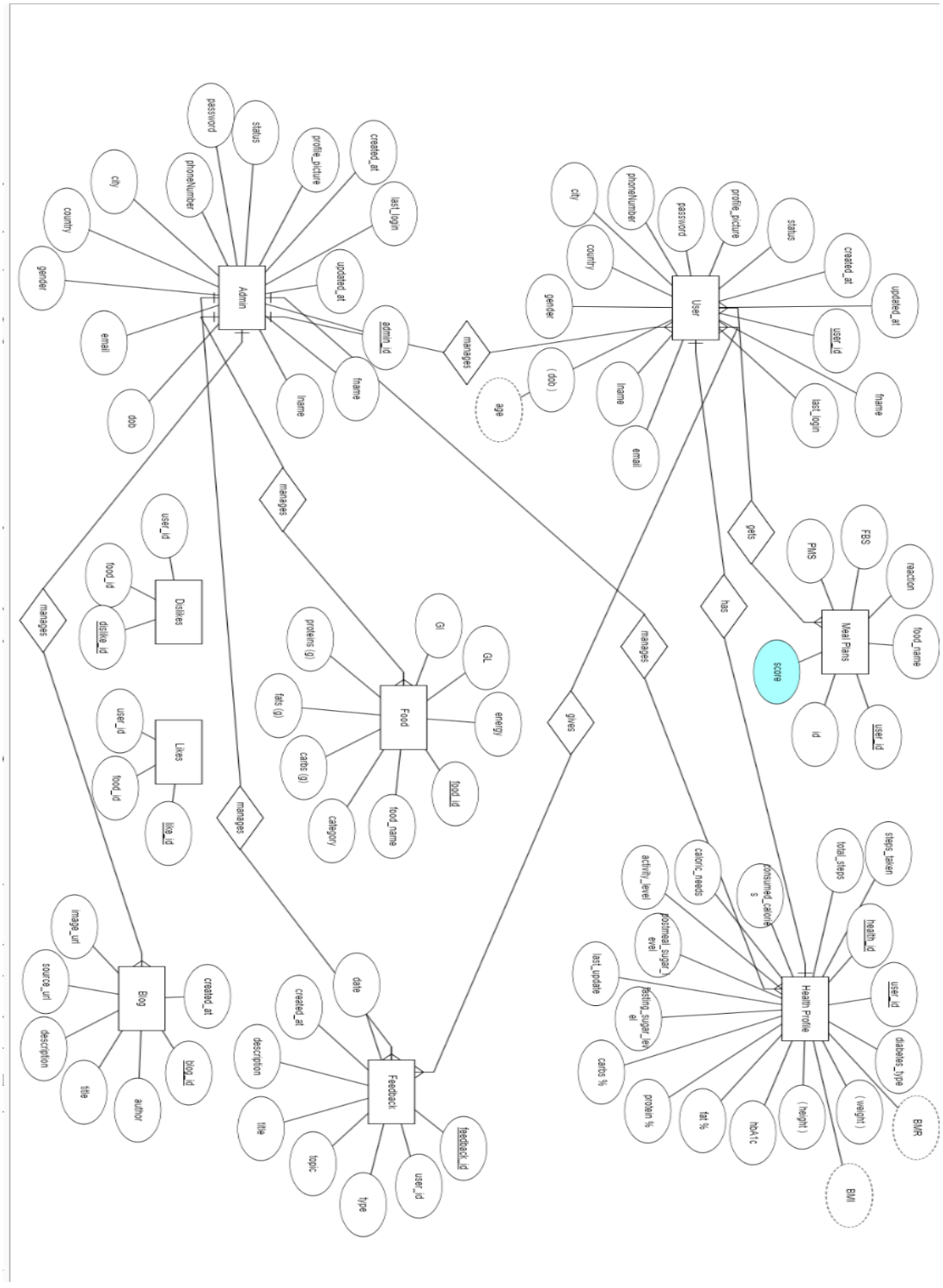
### 3.1.3.8 Name of Use-Case 22: Logout

| **Identifier** | UC-22 | |
|---|---|---|
| **Purpose** | Users can log out of the system | |
| **Priority** | Low | |
| **Pre-conditions** | • User should be logged in on the system<br>• Internet connectivity required. | |
| **Post- conditions** | User has successfully logged out. | |
| **Typical Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | The user clicks on "Logout" button on the sidebar. | The system will display a popup box with the message "Are you sure you want to logout?". |
| **2** | The user clicks the "Logout" button. | The system will end the user's session and redirect to the Login Screen. |
| **3** | The user clicks the "Cancel" button. | The system will close the popup box. |
| **Alternate Course of Action** | | |
| **S#** | **Actor Action** | **System Response** |
| **1** | N/A | N/A |

**Table 22: UC-22**

### 3.1.3 Requirements Analysis and Modeling

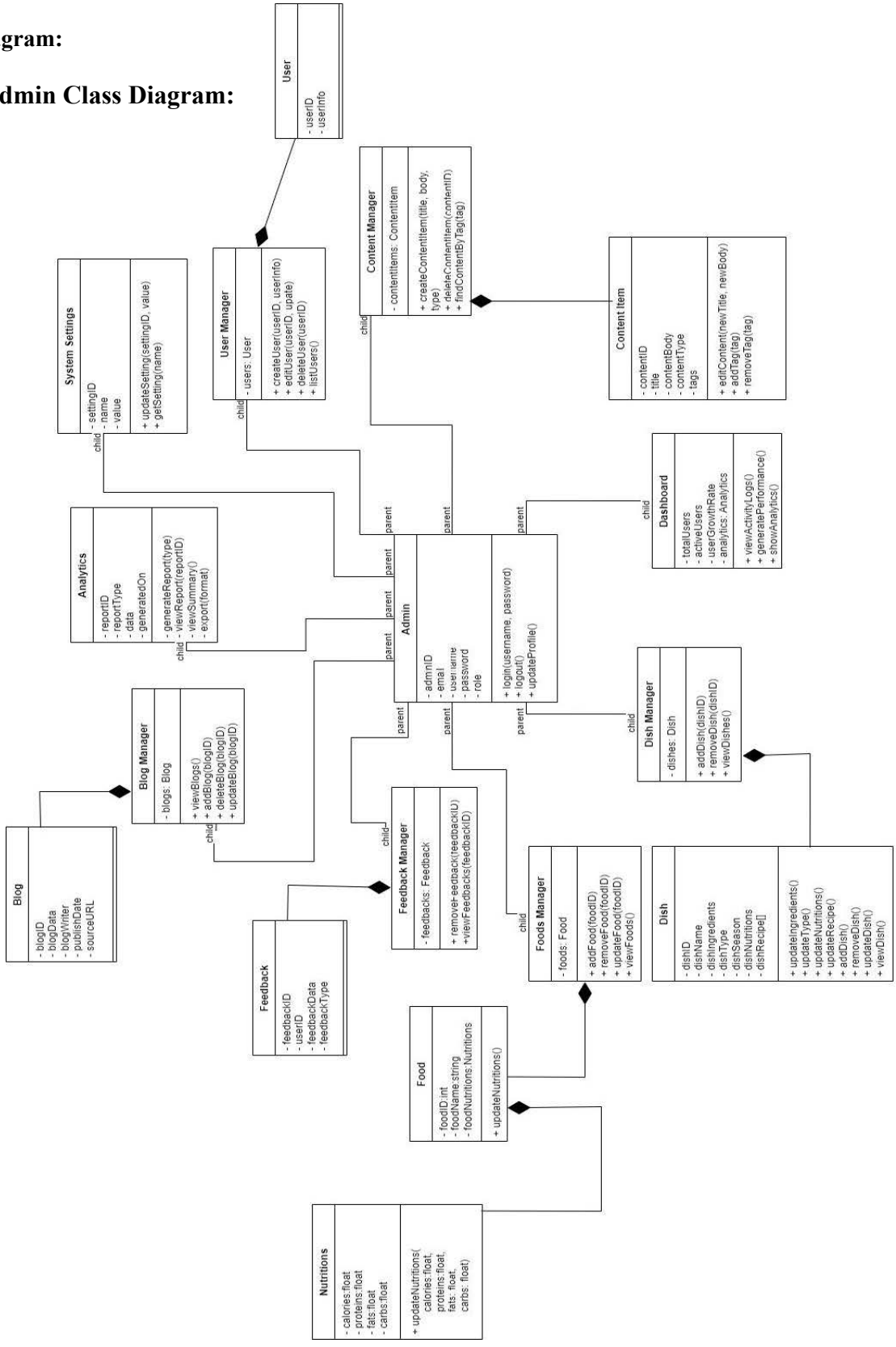
#### 3.1.3.1 ER Diagram:



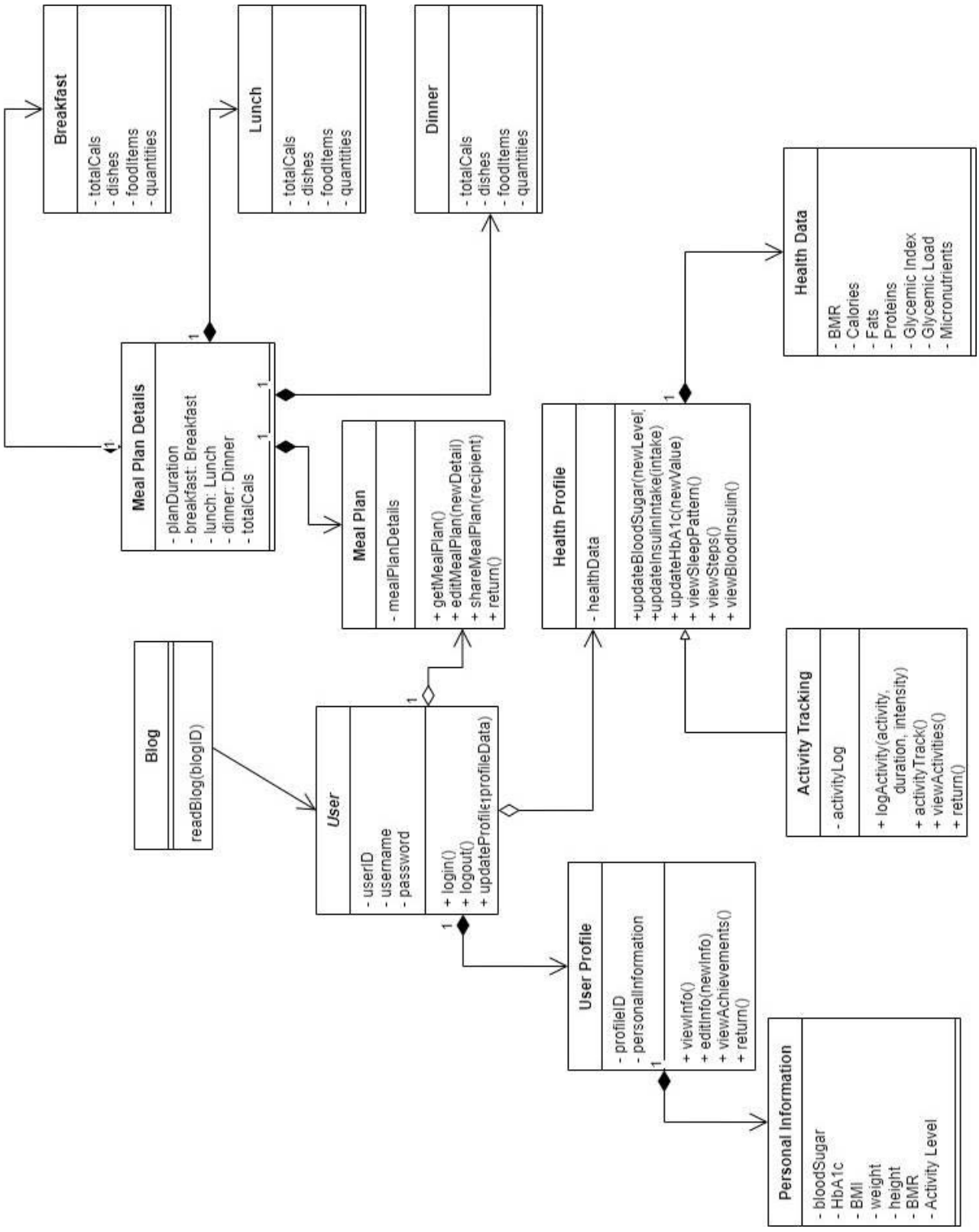

The Entity-Relationship (ER) Diagram depicts the data model of the SugarSage system, showing the entities (tables) like User Profiles, Admin, Blogs, Food, Dishes, and their relationships. It includes attributes of each entity and the types of relationships (one-to-many, many-to-many) between them.

### 3.1.3.2 Class Diagram:

- **Admin Class Diagram:**

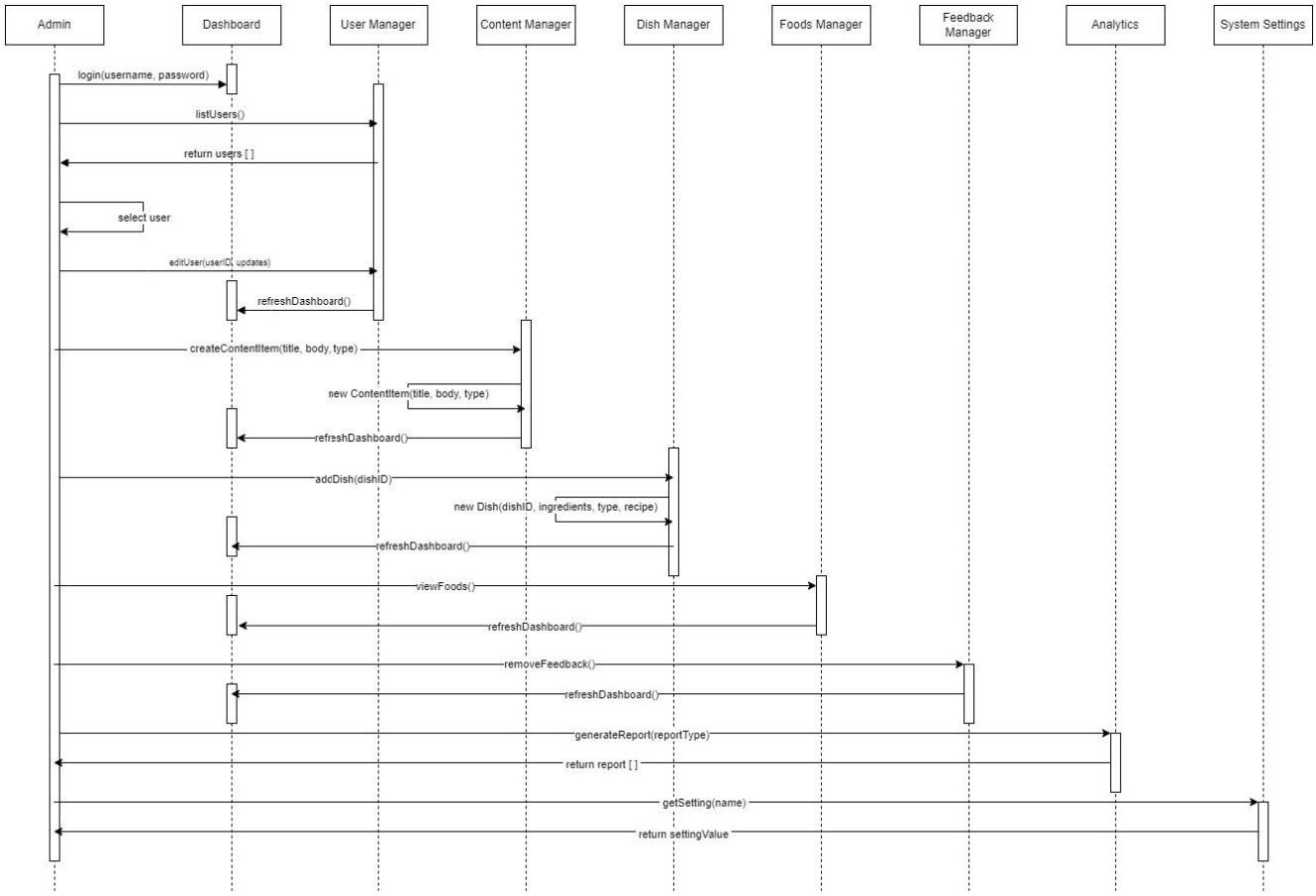


- **User Class Diagram:**



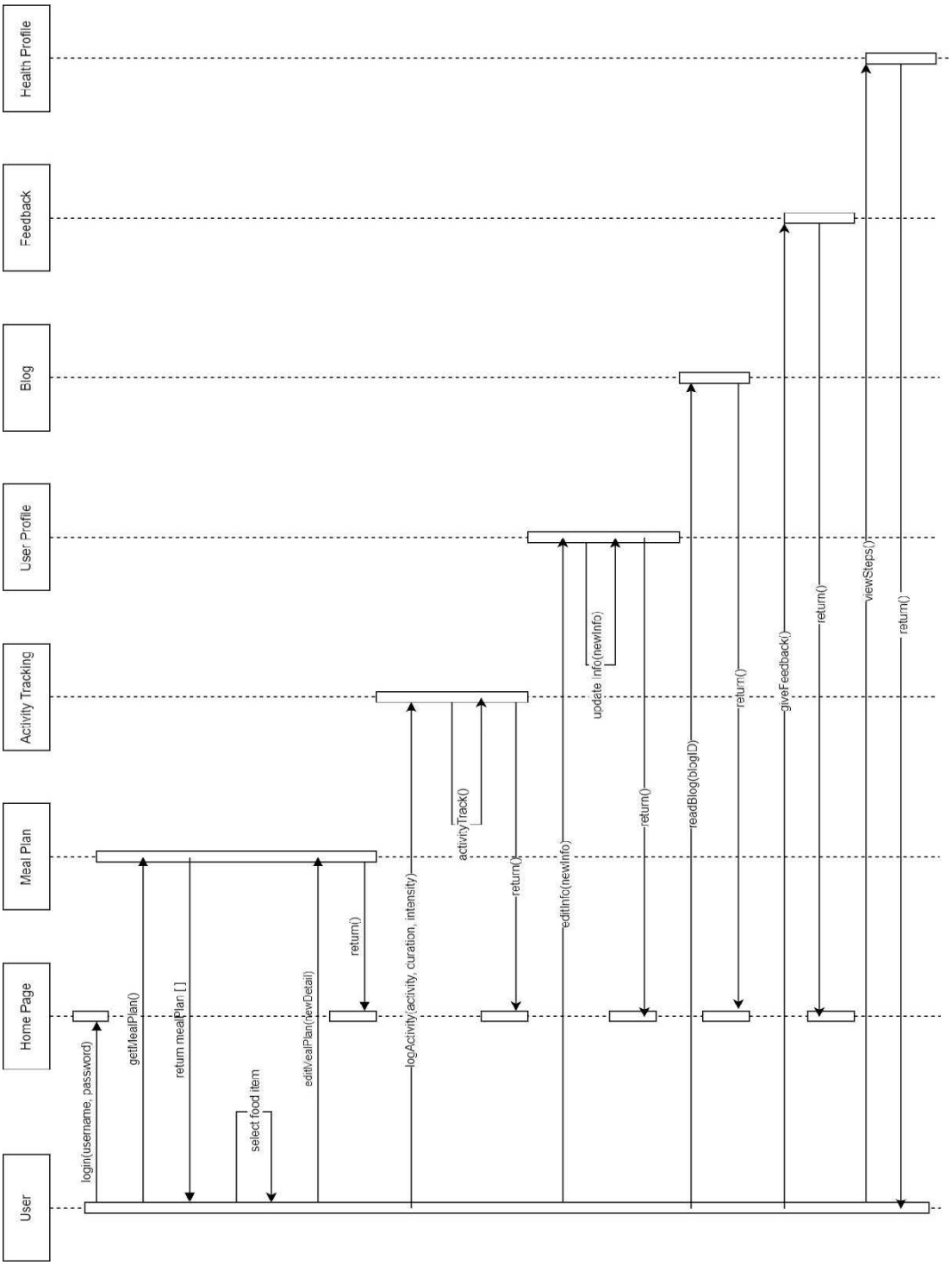
### 3.1.3.3 Sequence Diagram:

- **Admin Sequence Diagram:**



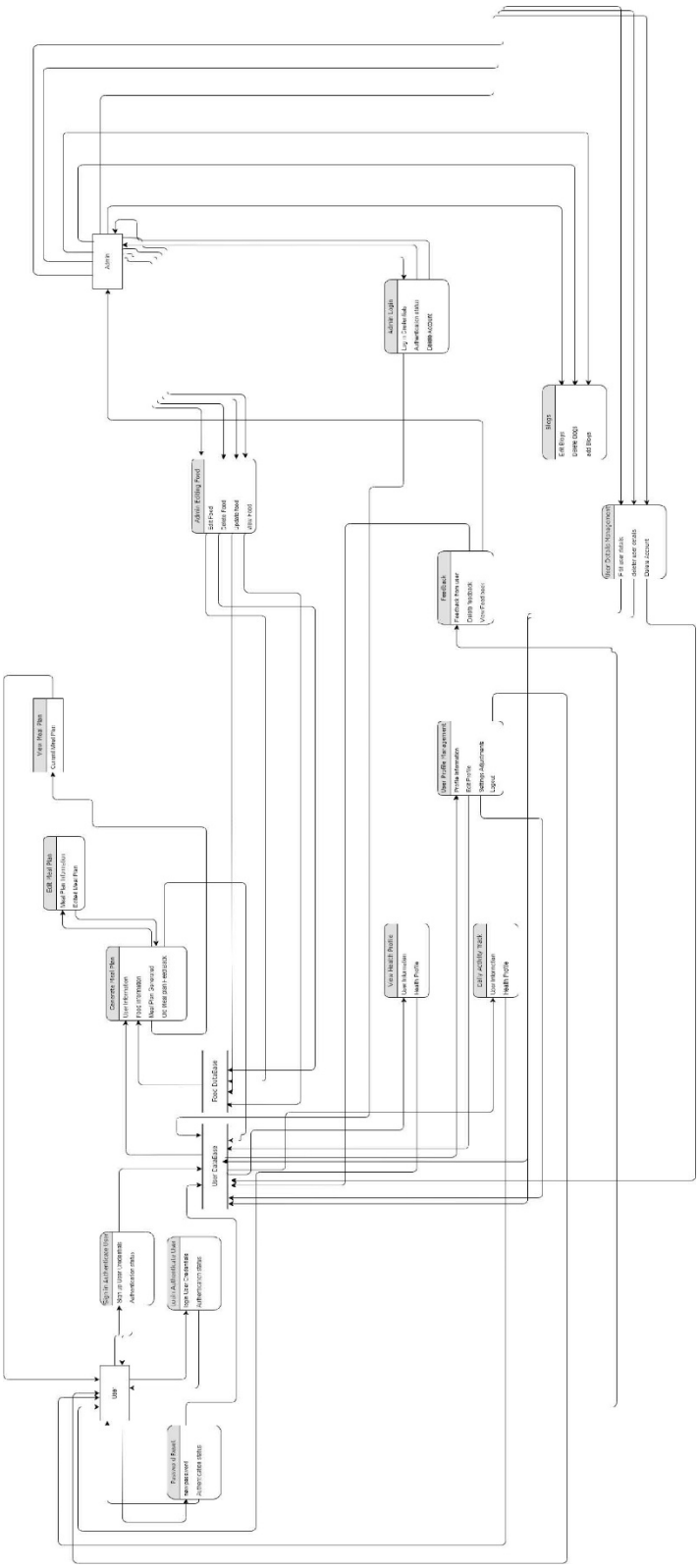

This sequence diagram provides a detailed interaction flow for administrative actions within the SugarSage system. It outlines the sequence of method calls between components such as the Admin, Dashboard, User Manager, Content Manager, Dish Manager, Foods Manager, Feedback Manager, Analytics, and System Settings when an admin performs tasks like user management or content creation.

- **User Sequence Diagram:**




This diagram displays the sequence of interactions from a user's perspective as they navigate the SugarSage system. It includes method calls among components like User, Home Page, Meal Plan, Activity Tracking, User Profile, Blog, Feedback, and Health Profile. It shows how users might log in, manage their meal plans, track activities, and read blogs.

### 3.1.3.4 Data Flow Diagram:



The Data Flow Diagram visually represents data flow within the SugarSage system. It maps out how data is input, processed, and output by different entities in the system, including user and admin interactions, information storage, and the processing of health data, meal plans, and feedbac

## 3.2 Nonfunctional Requirements

### 3.2.1 Performance Requirements

- **Response Time:** The mobile application should provide near-instantaneous response times when generating personalized diet plans and navigating through the interface. Response time should be within 1-2 seconds for typical user interactions.
- **Scalability:** The system should be able to handle a large number of users simultaneously without significant degradation in performance. It should be capable of handling at least 10,000 concurrent users.
- **Data Processing Time:** The AI algorithm should process user data and generate personalized diet plans within a reasonable time frame.
- **Offline Capability:** The application should have the ability to provide basic functionality even in offline mode, allowing users to view previously generated diet plans and access educational resources.

### 3.2.2 Safety Requirements

- **Data Privacy:** The application must adhere to strict data privacy and protection standards. It should encrypt sensitive user information and comply with relevant data protection regulations.
- **Emergency Information:** The application should provide access to emergency contact information and guidelines for users experiencing critical health situations related to their diabetes.
- **Incorrect use of application:** In case of incorrect use of SugarSage, the application will provide warnings and disclaimers to warn the user that the application is not a doctor and should not be used for medical advice or medical assistance. These warnings will be presented as Terms and Conditions to the user at the time of sign in.

### 3.2.3 Security Requirements

- **User Authentication:** The application must implement robust user authentication mechanisms to ensure that only authorized users can access their personalized diet plans and health information.
- **Regular Security Audits:** The system should undergo periodic security audits and vulnerability assessments to identify and address potential security threats and vulnerabilities.

### 3.2.4 Additional Software Quality Attributes

- **Usability:** The application should have an intuitive and user-friendly interface, catering to users with varying levels of technological proficiency. It should be rated highly in user experience assessments.

- **Maintainability:** The codebase should be well-documented and organized, allowing for easy maintenance and updates by developers in the future.

- **Reliability:** The system should be highly reliable, with minimal downtime or disruptions in service. It should aim for an uptime of 99.9%.

- **Adaptability:** The application should be capable of adapting to evolving dietary guidelines, medical recommendations, and technological advancements in the field of diabetes management.

- **Portability:** The application should be designed for cross-platform compatibility, ensuring that it can run seamlessly on various devices and operating systems, including iOS and Android. It should offer a consistent user experience across different platforms, taking into consideration diverse screen sizes and resolutions.

# Chapter 4. Technical Architecture

## 4.1 Application and Data Architecture

The SugarSage system is a personalized diabetes management and monitoring application that will be custom-built for users. It will consist of several application components, including the User Profile, Health Tracker, Meal Planner, Feedback, and Blog Reading modules. The application will primarily collect and manage data related to user health metrics, dietary intake, user feedback, and educational content. The critical data components will be User Data, Health Metrics, Meal Plans, Feedback, and Blogs.

The application will be developed using a layered architecture, utilizing a client/server model. The client will be a mobile application or web interface, while the server will provide API endpoints for data processing and storage. We will use programming languages suitable for mobile and web development, such as Flutter for mobile and JavaScript (Node.js or React) for the web client, with a server backend possibly in Python or Java.

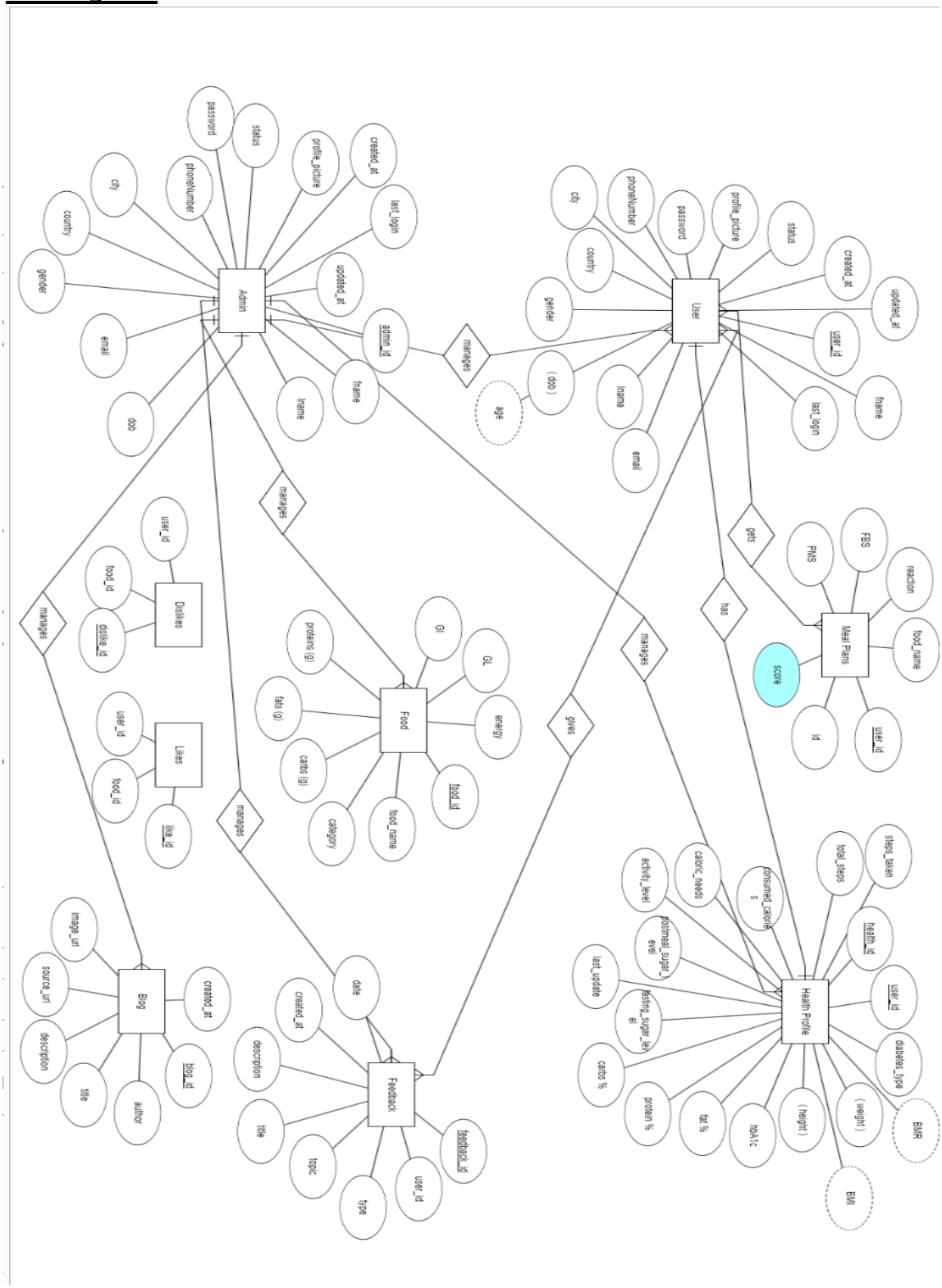
The hardware platform will not be device-specific, as the system is designed to be compatible with various smartphones and personal computers. Our backend infrastructure will be cloud-based, allowing for scalability and reliability. Depending on the data needs, we will use a relational database management system (RDBMS) like PostgreSQL or MySQL or a NoSQL database like MongoDB.

Our system will have a user-friendly interface accessible via mobile devices and web browsers. It is designed to be accessible online, ensuring users can manage their diabetes care anytime, anywhere.

The system will be hosted on a cloud platform (e.g., AWS, Azure) that provides the necessary computing resources, storage, and networking capabilities to support the application and data architecture.

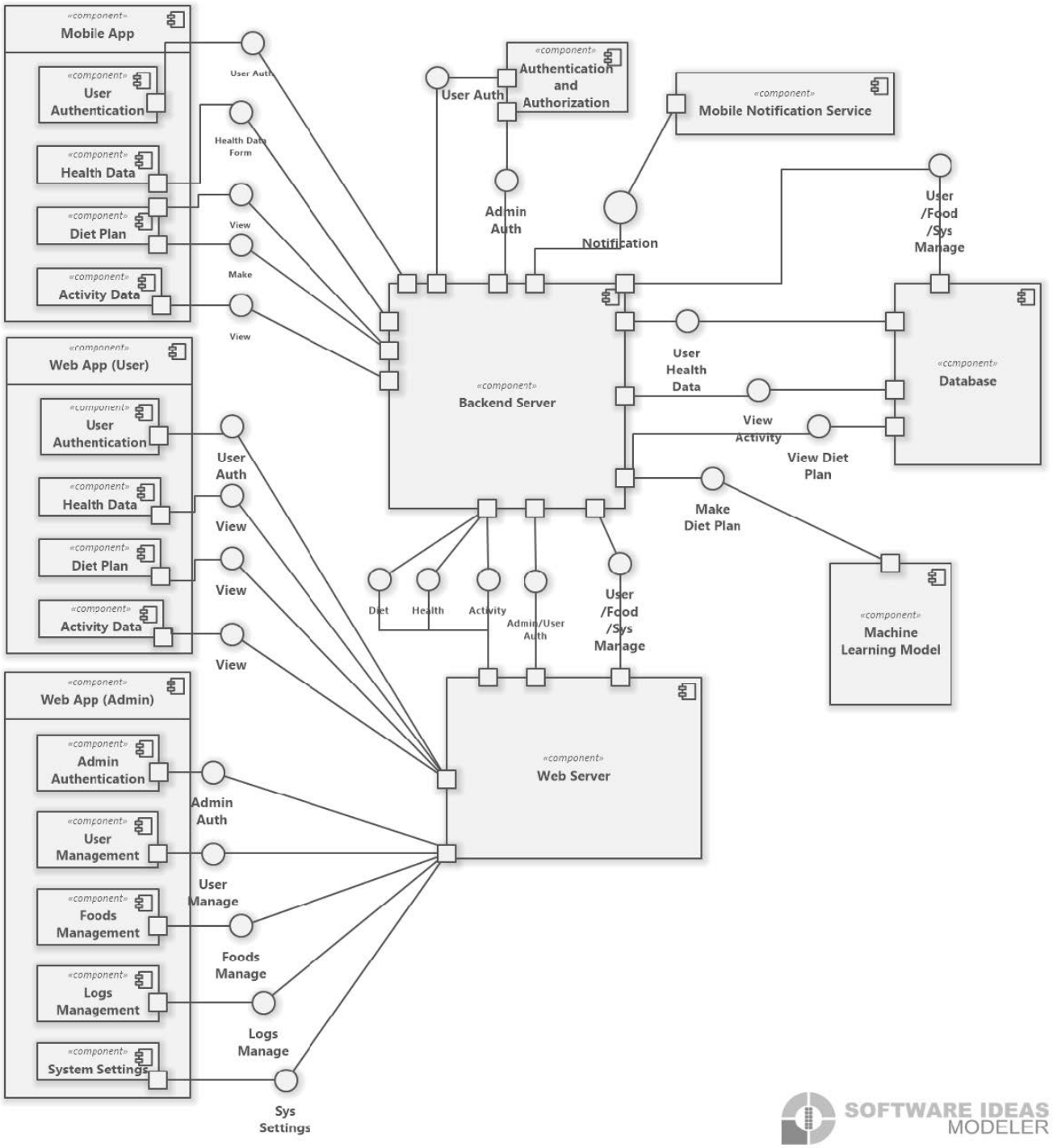
**Diagrams and Descriptions:**

## 1. ER Diagram:




The Entity-Relationship (ER) Diagram depicts the data model of the SugarSage system, showing the entities (tables) like User Profiles, Admin, Blogs, Food, Dishes, and their relationships. It includes attributes of each entity and the types of relationships (one-to-many, many-to-many) between them.

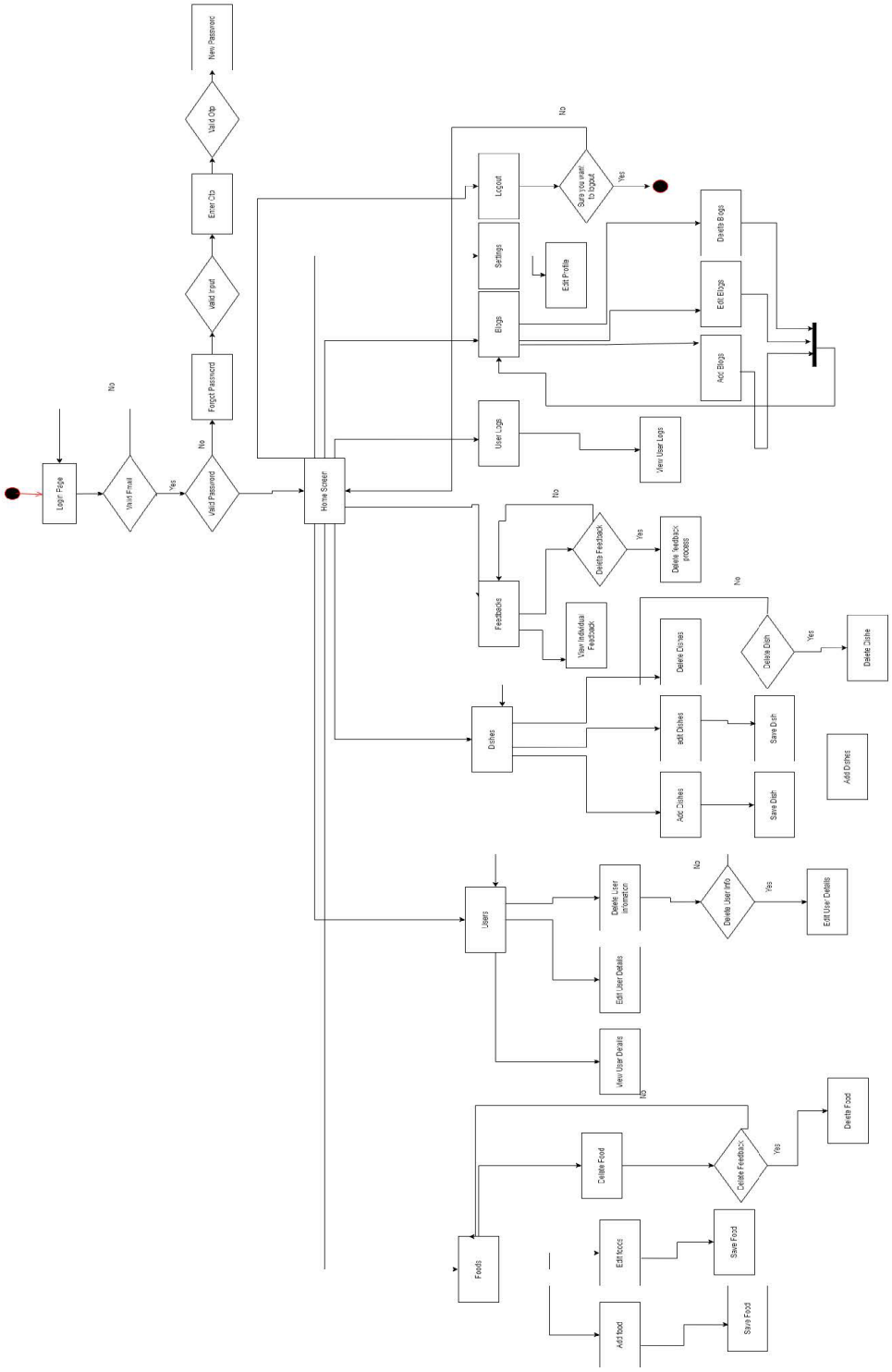
## 2. Component Diagram:




The Component Diagram illustrates the high-level architecture of the SugarSage system, showing the main components like the mobile app (for both users and admins), backend server, database, and machine learning model. It highlights the interaction between these components and subsystems, such as authentication, health data, diet plan, activity data, and notification service

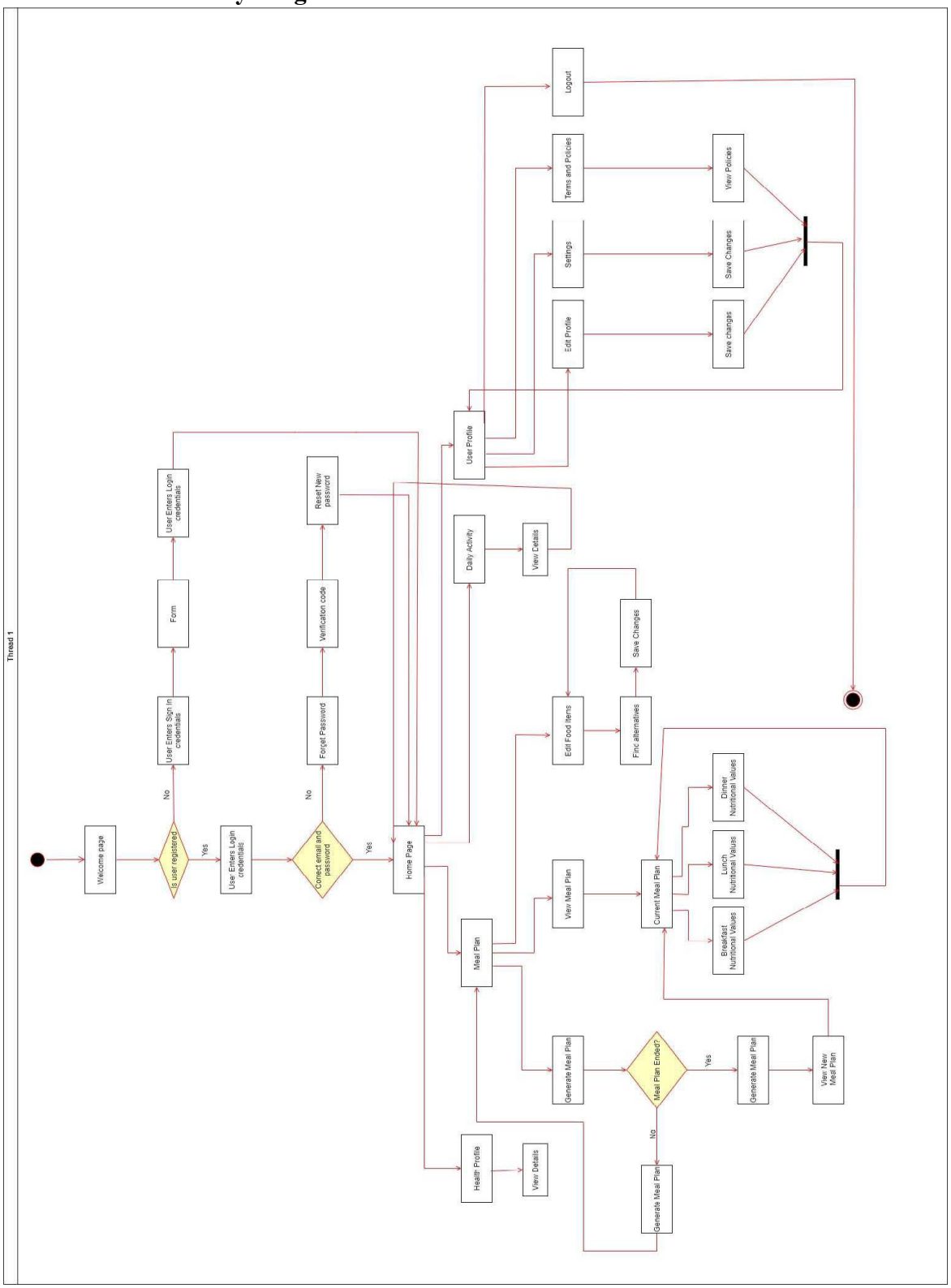
## 3. Activity Diagram:

- **Admin Activity Diagram:**




The Admin Activity Diagram is a flowchart that details the various actions an admin can perform within the SugarSage system. It covers processes such as logging in, managing user information, editing foods, handling feedback, and managing blog content, providing a visual guide to the admin's workflow and decision points.

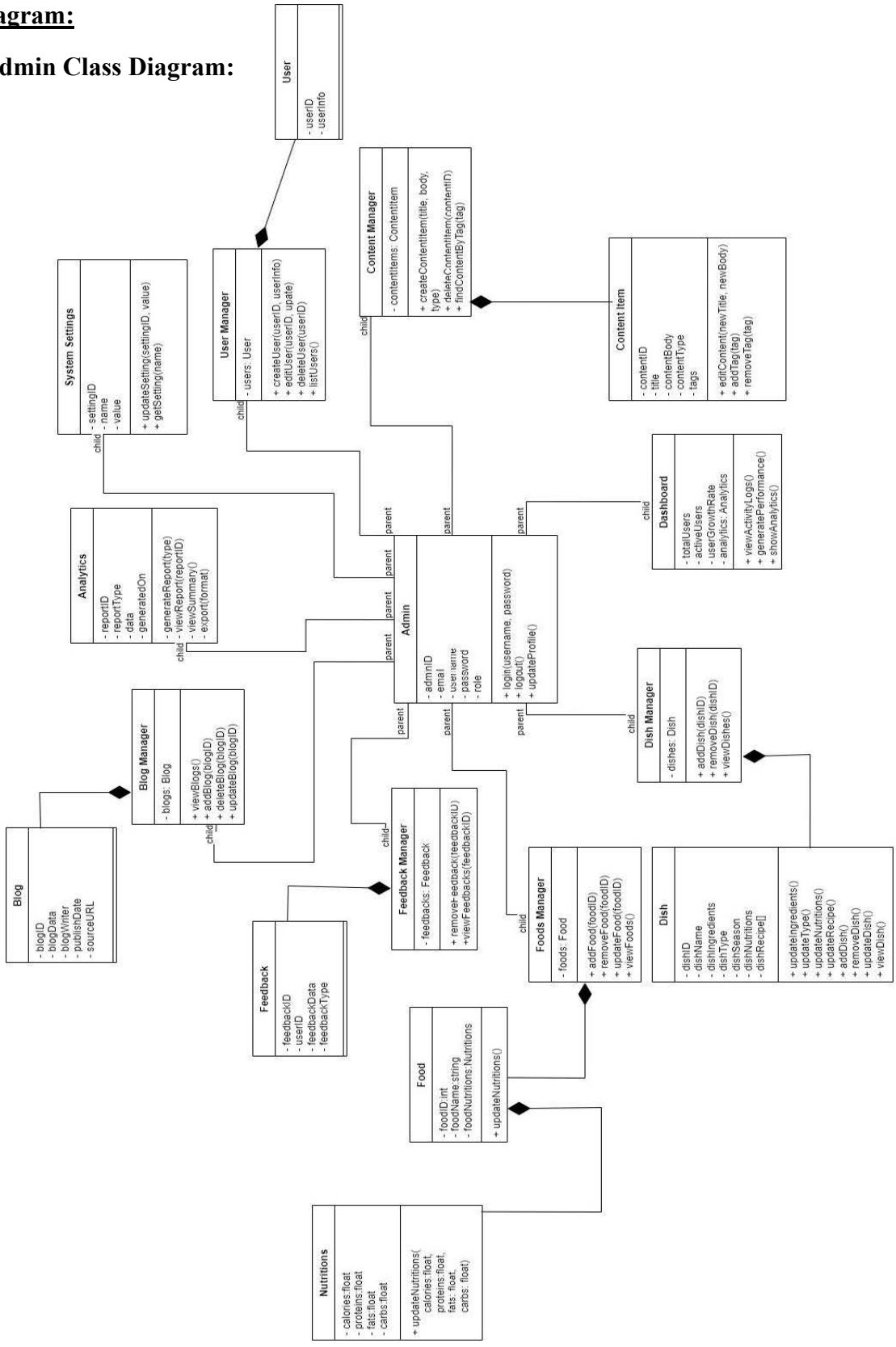
- **User Activity Diagram:**



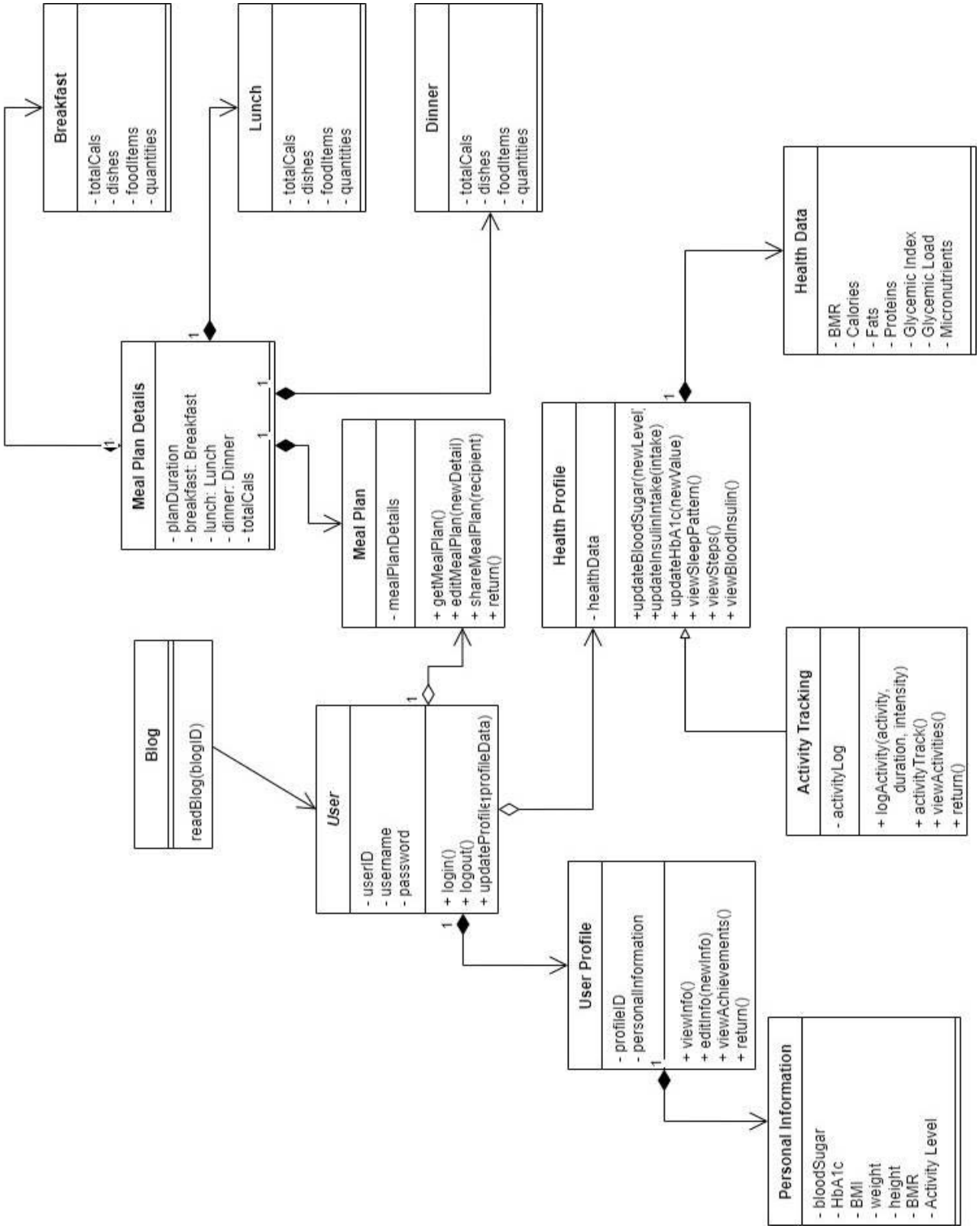

This diagram shows how users navigate the SugarSage system, from the welcome page to managing their health profile, meal plans, daily activities, and profile settings.

## 4. Class Diagram:

- **Admin Class Diagram:**



- **User Class Diagram:**



## 4.2 Component Interactions and Collaborations

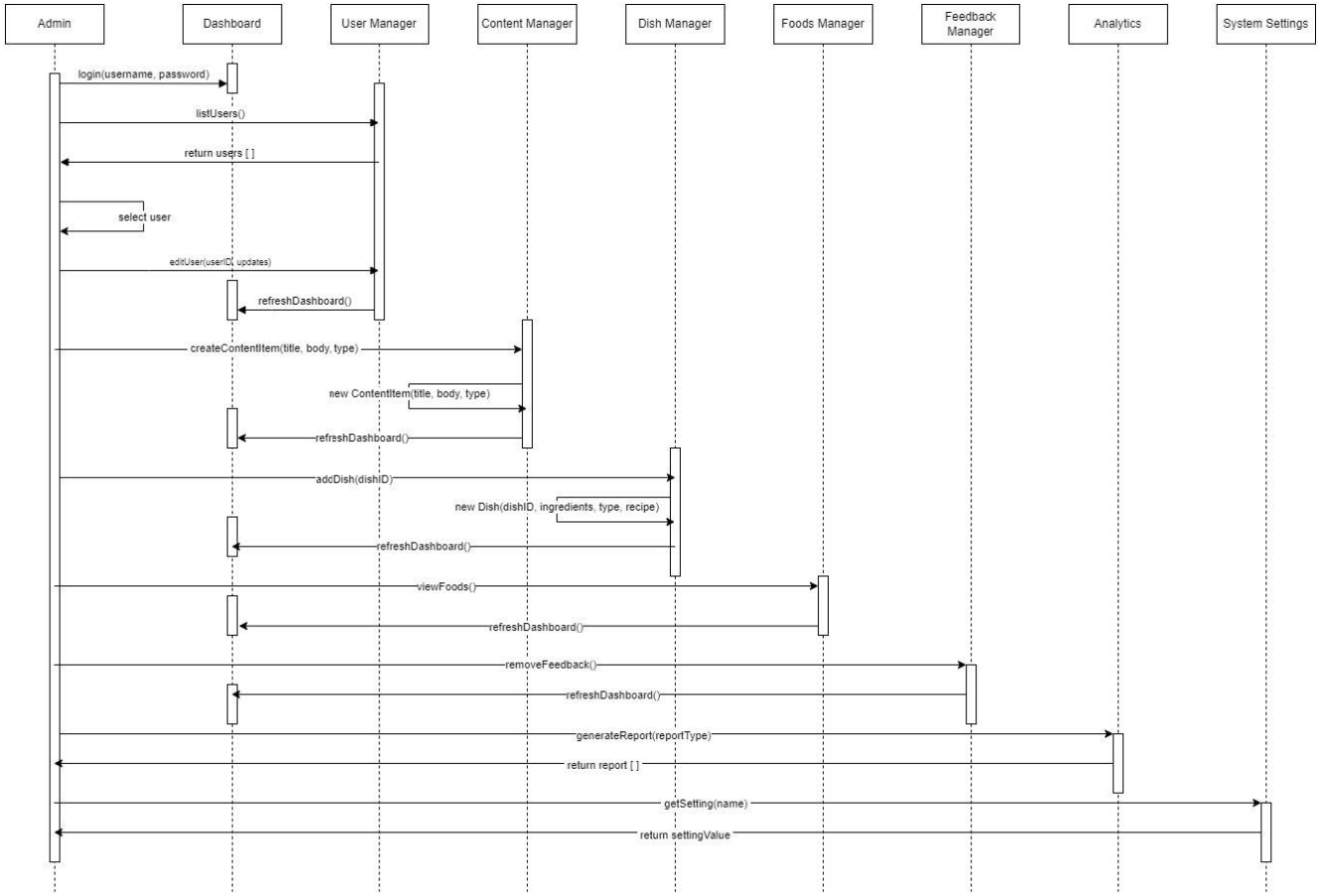
The components interact through well-defined API calls and responses. The User Profile component communicates with the Health Tracker to update and retrieve health metrics. The Meal Planner interacts with the User Profile to tailor diet plans based on user health data. The Feedback module lets users communicate their experience directly to the system administrators. Blog Reading enables users to access educational content provided by the system.

Design Level Sequence Diagrams, Collaboration Diagrams, and Event Trace diagrams will illustrate these components' detailed interactions and collaborations. These diagrams will show more details than in Phase 1, with explicit message calls, return values, and sequence flows.

**Diagrams and Descriptions:**

## 1. Sequence Diagram:

- **Admin Sequence Diagram:**

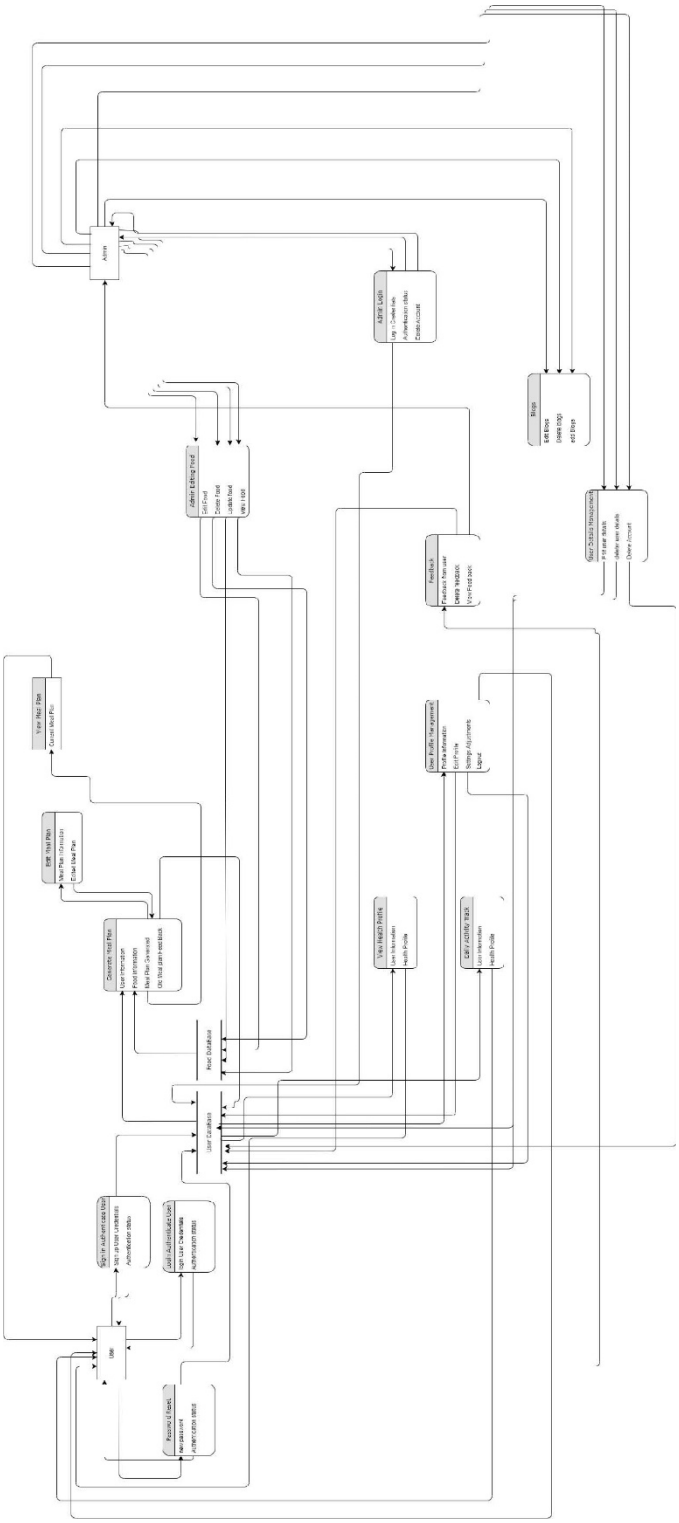



This sequence diagram provides a detailed interaction flow for administrative actions within the SugarSage system. It outlines the sequence of method calls between components such as the Admin, Dashboard, User Manager, Content Manager, Dish Manager, Foods Manager, Feedback Manager, Analytics, and System Settings when an admin performs tasks like user management or content creation.

- **User Sequence Diagram:**


This diagram displays the sequence of interactions from a user's perspective as they navigate the SugarSage system. It includes method calls among components like User, Home Page, Meal Plan, Activity Tracking, User Profile, Blog, Feedback, and Health Profile. It shows how users might log in, manage their meal plans, track activities, and read blogs.

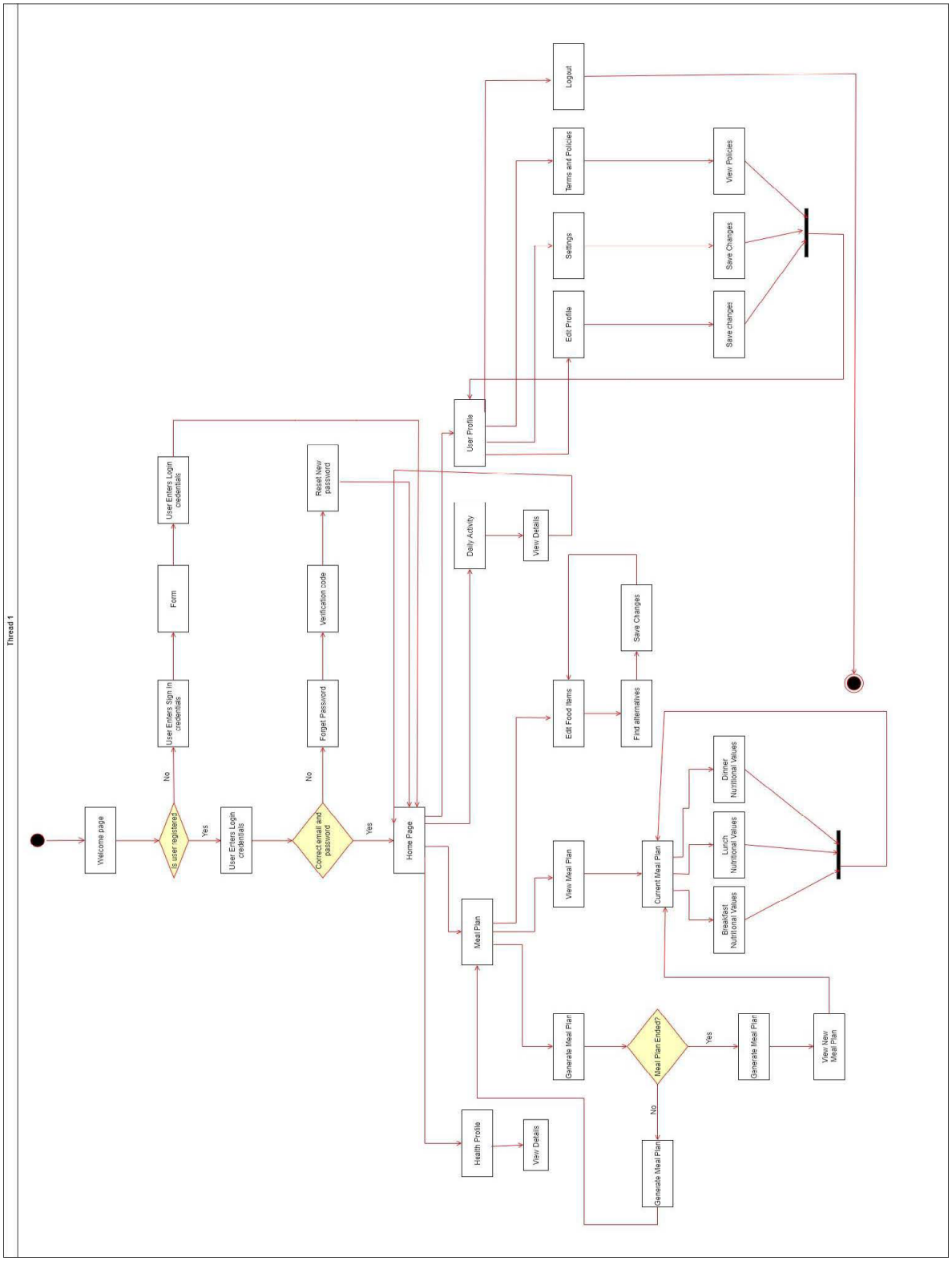
## 2. Data Flow Diagram:



The Data Flow Diagram visually represents data flow within the SugarSage system. It maps out how data is input, processed, and output by different entities in the system, including user and admin interactions, information storage, and the processing of health data, meal plans, and feedback.

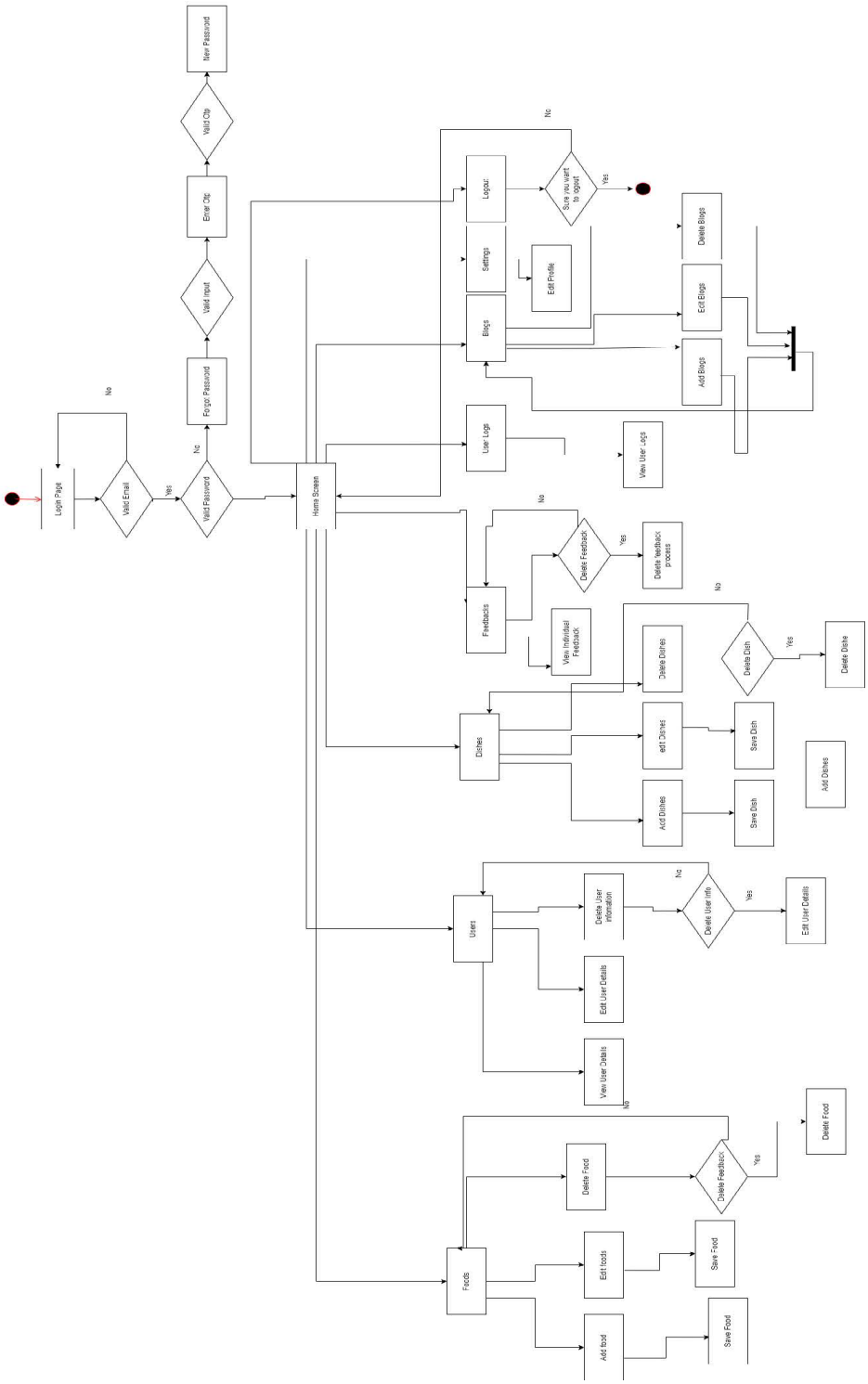
## 3. Activity Diagram:

- **User**




This diagram shows how users navigate the SugarSage system, from the welcome page to managing their health profile, meal plans, daily activities, and profile settings.

- **Admin:**




The Admin Activity Diagram is a flowchart that details the various actions an admin can perform within the SugarSage system. It covers processes such as logging in, managing user information, editing foods, handling feedback, and managing blog content, providing a visual guide to the admin's workflow and decision points.

## 4.3 Design Reuse and Design Patterns

In developing our SugarSage system, we will incorporate strategies for design reuse and apply design patterns to ensure efficiency, maintainability, and scalability.

**Design Reuse:**

- **Common Libraries:**
  We will abstract functions and classes that are common across different parts of the system into common libraries. This will include components for user authentication, data validation routines, and error handling. Reusing these libraries will decrease development time and increase code consistency.

- **UI Component Library:**
  A set of reusable user interface components will be created to ensure a consistent user experience across the mobile and web applications. These components will range from form elements to complex data visualization tools.

- **Service Layer:**
  Our system will feature a service layer that abstracts business logic from the controllers, promoting reuse for different parts of the application that require similar business rules processing.

**Design Patterns:**

- **Model-View-Controller (MVC):**
  We will employ the MVC pattern to separate concerns, making our system easier to maintain and extend. The Model will manage the data and business logic; the View will handle the presentation of information to the user; the Controller will process user input and update the Model and View accordingly.

- **Observer Pattern:**
  We will utilize the Observer pattern to enable a publish-subscribe model, allowing components to listen and react to events or changes in other components. This pattern will be particularly useful in the SugarSage system for updating user interfaces in response to changes in health data or for dispatching notifications.

- **Singleton Pattern:**
  We will ensure that classes such as the database connection manager or the configuration manager are instantiated only once throughout the application to manage shared resources effectively

## 4.4 Technology Architecture

Our anticipated infrastructure for the SugarSage system is designed to be robust and adaptable to the dynamic needs of users and system administrators. The core of our infrastructure will rely on a cloud-based environment, which provides several key advantages:

**Cloud-Based Environment:**

- **Auto-Scaling:**
  Our system will leverage the auto-scaling capabilities of the cloud to adjust resources dynamically based on the current load. This ensures the system remains responsive during peak usage times without incurring unnecessary costs during off-peak times.

- **High Availability:**
  The cloud infrastructure is designed for high availability, with redundancy and failover mechanisms to minimize downtime and ensure users have continuous access to the system.

**Connectivity Requirements:**

- **Internet Access:**
  Reliable internet access is a foundational requirement for SugarSage, enabling users to interact with the system from anywhere. Our system will utilize HTTPS protocols to secure all client and server communications.

**Modes of Operation:**

- **Synchronous Online Transactions:**
  Our system will support synchronous online transactions, essential for immediate and interactive user experiences. This includes actions like logging meals, tracking blood sugar levels, and receiving real-time feedback.

- **Asynchronous Batch Processing:**
  In addition to real-time interactions, our system will also handle asynchronous batch processing tasks. These tasks are critical for data analysis, health report generation, and other processes that do not require immediate user interaction.

**Technology Stack:**

- **Mobile Application Development:**
  **Flutter:** Chosen for its ability to compile into native code, which enhances performance and ensures a smooth user experience on both iOS and Android platforms. Flutter's extensive widget library facilitates the creation of visually appealing and responsive UIs, critical for engaging diabetic patients who require frequent interaction with the app.

- **Web Application Development:**
  **ReactJs:** Selected for its efficiency in updating and rendering components, which is essential for web applications requiring real-time data updates. React's component-based architecture makes it scalable, aiding in developing complex features such as diet tracking and progress monitoring. Its strong community support ensures access to a wide range of libraries and tools, enhancing development speed and capabilities.

- **Backend Development:**
  **Node.js:** Utilized for its non-blocking, event-driven architecture, which is excellent for asynchronous requests and handles multiple connections simultaneously. This is crucial for handling the intensive load of user interactions and data processing requests from both mobile and web platforms.

  **Python:** Integrated specifically for its robustness in scientific computing and machine learning. Python is used to handle the computational aspects of our machine learning models, which suggest personalized diet plans based on user health data. Python's extensive ecosystem of libraries, such as TensorFlow and Scikit-learn, facilitates efficient model development and integration.

- **Database Management:**
  SQL Database: Employed to manage structured data storage, offering robust transaction support and reliability. SQL databases ensure data integrity and security, which are paramount when dealing with sensitive medical information of users. They also provide powerful query capabilities that enable quick retrieval of user records and insights into app usage patterns.

**System Hosting and Platform:**

- **Cloud Platforms:**
  Platforms such as AWS, Azure or Google Cloud will be used to host our system, offering a range of services from compute instances to managed databases, which are essential for building a scalable and secure system.

## 4.5 Architecture Evaluation

### 4.5.1 Cloud-Based Infrastructure (AWS, Azure, Google Cloud):

1. **Pros:**

    - **Scalability:** Cloud platforms provide the ability to scale resources dynamically. This flexibility is crucial for handling fluctuations in system demand without needing physical infrastructure expansion.

    - **Cost-Effectiveness:** Using cloud services allows for a pay-as-you-go pricing model, which means the system incurs costs only based on actual usage, reducing unnecessary expenditure.

    - **High Availability and Reliability:** Cloud platforms offer built-in redundancy and disaster recovery capabilities, ensuring the system remains operational and accessible without significant downtime.

2. **Cons:**

    - **Vendor Lock-in:** Depending on proprietary services offered by specific cloud providers can make it challenging to migrate to other platforms in the future.

    - **Complexity in Management:** Managing a cloud-based infrastructure requires specialized skills and understanding of cloud services, which can introduce deployment and ongoing management complexity.

    - **Reason for Selection:** The decision to use cloud platforms like AWS, Azure, or Google Cloud was based on their robustness in handling large-scale deployments and their extensive suite of services that cater to all aspects of the system's needs, from computing power to database management.

### 4.5.2 Programming Language and Framework:

1. **Flutter for Mobile Development:**

    - **Pros:** Provides a single codebase for both iOS and Android platforms, which simplifies developmentand maintenance. It also supports native performance, which is critical for ensuring a smooth user experience.
    - **Cons:** Flutter is relatively new, and while it has a growing community, it lacks some of the extensiveresources and third-party libraries available to more established platforms like React Native.
    - **Reason for selection:** Chosen for its rapid development capabilities and excellent performance across both major mobile platforms, which is essential for a healthcare application requiring highresponsiveness and reliability.

2. **ReactJs for Web Development:**

    - **Pros:** Known for its virtual DOM, which makes it highly efficient for dynamic content updates crucial for real-time web applications.
    - **Cons:** React's JSX syntax and architectural depth can present a steep learning curve for new developers.
    -

- **Reason for Selection:** React's component-based architecture allows for efficient development of complex user interfaces, and its widespread adoption ensures strong community support and continual updates.

### 4.5.3 Node.js and Python in Backend Development:

- **Pros:** Node.js offers non-blocking I/O operations that efficiently handle concurrent requests, making it ideal for high-performance backends. Python is renowned for its simplicity and robust library support, particularly for scientific computing and machine learning tasks.
- **Cons:** Python can be slower than some alternatives like Java or C++ for certain backend operations. Node.js, being single-threaded, might require additional considerations for CPU-bound tasks.
- **Reason for Selection:** Node.js was selected for its efficiency with I/O-bound tasks and Python for its superiority in data analytics and machine learning, both critical for a data-driven application like SugarSage.

### 4.5.4 SQL Database:

- **Pros:** Offers strong consistency, transactions, and robustness for complex queries, which are indispensable for medical data handling.
- **Cons:** SQL databases can be less flexible regarding schema modifications compared to NoSQL options.
- **Reason for Selection:** The relational nature of medical data, such as user records and health metrics,

# Chapter 5. Detailed Design and Implementation

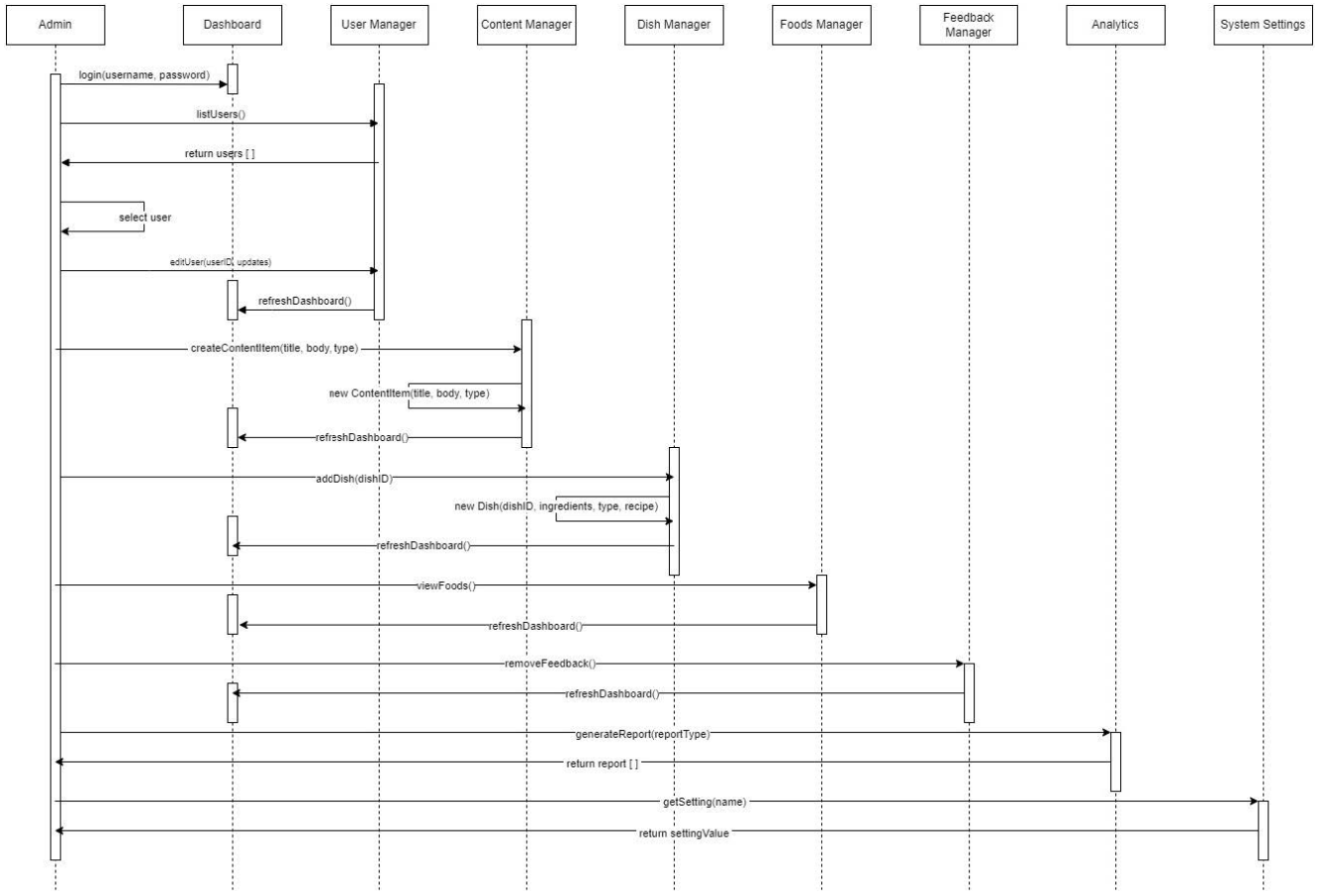
## 5.1 Component-Component Interface

- **User Profile and Health Tracker:**
  The User Profile component sends user demographic and health status data to the Health Tracker component, which uses this information to monitor and analyze health metrics over time.

- **Meal Planner and User Profile:**
  The Meal Planner component retrieves user-specific health data from the User Profile component to create personalized meal plans based on dietary needs and health objectives.

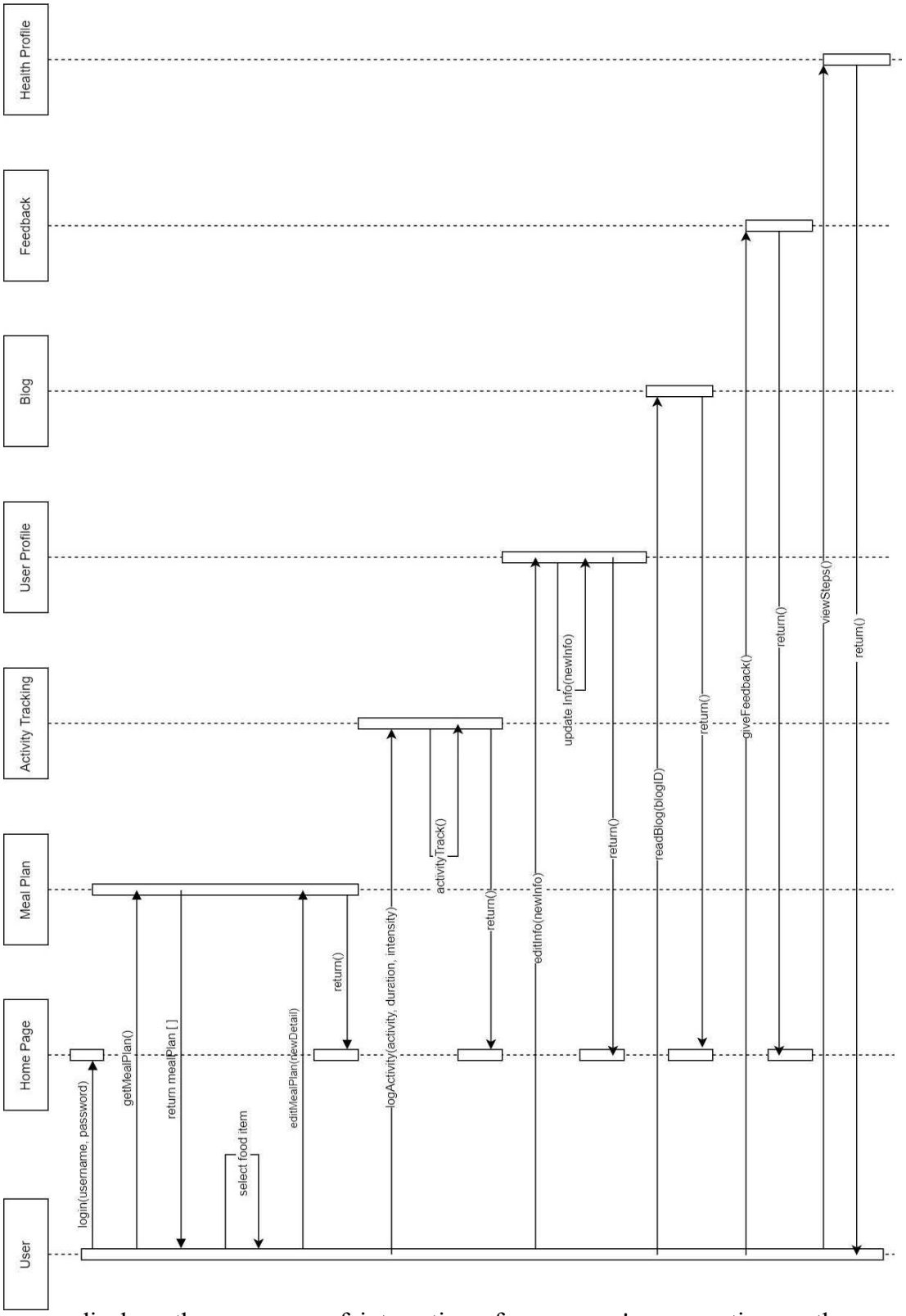
**Sequence Diagram:**

- **Admin Sequence Diagram:**




This sequence diagram provides a detailed interaction flow for administrative actions within the SugarSage system. It outlines the sequence of method calls between components such as the Admin, Dashboard, User Manager, Content Manager, Dish Manager, Foods Manager, Feedback Manager, Analytics, and System Settings when an admin performs tasks like user management or content creation.

- **User Sequence Diagram:**




This diagram displays the sequence of interactions from a user's perspective as they navigate the SugarSage system. It includes method calls among components like User, Home Page, Meal Plan, Activity Tracking, User Profile, Blog, Feedback, and Health Profile. It shows how users might log in, manage their meal plans, track activities, and read blogs.

## 5.2 Component-External Entities Interface

There are no External Entities used in the project. Everything we used is within libraries.

## 5.3 Component-Human Interface

1. **Screens and Interaction Points:**

    - **Login/Signup Screens:** Users receive forms to input their credentials or register a new account. Error messages are displayed if inputs are invalid or if credentials do not match existing records.

    - **Dashboard:** After login, users view their health dashboard which displays recent health metrics, suggested meal plans, and recent activities.

    - **Meal Plan Creation:** Users input their dietary preferences, health information, and other relevant data to generate customized meal plans.

    - **Health Tracking:** Users enter or sync health data such as daily physical activity, blood glucose levels, and meal intake. The system provides visual feedback on their health trends over time.

    - **Feedback Submission:** Users can submit feedback through a dedicated interface, which directly communicates with the backend to store and process these inputs for system evaluation and response.

2. **HCI Principles Applied:**

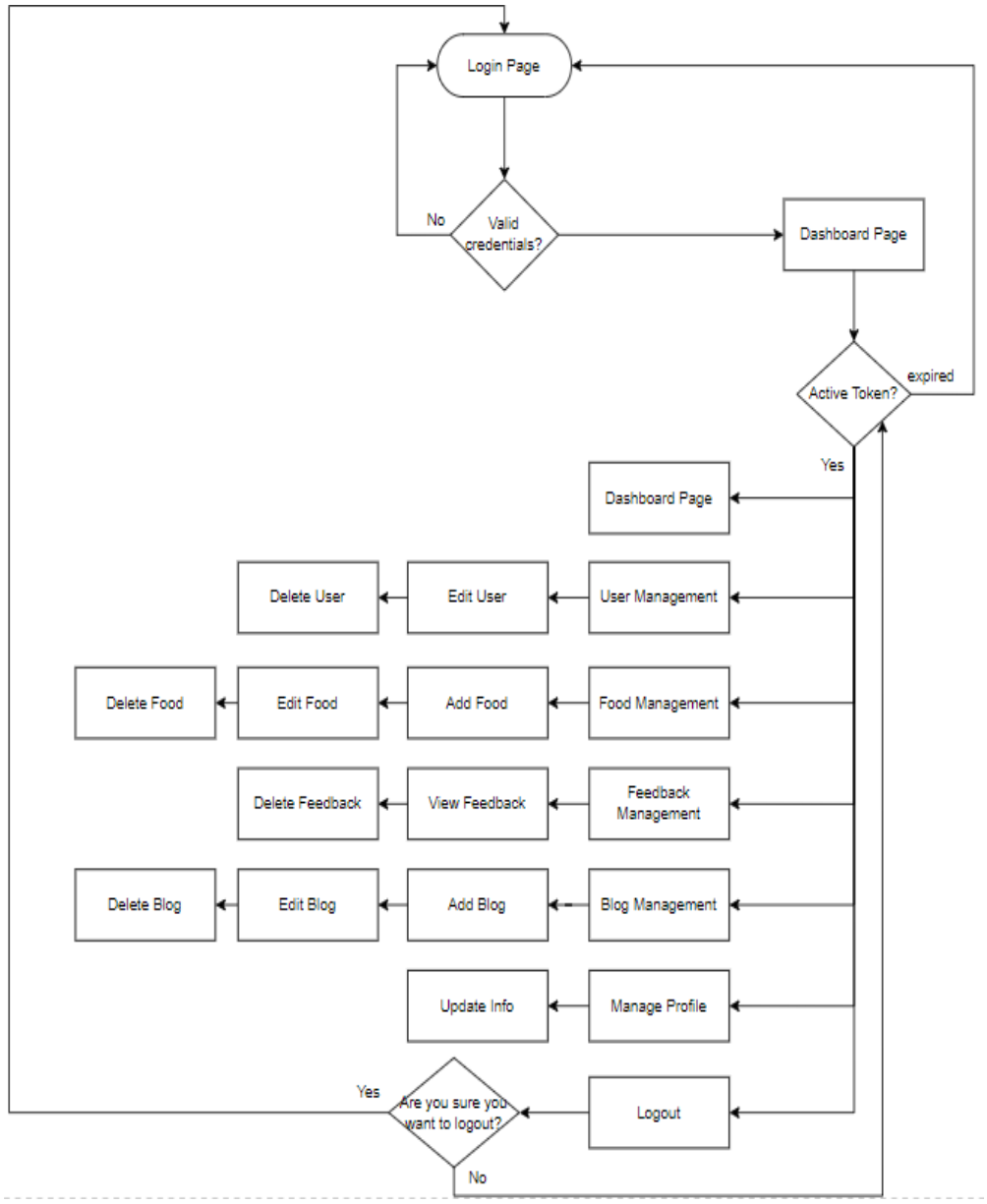
    - **Consistency:** The interface maintains consistent navigation and interaction patterns across different screens to reduce the learning curve and enhance user experience.

    - **Error Prevention and Recovery:** Forms and inputs are designed to prevent errors by using dropdowns for specific inputs and providing clear error messages to assist users in correcting entries before submission.

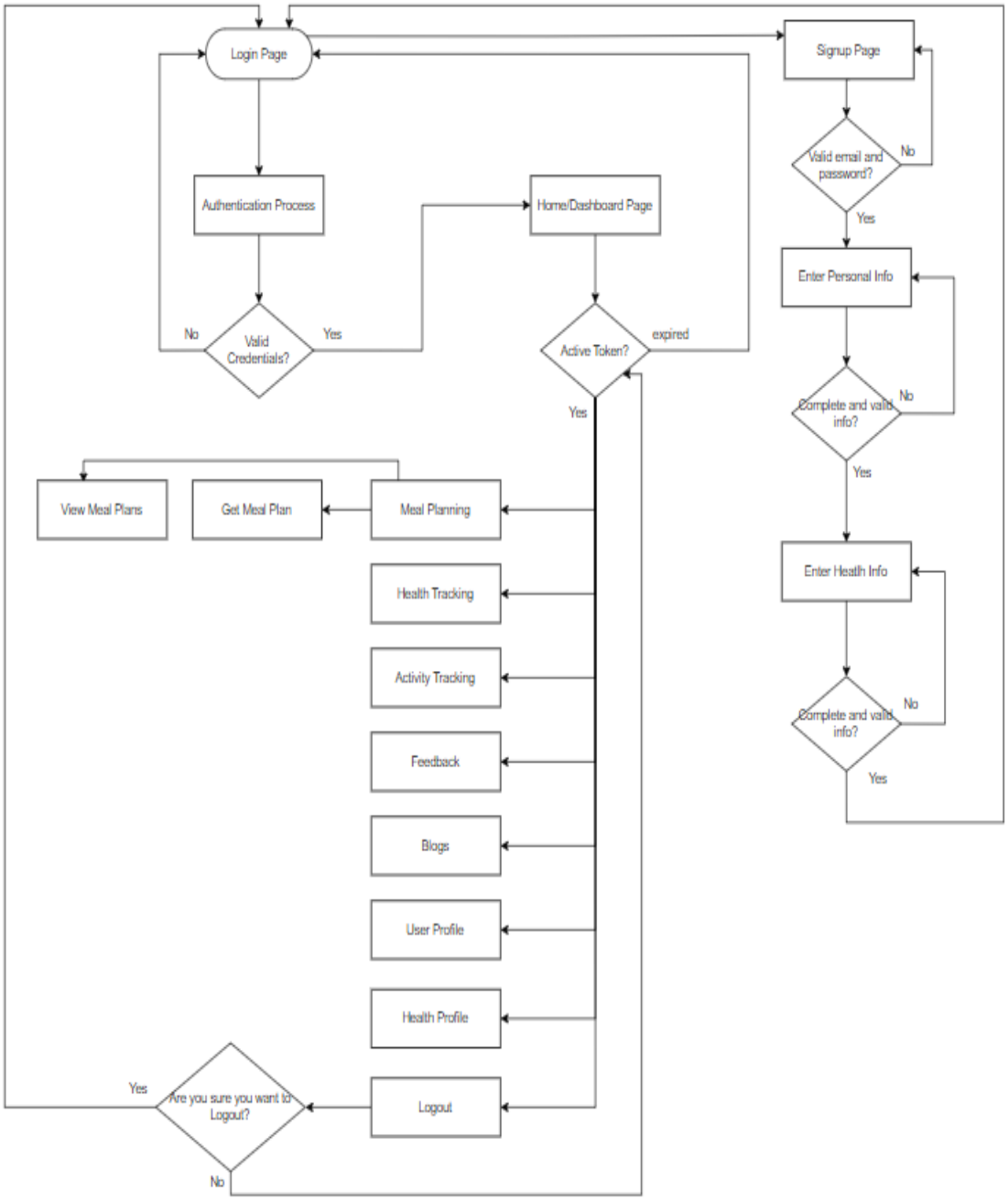
# 5.4 Screenshots/Prototype

## 5.4.1 Workflow

### 5.4.1.1 Admin Work Flow:

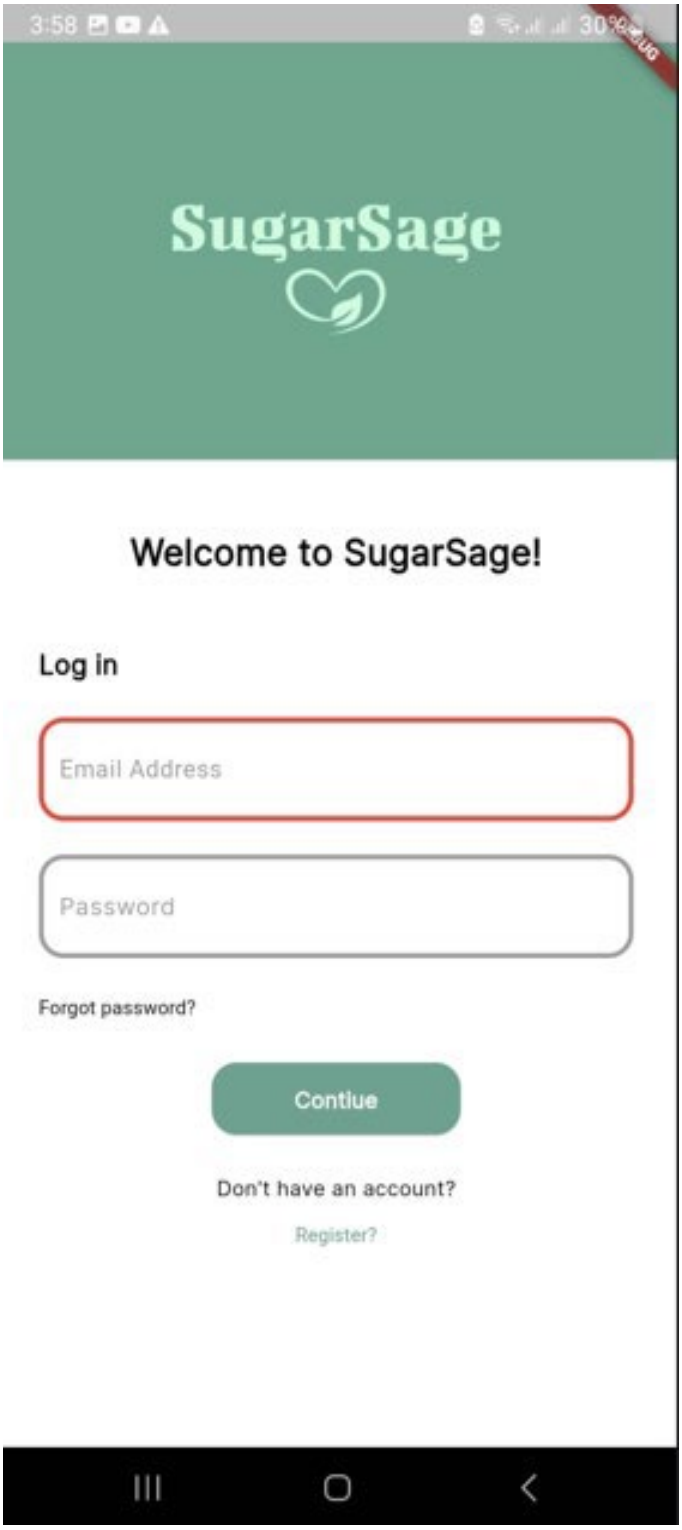


### 4.1.1.2 User Work Flow:

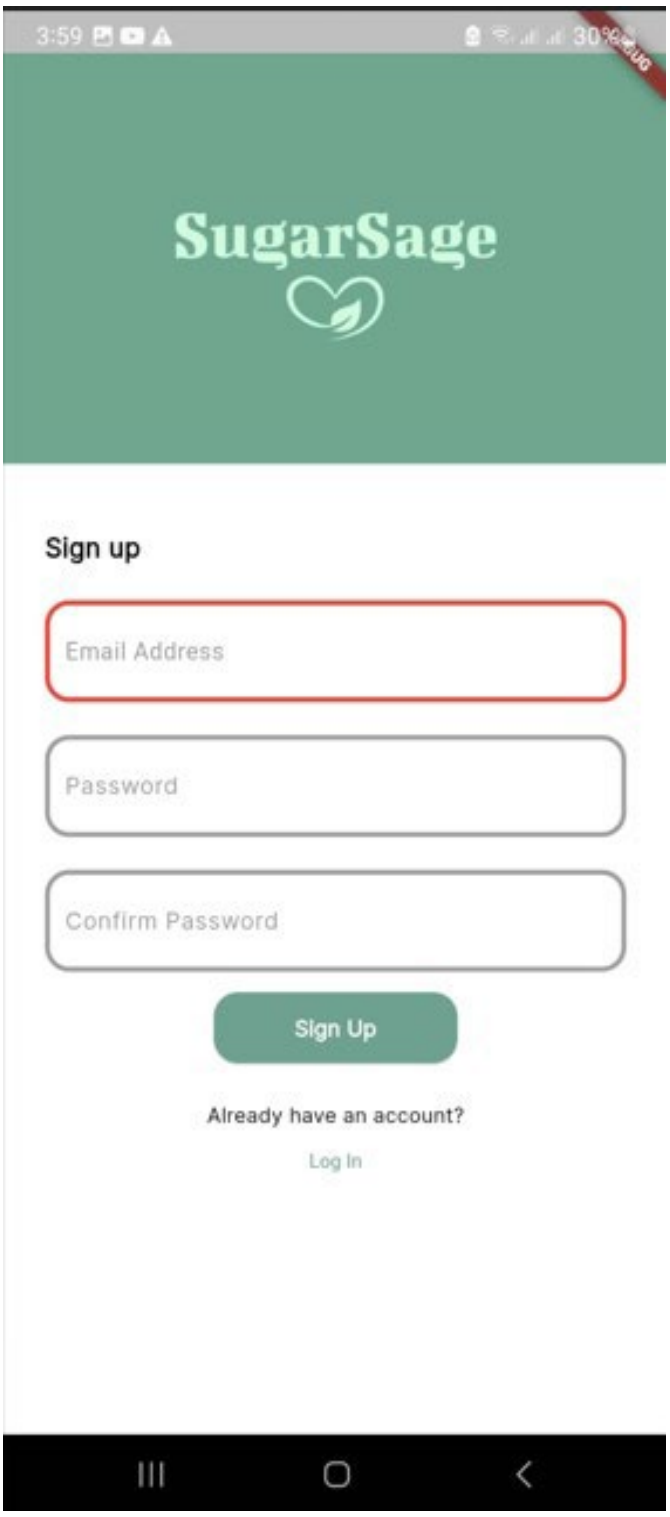


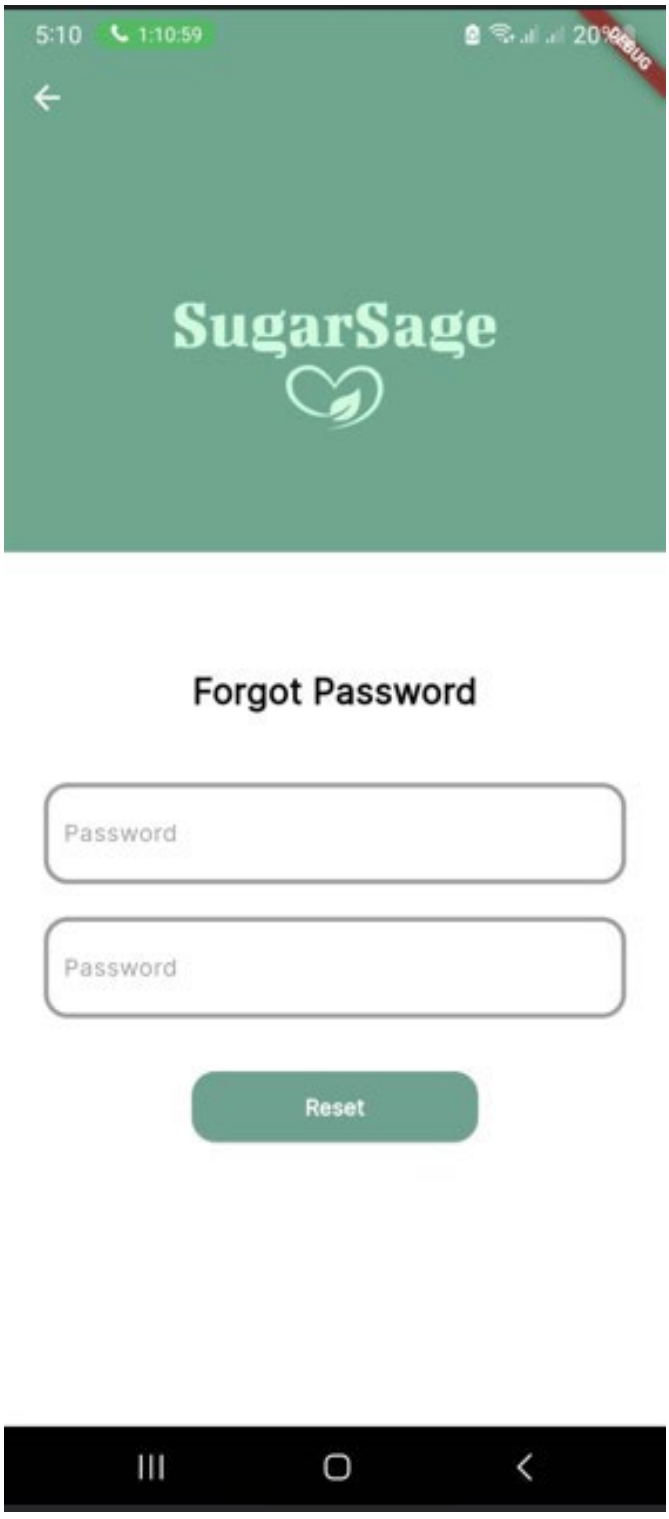
### 5.4.2 Screens

#### 5.4.2.1 Mobile App Screen:

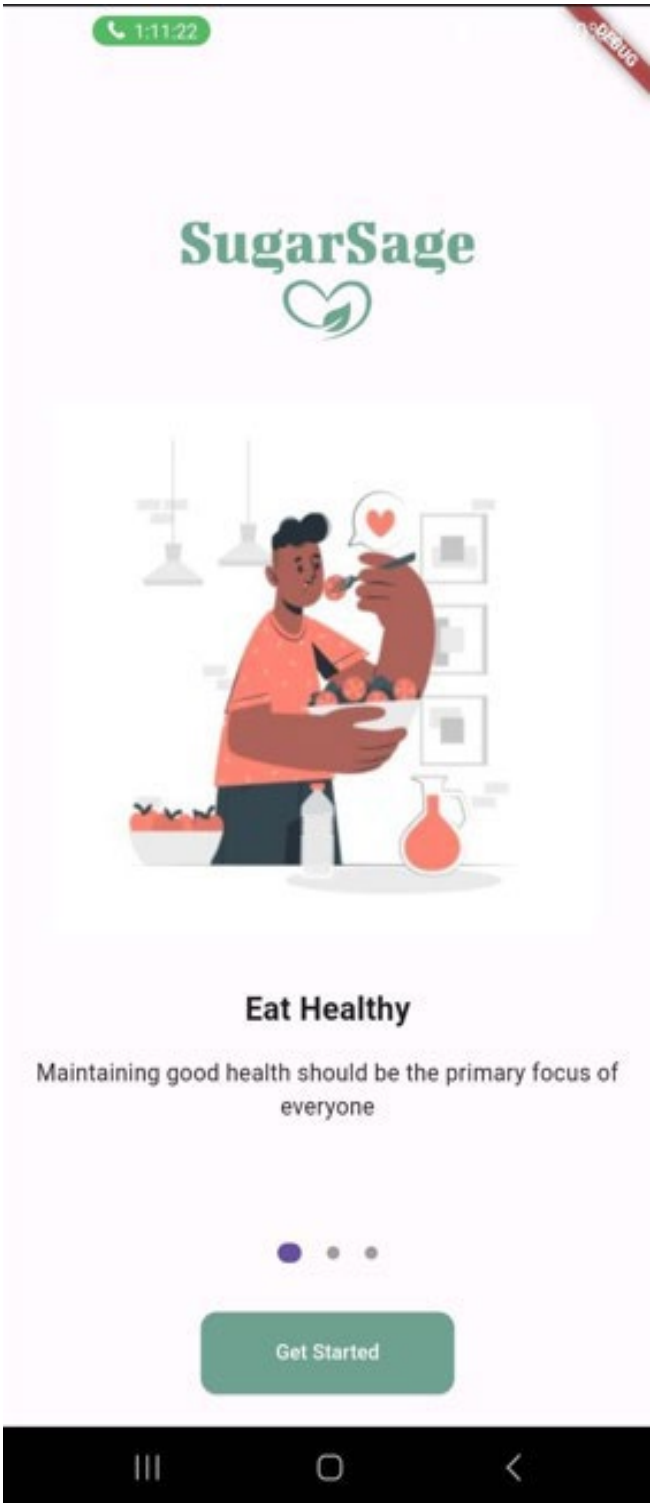



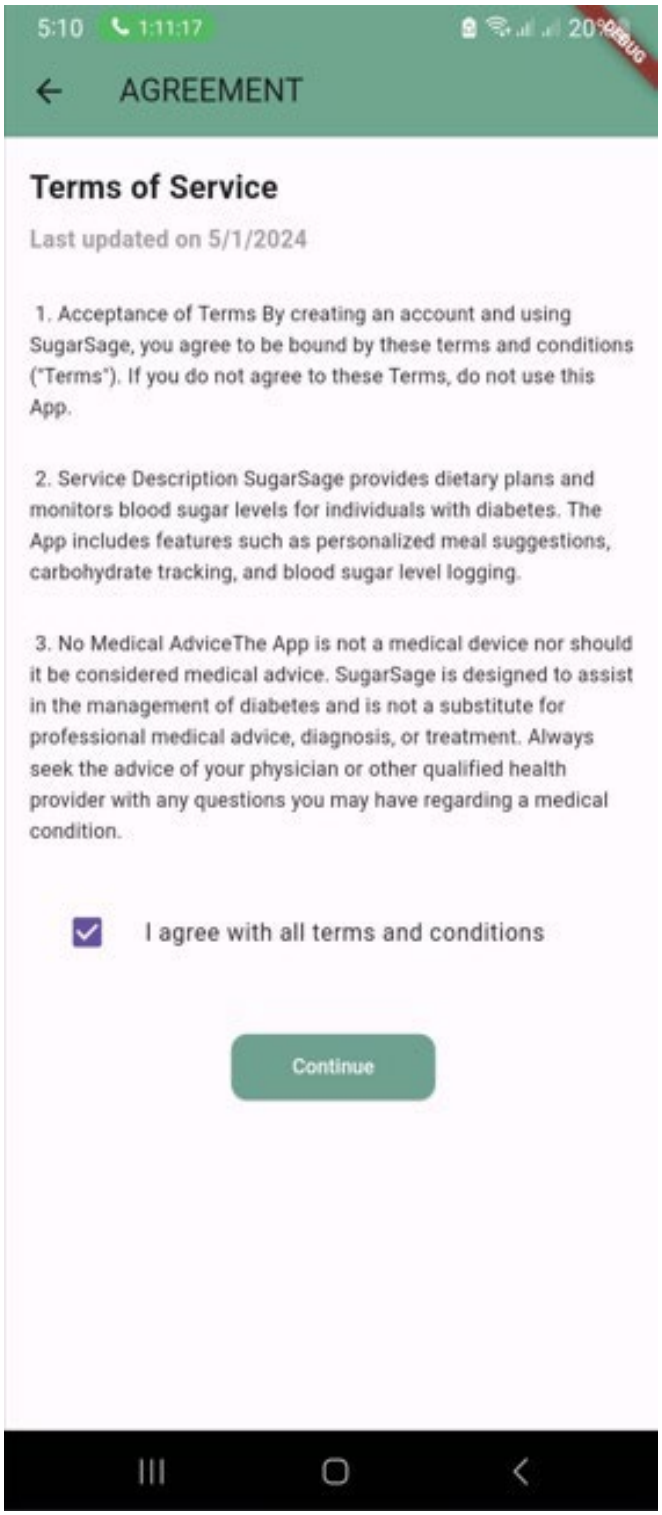

1:10:06
DEBUG
SugarSage
Welcome to SugarSage!
Log in
Email Address
safoora9770@gmail.com
Password
Forgot password?
Contiue
Don't have an account?
Register?


3:59
DEBUG
SugarSage
Sign up
Email Address
Password
Confirm Password
Sign Up
Already have an account?
Log In

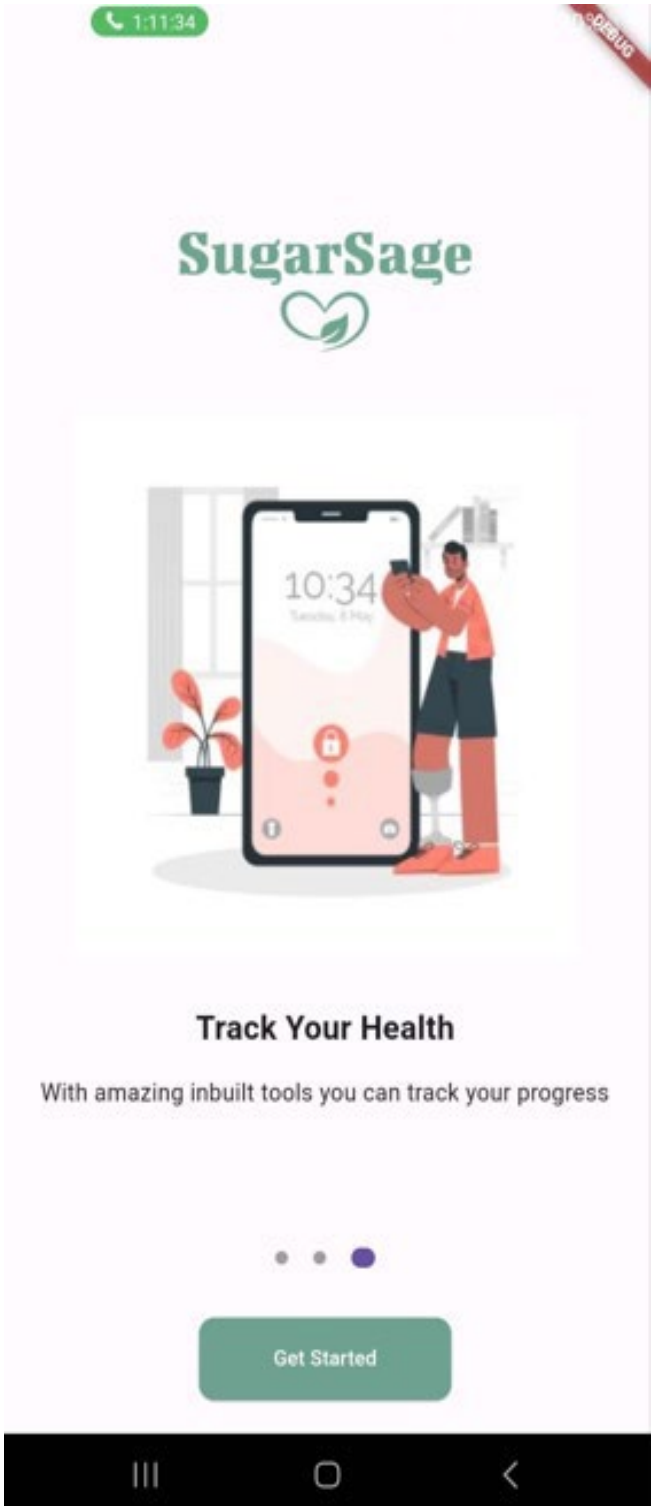

5:10
1:10:59
SugarSage
Forgot Password
Password
Password
Reset

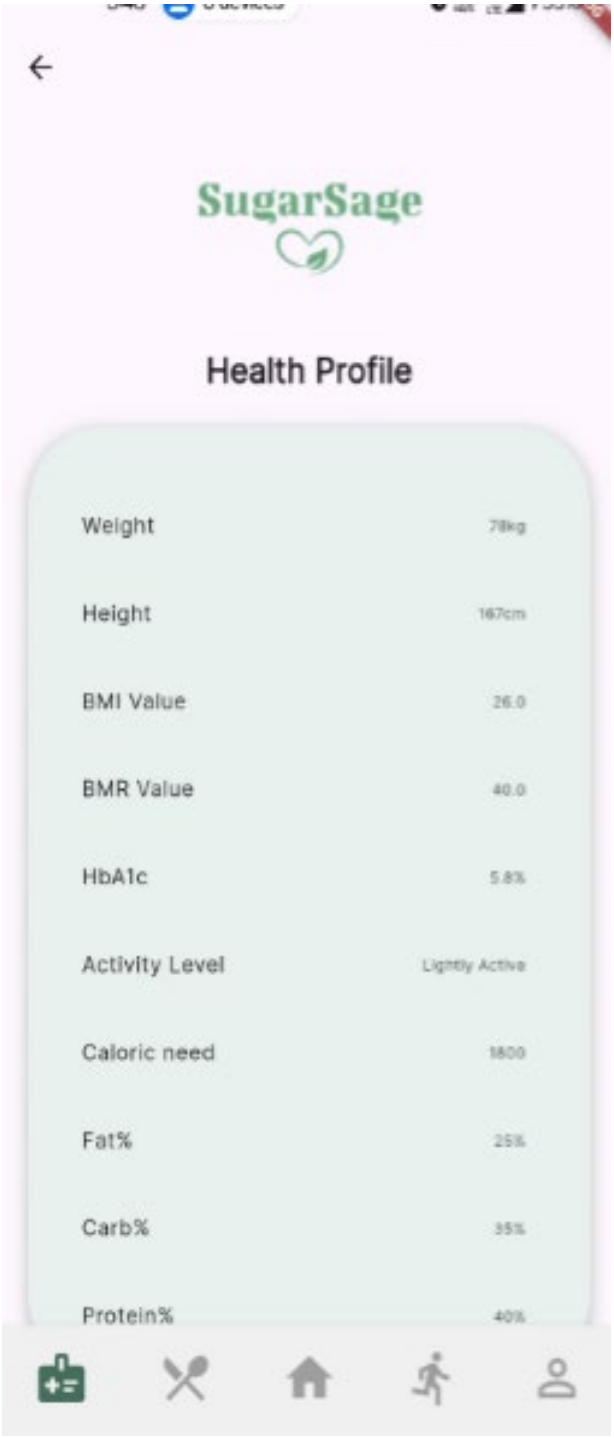

5:10
1:10:49
SugarSage
Forgot Password
Password
Password
Password
Reset

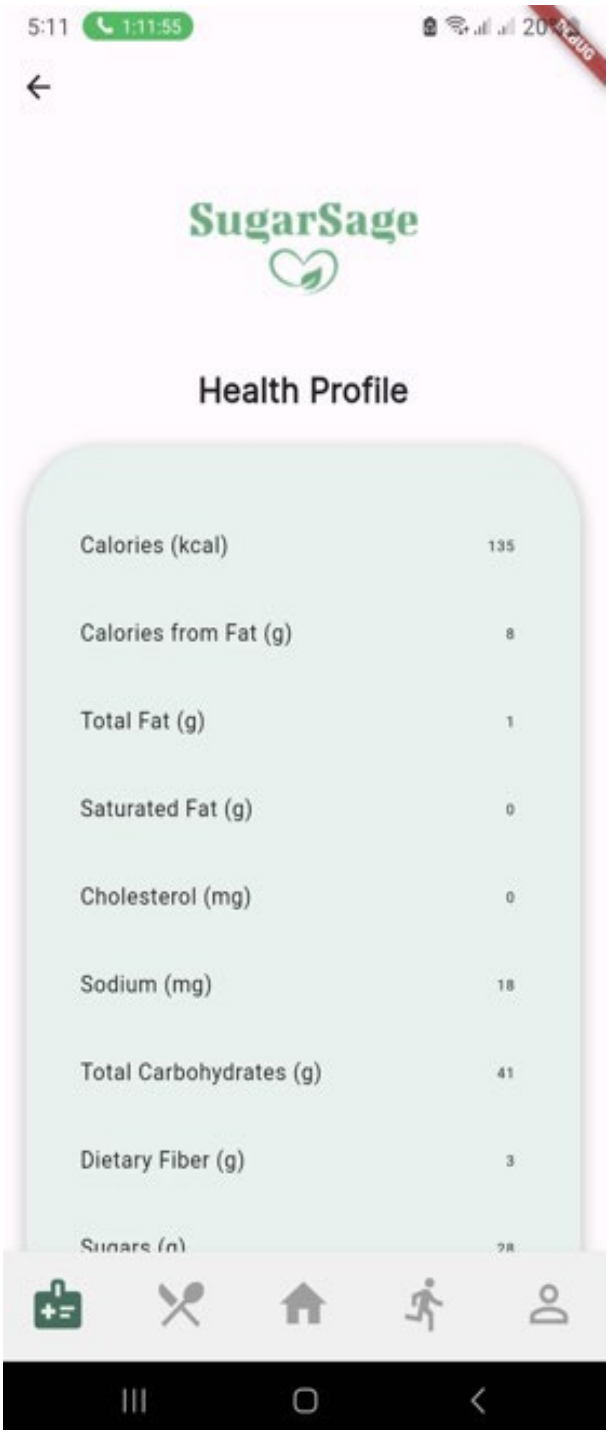

1:11:22
DEBUG
SugarSage
Eat Healthy
Maintaining good health should be the primary focus of everyone
Get Started

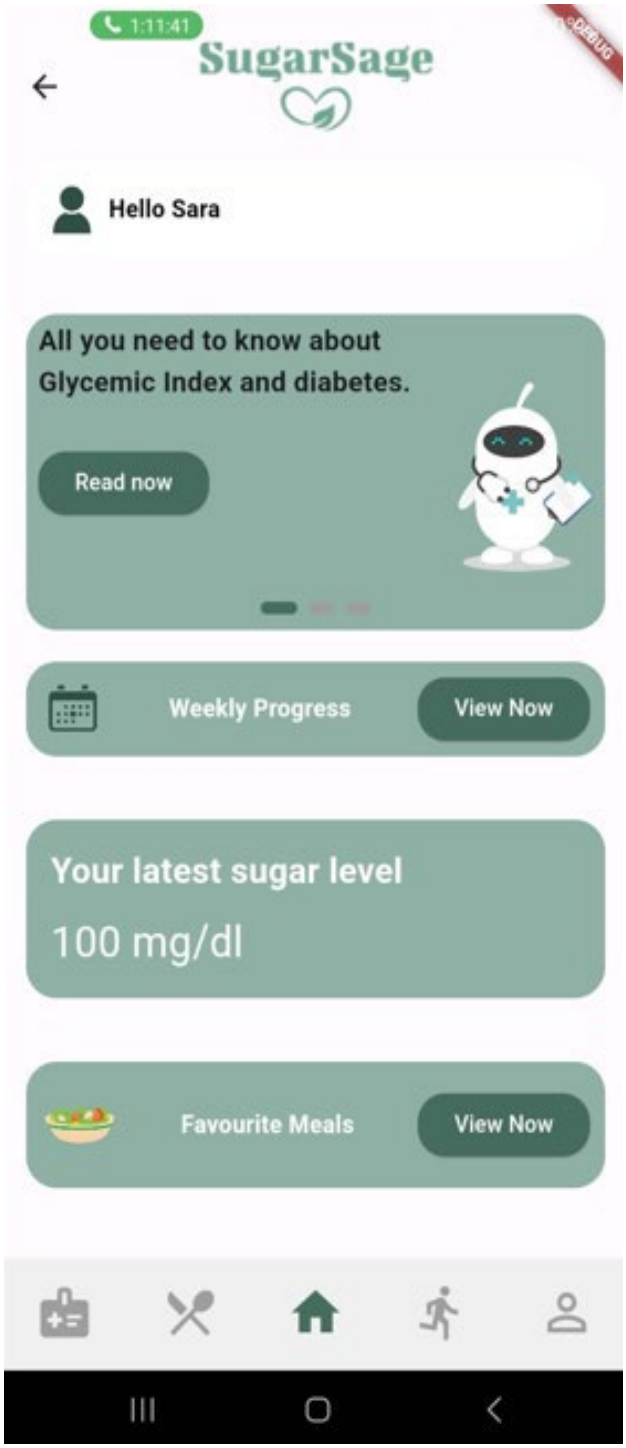

5:10
1:11:17
DEBUG
AGREEMENT
Terms of Service
Last updated on 5/1/2024
1. Acceptance of Terms By creating an account and using SugarSage, you agree to be bound by these terms and conditions ("Terms"). If you do not agree to these Terms, do not use this App.
2. Service Description SugarSage provides dietary plans and monitors blood sugar levels for individuals with diabetes. The App includes features such as personalized meal suggestions, carbohydrate tracking, and blood sugar level logging.
3. No Medical AdviceThe App is not a medical device nor should it be considered medical advice. SugarSage is designed to assist in the management of diabetes and is not a substitute for professional medical advice, diagnosis, or treatment. Always seek the advice of your physician or other qualified health provider with any questions you may have regarding a medical condition.
I agree with all terms and conditions
Continue


1:11:28
DEBUG
SugarSage
Healthy and Local Recipes
Have access to healthy and local recipes
Get Started


1:11:34
DEBUG
SugarSage
10:34
Track Your Health
With amazing inbuilt tools you can track your progress
Get Started

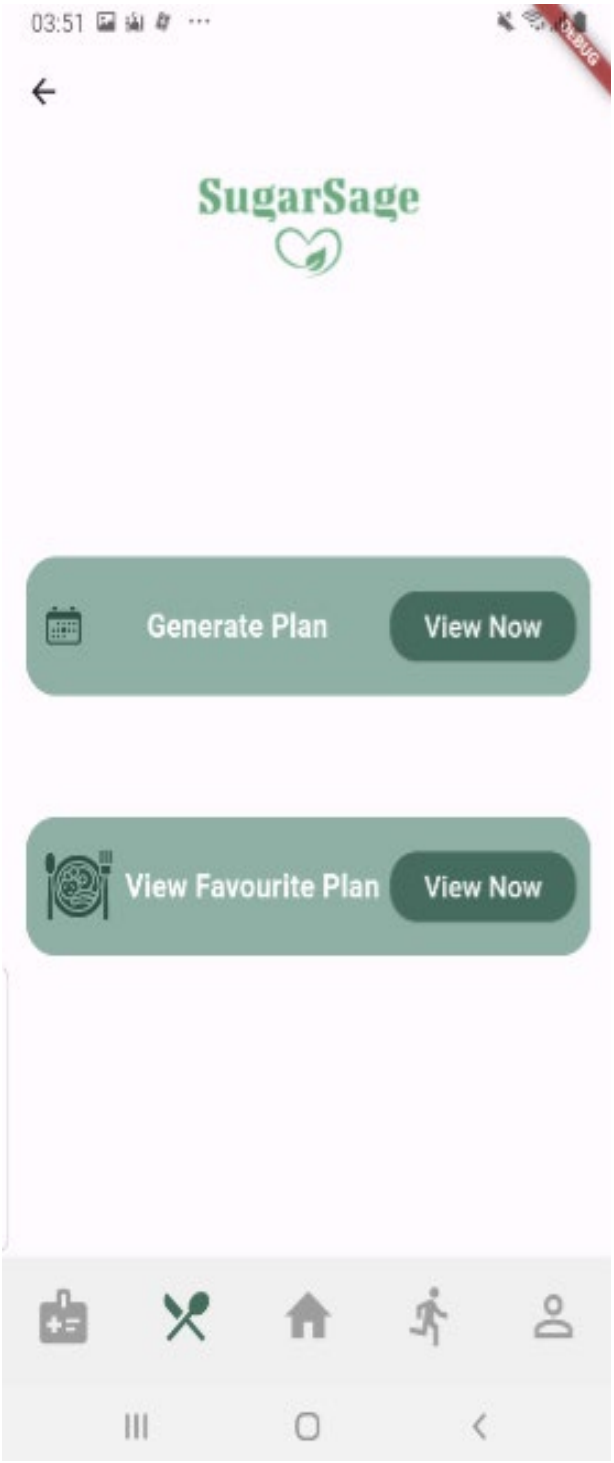

SugarSage
Health Profile
Weight
78kg
Height
167cm
BMI Value
26.0
BMR Value
40.0
HbA1c
5.8%
Activity Level
Lightly Active
Caloric need
1800
Fat%
25%
Carb%
35%
Protein%
40%

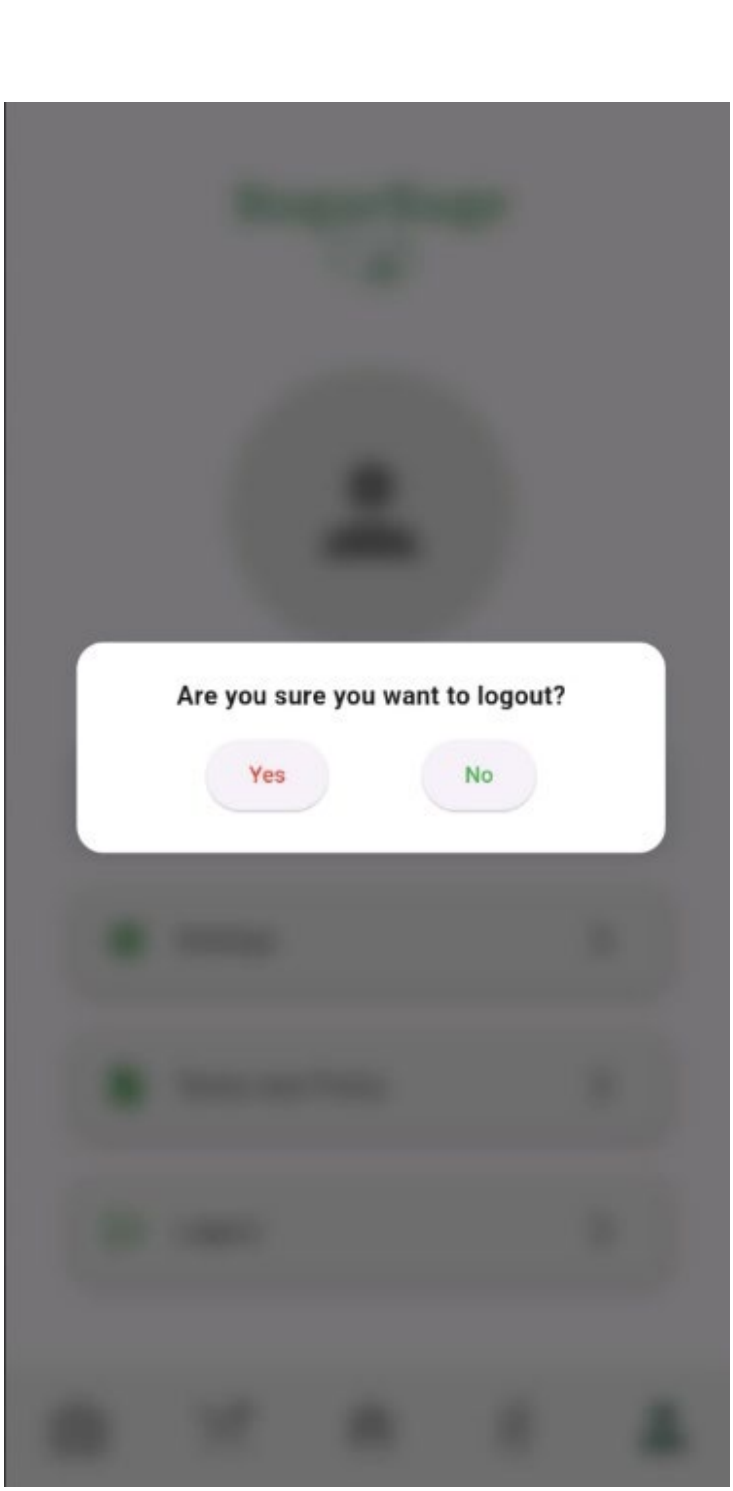

5:11
1:11:55
SugarSage
Health Profile
Calories (kcal)
135
Calories from Fat (g)
8
Total Fat (g)
1
Saturated Fat (g)
0
Cholesterol (mg)
0
Sodium (mg)
18
Total Carbohydrates (g)
41
Dietary Fiber (g)
3

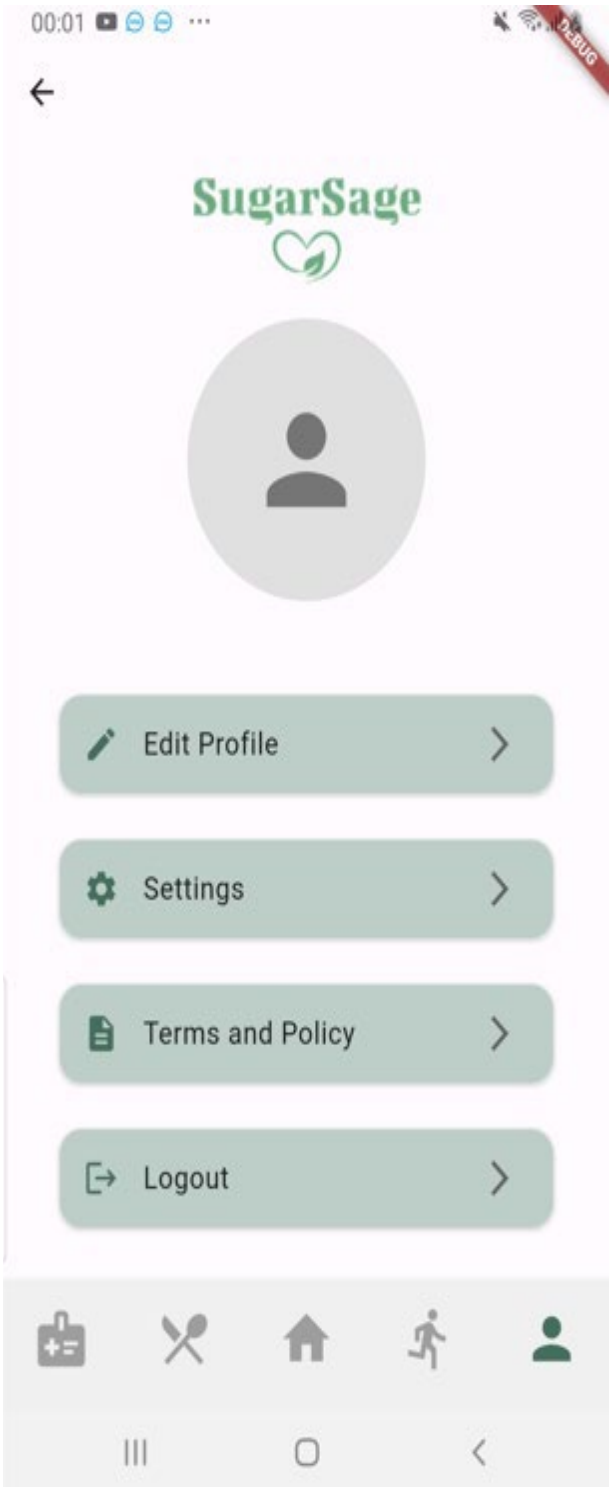

1:11:41
SugarSage
Hello Sara
All you need to know about
Glycemic Index and diabetes.
Read now
Weekly Progress
View Now
Your latest sugar level
100 mg/dl
Favourite Meals
View Now

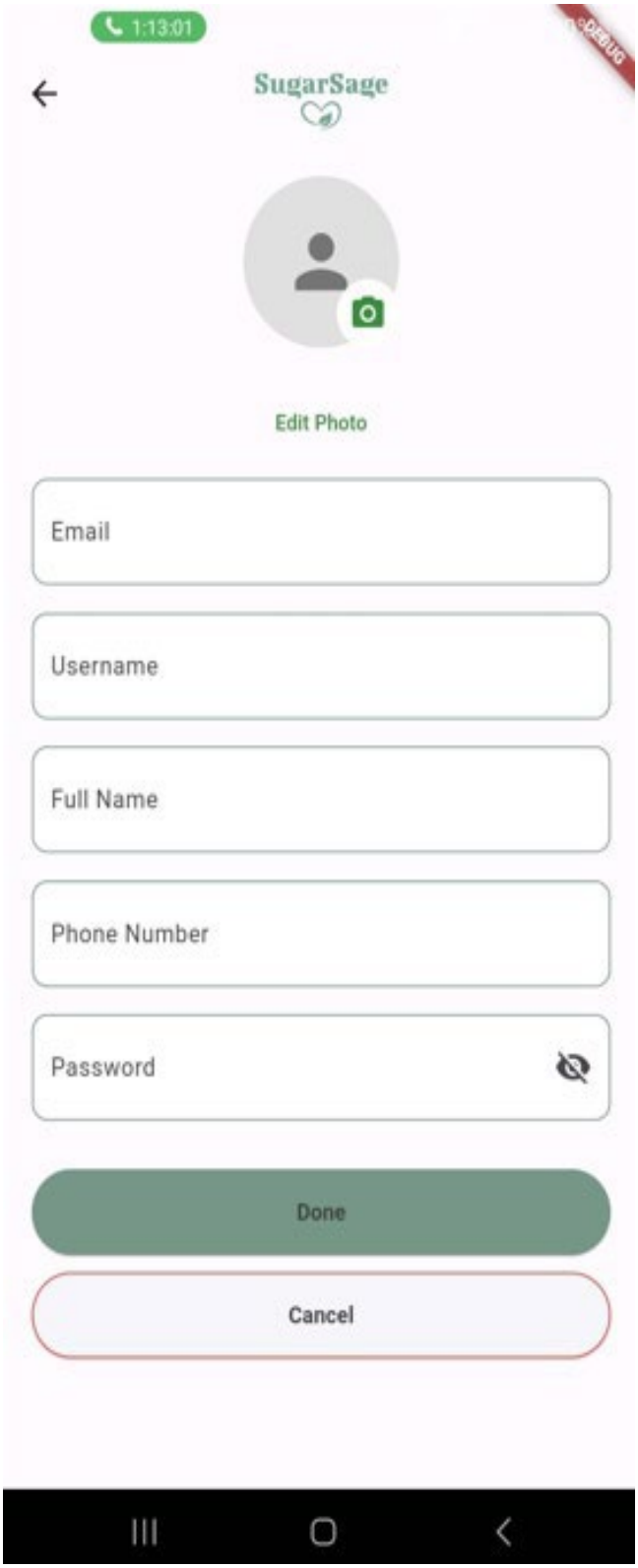

5:12
1:12:34
SugarSage
1. Banana Smoothie
Calories (kcal) 135
Calories from Fat (g) 8
Total Fat (g) 1
Saturated Fat (g) 0
Cholesterol (mg) 0
Sodium (mg) 18
Total Carbohydrates (g) 41
Dietary Fiber (g) 3
Sugars (g) 28
Protein (g) 3

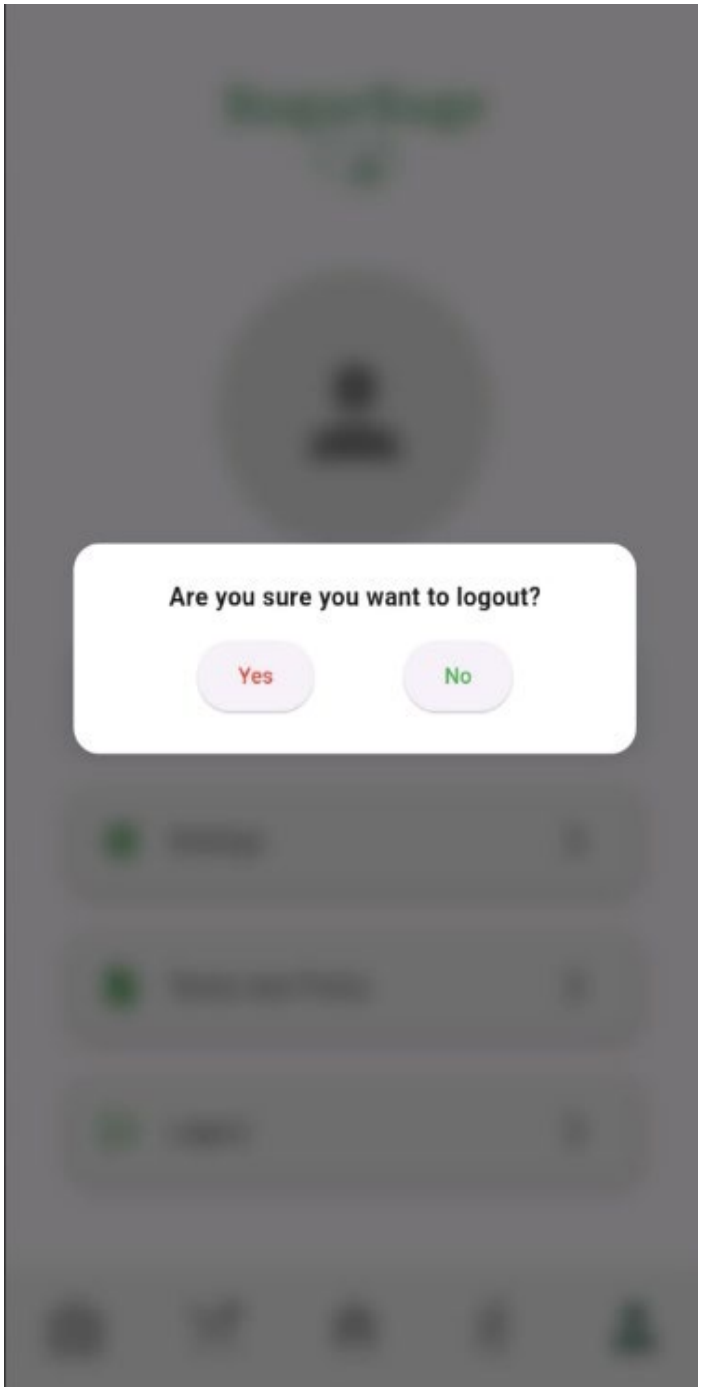

00:02
SugarSage
Breakfast
Banana Smoothie
1 medium banana
2 cups milk
1 tbsp peanut butter
Toast
2 slices of bran bread
1 tsp of salted butter
Nutritional Values
Lunch
Grilled Chicken
150g chicken breast
1 tbsp olive oil
Salad
1 cup mixed greens

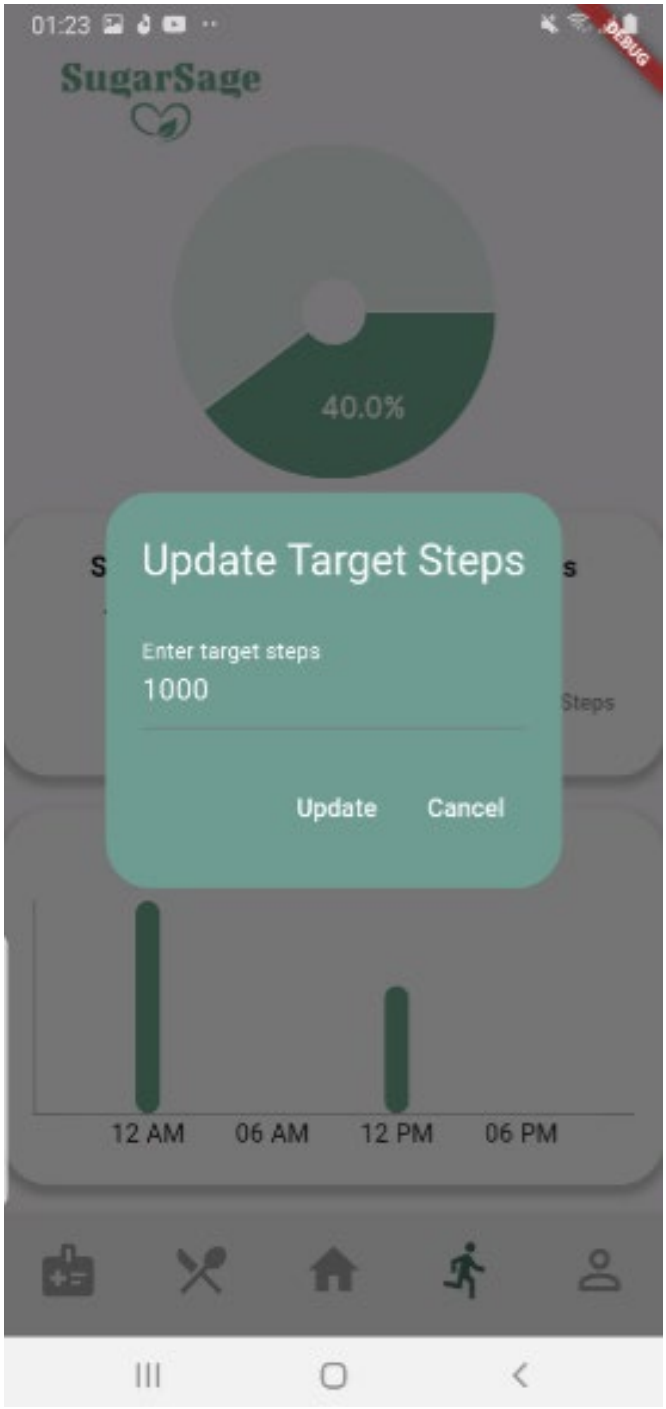

03:51
SugarSage
Generate Plan
View Now
View Favourite Plan
View Now

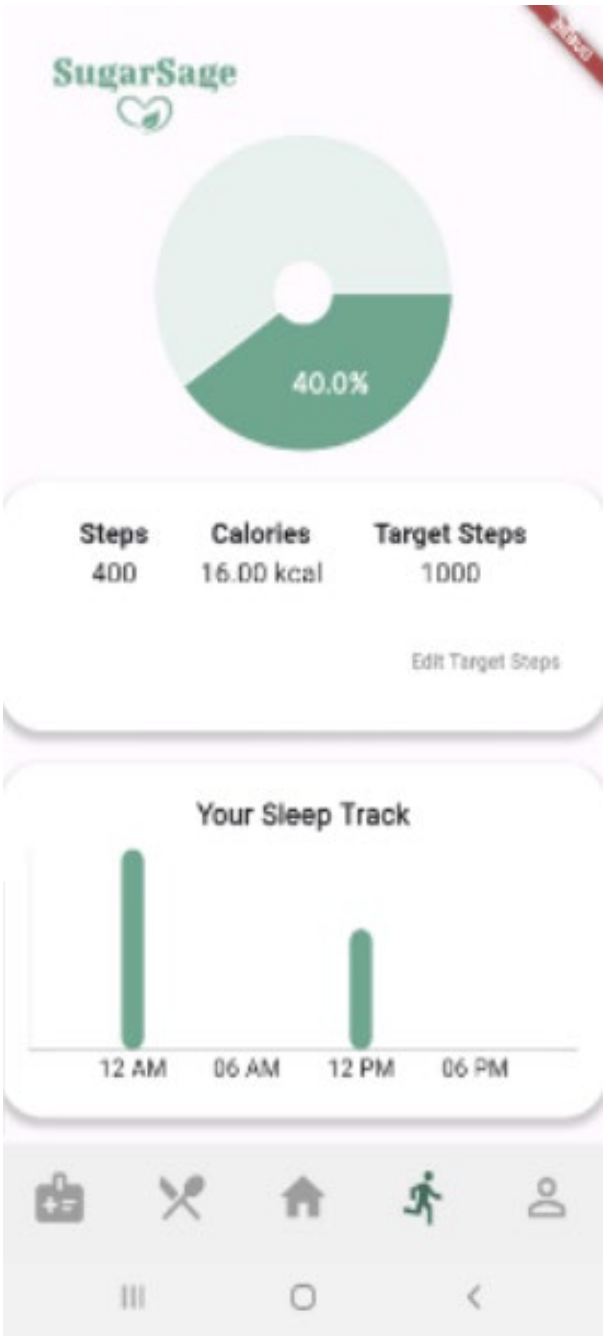

Are you sure you want to logout?
Yes
No

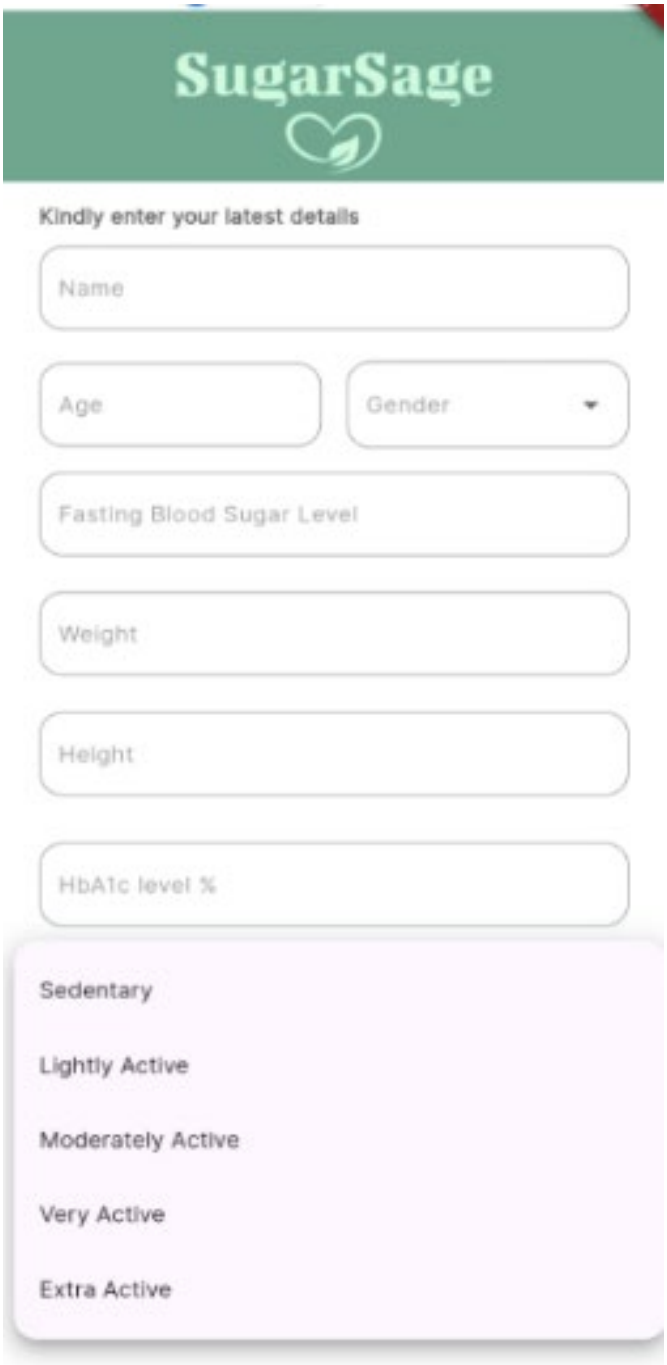

00:01
DEBUG
SugarSage
Edit Profile
Settings
Terms and Policy
Logout


1:13:01
SugarSage
Edit Photo
Email
Username
Full Name
Phone Number
Password
Done
Cancel

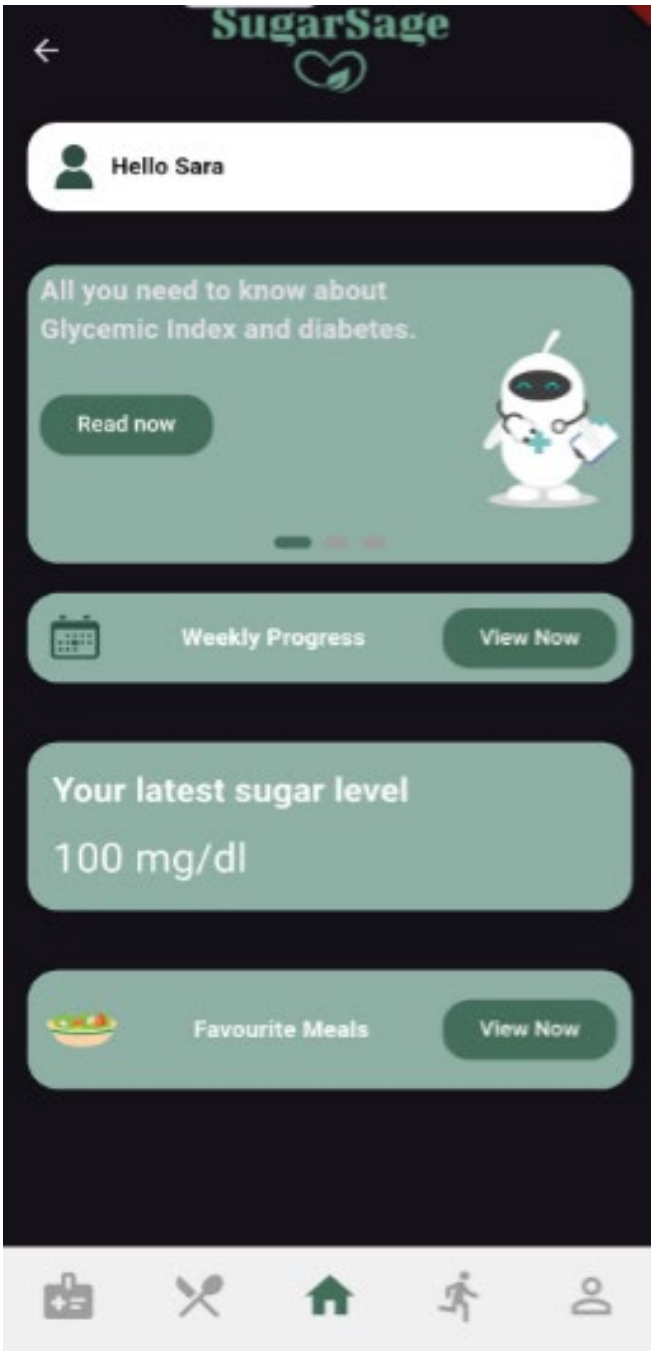

Are you sure you want to logout?
Yes
No


01:23
SugarSage
40.0%
Update Target Steps
Enter target steps
1000
Update
Cancel
12 AM
06 AM
12 PM
06 PM


SugarSage
40.0%
Steps
400
Calories
16.00 kcal
Target Steps
1000
Edit Target Steps
Your Sleep Track
12 AM
06 AM
12 PM
06 PM

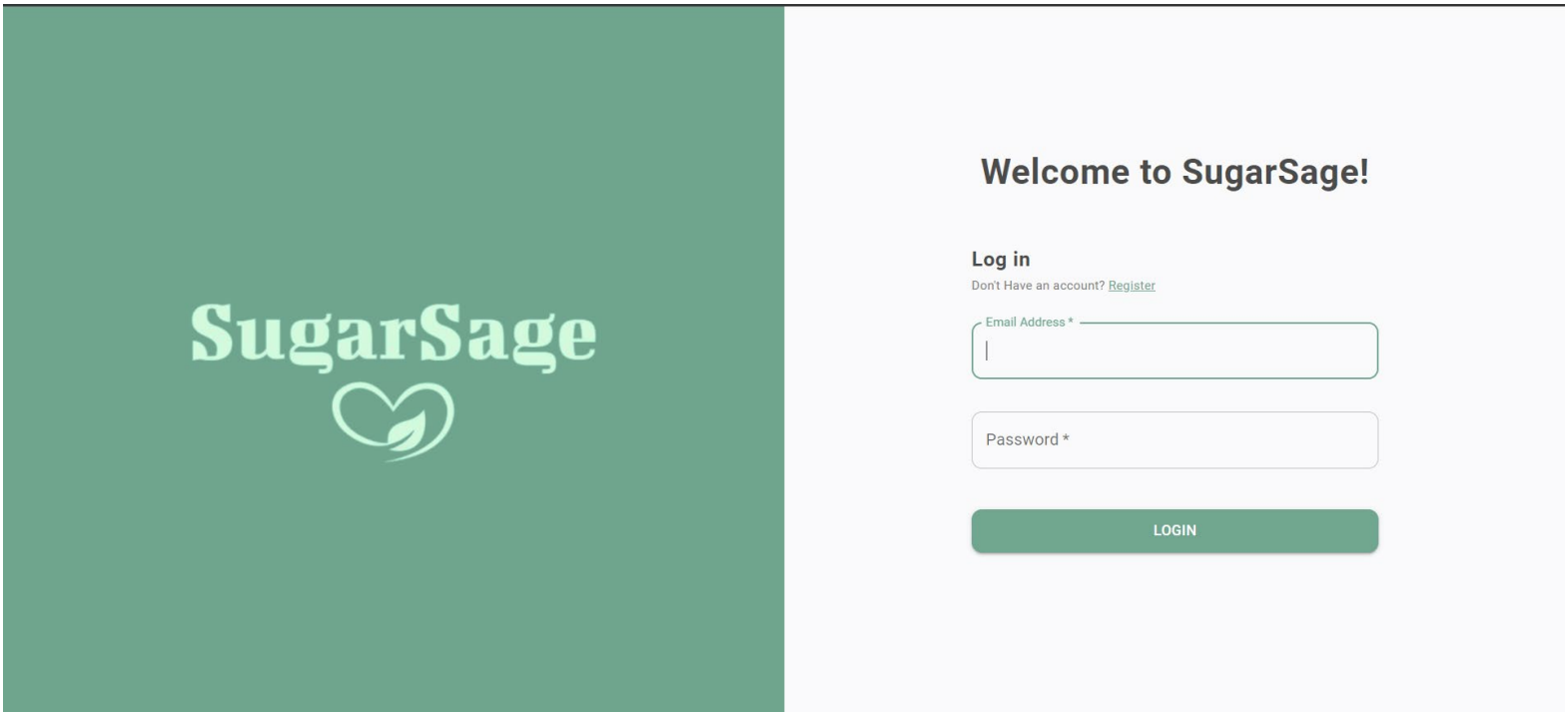

SugarSage
Kindly enter your latest details
Name
Age
Gender
Fasting Blood Sugar Level
Weight
Height
HbA1c level %
Sedentary
Lightly Active
Moderately Active
Very Active
Extra Active

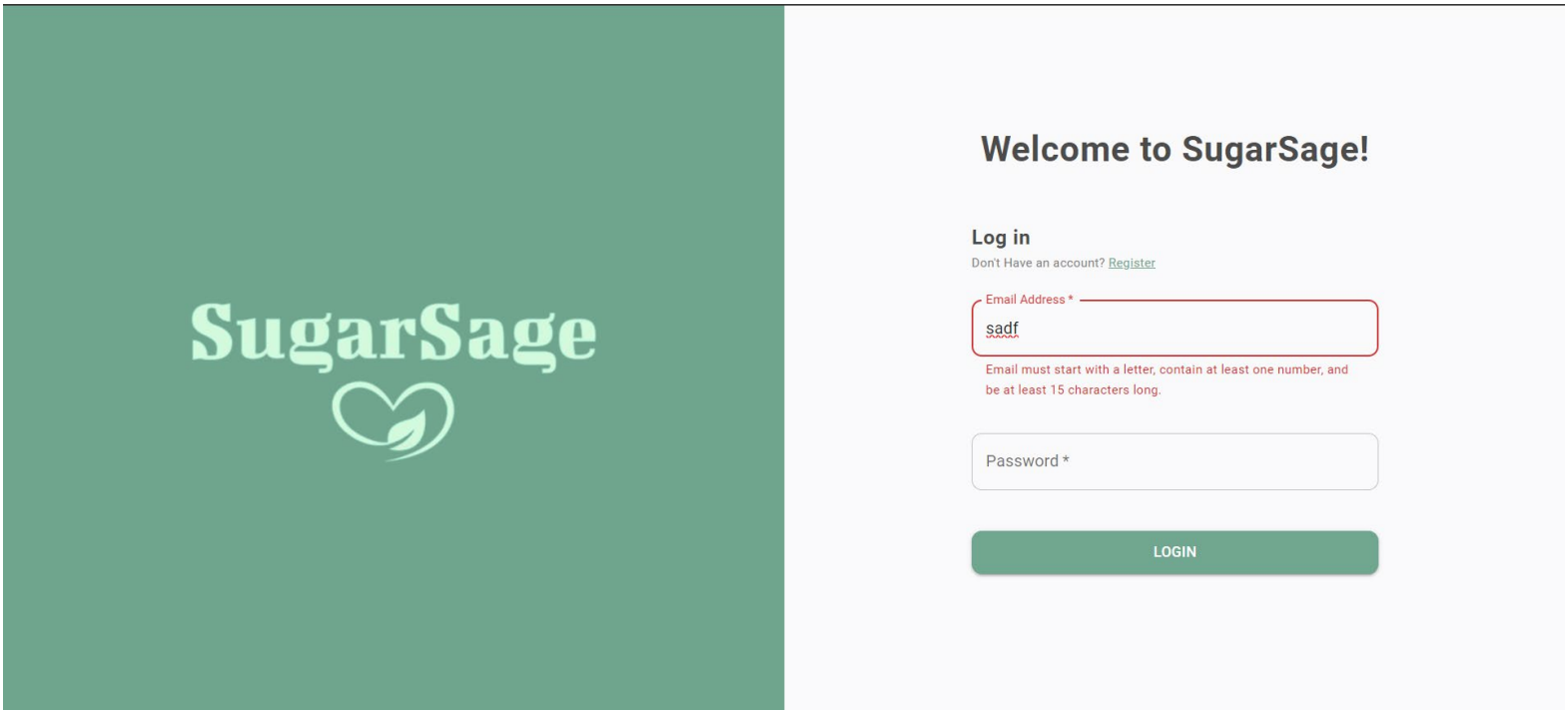

SugarSage
Kindly enter your latest details
Name
Age
Male
Female
Other
Weight
Height
HbA1c level %
Activity Level
Continue

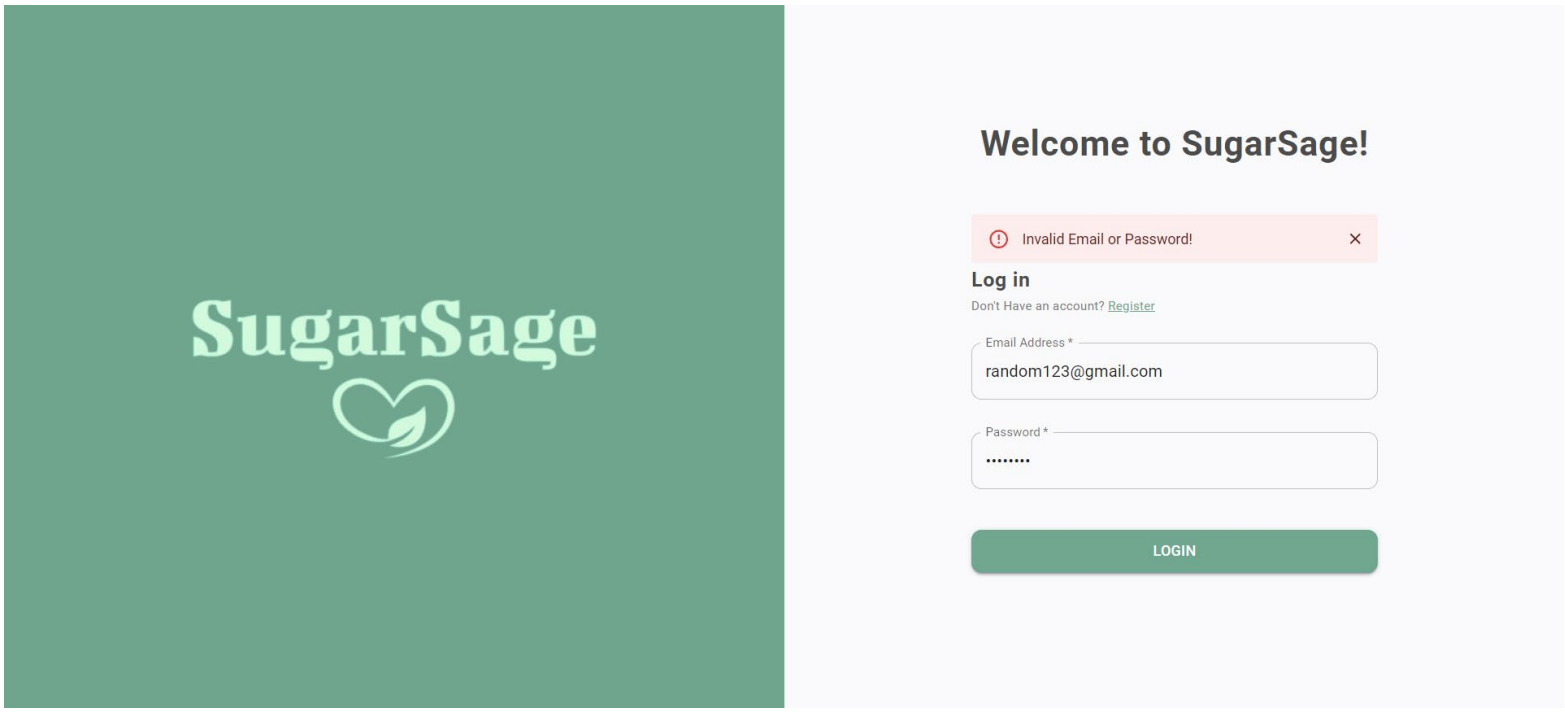

SugarSage
Hello Sara
All you need to know about
Glycemic Index and diabetes.
Read now
Weekly Progress
View Now
Your latest sugar level
100 mg/dl
Favourite Meals
View Now

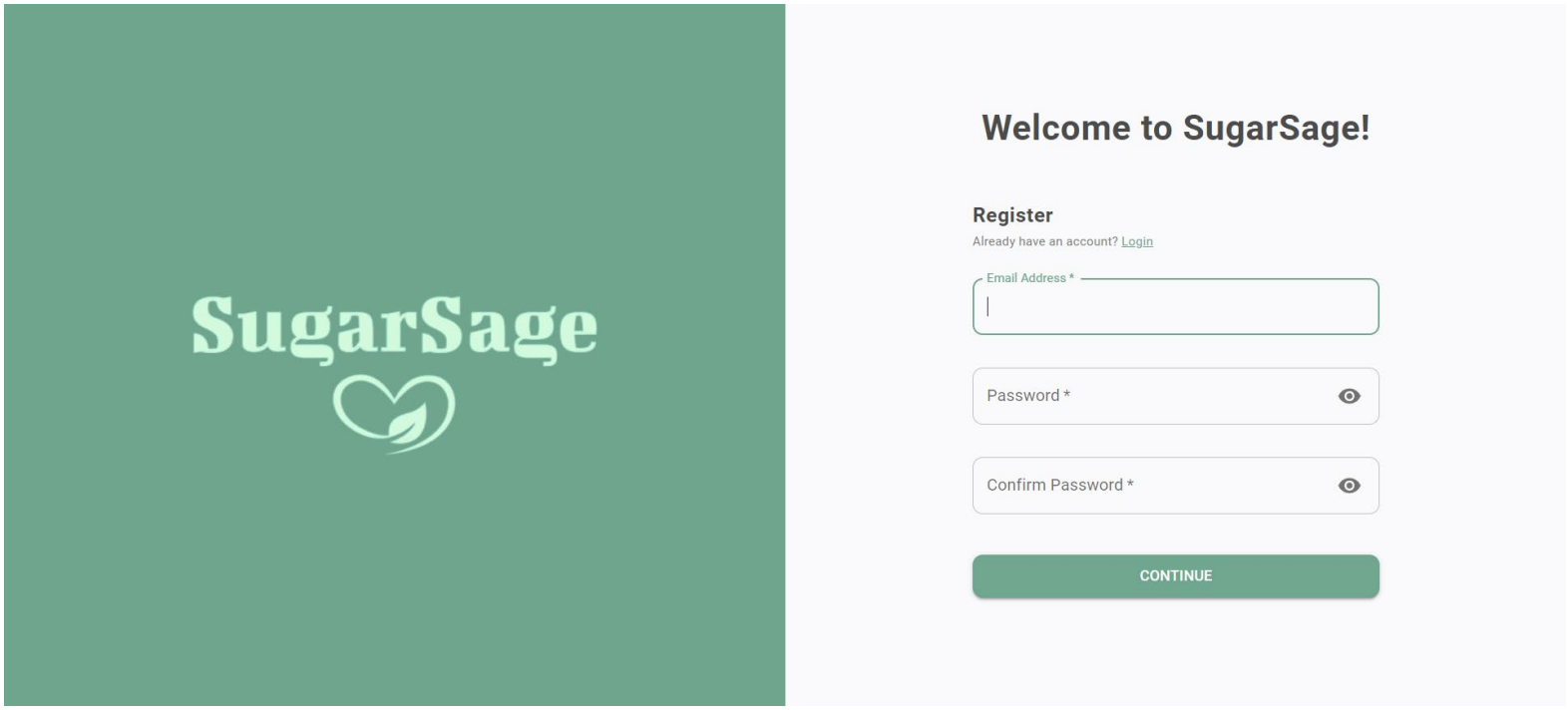

SugarSage
Setting
Change Password
Dark Mode

### 5.4.2.2 User Web App Screen:

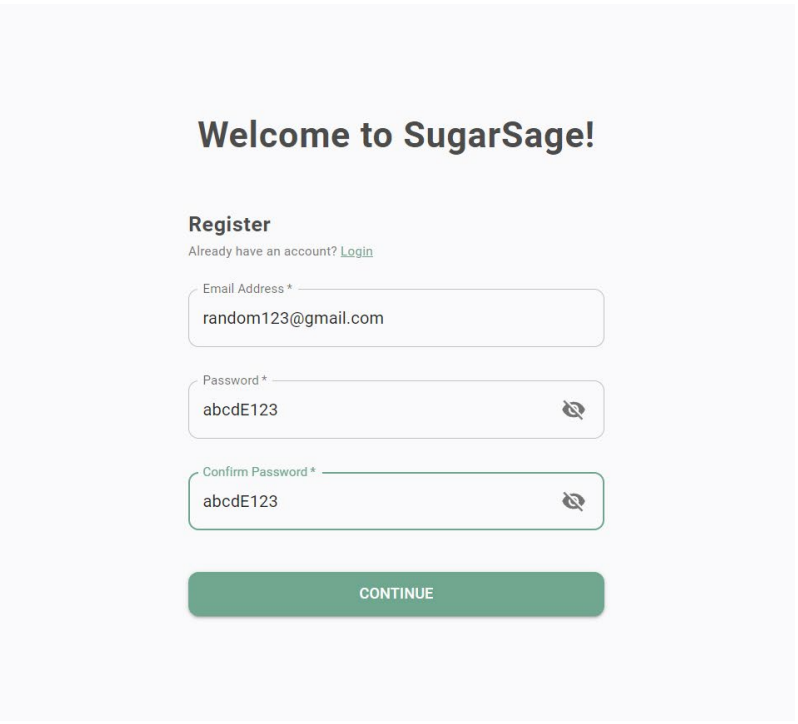


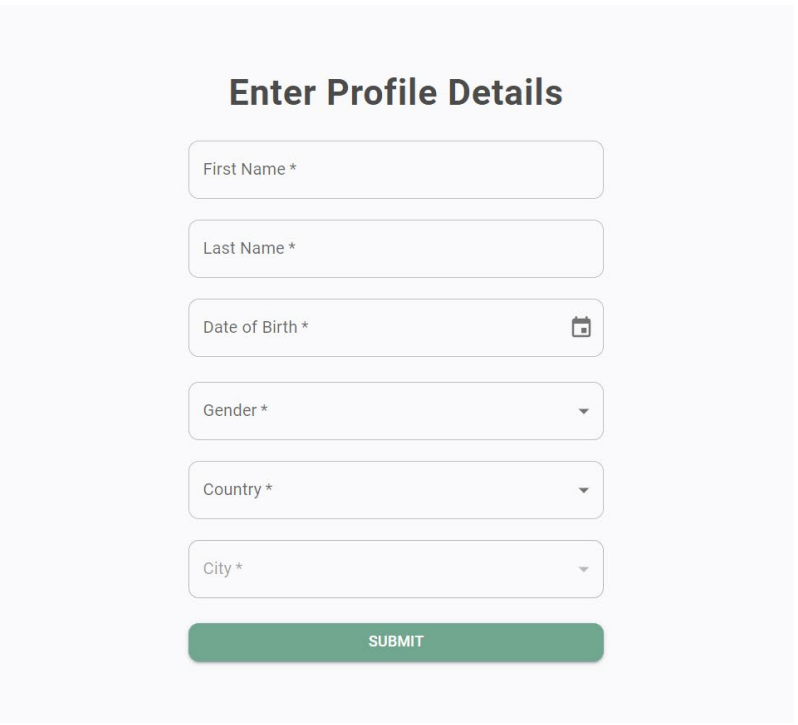

SugarSage
Welcome to SugarSage!
Invalid Email or Password!
Log in
Don't Have an account? Register
Email Address *
random123@gmail.com
Password *
LOGIN


SugarSage
Welcome to SugarSage!
Register
Already have an account? Login
Email Address *
Password *
Confirm Password *
CONTINUE

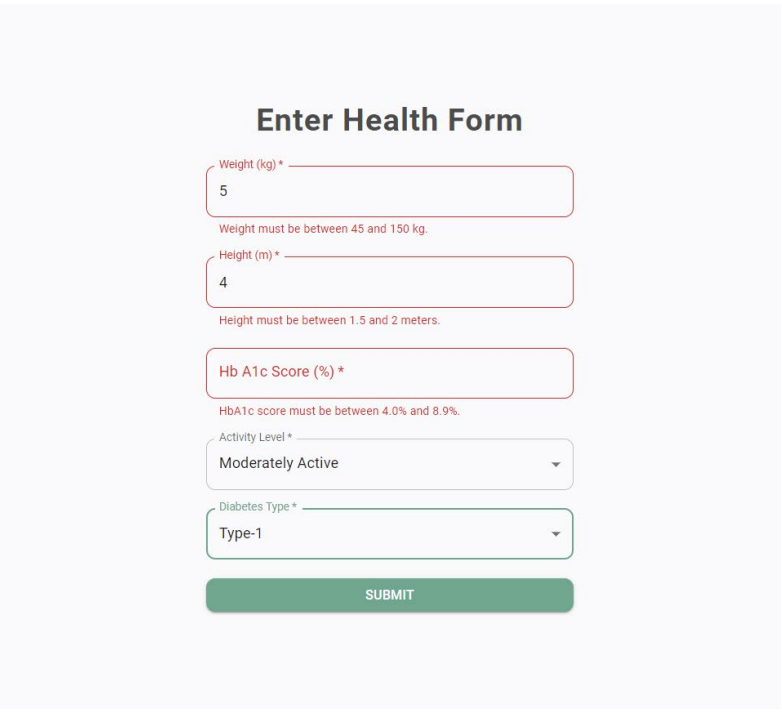

SugarSage

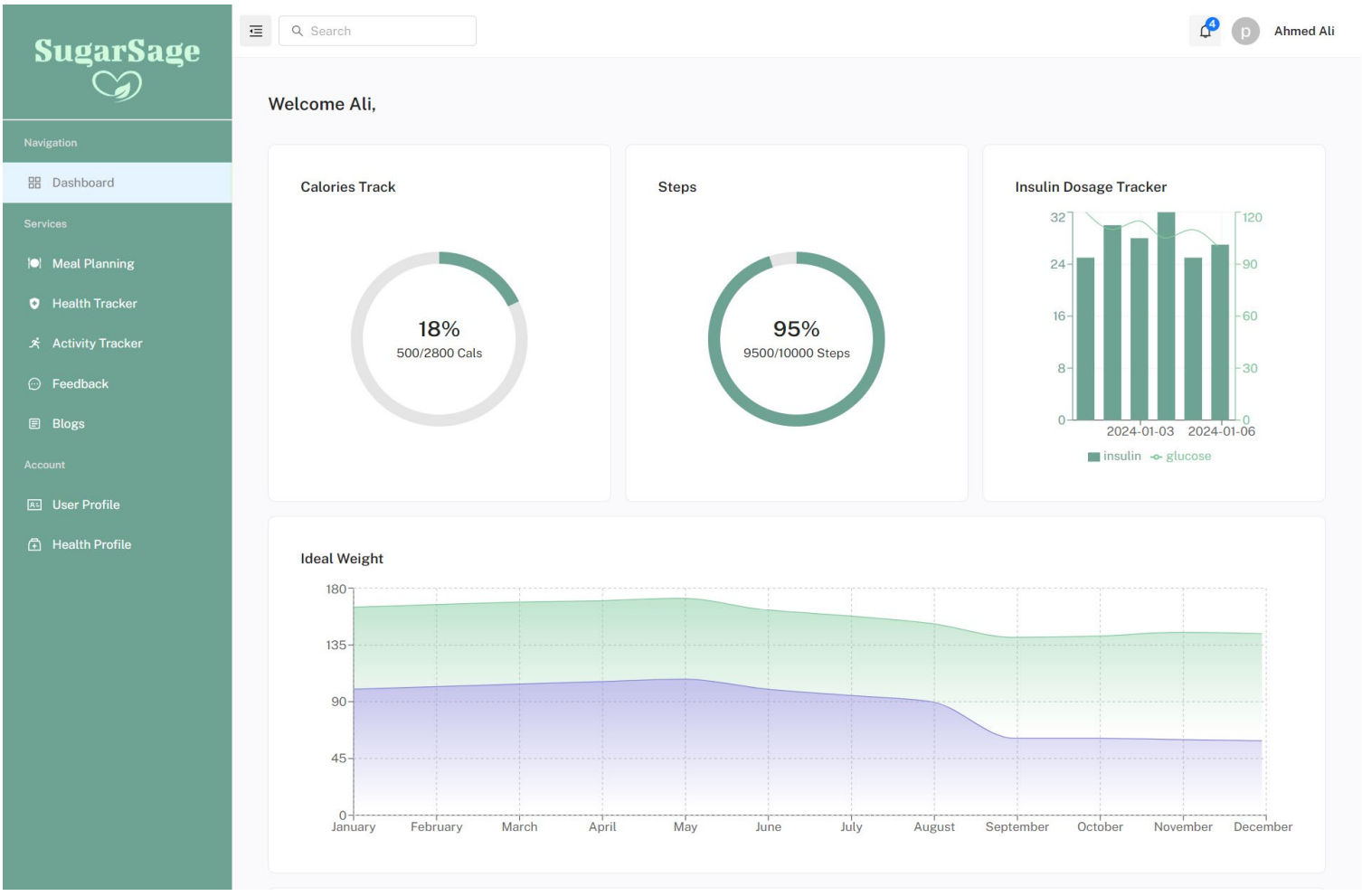

Welcome to SugarSage!
Register
Already have an account? Login
Email Address *
random123@gmail.com
Password *
abcdE123
Confirm Password *
abcdE123
CONTINUE

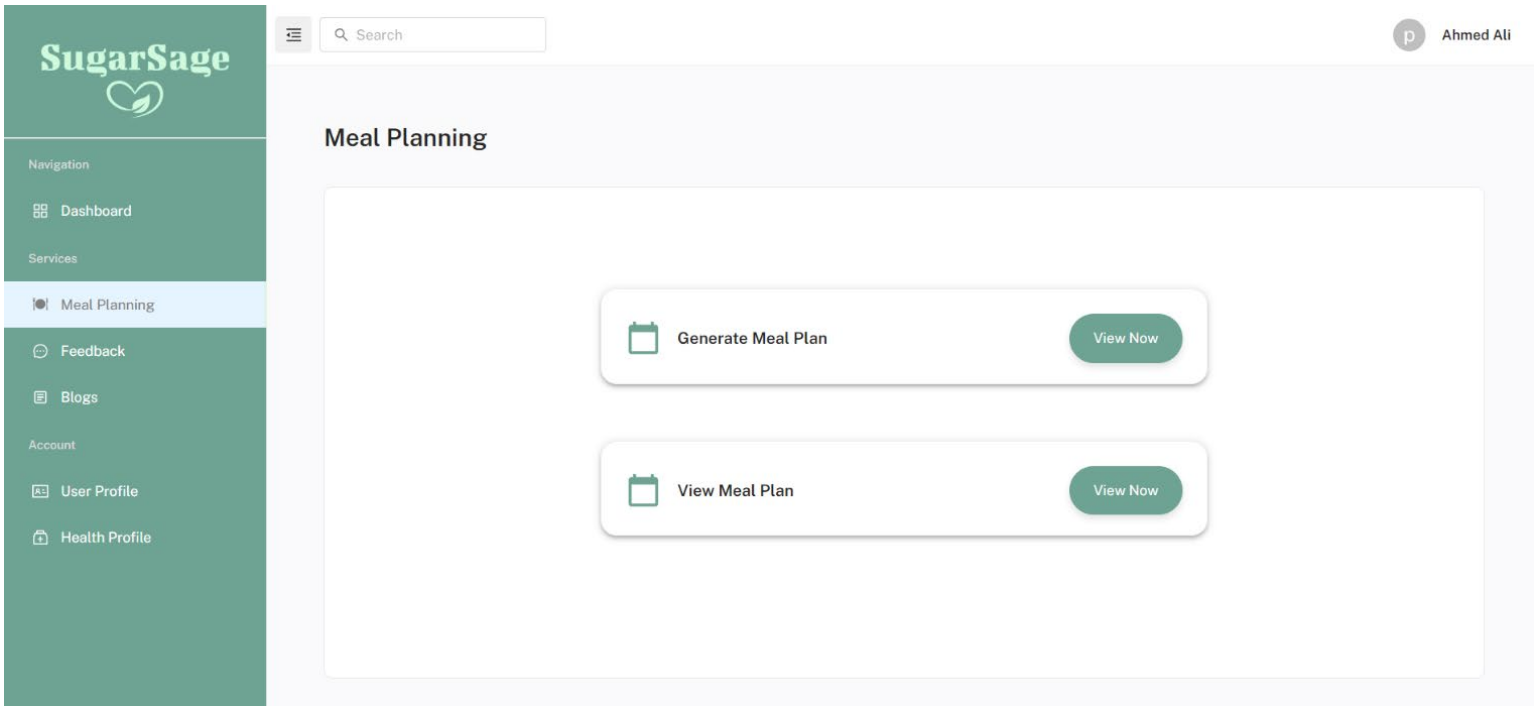

SugarSage

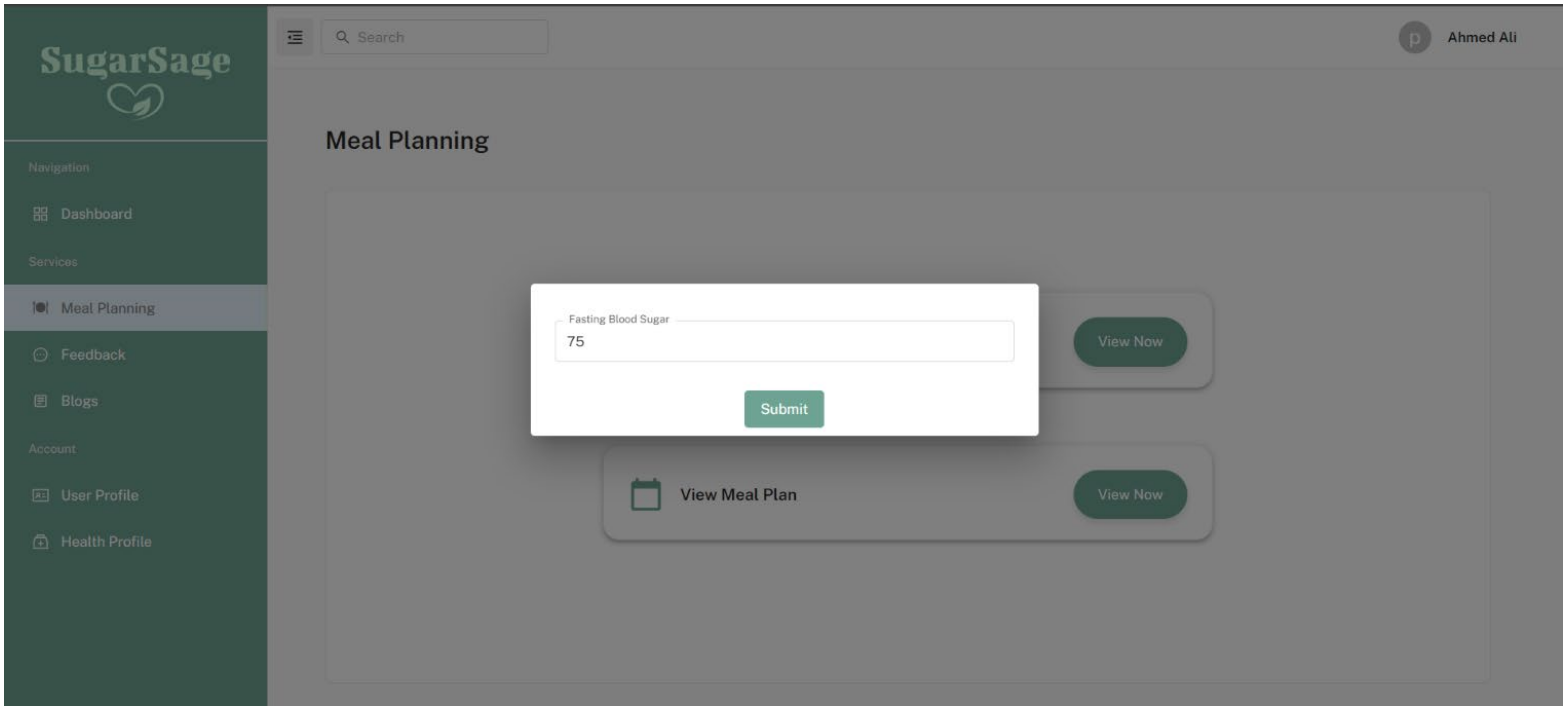

Enter Profile Details
First Name *
Last Name *
Date of Birth *
Gender *
Country *
City *
SUBMIT

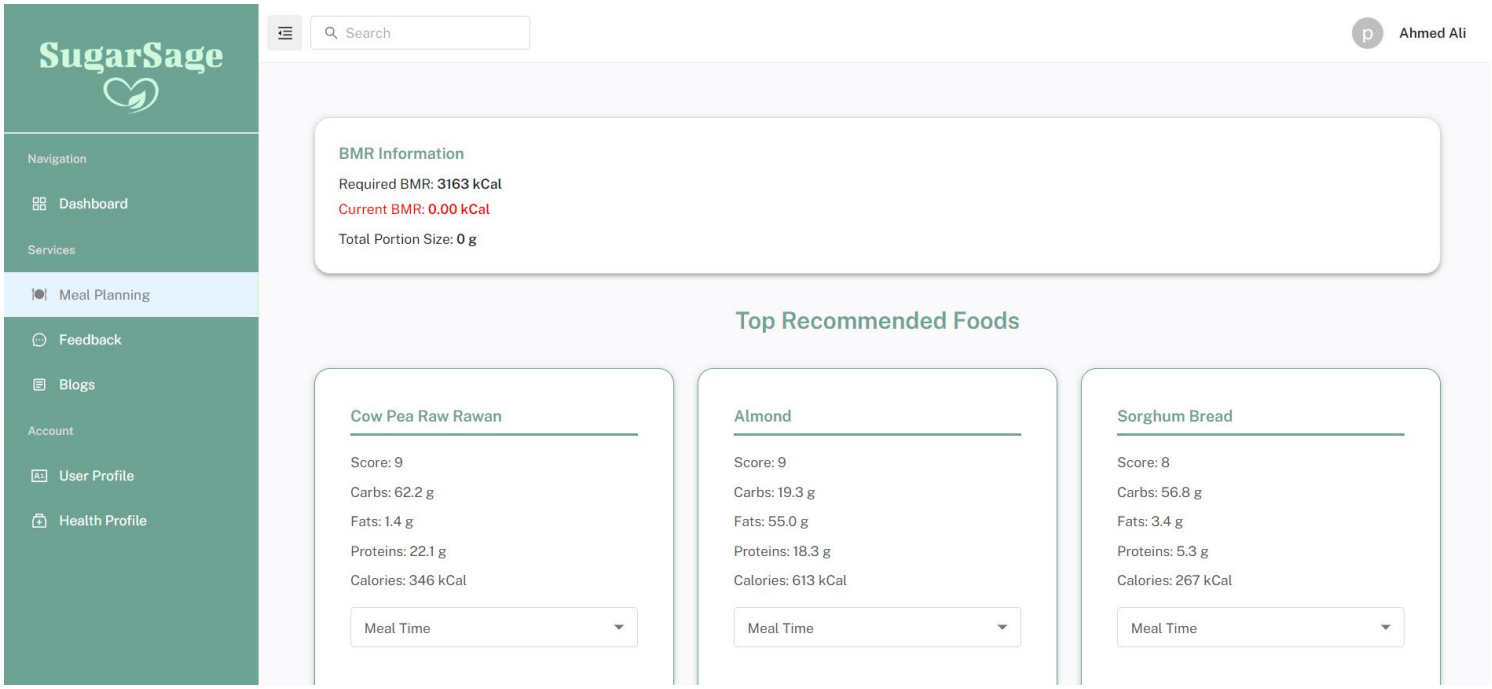

SugarSage

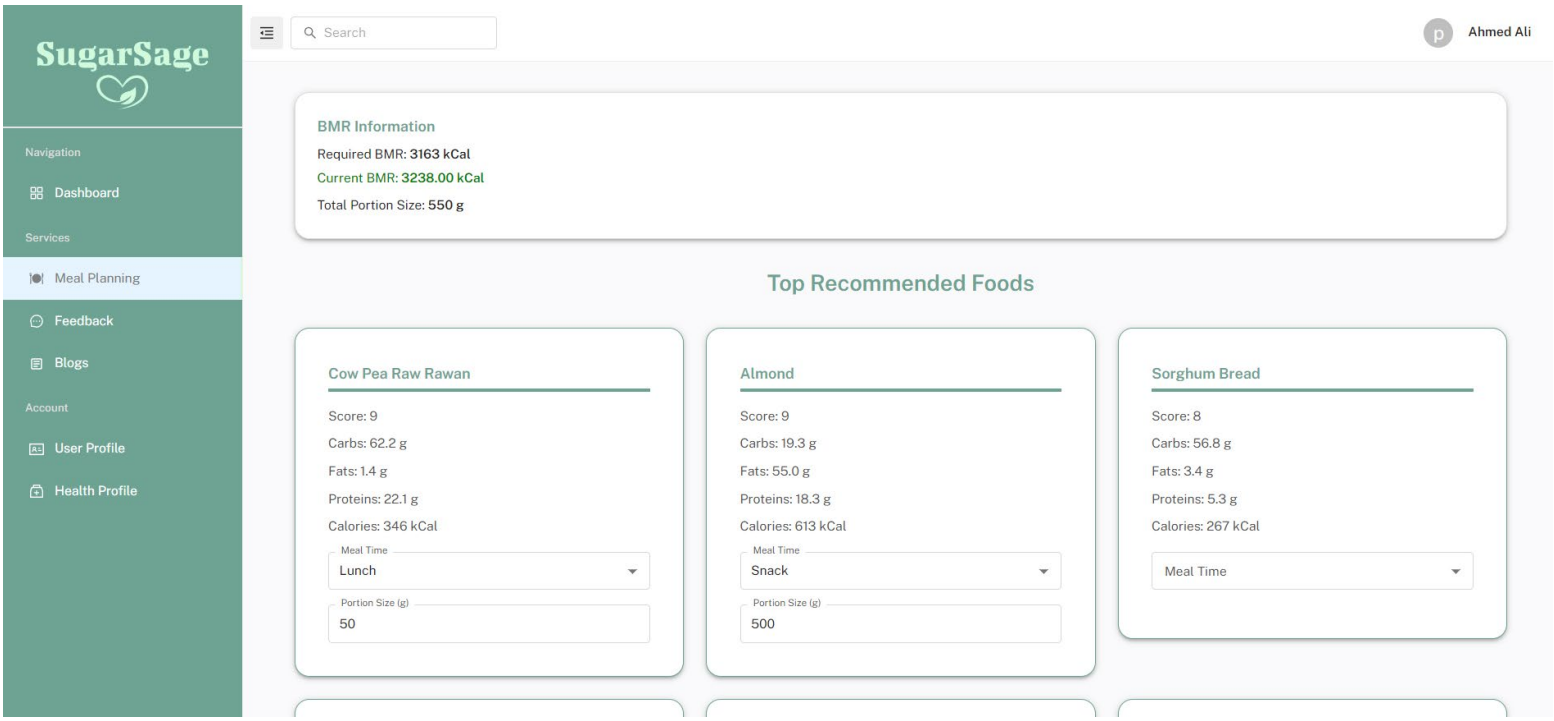

Enter Health Form
Weight (kg) *
5
Weight must be between 45 and 150 kg.
Height (m) *
4
Height must be between 1.5 and 2 meters.
Hb A1c Score (%) *
HbA1c score must be between 4.0% and 8.9%.
Activity Level *
Moderately Active
Diabetes Type *
Type-1
SUBMIT

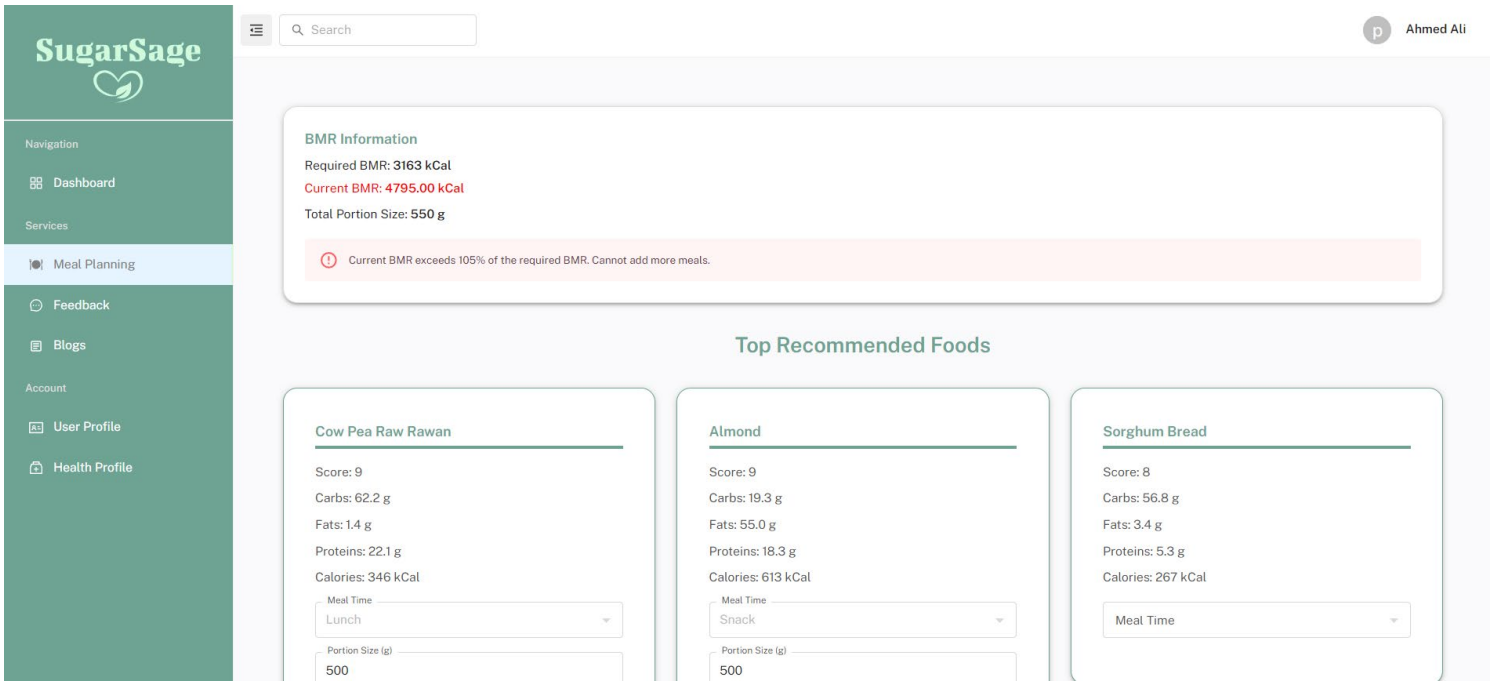

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Health Tracker
Activity Tracker
Feedback
Blogs
Account
User Profile
Health Profile
Welcome Ali,
Calories Track
18%
500/2800 Cals
Steps
95%
9500/10000 Steps
Insulin Dosage Tracker
2024-01-03
2024-01-06
insulin
glucose
Ideal Weight
January
February
March
April
May
June
July
August
September
October
November
December

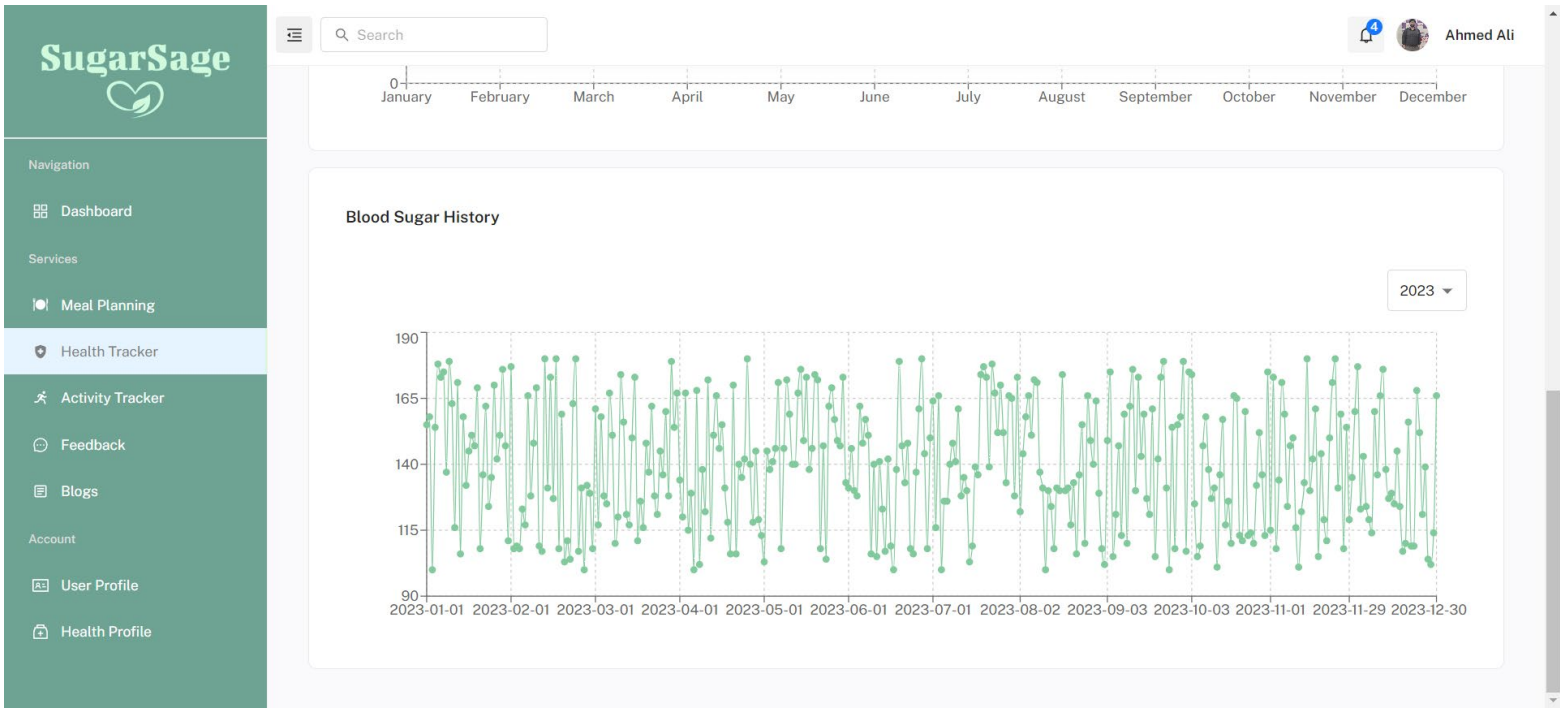

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Feedback
Blogs
Account
User Profile
Health Profile
Meal Planning
Generate Meal Plan
View Now
View Meal Plan
View Now

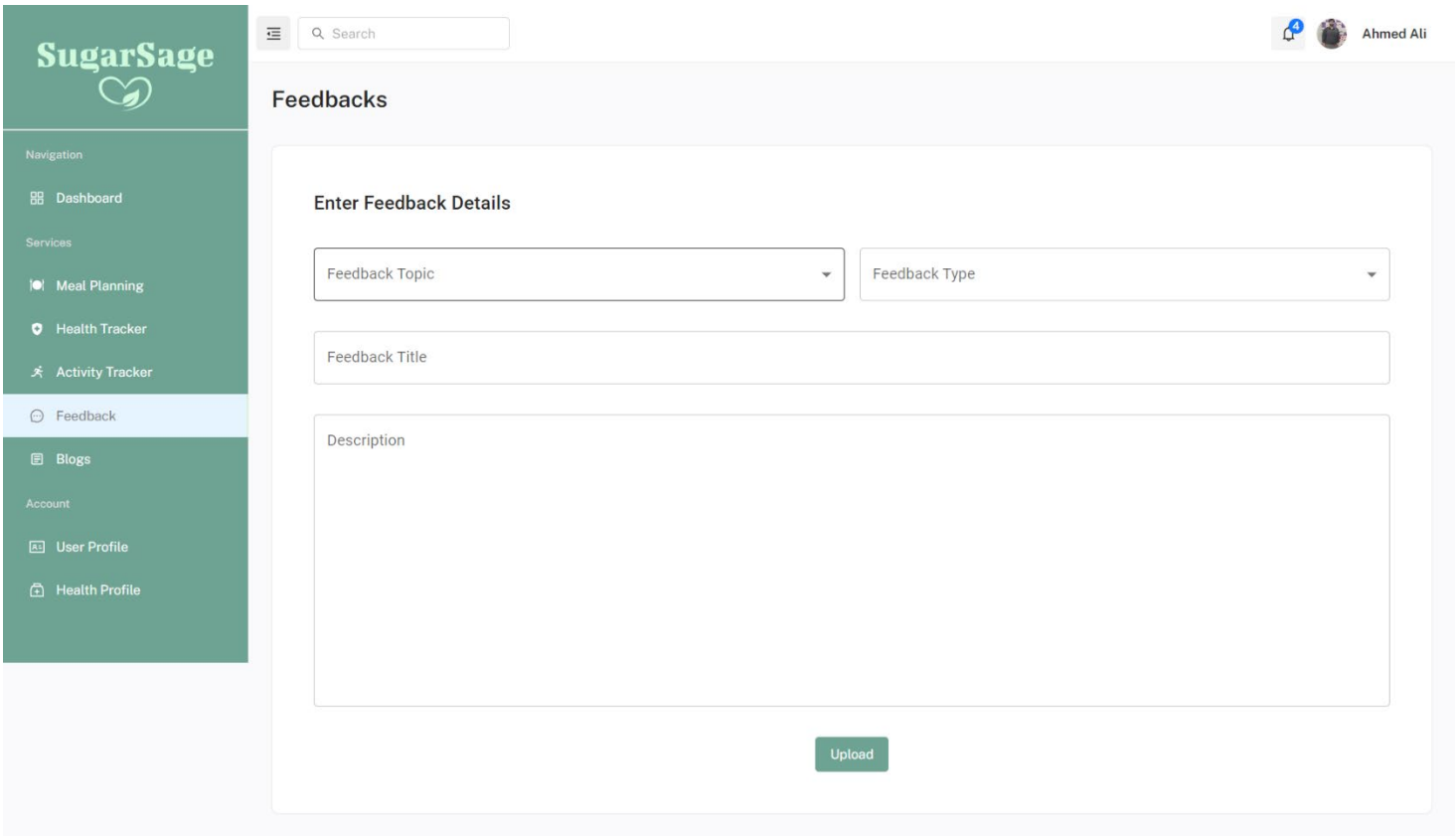

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Feedback
Blogs
Account
User Profile
Health Profile
Meal Planning
Fasting Blood Sugar
75
Submit
View Now
View Meal Plan
View Now

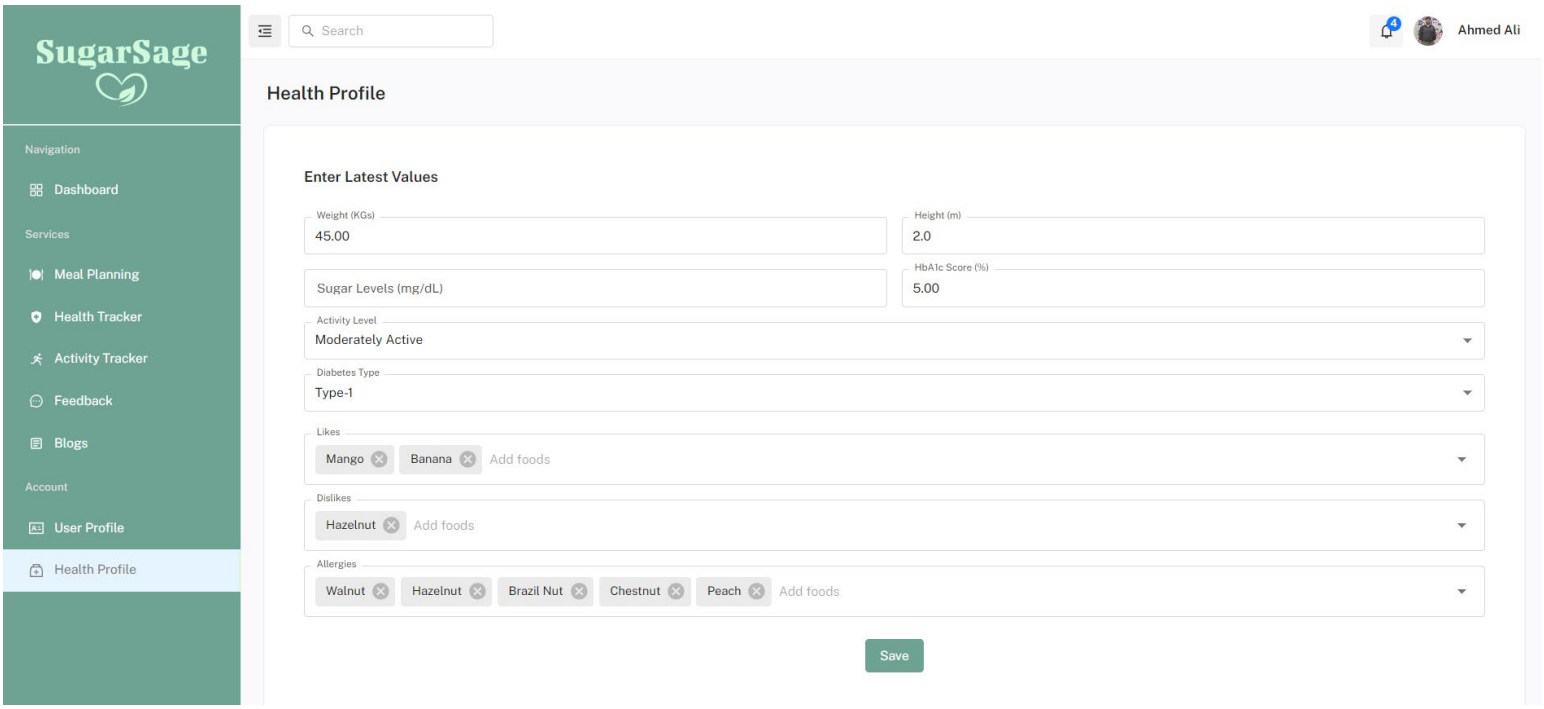

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Feedback
Blogs
Account
User Profile
Health Profile
BMR Information
Required BMR: 3163 kCal
Current BMR: 0.00 kCal
Total Portion Size: 0 g
Top Recommended Foods
Cow Pea Raw Rawan
Score: 9
Carbs: 62.2 g
Fats: 1.4 g
Proteins: 22.1 g
Calories: 346 kCal
Meal Time
Almond
Score: 9
Carbs: 19.3 g
Fats: 55.0 g
Proteins: 18.3 g
Calories: 613 kCal
Meal Time
Sorghum Bread
Score: 8
Carbs: 56.8 g
Fats: 3.4 g
Proteins: 5.3 g
Calories: 267 kCal
Meal Time

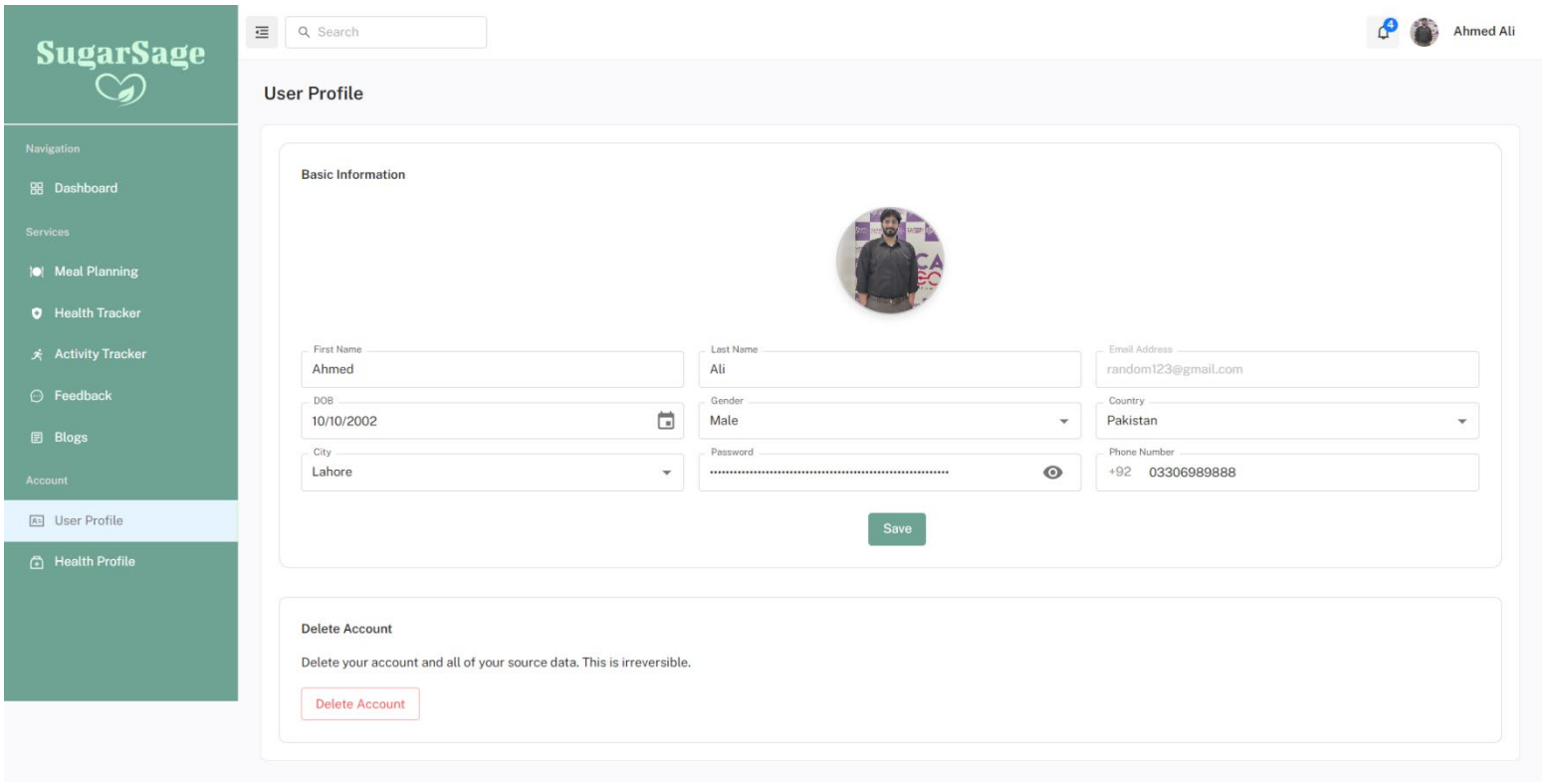

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Feedback
Blogs
Account
User Profile
Health Profile
BMR Information
Required BMR: 3163 kCal
Current BMR: 3238.00 kCal
Total Portion Size: 550 g
Top Recommended Foods
Cow Pea Raw Rawan
Score: 9
Carbs: 62.2 g
Fats: 1.4 g
Proteins: 22.1 g
Calories: 346 kCal
Meal Time
Lunch
Portion Size (g)
50
Almond
Score: 9
Carbs: 19.3 g
Fats: 55.0 g
Proteins: 18.3 g
Calories: 613 kCal
Meal Time
Snack
Portion Size (g)
500
Sorghum Bread
Score: 8
Carbs: 56.8 g
Fats: 3.4 g
Proteins: 5.3 g
Calories: 267 kCal
Meal Time

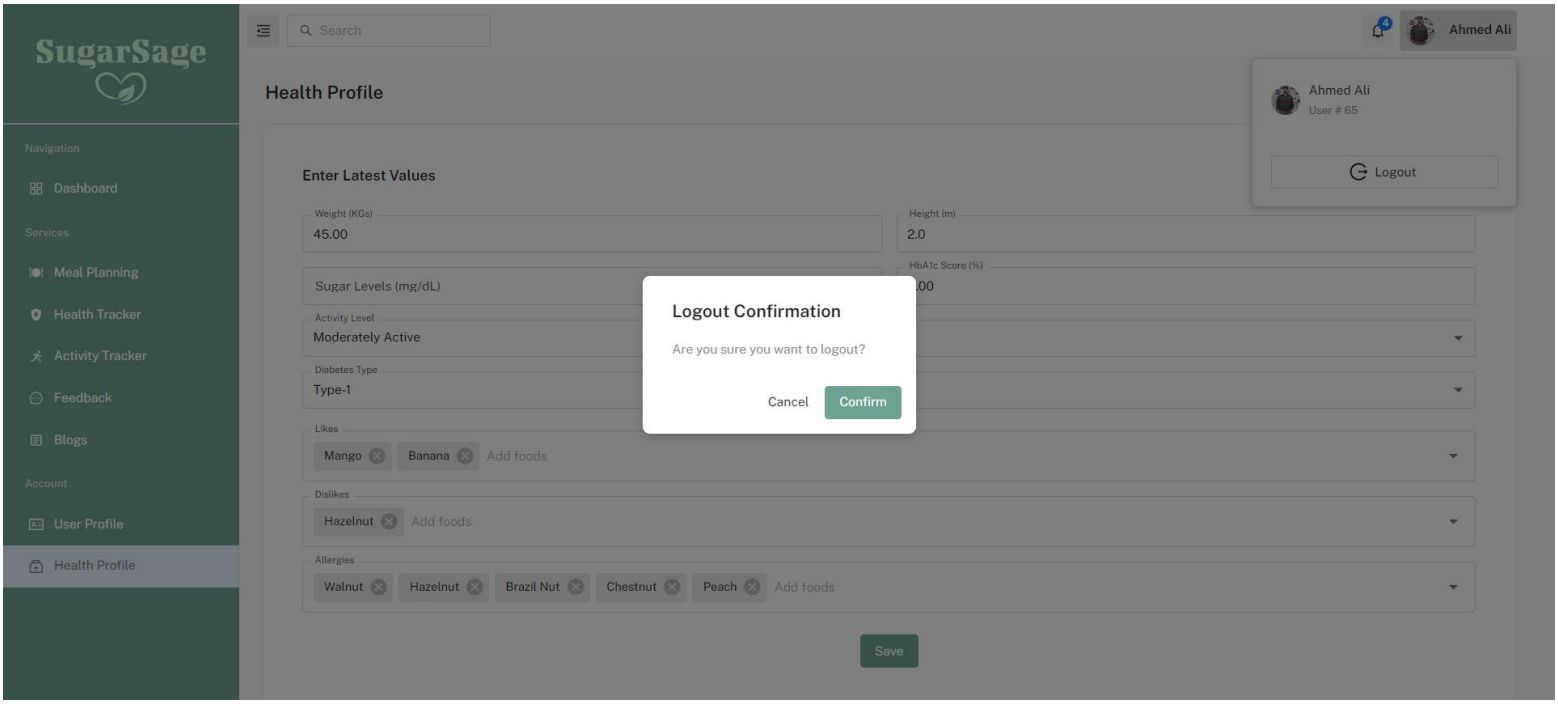

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Feedback
Blogs
Account
User Profile
Health Profile
BMR Information
Required BMR: 3163 kCal
Current BMR: 4795.00 kCal
Total Portion Size: 550 g
Current BMR exceeds 105% of the required BMR. Cannot add more meals.
Top Recommended Foods
Cow Pea Raw Rawan
Score: 9
Carbs: 62.2 g
Fats: 1.4 g
Proteins: 22.1 g
Calories: 346 kCal
Meal Time
Lunch
Portion Size (g)
500
Almond
Score: 9
Carbs: 19.3 g
Fats: 55.0 g
Proteins: 18.3 g
Calories: 613 kCal
Meal Time
Snack
Portion Size (g)
500
Sorghum Bread
Score: 8
Carbs: 56.8 g
Fats: 3.4 g
Proteins: 5.3 g
Calories: 267 kCal
Meal Time

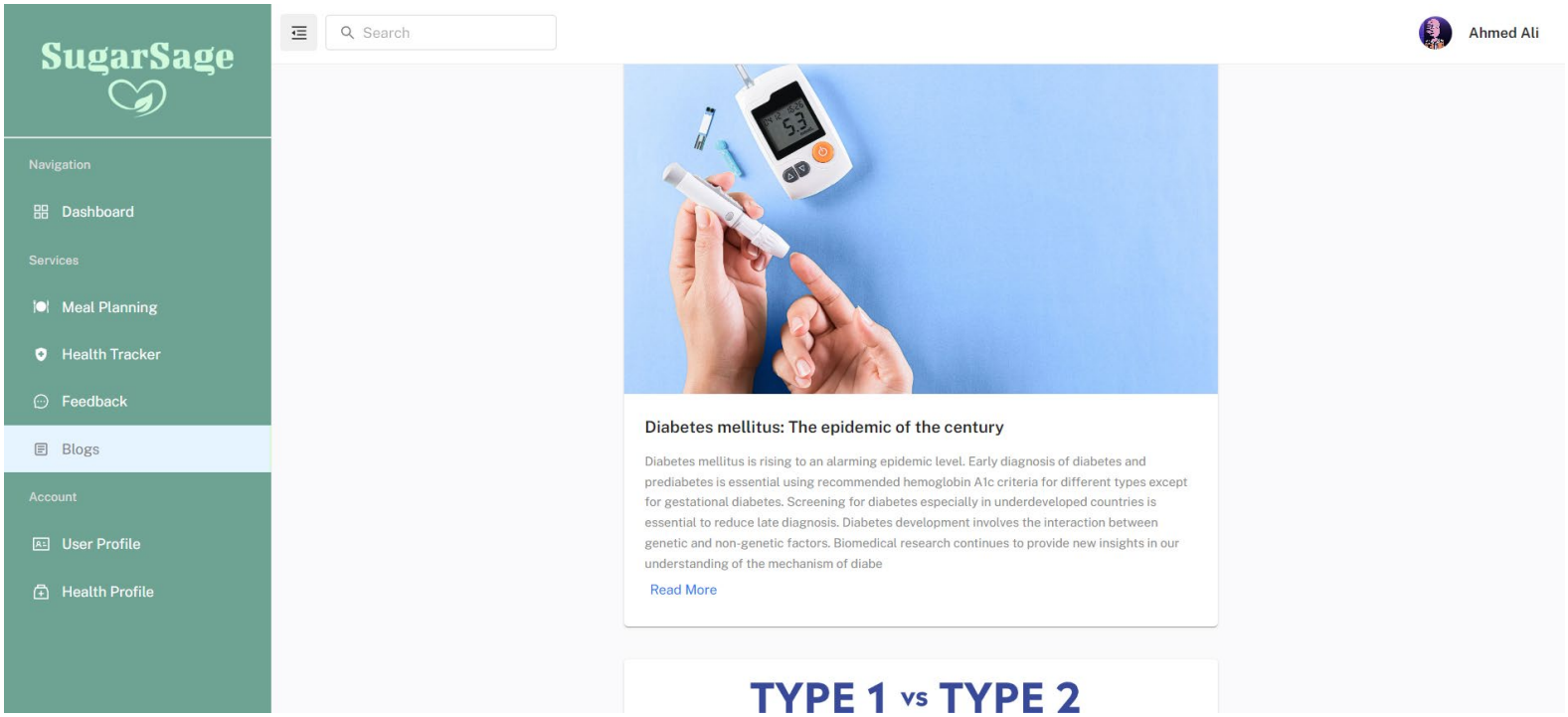

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Health Tracker
Activity Tracker
Feedback
Blogs
Account
User Profile
Health Profile
0
January
February
March
April
May
June
July
August
September
October
November
December
Blood Sugar History
2023
190
165
140
115
90
2023-01-01
2023-02-01
2023-03-01
2023-04-01
2023-05-01
2023-06-01
2023-07-01
2023-08-02
2023-09-03
2023-10-03
2023-11-01
2023-11-29
2023-12-30


SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Health Tracker
Activity Tracker
Feedback
Blogs
Account
User Profile
Health Profile
Feedbacks
Enter Feedback Details
Feedback Topic
Feedback Type
Feedback Title
Description
Upload

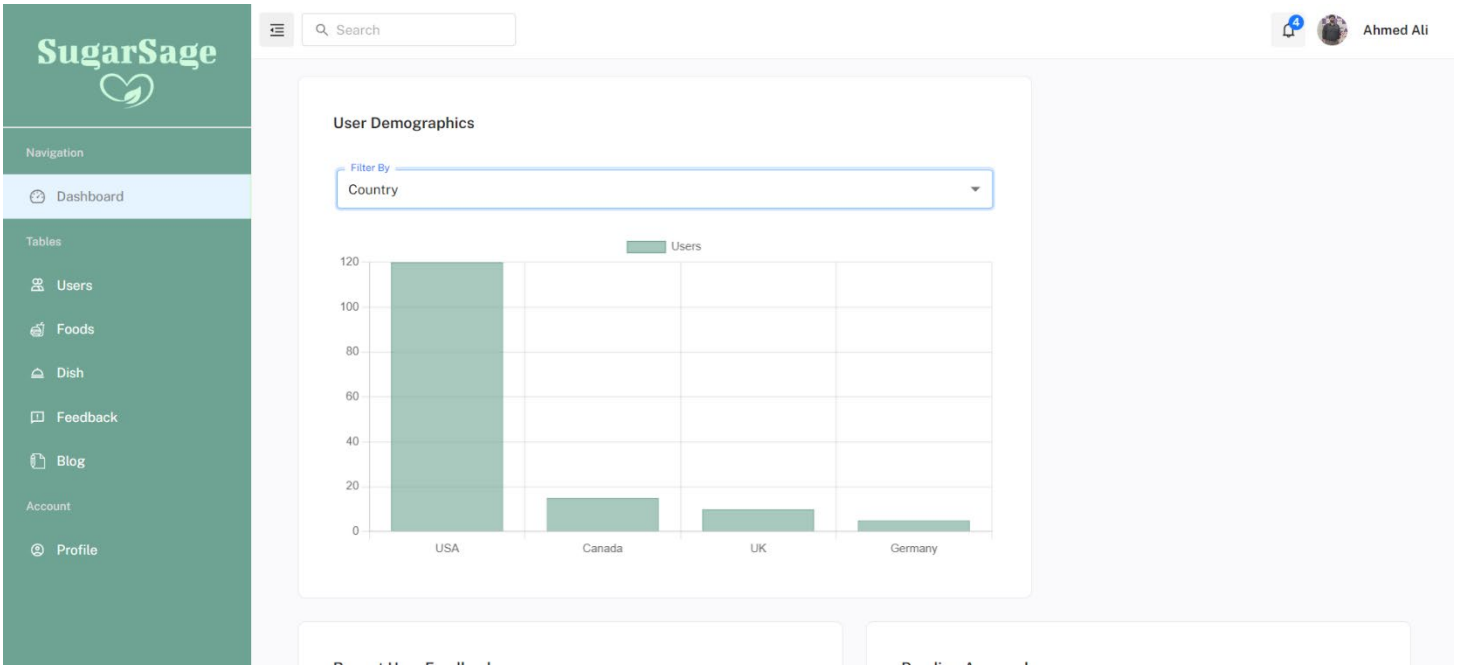

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Services
Meal Planning
Health Tracker
Activity Tracker
Feedback
Blogs
Account
User Profile
Health Profile
Health Profile
Enter Latest Values
Weight (KGs)
45.00
Height (m)
2.0
Sugar Levels (mg/dL)
HbA1c Score (%)
5.00
Activity Level
Moderately Active
Diabetes Type
Type-1
Likes
Mango
Banana
Add foods
Dislikes
Hazelnut
Add foods
Allergies
Walnut
Hazelnut
Brazil Nut
Chestnut
Peach
Add foods
Save

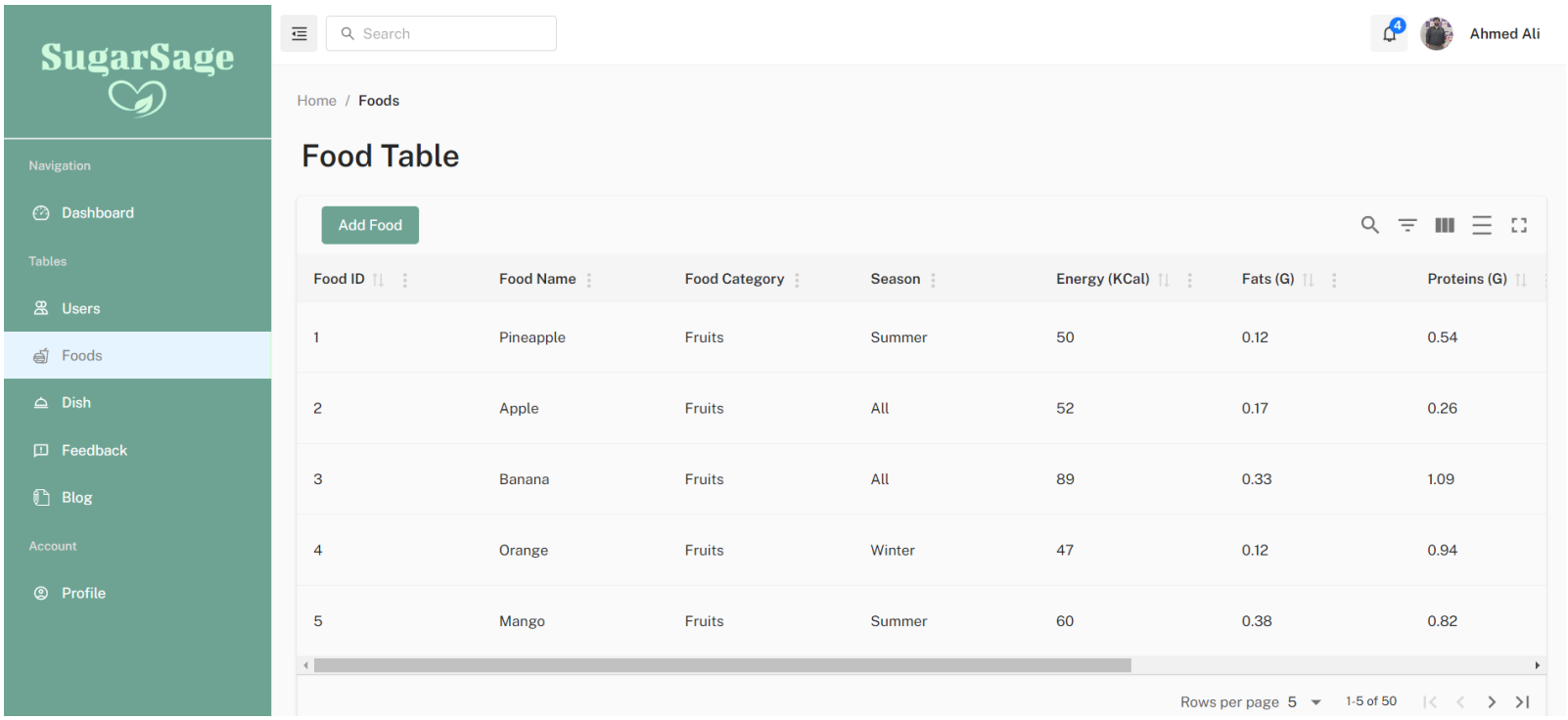

SugarSage
Search
Ahmed Ali
User Profile
Navigation
Dashboard
Services
Meal Planning
Health Tracker
Activity Tracker
Feedback
Blogs
Account
User Profile
Health Profile
Basic Information
First Name
Ahmed
Last Name
Ali
Email Address
random123@gmail.com
DOB
10/10/2002
Gender
Male
Country
Pakistan
City
Lahore
Password
Phone Number
+92 03306989888
Save
Delete Account
Delete your account and all of your source data. This is irreversible.
Delete Account

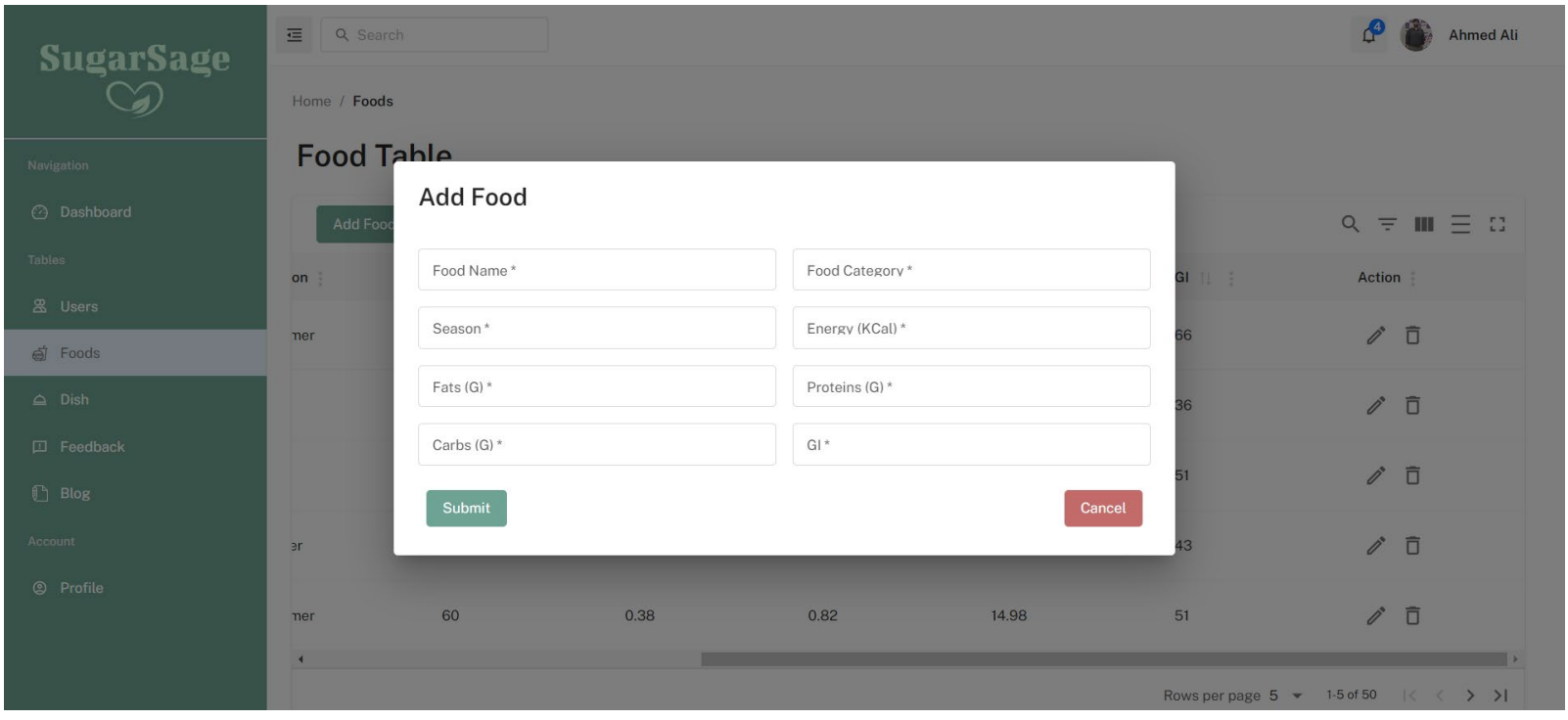

SugarSage
Search
Ahmed Ali
Health Profile
Ahmed Ali
User # 65
Logout
Enter Latest Values
Weight (KGs)
45.00
Height (m)
2.0
Sugar Levels (mg/dL)
HbA1c Score (%)
Activity Level
Moderately Active
Diabetes Type
Type-1
Likes
Mango
Banana
Add foods
Dislikes
Hazelnut
Add foods
Allergies
Walnut
Hazelnut
Brazil Nut
Chestnut
Peach
Add foods
Save
Logout Confirmation
Are you sure you want to logout?
Cancel
Confirm

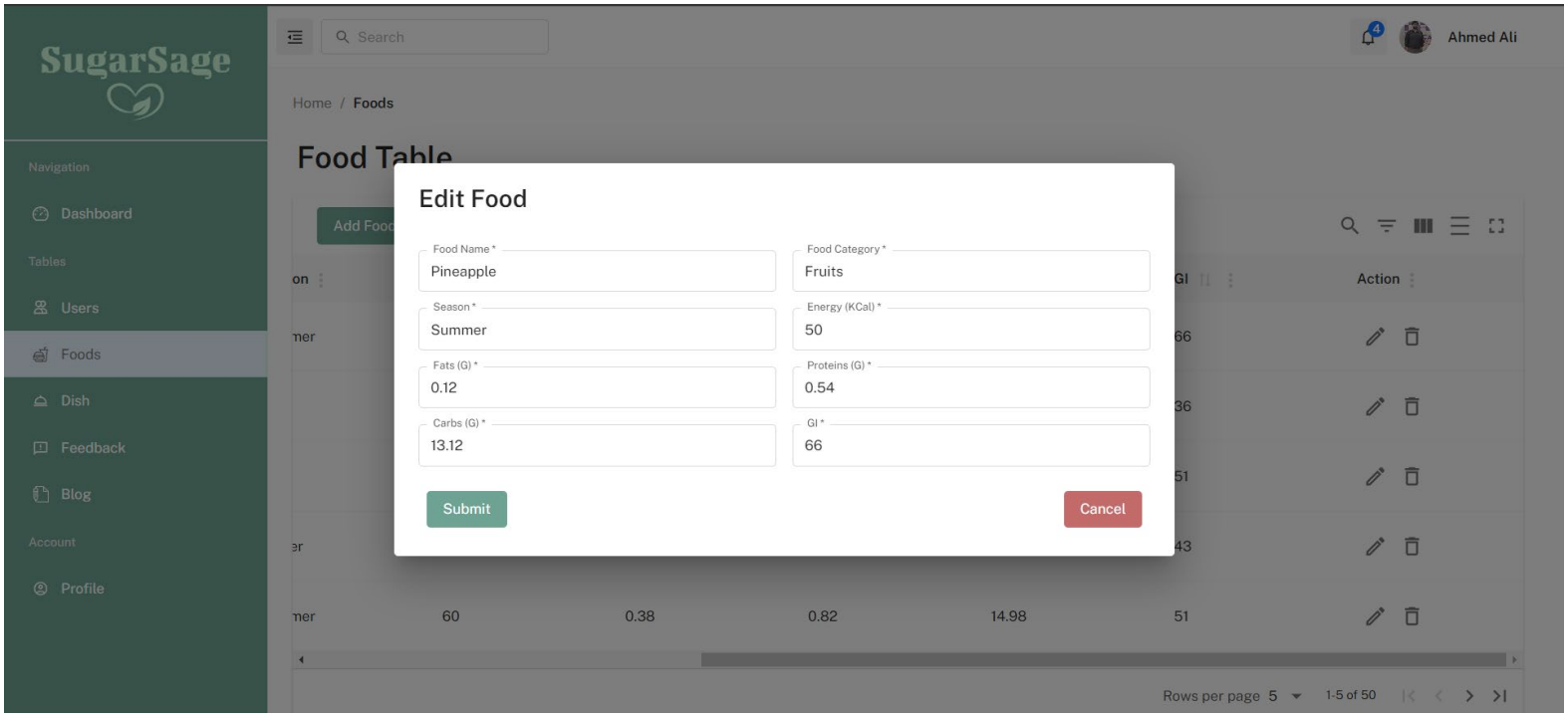


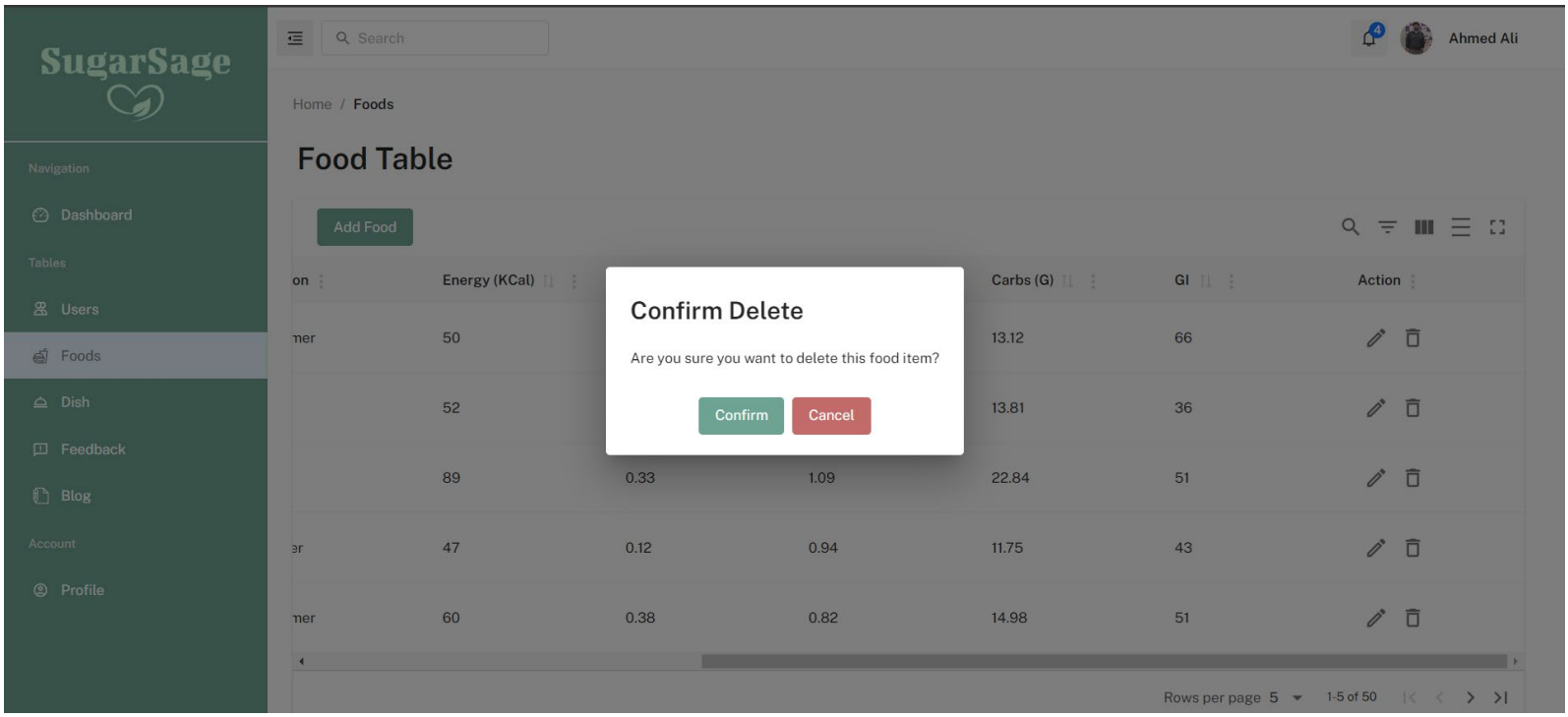

### 5.4.2.3 Admin Web App Screen:

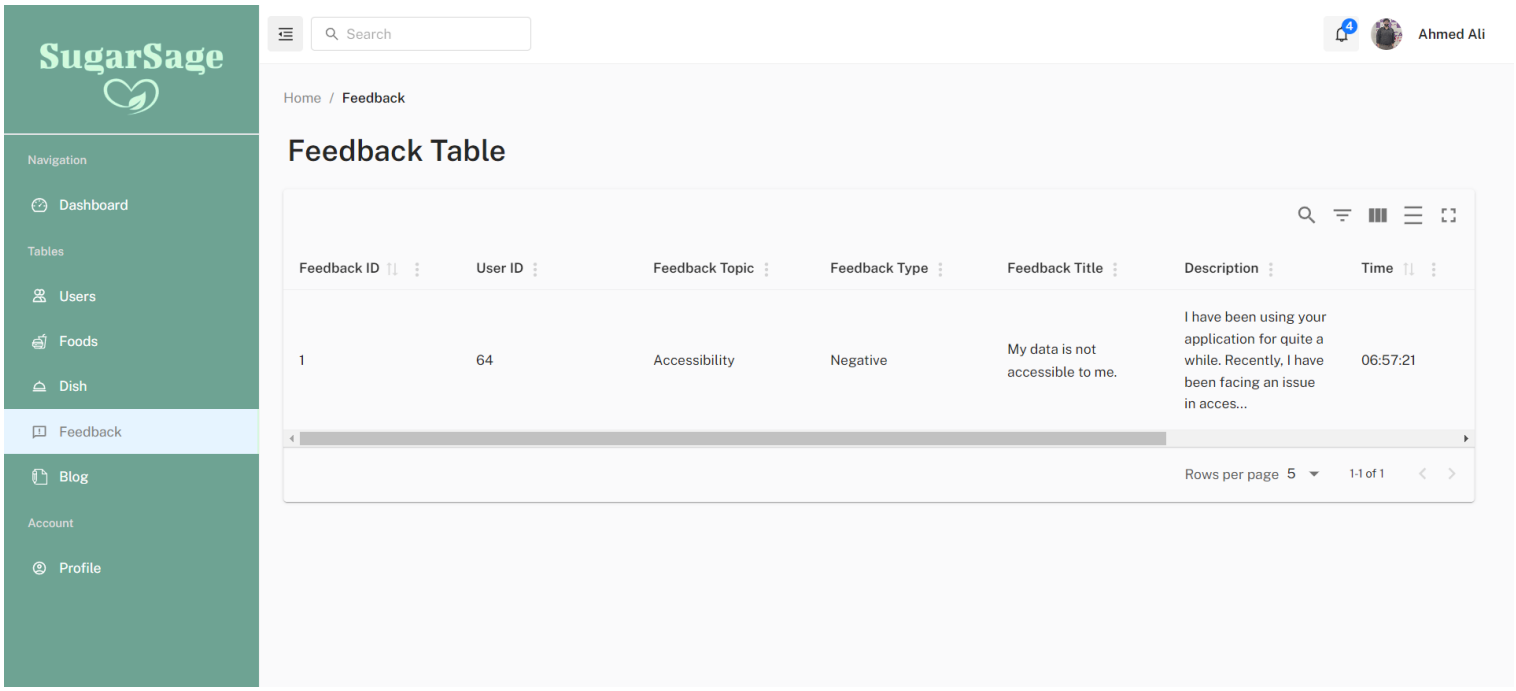


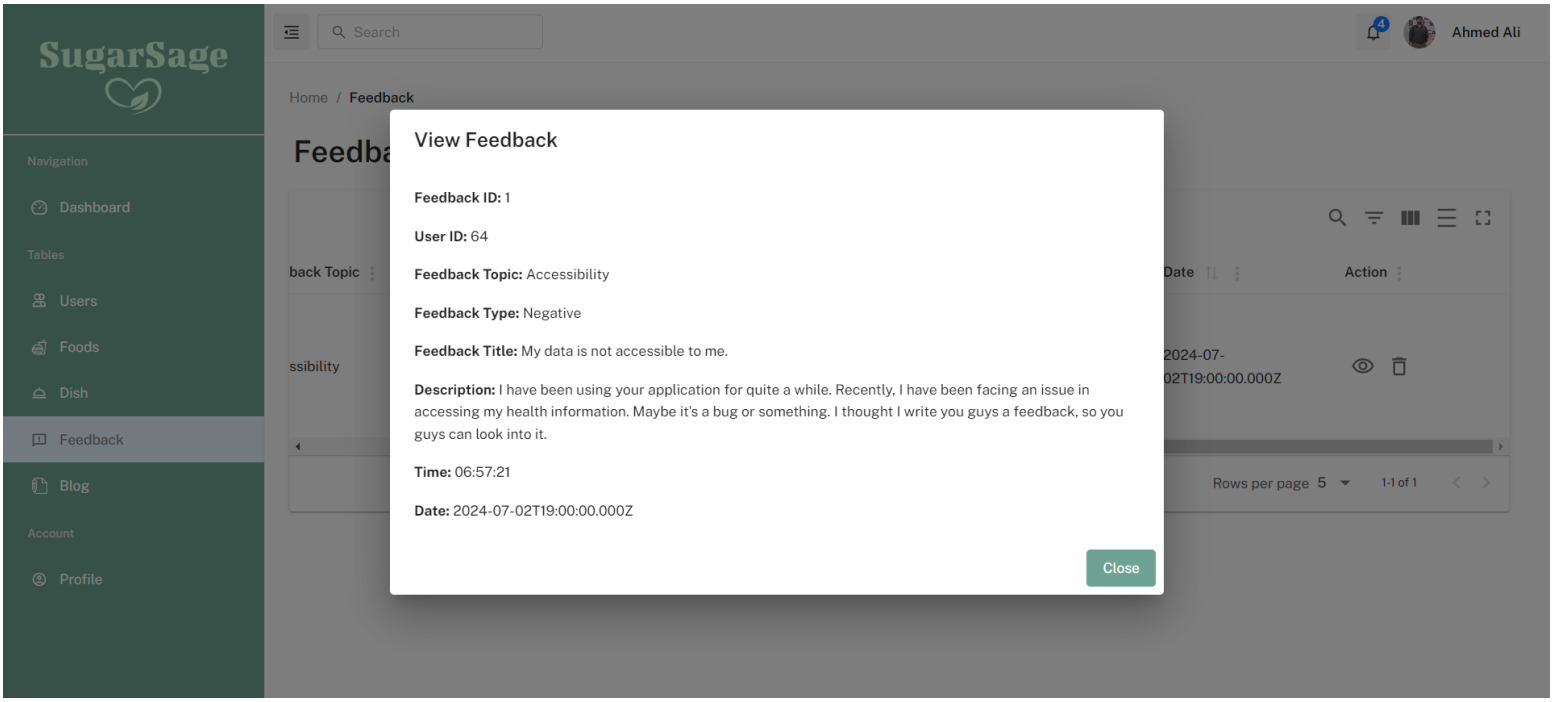

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Home / Foods
Food Table
Add Food
Food ID
Food Name
Food Category
Season
Energy (KCal)
Fats (G)
Proteins (G)
1 Pineapple Fruits Summer 50 0.12 0.54
2 Apple Fruits All 52 0.17 0.26
3 Banana Fruits All 89 0.33 1.09
4 Orange Fruits Winter 47 0.12 0.94
5 Mango Fruits Summer 60 0.38 0.82
Rows per page 5
1-5 of 50

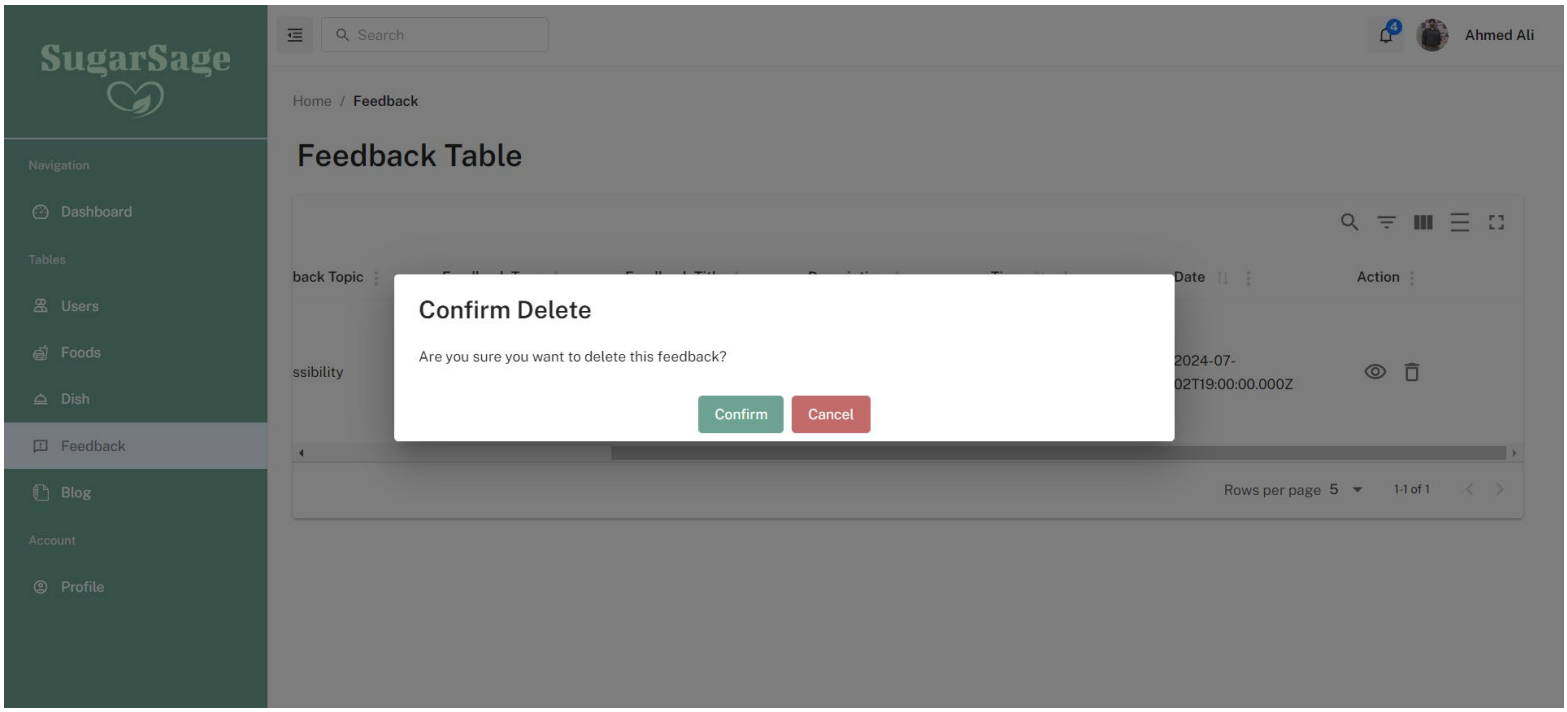

SugarSage
Search
Ahmed Ali
Home / Foods
Food Table
Add Food
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Food Name *
Food Category *
Season *
Energy (KCal) *
Fats (G) *
Proteins (G) *
Carbs (G) *
GI *
Submit
Cancel
Action
Rows per page 5
1-5 of 50

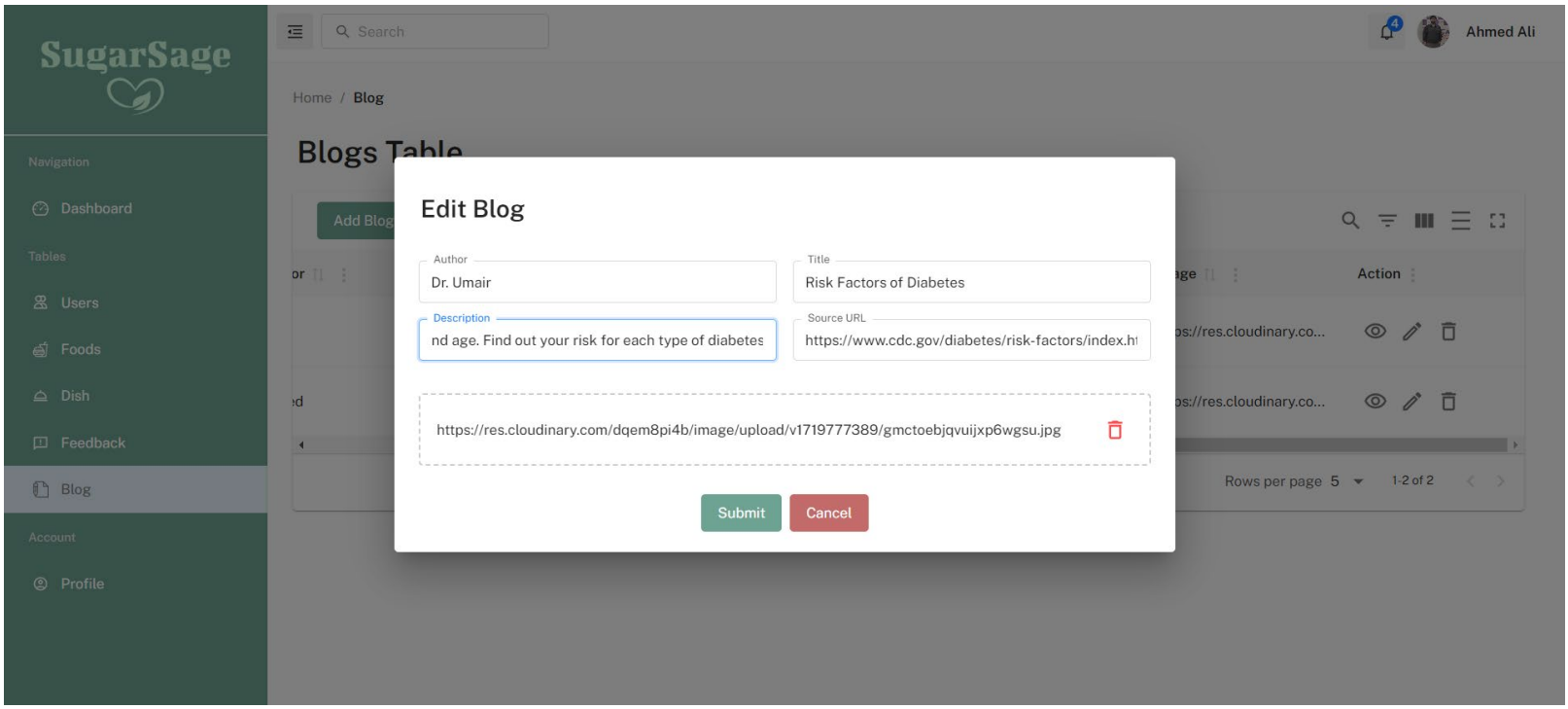

SugarSage
Search
Ahmed Ali
Home / Foods
Food Table
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Edit Food
Food Name *
Pineapple
Food Category *
Fruits
Season *
Summer
Energy (KCal) *
50
Fats (G) *
0.12
Proteins (G) *
0.54
Carbs (G) *
13.12
GI *
66
Submit
Cancel
Action
60
0.38
0.82
14.98
51
Rows per page 5
1-5 of 50

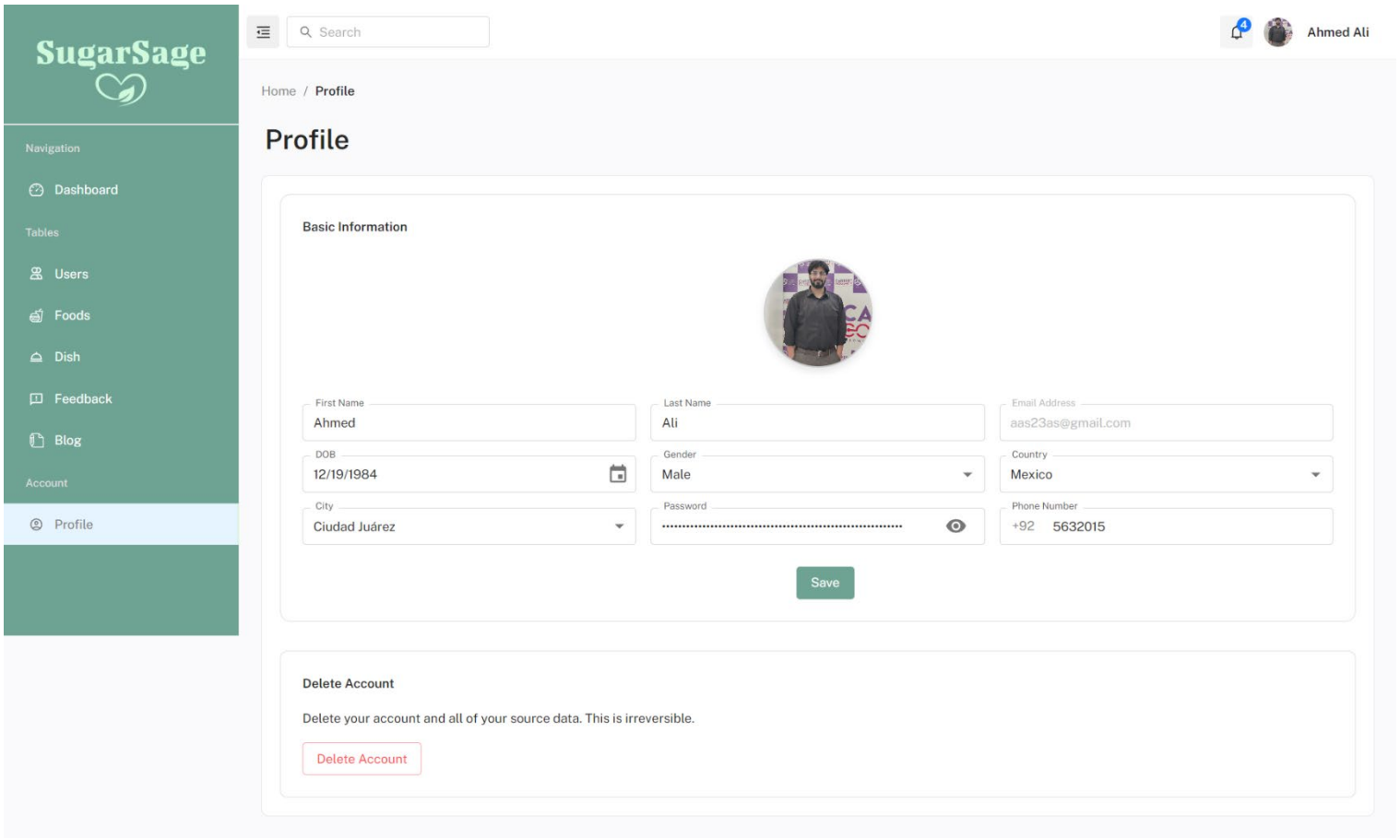

SugarSage
Search
Ahmed Ali
Home / Foods
Food Table
Add Food
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Energy (KCal)
Carbs (G)
GI
Action
Confirm Delete
Are you sure you want to delete this food item?
Confirm
Cancel
50
13.12
66
52
13.81
36
89
0.33
1.09
22.84
51
47
0.12
0.94
11.75
43
60
0.38
0.82
14.98
51
Rows per page 5
1-5 of 50

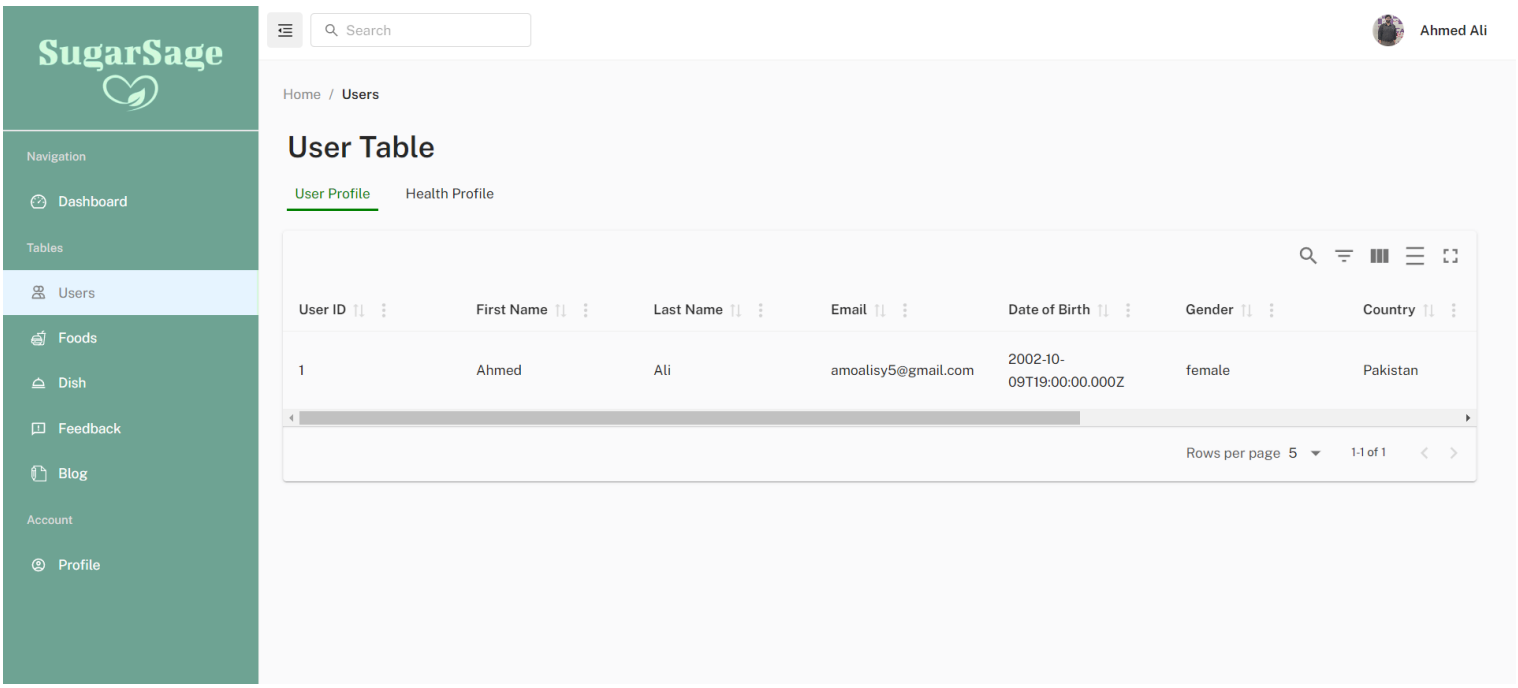

SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Home / Feedback
Feedback Table
Feedback ID
User ID
Feedback Topic
Feedback Type
Feedback Title
Description
Time
1
64
Accessibility
Negative
My data is not accessible to me.
I have been using your application for quite a while. Recently, I have been facing an issue in acces...
06:57:21
Rows per page 5
1-1 of 1


SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Home / Feedback
View Feedback
Feedback ID: 1
User ID: 64
Feedback Topic: Accessibility
Feedback Type: Negative
Feedback Title: My data is not accessible to me.
Description: I have been using your application for quite a while. Recently, I have been facing an issue in accessing my health information. Maybe it's a bug or something. I thought I write you guys a feedback, so you guys can look into it.
Time: 06:57:21
Date: 2024-07-02T19:00:00.000Z
Close


SugarSage
Search
Ahmed Ali
Home / Feedback
Feedback Table
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
back Topic
Date
Action
Confirm Delete
Are you sure you want to delete this feedback?
Confirm
Cancel
ssibility
2024-07-
02T19:00:00.000Z
Rows per page 5
1-1 of 1


SugarSage
Search
Ahmed Ali
Home / Blog
Blogs Table
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Add Blog
Action
Edit Blog
Author
Dr. Umair
Title
Risk Factors of Diabetes
Description
nd age. Find out your risk for each type of diabetes
Source URL
https://www.cdc.gov/diabetes/risk-factors/index.ht
https://res.cloudinary.com/dqem8pi4b/image/upload/v1719777389/gmctoebjqvuijxp6wgsu.jpg
Submit
Cancel
Rows per page 5
1-2 of 2


SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Home / Profile
Profile
Basic Information
First Name
Ahmed
Last Name
Ali
Email Address
aas23as@gmail.com
DOB
12/19/1984
Gender
Male
Country
Mexico
City
Ciudad Juárez
Password
Phone Number
+92 5632015
Save
Delete Account
Delete your account and all of your source data. This is irreversible.
Delete Account


SugarSage
Search
Ahmed Ali
Navigation
Dashboard
Tables
Users
Foods
Dish
Feedback
Blog
Account
Profile
Home / Users
User Table
User Profile
Health Profile
User ID
First Name
Last Name
Email
Date of Birth
Gender
Country
1
Ahmed
Ali
amoalisy5@gmail.com
2002-10-09T19:00:00.000Z
female
Pakistan
Rows per page 5
1-1 of 1

# Chapter 6. Machine Learning Model 01

## 6.1 Dataset

### 6.1.1 Overview

The dataset utilized for our machine learning model comprises 12 features and 3 target variables to predict the percentage distribution of carbohydrates, fats, and proteins in an individual's diet. The features encompass a broad spectrum of biometric and lifestyle characteristics influencing nutritional needs. Here's a snapshot of the dataset structure:

#### *6.1.1.1 Features*

| Age | Gender | Height (m) | Weight (kg) | BMI | HbA1c Level | Blood Type | Resting Heart Rate (bpm) | Water Intake (liters/day) | Sleep (hours) | Activity Level | BMR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | Female | 1.5 | 51 | 22.7 | 4.5 | AB+ | 63 | 1 | 6 | Lightly Active | 1640 |
| 31 | Male | 1.57 | 45 | 18.3 | 5.3 | B+ | 68 | 1 | 9 | Moderately Active | 2060 |
| 19 | Male | 1.76 | 70 | 22.6 | 5.3 | O- | 72 | 2 | 6 | Sedentary | 2125 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

#### *6.1.1.2 Targets*

| Carb (%) | Fats (%) | Protein (%) |
|---|---|---|
| 46 | 32 | 22 |
| 54 | 20 | 26 |
| 52 | 25 | 22 |
| ... | ... | ... |

### 6.1.2 Feature Details

#### *6.1.2.1 Biometric and lifestyle feature:*

**Age and Gender**: These features account for variations in metabolic rates and nutritional needs across different age groups and between genders. Age influences basal metabolic rate (BMR) and an individual's overall energy requirements. [2], [3], [4]

**Height and Weight**: These measurements are used to calculate Body Mass Index (BMI), a crucial indicator of body fat, and help assess dietary needs.

**BMI (Body Mass Index)**: BMI is calculated as weight in kilograms divided by height in square meters. It provides a standardized measure to classify underweight, normal weight, overweight, and obesity, which are critical for determining appropriate nutritional guidelines. [5]

**HbA1c Level**: This feature reflects the average blood glucose levels over the past two to three months. It is particularly important for understanding carbohydrate metabolism and managing dietary recommendations for individuals with varying glucose tolerance levels. [6]

**Blood Type**: Although not directly linked to nutritional needs, blood type is included for comprehensive health profiling. Certain studies suggest minor dietary variations based on blood type.

**Resting Heart Rate, Water Intake, and Sleep**: These lifestyle factors provide insights into overall health status and energy expenditure. A resting heart rate can indicate cardiovascular fitness, while water intake and sleep duration are essential for maintaining metabolic balance and overall well-being. [7]

**Activity Level**: This feature is crucial for adjusting caloric intake. The activity level is determined based on the Harris-Benedict equation, which helps estimate an individual's Basal Metabolic Rate (BMR) adjusted for physical activity. [8], [9]

**BMR (Basal Metabolic Rate)**: BMR represents the calories required to keep the body functioning at rest. It is a fundamental metric for determining daily calorie needs and is significant in personalizing dietary recommendations. [10]

#### *6.1.2.2 Targets*

**Carb (%):** The percentage of total daily calories that should come from carbohydrates. This target is influenced by activity level, BMI, and HbA1c levels.

**Fats (%):** The percentage of total daily calories that should come from fats. This target is adjusted based on individual metabolic needs and health goals.

**Protein (%):** The percentage of total daily calories that should come from proteins. Protein requirements are influenced by age, gender, activity level, and overall health status.

### 6.1.3 Feature Identification and Categorization

Attributes can be described by corresponding values and categorized as discrete (categorical) or continuous. Discrete features have a finite number of values, while continuous features cover a specific interval or range and can take on infinite values within that range.
The following attributes are gathered from the dataset:

#### *6.1.3.1 Discrete Features*

- **Gender**: Categorical feature with values 'Male' and 'Female.'
- **Blood Type**: Categorical feature with possible blood types such as 'A+,' 'A-,' 'B+,' 'B-,' 'AB+,' 'AB-,' 'O+,' and 'O-.'
- **Activity Level**: Categorical feature with levels 'Sedentary,' 'Lightly Active,' 'Moderately Active,' 'Very Active,' and 'Extra Active.'

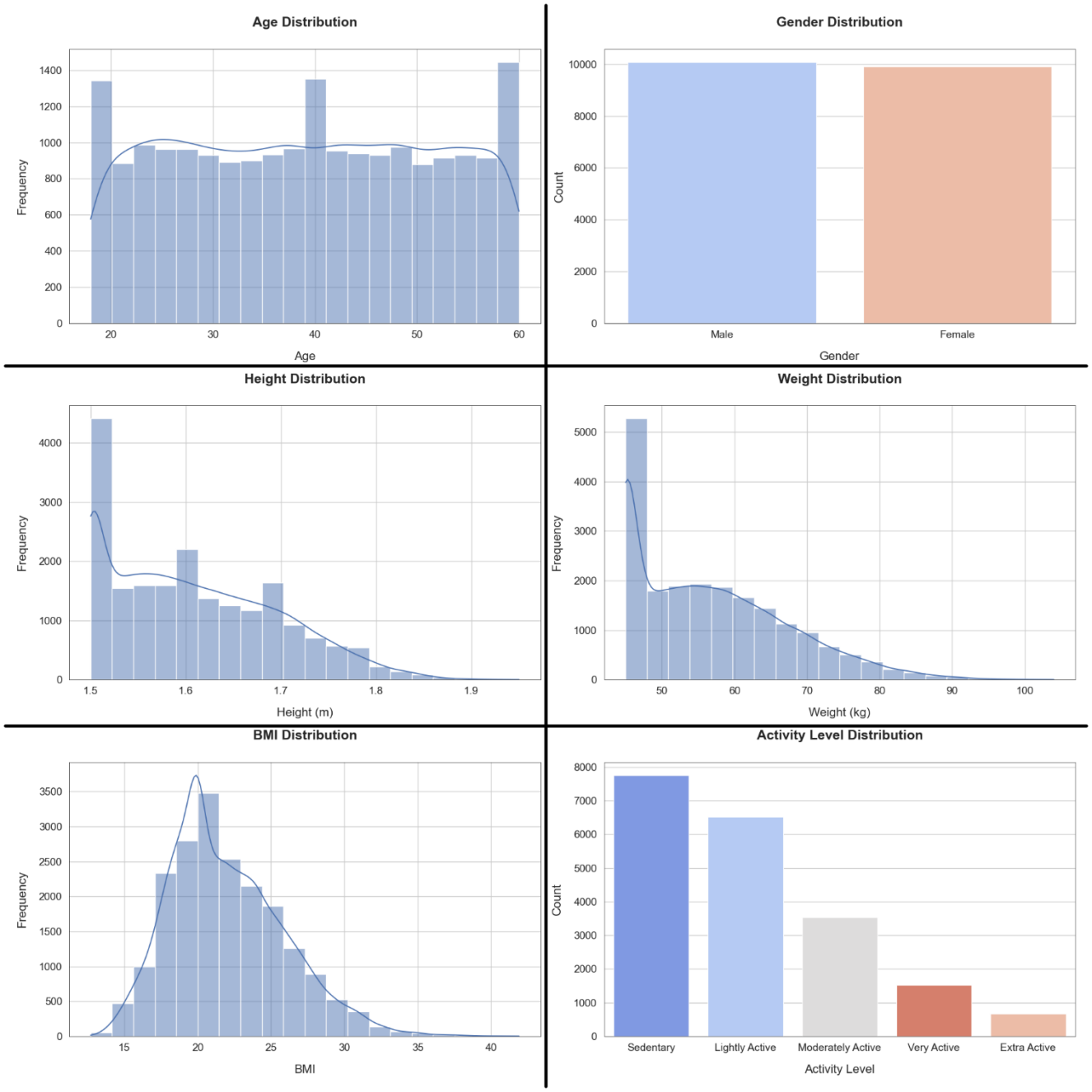
#### *6.1.3.2 Continuous Features*

- **Age**: Numerical features represent the age of the individual in years.

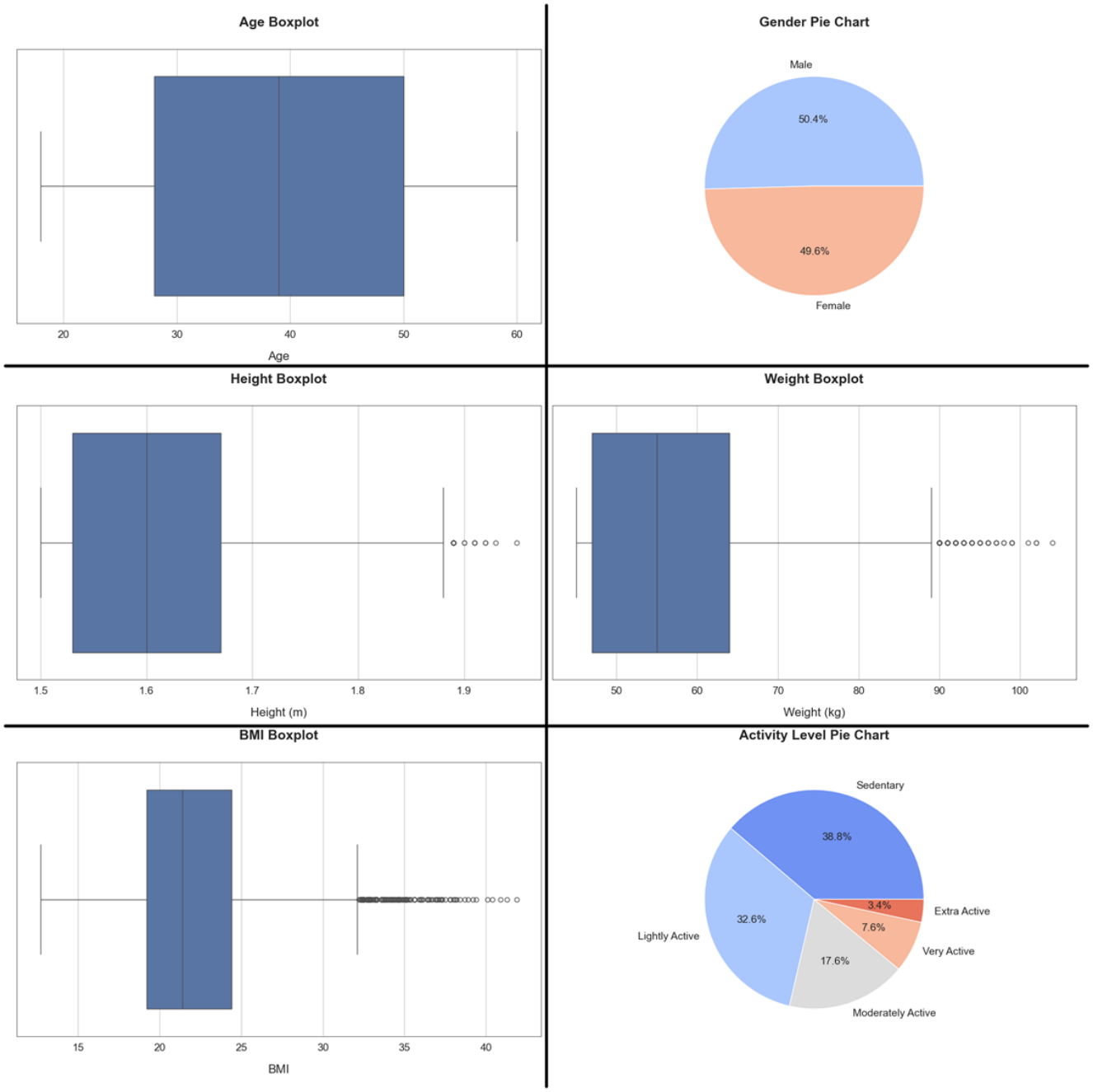
- **Height (m)**: Numerical feature representing the individual's height in meters.
- **Weight (kg)**: Numerical feature representing the individual's weight in kilograms.
- **BMI**: Numerical feature calculated as weight divided by the square of height.
- **HbA1c Level**: Numerical feature indicating the average blood glucose level over the past two to three months.
- **Resting Heart Rate (bpm)**: Numerical feature representing the number of heartbeats per minute while at rest.
- **Water Intake (liters/day)**: Numerical feature representing daily water intake in liters.
- **Sleep (hours)**: Numerical feature representing the number of hours of sleep per night.
- **BMR**: Numerical feature representing the basal metabolic rate in calories.

## 6.1.4 Graphs

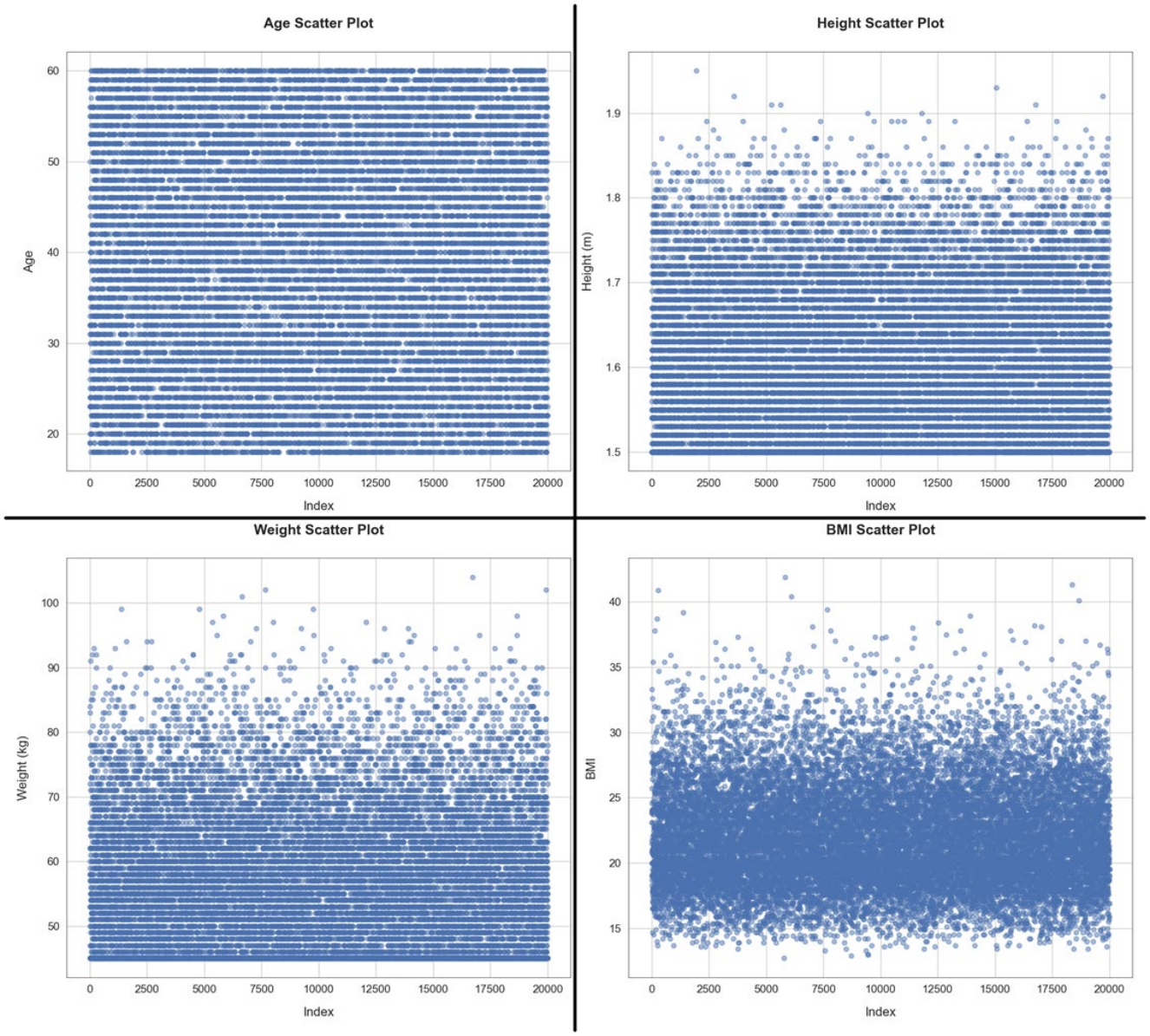
### *6.1.4.1 Distributions using Bar Graph*



### *6.1.4.2 Distributions using Box Plot*

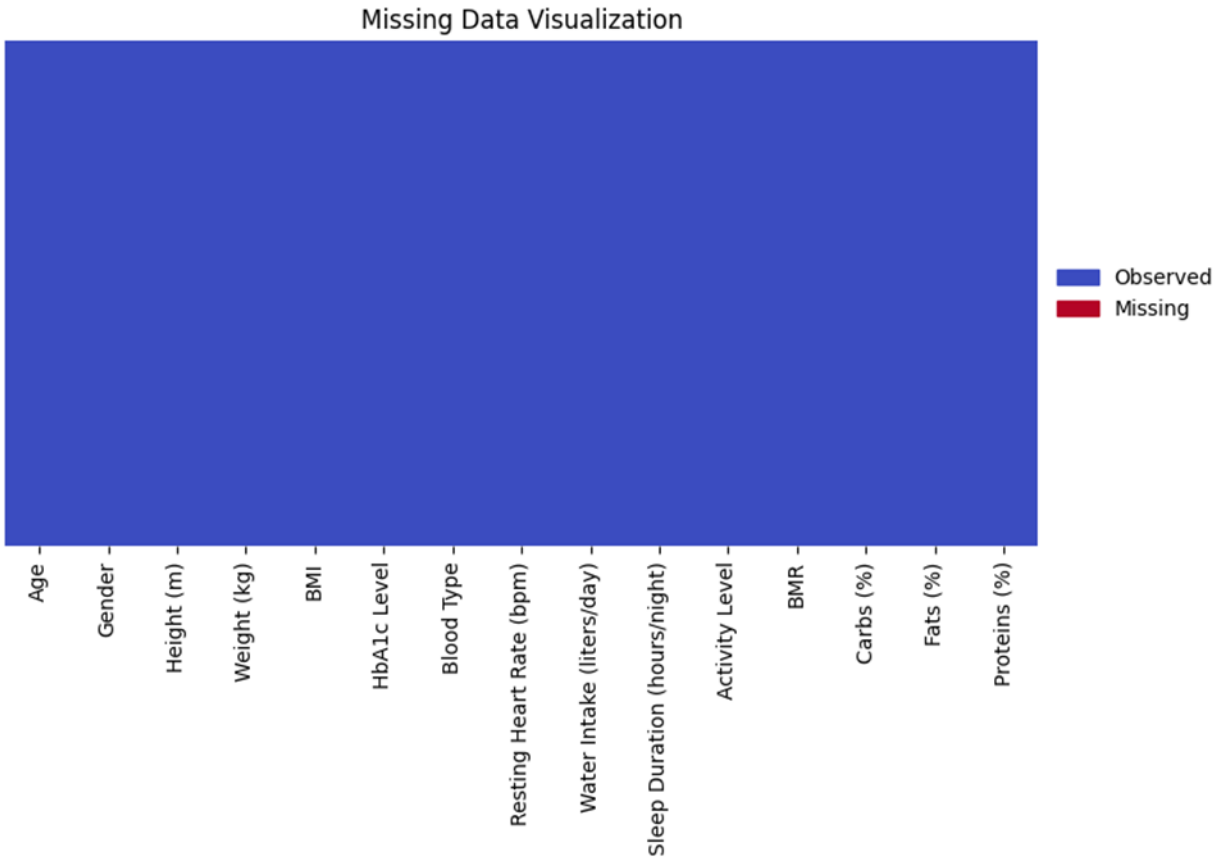


### *6.1.4.3 Scatter Plots*



## 6.2 Pre-Processing

### 6.2.1 Missing Values



### 6.2.2 Outliers

**Method Used: Interquartile Range (IQR)**

Outliers can significantly skew the results of a machine-learning model, leading to inaccurate predictions and reduced model performance. Detecting and removing outliers is a critical preprocessing step to ensure the quality and reliability of the dataset.

**Rationale for Finding and Removing Outliers**

Outliers are data points that significantly differ from the rest of the dataset. They can arise due to various reasons, such as data entry errors, measurement errors, or genuine variability in the data. While some outliers may provide valuable information, others can distort statistical analyses and machine learning models.

Removing outliers helps in:

1. **Improving Model Performance:** Outliers can bias the model, leading to poor generalization of new data. Removing them can enhance the accuracy and robustness of the model.
2. **Enhancing Data Quality:** Eliminating anomalies makes the dataset cleaner and more representative of the underlying population.
3. **Facilitating Better Insights:** Outlier removal aids in gaining more meaningful insights and understanding of the data trends and patterns.

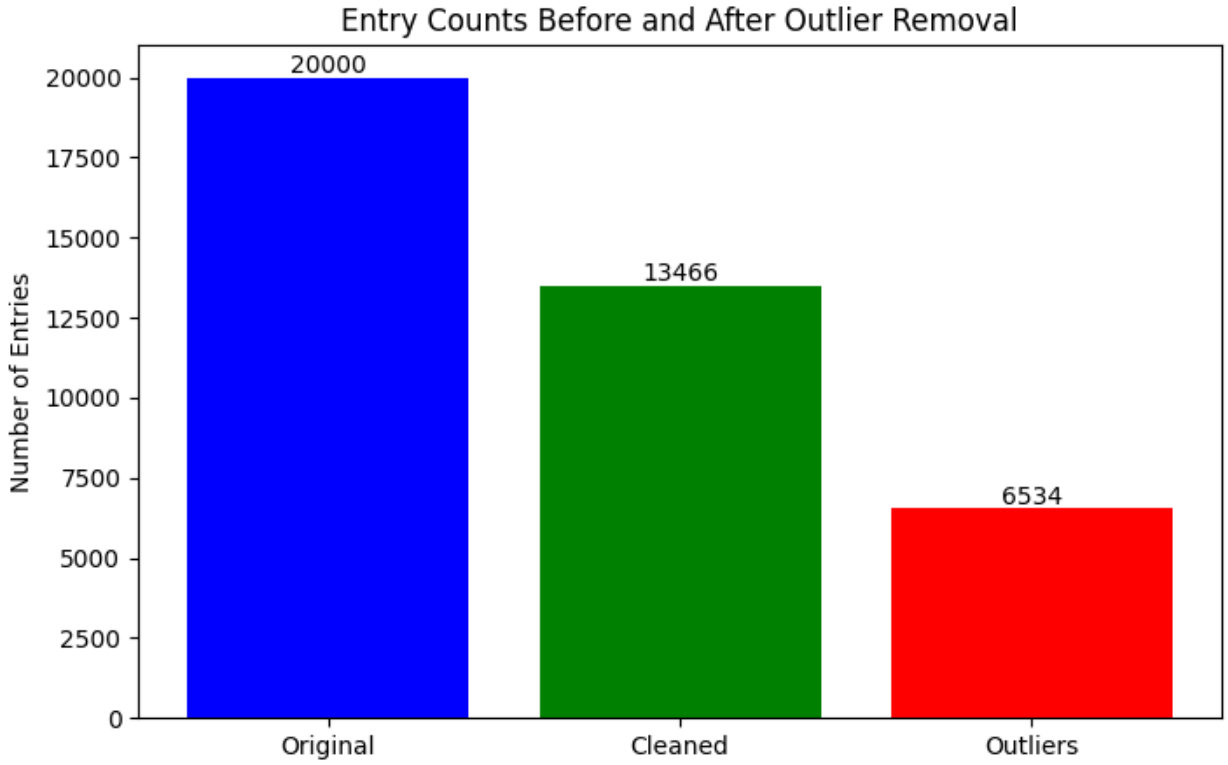
**Interquartile Range (IQR) Method**

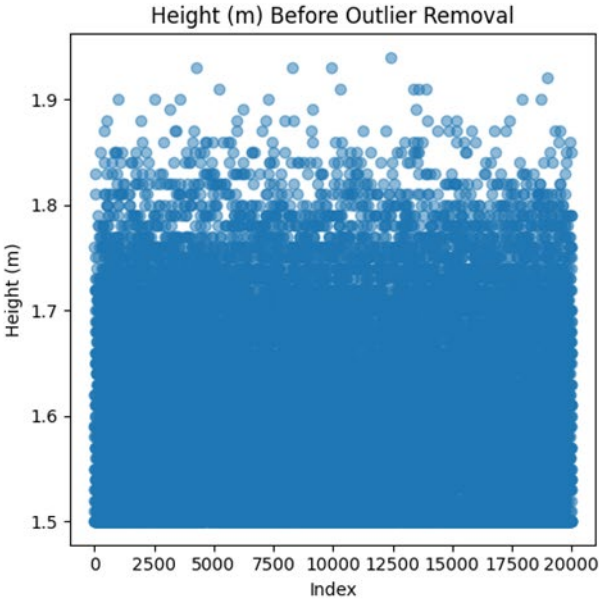
The IQR method is a popular technique for detecting outliers in a dataset. It measures the statistical dispersion and identifies data points outside a defined range.

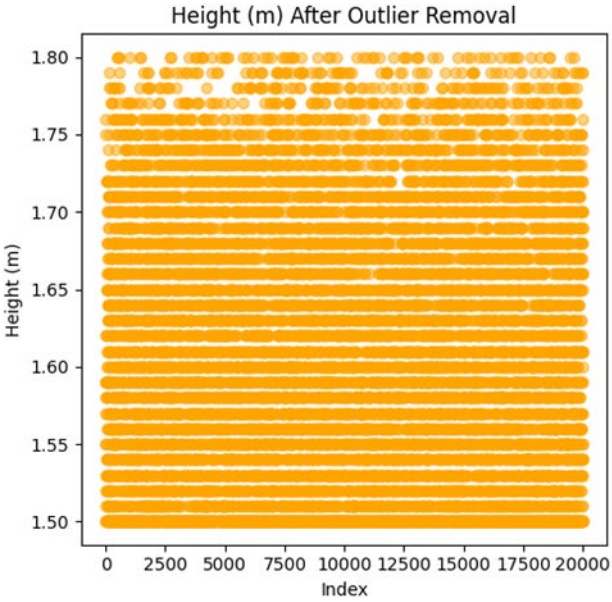
Steps involved in the IQR method:

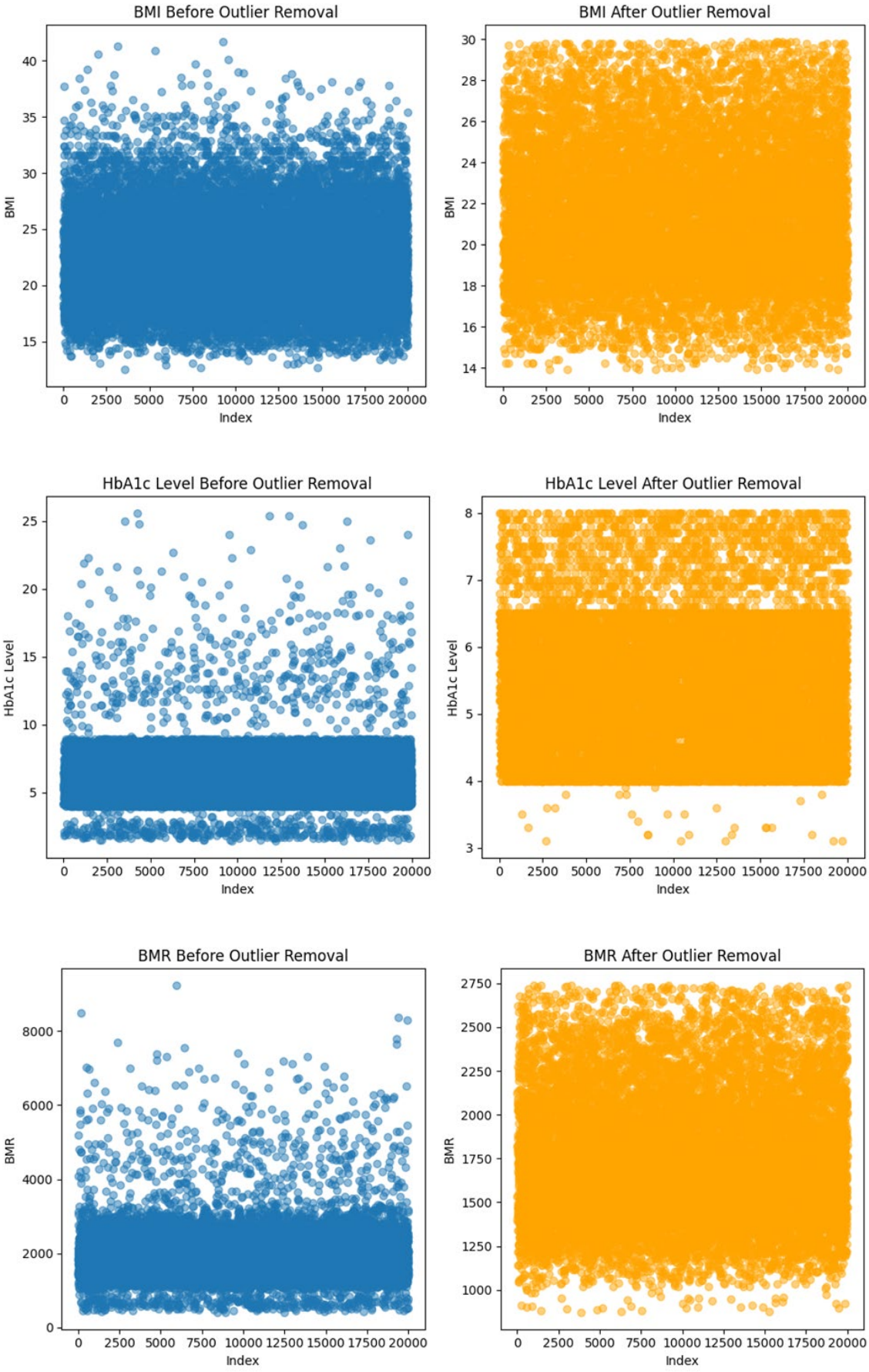
1. **Calculate the Quartiles:** Determine the data's first quartile (Q1) and the third quartile (Q3). Q1 is the 25th percentile, and Q3 is the 75th percentile.
2. **Compute the IQR:** The IQR is the difference between Q3 and Q1 (IQR = Q3 - Q1).
3. **Define the Bounds:** Set the lower and upper bounds for outlier detection. Any data point below Q1 - 1.5*IQR or above Q3 + 1.5*IQR is an outlier.

### *6.2.2.1 Graphical Comparisons*



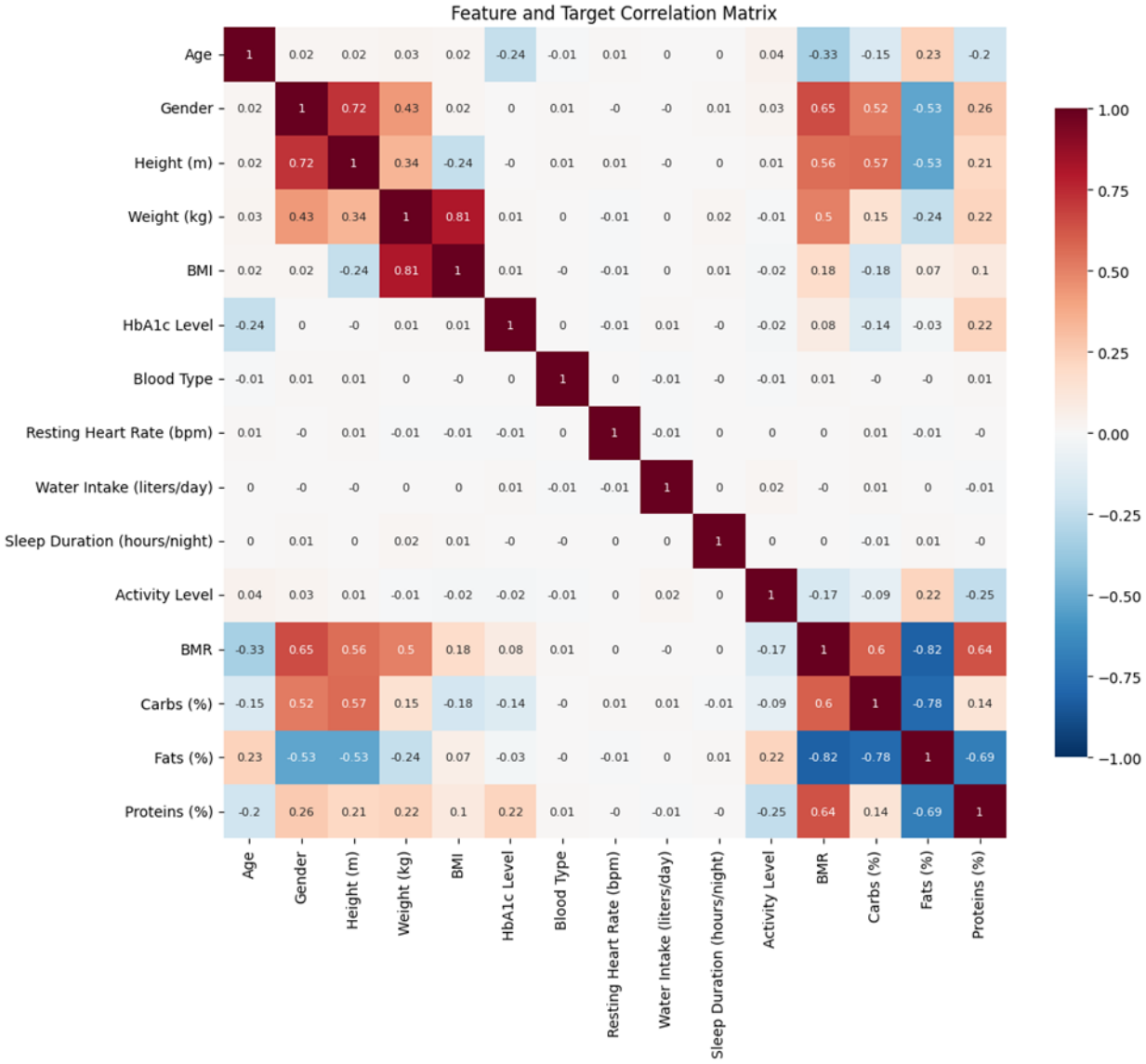




BMI Before Outlier Removal
BMI
Index
BMI After Outlier Removal
BMI
Index
HbA1c Level Before Outlier Removal
HbA1c Level
Index
HbA1c Level After Outlier Removal
HbA1c Level
Index
BMR Before Outlier Removal
BMR
Index
BMR After Outlier Removal
BMR
Index

### 6.2.3 Correlation

We conducted a correlation analysis to understand the relationships between the features and the target variables in our dataset. The correlation matrix below shows the correlation coefficients between all features and targets. This analysis helps identify which features have significant linear relationships with the target variables, aiding in feature selection and model optimization.




**Key Findings from the Correlation Matrix:**

1. **Age:**

- **BMR:** There is a negative correlation (-0.33) between Age and BMR, indicating that BMR tends to decrease as age increases. This is expected as the metabolic rate generally slows down with age.
- **Carbs (%):** A slight negative correlation (-0.15) suggests that older individuals might have a lower percentage of their diet from carbohydrates.
- **Fats (%):** A moderate positive correlation (0.23) indicates that older individuals may consume more fats.

- **Proteins (%):** A small negative correlation (-0.20) with protein percentage, implying that protein intake might decrease with age.

2. **Gender:**

- **Weight (kg) and BMI:** Gender shows a significant correlation with Weight (0.43) and BMI (0.02). This could be due to physiological differences between males and females.
- **Carbs (%):** Gender has a moderate positive correlation (0.52) with carbohydrate intake, suggesting dietary differences between genders.
- **Fats (%):** Gender shows a strong negative correlation (-0.53) with fat intake, indicating different dietary fat consumption patterns between males and females.
- **Proteins (%):** A moderate positive correlation (0.26) suggests that protein consumption varies by gender.

3. **Height (m):**

- **BMI:** Height has a negative correlation (-0.24) with BMI, which is expected since BMI is inversely related to height for a given weight.
- **BMR:** A strong positive correlation (0.56) with BMR indicates that taller individuals have higher basal metabolic rates.
- **Carbs (%):** Height shows a moderate positive correlation (0.57) with carbohydrate intake.
- **Fats (%):** Height has a significant negative correlation (-0.53) with fat intake.
- **Proteins (%):** Height also correlates positively (0.21) with protein intake.

4. **Weight (kg):**

- **BMI:** As expected, there is a very strong positive correlation (0.81) with BMI.
- **BMR:** Weight shows a significant positive correlation (0.65) with BMR, indicating that heavier individuals have higher metabolic rates.
- **Carbs (%):** A positive correlation (0.57) with carbohydrate intake.
- **Fats (%):** Weight has a negative correlation (-0.53) with fat intake.
- **Proteins (%):** Weight positively correlates (0.22) with protein intake.

5. **BMI:**

- **BMR:** BMI has a moderate positive correlation (0.18) with BMR.
- **Carbs (%):** A positive correlation (0.50) indicates that individuals with higher BMI consume more carbohydrates.
- **Fats (%):** BMI shows a small negative correlation (-0.14) with fat intake.
- **Proteins (%):** BMI shows a small positive correlation (0.10) with protein intake.

6. **HbA1c Level:**

    - **Carbs (%):** A negative correlation (-0.18) suggests that individuals with higher HbA1c levels tend to consume fewer carbohydrates.
    - **Fats (%):** There is a small positive correlation (0.07) with fat intake.
    - **Proteins (%):** HbA1c Level shows a very small negative correlation (-0.03) with protein intake.

7. **Resting Heart Rate (bpm):**

    - **BMR:** A slight negative correlation (-0.01) with BMR exists.
    - **Carbs (%):** No significant correlation with carbohydrate intake.
    - **Fats (%):** A small positive correlation (0.01) with fat intake.
    - **Proteins (%):** Resting Heart Rate shows a small negative correlation (-0.01) with protein intake.

8. **Water Intake (liters/day):**

    - **BMR:** No significant correlation with BMR.
    - **Carbs (%):** A small positive correlation (0.01) with carbohydrate intake.
    - **Fats (%):** A very small negative correlation (-0.01) with fat intake.
    - **Proteins (%):** Water Intake shows a very small positive correlation (0.01) with protein intake.

9. **Sleep Duration (hours/night):**

    - **BMR:** No significant correlation with BMR.
    - **Carbs (%):** A small negative correlation (-0.09) with carbohydrate intake.
    - **Fats (%):** A positive correlation (0.22) with fat intake.
    - **Proteins (%):** Sleep Duration shows a very small positive correlation (0.02) with protein intake.

10. **Activity Level:**

    - **BMR:** A small positive correlation (0.02) with BMR.
    - **Carbs (%):** A positive correlation (0.60) indicates that more active individuals consume more carbohydrates.
    - **Fats (%):** A strong negative correlation (-0.78) with fat intake, suggesting that more active individuals consume less fat.
    - **Proteins (%):** A moderate positive correlation (0.25) with protein intake.

11. **BMR:**

    - **Carbs (%):** A positive correlation (0.60) with carbohydrate intake.
    - **Fats (%):** A strong negative correlation (-0.78) with fat intake.

- **Proteins (%):** A strong positive correlation (0.64) with protein intake.

These correlations provide valuable insights into how different features interact with each other and influence the target variables. For instance, the strong correlations between BMR and the target variables (Carbs, Fats, Proteins) highlight the importance of metabolic rate in dietary recommendations. Similarly, the significant correlations involving Age, Gender, Height, and Weight suggest that these features are critical in predicting nutritional needs.

### 6.2.4 Feature selection and removal

**Method Used: Elastic Net Model**

To determine the importance of each feature in predicting the target variables (Carbs (%), Fats (%), and Proteins (%)), we employed the Elastic Net model, a robust and reliable tool. This model combines the penalties of the Lasso and Ridge regression techniques, making it suitable for feature selection when dealing with multicollinear datasets or a mix of important and irrelevant features.

**The rationale for Using Elastic Net**

The Elastic Net model is advantageous because it effectively balances the penalties of the Lasso and Ridge regression techniques, ensuring a robust feature selection process. It is particularly suitable for feature selection when dealing with multicollinear datasets or a mix of important and irrelevant features.

1. **Balances L1 and L2 Penalties:** This balance helps handle correlated features and select groups of related features.
2. **Feature Selection:** It can shrink the coefficients of less important features to zero, removing them from the model and highlighting the most influential features.
3. **Robustness:** It is robust against overfitting and performs well when the number of predictors exceeds the number of observations.

**Implementation and Results**

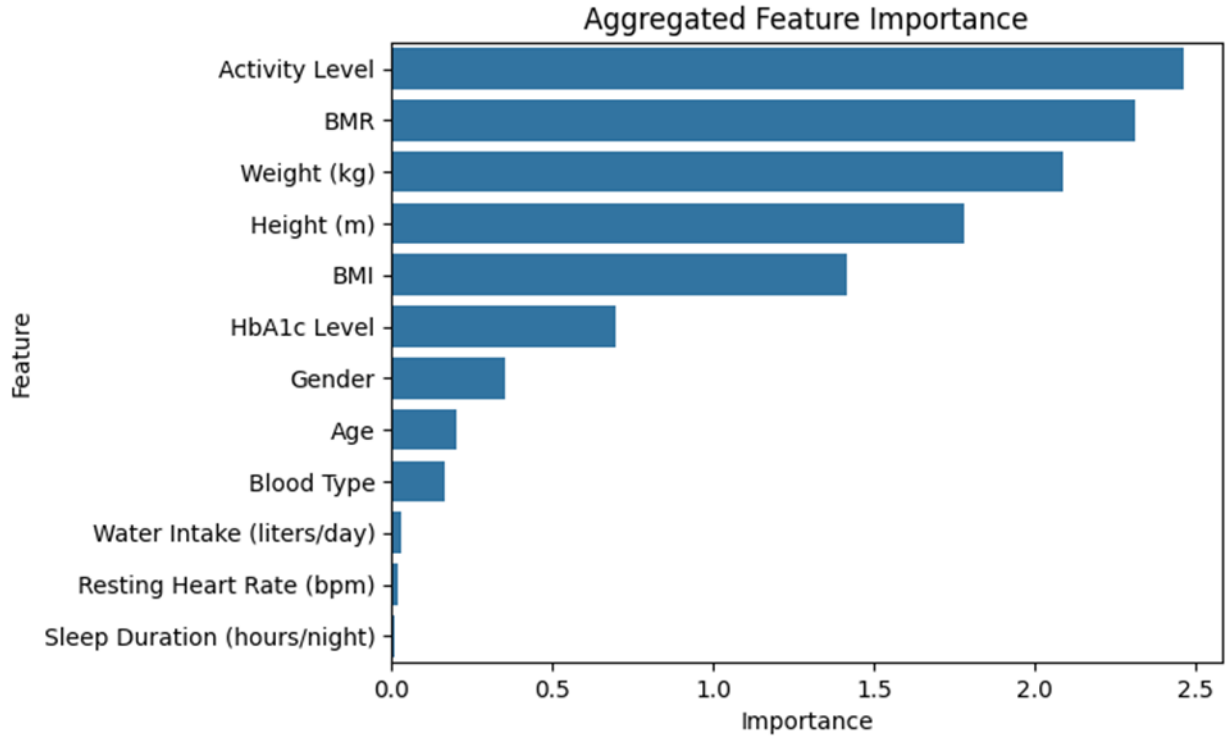
Here is the process followed and the key findings:

1. **Data Preparation:**
    - The dataset was pre-processed to handle categorical variables through one-hot encoding.
    - Features were normalized using standard scaling.
    - The dataset was split into training and testing sets.
2. **Model Training:**
    - Separate Elastic Net models were trained for each target variable: Carbs (%), Fats (%), and Proteins (%).
    - The feature's importance was extracted from these models' coefficients.
3. **Feature Importance Calculation:**
    - The importance values for each feature were averaged across the three target variables to obtain a consolidated importance score.
    - Aggregated importance scores were calculated for categorical features by summing the importance values of their one-hot encoded components.
4. **Results:**
    - The least essential features had very low values, indicating minimal impact on the target variables.

- The most important features were identified based on their higher average importance scores.

**Visualizing Feature Importance:**

The bar chart below shows the aggregated importance of each feature. Features with higher bars significantly impact predicting the target variables, while those with lower bars are less influential.




**Conclusion:**

The Elastic Net model ranked feature importance, allowing us to identify and remove the least important features. This step simplified the model and maintained its accuracy, ensuring that only the most relevant features were used for predictions. Visualizing feature importance helps us understand the relative impact of each feature, guiding future data collection and model improvement efforts.

### 6.2.5 Normalization

**Normalization Process**

Normalization is a crucial step in data preprocessing, especially when dealing with features that have different scales. Without normalization, features with larger ranges could dominate the model training process, leading to biased results. For our model, we chose the Standard Scaler for normalization.

**Choice of Scaling Method: Standard Scaler**

The Standard Scaler was selected to ensure that each feature contributes equally to the model. This scaler standardizes the features by removing the mean and scaling to unit variance, particularly useful when the features have different units or scales. The formula used by the Standard Scaler is:

$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean
$\sigma =$ Standard Deviation

**Normalization Implementation**

1. **Loading the Dataset:**
   - The dataset was loaded and cleaned by removing unnamed columns and handling missing values.
2. **Splitting the Data:**
   - The dataset was split into features (X) and targets (y). The features include biometric and lifestyle attributes, while the targets are the percentages of carbohydrates, fats, and proteins.
3. **Handling Categorical and Numerical Features:**
   - The features were divided into categorical and numerical categories. Categorical features, such as 'Gender' and 'Activity Level,' were handled separately from numerical features.
4. **Applying Standard Scaler:**
   - Numerical features were normalized using the Standard Scaler, ensuring each numerical feature had a mean of 0 and a standard deviation of 1. This transformation helps make the data suitable for machine learning algorithms by eliminating the bias caused by the different scales of the features.

**Benefits of Using Standard Scaler:**

- **Equal Contribution:** Ensures that all features contribute equally to the model training process.
- **Improved Convergence:** Helps in faster and more stable convergence of gradient descent-based algorithms.
- **Enhanced Performance:** Reduces the impact of outliers and improves the overall performance and accuracy of the model.

## 6.3 Machine Learning Model Training

The machine learning model training process is critical to developing an effective predictive model. This section explains the model training steps, including data preprocessing, model selection, hyperparameter tuning, and validation.

### 6.3.1 Model Selection

**Overview**
Choosing the right machine learning model is crucial for achieving accurate predictions. The model selection process involves evaluating different algorithms to determine which performs best on the dataset.

**Chosen Model: XGBoost Regressor**

1. **Reason for Selection:** The XGBoost Regressor was chosen for its ability to handle large datasets, capture complex patterns, and provide robustness against overfitting. XGBoost is known for its efficiency and performance in various machine-learning tasks.

2. **Comparison with Other Models:** While other models, such as linear regression and Decision Trees, were considered, XGBoost provided superior performance and better handling of non- linear relationships in the data.

### 6.3.2 Explanation of XGBoost Regressor

**Overview**
XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm that has gained popularity for its speed and performance. It is an implementation of gradient-boosted decision trees designed for speed and performance.

**How It's Made:**

1. **Gradient Boosting Framework:** XGBoost is built on the gradient boosting framework, which combines the predictions of multiple weak learners (usually decision trees) to create a strong predictive model.

2. **Decision Trees:** The core model is a decision tree, and XGBoost builds an ensemble of trees sequentially, where each tree attempts to correct the errors of the previous ones.

**How It Works:**

1. **Model Training:**

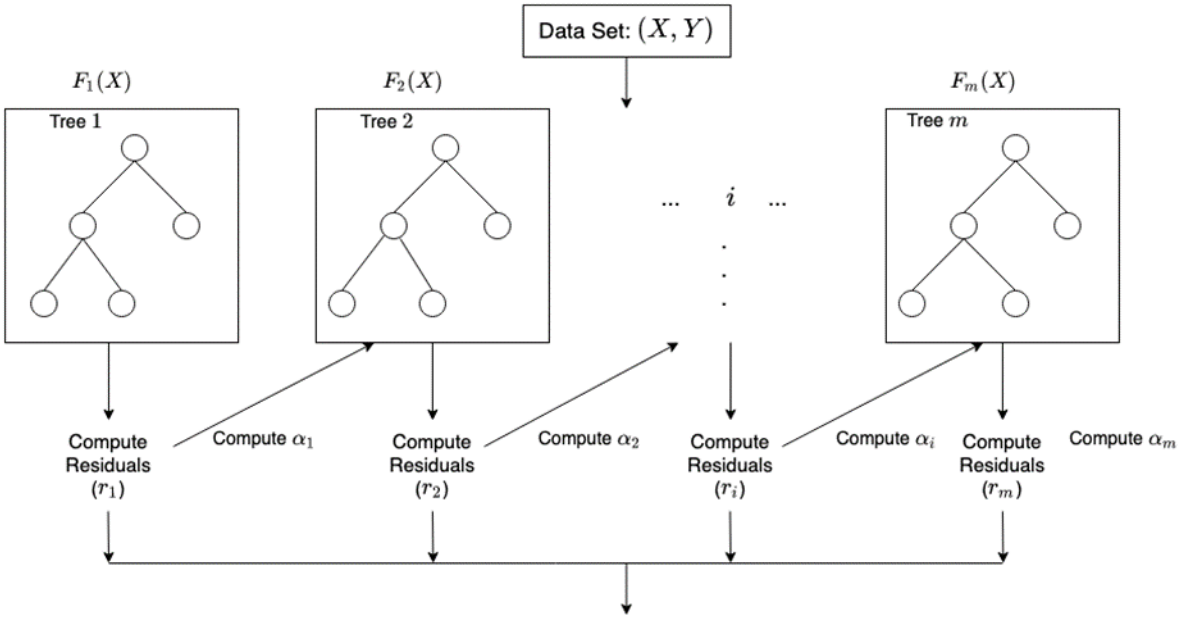
   - **Initialization:** Start with an initial prediction, often the mean of the target variable.
   - **Tree Building:** Sequentially add decision trees. Each tree is trained on the residual errors of the previous trees.
   - **Gradient Descent:** Use gradient descent to minimize the loss function. The gradients indicate how to adjust the model to reduce errors.

- **Weight Updates:** Update the weights of the trees to reduce the residuals iteratively.

2. **Regularization:** XGBoost includes regularization parameters to prevent overfitting, making it robust and generalizable.

3. **Parallel Processing:** It supports parallel processing to speed up computation, making it efficient for large datasets.

## Visual Example:




$$F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

where $\alpha_i$, and $r_i$ are the regularization parameters and residuals computed with the $i^{th}$ tree respectfully, and $h_i$ is a function that is trained to predict residuals, $r_i$ using $X$ for the $i^{th}$ tree. To compute $\alpha_i$ we use the residuals computed, $r_i$ and compute the following: $arg \min_{\alpha} = \sum_{i=1}^{m} L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where $L(Y, F(X))$ is a differentiable loss function.

### 6.3.3 Hyperparameter Tuning

**Overview**
Hyperparameter tuning involves optimizing the model's parameters to improve its performance. This step is essential for achieving the best possible results from the chosen model.

**Steps:**

1. **Grid Search:** Grid Search was used for hyperparameter tuning, which systematically tests combinations of parameters to find the best settings.
2. **Parameters Tuned:** Key hyperparameters tuned for the XGBoost Regressor included:
    - **n_estimators:** Number of trees (tested values: 300, 400).
    - **max_depth:** Maximum depth of each tree (tested values: 1, 3, 4).
    - **learning_rate:** Step size shrinkage (tested value: 0.2).
    - **subsample:** Subsample ratio of the training instances (tested values: 0.5, 1.0).
    - **colsample_bytree:** Subsample ratio of columns when constructing each tree (tested values: 0.7, 1.0).
3. **Results:** The best hyperparameters were found to be:
    - **n_estimators:** 400
    - **max_depth:** 1
    - **learning_rate:** 0.2
    - **subsample:** 1.0
    - **colsample_bytree:** 1.0 These settings improved the model's performance, as indicated by the evaluation metrics.

### 6.3.4 Model Training

**Overview**
Model training feeds the pre-processed data into the chosen algorithm to learn patterns and make predictions.

**Steps:**
1. **Training Process:** The training data was used to train the XGBoost Regressor. The model was fitted to the data using the best hyperparameters identified through Grid Search.

2. **Cross-Validation:** Five-fold cross-validation was used to ensure the model generalizes well to unseen data. This involves splitting the training data into five subsets, training on four subsets, and validating the remaining subset. This process is repeated five times, each time with a different validation set, to ensure robustness.

### 6.3.5 Model Evaluation

**Overview**
Evaluating the trained model is crucial to understanding its performance and identifying any potential issues.

**Metrics Used:**
- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values.
- **Root Mean Squared Error (RMSE):** Provides a measure of the differences between predicted and actual values in the same units as the target variable.
- **Mean Absolute Percentage Error (MAPE):** Indicates the accuracy of predictions as a percentage.
- **R-squared ($R^2$) measures** the proportion of variance in the target variable explained by the model.

**Evaluation Process:**
- **Training Set Performance:** The model was evaluated on the training set to check for any signs of overfitting or underfitting.
- **Validation Set Performance:** Cross-validation results ensured the model's generalization ability to new data.
- **Interpretation:** The evaluation metrics showed that the model was well-fitting, with an $R^2$ score of 0.7753 and a MAPE of 5.11% on the test set.

**Evaluation Results:**
- **Test MSE:** 4.8126
- **Test RMSE:** 2.1938
- **Test MAPE:** 5.11%
- **R^2 Score:** 0.7753

### 6.3.6 Model Testing

**Overview**
Model testing involves assessing the model's performance on a separate test set not used during training or validation. This step helps understand the model's real-world applicability.

**Steps:**

1. **Test Set Evaluation:** The model was evaluated using the same metrics on a separate test set.
2. **Comparison with Training/Validation Performance:** The test set results were compared with the training and validation performance to identify any overfitting or underfitting issues.

**Test Evaluation Results:**

- **Test MSE:** 5.3274
- **Test RMSE:** 2.3081
- **Test MAPE:** 5.43%
- **Test R^2 Score:** 0.7843

## 6.4 Results

The results section provides a comprehensive overview of the model's performance, presented through tables and figures. This section helps in understanding the trained model's effectiveness and predictive accuracy.

### 6.4.1 Tables

**Overview**
The tables below summarize the model's evaluation metrics on both the training and test datasets. These metrics include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared ($R^2$) Score.

**Training and Test Set Performance:**

| Metric | Training Set | Test Set |
|---|---|---|
| MSE | 4.8126 | 5.3274 |
| RMSE | 2.1938 | 2.3081 |
| MAPE | 5.11% | 5.43% |
| $R^2$ Score | 0.7753 | 0.7843 |

**Interpretation:**

- **MSE (Mean Squared Error):** The MSE values indicate the average squared difference between the actual and predicted values. Lower MSE values suggest better model performance.
- **RMSE (Root Mean Squared Error):** The RMSE values measure the differences between the actual and predicted values in the same units as the target variables. Lower RMSE values indicate better predictive accuracy.
- **MAPE (Mean Absolute Percentage Error):** The MAPE values indicate the accuracy of the predictions as a percentage. Lower MAPE values represent higher accuracy.
- **$R^2$ Score:** The $R^2$ scores measure the proportion of variance in the target variables explained by the model. Higher $R^2$ scores indicate better model performance.

**Comparison of Model Performance:**
The table below compares the accuracies of all the models tried during the project. This comparison helps understand the relative performance of different algorithms and the effectiveness of the chosen model.

| Model Name | Mean Squared Error (MSE) | Root Mean Squared Error (RMSE) | Mean Absolute Percentage Error (MAPE) |
|---|---|---|---|
| Linear Regression | 6.93 | 2.63 | 7.23% |
| Ridge Regression | 7.12 | 2.67 | 7.31% |
| Lasso Regression | 6.92 | 2.63 | 7.23% |
| Elastic Net Regression | 7.05 | 2.66 | 7.28% |

| | | | |
|---|---|---|---|
| Polynomial Regression | 6.93 | 2.63 | 7.23% |
| Decision Tree Regression | 7.06 | 2.66 | 7.31% |
| Random Forest Regression | 6.22 | 2.49 | 6.96% |
| AdaBoost Regression | 6.27 | 2.50 | 6.98% |
| **XGBoost Regression** | **5.32** | **2.3** | **5.43%** |
| K-Nearest Neighbours | 11.02 | 3.32 | 8.69% |

## Interpretation:

- **XGBoost Regression:** Achieved the lowest MSE (5.32), RMSE (2.3), and MAPE (5.43%), making it the best-performing model among those tested.
- **Random Forest Regression and AdaBoost Regression:** They also performed well but were slightly less accurate than XGBoost.
- **K-Nearest Neighbours Regression:** The highest error metrics indicated it was the least effective model for this dataset.

### 6.4.2 Figures

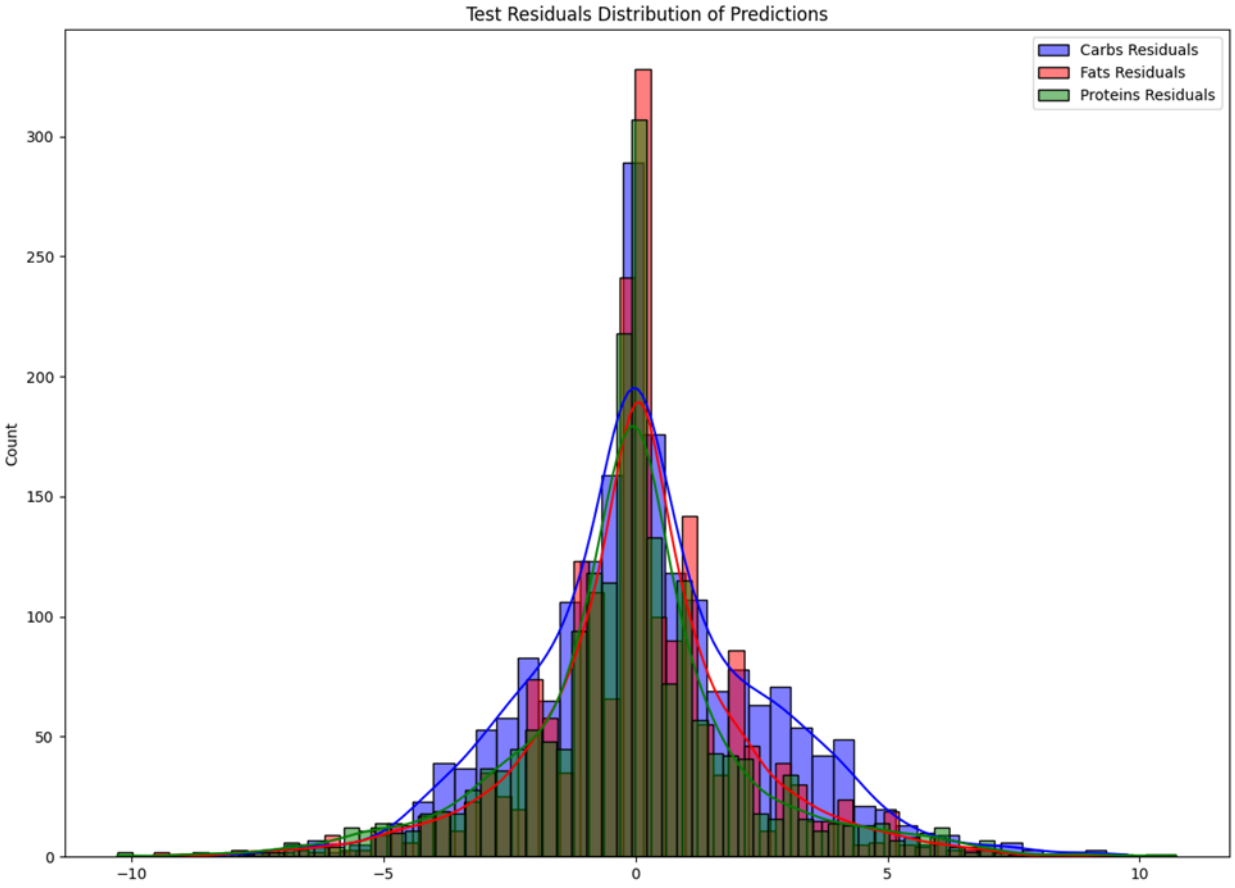
**Overview**

The figures below visualize the model's performance through various plots, including the residuals distribution, actual vs. predicted values, and residuals for each target variable. These visualizations help understand the model's behavior and identify any potential issues.

**Figures:**

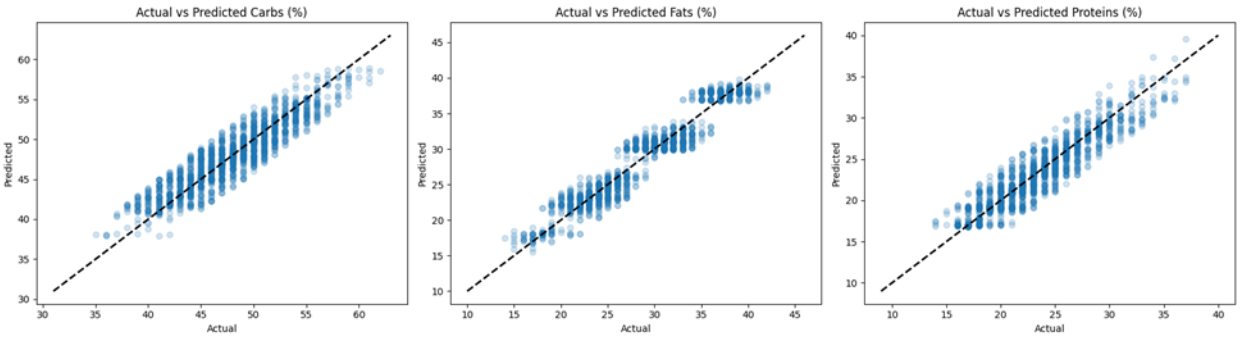
1. **Residuals Distribution:**

   The residuals distribution plot shows the differences between each target variable's actual and predicted values. A narrow, centered distribution around zero indicates good model performance.



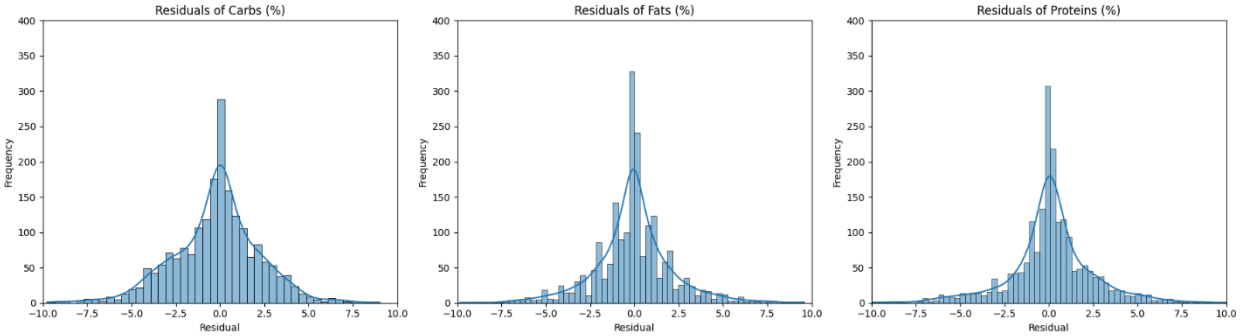

2. **Actual vs Predicted Values:**

   These scatter plots compare the actual values of the target variables (Carbs (%), Fats (%), and Proteins (%)) with the predicted values. The dashed line represents the ideal scenario where actual and predicted values are equal. Points closely aligned with the dashed line indicate high predictive accuracy.




3. **Residuals for Each Target Variable:**

The histograms for the residuals of Carbohydrates (%), Fats (%), and Proteins (%) show the frequency distribution of the residuals. These plots help understand each target variable's distribution and spread of prediction errors.




**Conclusion:**

The tables and figures provide a detailed overview of the model's performance, highlighting its accuracy and reliability in predicting the distribution of carbohydrates, fats, and proteins in an individual's diet. The results indicate that the XGBoost Regressor model performs well, with low error metrics and high $R^2$ scores, making it a suitable choice for this predictive task. The visualizations further confirm the model's effectiveness and help identify potential improvement areas.

## 6.5 Discussion

The discussion section provides a comprehensive analysis of the model-building process, wrapping up everything done in the model-development process. It highlights the best and worst-performing algorithms, reasons for their performance, and insights for future work. This critical evaluation helps understand the chosen approach's strengths and limitations and identifies improvement opportunities.

**Best Performing Algorithm: XGBoost Regressor**

The XGBoost Regressor emerged as the best-performing model in our project. Its evaluation metrics on the test set were as follows:

- **MSE:** 5.32
- **RMSE:** 2.30
- **MAPE:** 5.43%

**Reasons for Superior Performance:**

1. **Complex Pattern Recognition:** XGBoost is highly effective at capturing complex, non-linear relationships within the data. This capability was crucial given our dataset's diverse biometric and lifestyle features.
2. **Regularization:** Including L1 and L2 regularization techniques helped mitigate overfitting, ensuring the model's generalization to new, unseen data.
3. **Gradient Boosting Framework:** The gradient boosting mechanism builds an ensemble of weak learners, each correcting the errors of the previous ones. This iterative improvement contributed significantly to the model's robustness.
4. **Hyperparameter Optimization:** Extensive hyperparameter tuning using Grid Search allowed the identification of optimal parameters, enhancing model performance.
5. **Parallel Processing:** XGBoost's support for parallel processing made it efficient and capable of handling large datasets swiftly.

These features collectively made XGBoost the most reliable and accurate model for predicting the distribution of carbohydrates, fats, and proteins in an individual's diet.

**Least Performing Algorithm: K-Nearest Neighbors (KNN) Regression**

In contrast, the K-Nearest Neighbors (KNN) Regression model demonstrated the poorest performance with the following metrics:

- **MSE:** 11.02
- **RMSE:** 3.32
- **MAPE:** 8.69%

### Reasons for Poor Performance:

1. **Sensitivity to Outliers:** KNN is highly sensitive to outliers, which can disproportionately influence the model's predictions. Despite preprocessing steps to handle outliers, KNN still struggled with anomalies in the dataset.

2. **Curse of Dimensionality:** The effectiveness of distance metrics diminishes in high-dimensional spaces, leading to less meaningful predictions. Our dataset's multiple features exacerbated this issue for KNN.

3. **Lack of Complexity Handling:** KNN's simplicity in dealing with non-linear relationships and reliance on local neighborhood structures were inadequate compared to more sophisticated models like XGBoost.

4. **Uniform Weighting:** In KNN, each neighbor's influence is equally weighted, which can misrepresent the true importance of closer versus farther points in a high-dimensional context.

These limitations made KNN unsuitable for our predictive task, highlighting the need for more advanced algorithms for complex datasets.

### Key Insights:

- **Tree-Based Models:** Random Forest and AdaBoost also performed well but were slightly less accurate than XGBoost. Their ability to handle non-linear relationships and ensemble nature contributed to their robustness.

- **Linear Models:** Linear Regression, Ridge, Lasso, and Elastic Net performed similarly, with moderate error metrics. However, they were less capable of capturing non-linear patterns than tree-based models.

- **Polynomial Regression:** Performed similarly to linear models, indicating that higher-order polynomials did not significantly enhance model accuracy.

- **Decision Trees:** While better than linear models, single decision trees lacked the ensemble strength of Random Forest and AdaBoost, resulting in higher errors.

### Future Work and Improvements

### Potential Improvements:

1. **Feature Engineering:** Creating new features or transforming existing ones could help improve model performance. Incorporating interaction terms or higher-order polynomial features might capture more complex patterns.

2. **Algorithm Exploration:** Exploring advanced algorithms like LightGBM or CatBoost, like XGBoost, but may offer improved performance or efficiency in specific scenarios, could be beneficial.

3. **Hyperparameter Tuning:** More sophisticated hyperparameter tuning techniques, such as Bayesian Optimization or Random Search, might yield better results than Grid Search.

4. **Ensemble Methods:** Combining multiple models to form an ensemble could leverage the strengths of different algorithms, potentially improving overall performance.

### Future Directions:

1. **Expand Dataset:** Collecting more data, especially with a broader range of variations, can enhance the model's robustness and generalizability.

2. **Real-Time Predictions:** Implementing the model in a real-time system for live dietary recommendations could provide valuable user feedback and refine the model.

3. **User Personalization:** Enhancing the model to incorporate user preferences and feedback for more personalized dietary recommendations.

## Conclusion

The XGBoost Regressor demonstrated exceptional performance, making it the best choice for predicting the distribution of carbohydrates, fats, and proteins in an individual's diet. The KNN Regression model, while simple, struggled with high-dimensional data and sensitivity to outliers. By leveraging the strengths of advanced algorithms and exploring potential improvements, the predictive model can be further refined to provide even more accurate and personalized dietary recommendations. The insights gained from this project pave the way for future enhancements and practical applications in personalized nutrition planning.

# Chapter 7. Machine Learning Model 02

## 7.1 Dataset

### 7.1.1 Overview

The dataset used for the second model focuses on classifying food items' suitability for diabetic patients based on various features. The goal is to predict the 'RecScore' (recommendation score) for each food item. Here's a snapshot of the dataset structure:

#### *7.1.1.1 Features*

| Age | Gender | Height (m) | Weight (kg) | BMI | HbA1c Level | Activity Level | Food | GL | Reaction | FBS | PMS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | Female | 1.5 | 51 | 22.7 | 4.5 | Very Active | Cashew Nut | 7.1 | 4 | 97 | 108 |
| 31 | Male | 1.57 | 45 | 18.3 | 5.3 | Lightly Active | Carrots | 1.5 | 2 | 111 | 116 |
| 19 | Male | 1.76 | 70 | 22.6 | 5.3 | Moderately Active | Fish Shanghara | 0 | 4 | 81 | 81 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

#### *7.1.1.2 Targets*

| RecScore |
|---|
| 8 |
| 9 |
| 9 |
| ... |

### 7.1.2 Feature Details

#### *7.1.2.1 Biometric and Lifestyle Features*

**Age and Gender**: These features account for variations in metabolic rates and nutritional needs across different age groups and between genders. Age influences basal metabolic rate (BMR) and an individual's overall energy requirements. [1]

**Height and Weight**: These measurements are used to calculate Body Mass Index (BMI), a crucial indicator of body fat, and help assess dietary needs.

**BMI (Body Mass Index)**: BMI is calculated as weight in kilograms divided by height in square meters. It provides a standardized measure to classify underweight, normal weight, overweight, and obesity, which are critical for determining appropriate nutritional guidelines.[1], [2]

**HbA1c Level**: This feature reflects the average blood glucose levels over the past two to three months. It is particularly important for understanding carbohydrate metabolism and managing dietary recommendations for individuals with varying glucose tolerance levels.[3]

**Activity Level**: This feature is crucial for adjusting caloric intake. The activity level is determined based on the Harris-Benedict equation, which helps estimate an individual's Basal Metabolic Rate (BMR) adjusted for physical activity.

**Food**: Represents different food items consumed by individuals, each with unique nutritional values.

**GL (Glycemic Load)**: Measures the impact of carbohydrate consumption on blood sugar levels.

**Reaction**: Captures the body's immediate response to food consumption.

**FBS (Fasting Blood Sugar)**: Indicates blood glucose levels after fasting, crucial for tailoring dietary recommendations.

**PMS (Post-Meal Sugar)**: Indicates blood glucose levels after consuming food, essential for assessing the impact of meals.

#### *7.1.2.2 Targets*

**RecScore (Recommendation Score):** Indicates the suitability of a food item for diabetic patients based on various features. Helps in making personalized dietary recommendations.

### 7.1.3 Feature Identification and Categorization

Attributes can be described by corresponding values and categorized as discrete (categorical) or continuous. Discrete features have a finite number of values, while continuous features cover a specific interval or range and can take on infinite values within that range.

**Discrete Features:**

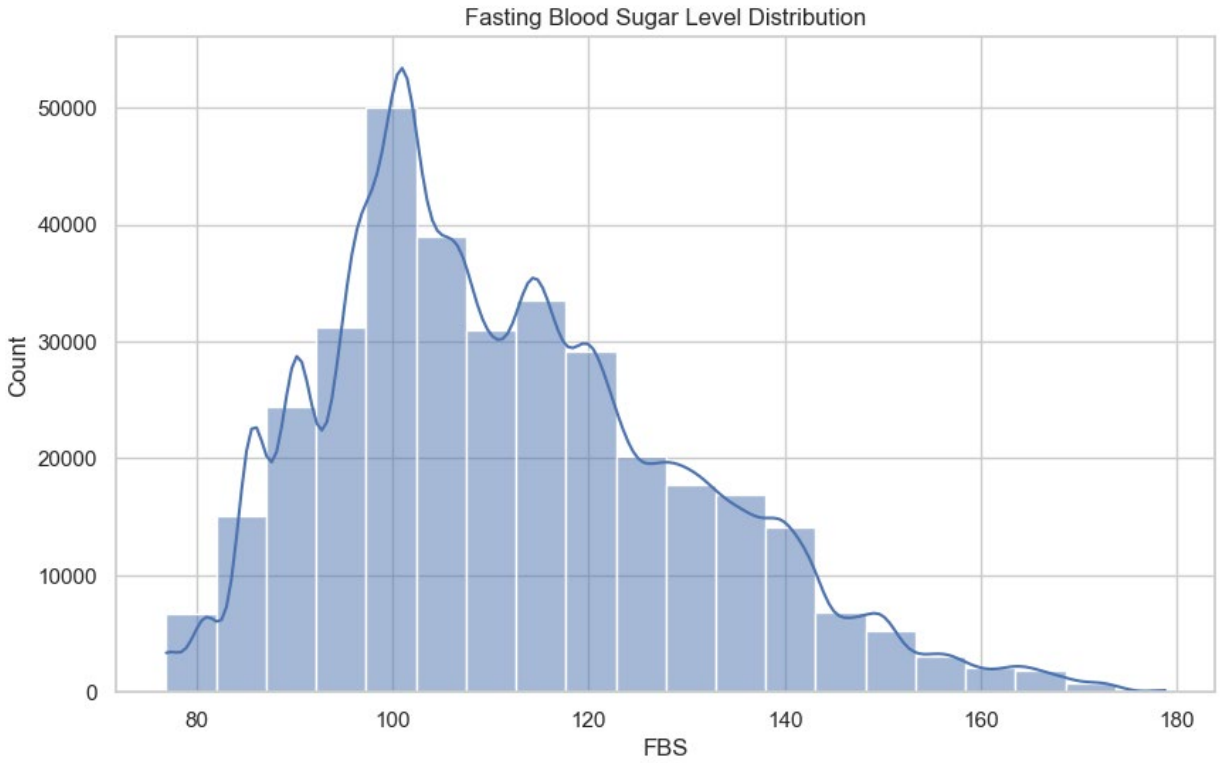
- **Gender**: Categorical feature with values 'Male' and 'Female.'
- **Food**: Categorical feature representing different food items.
- **Activity Level**: Categorical feature with levels 'Sedentary,' 'Lightly Active,' 'Moderately Active,' 'Very Active.'

**Continuous Features:**

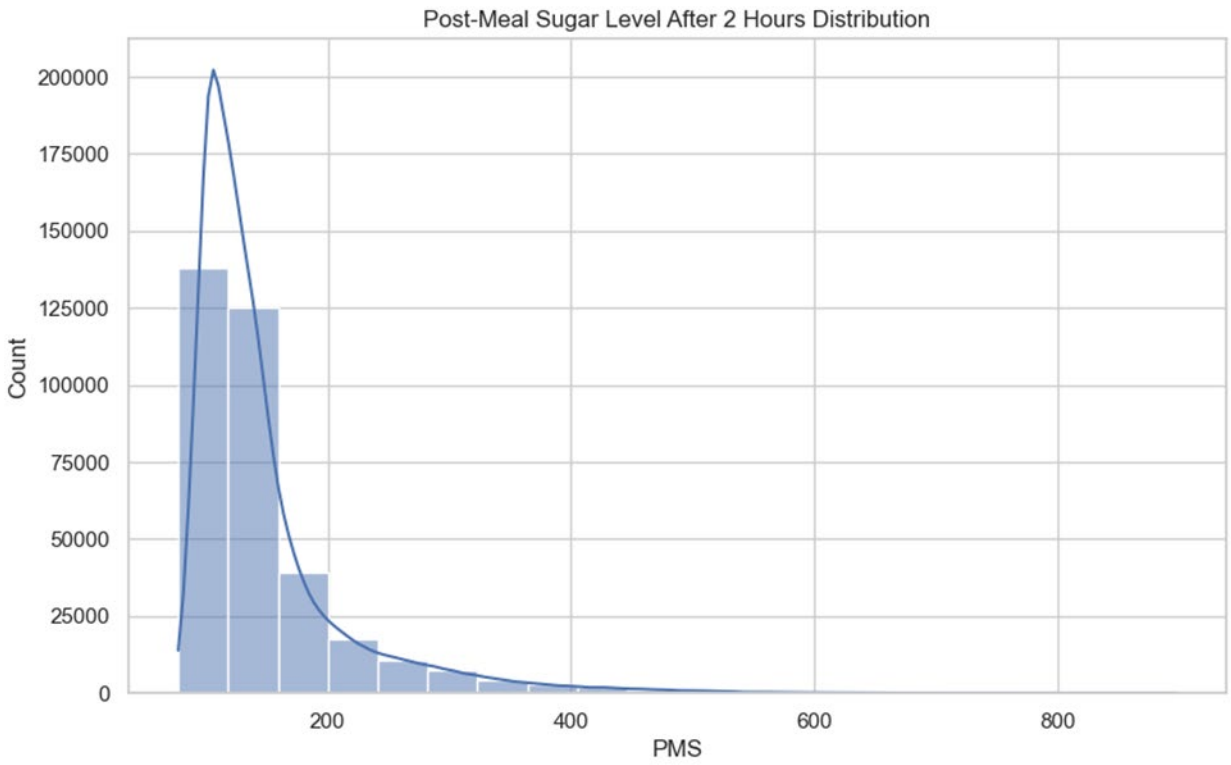
- **Age**: Numerical feature representing the age of the individual in years.
- **Height (m)**: Numerical feature representing the individual's height in meters.
- **Weight (kg)**: Numerical feature representing the individual's weight in kilograms.
- **BMI**: Numerical feature calculated as weight divided by the square of height.
- **HbA1c Level**: Numerical feature indicating the average blood glucose level over the past two to three months.
- **GL**: Numerical feature representing the glycemic load of food items.
- **Reaction**: Numerical feature indicating the body's immediate response to food.
- **FBS**: Numerical feature representing fasting blood sugar levels.
- **PMS**: Numerical feature representing post-meal sugar levels.

### 7.1.4 Graphs

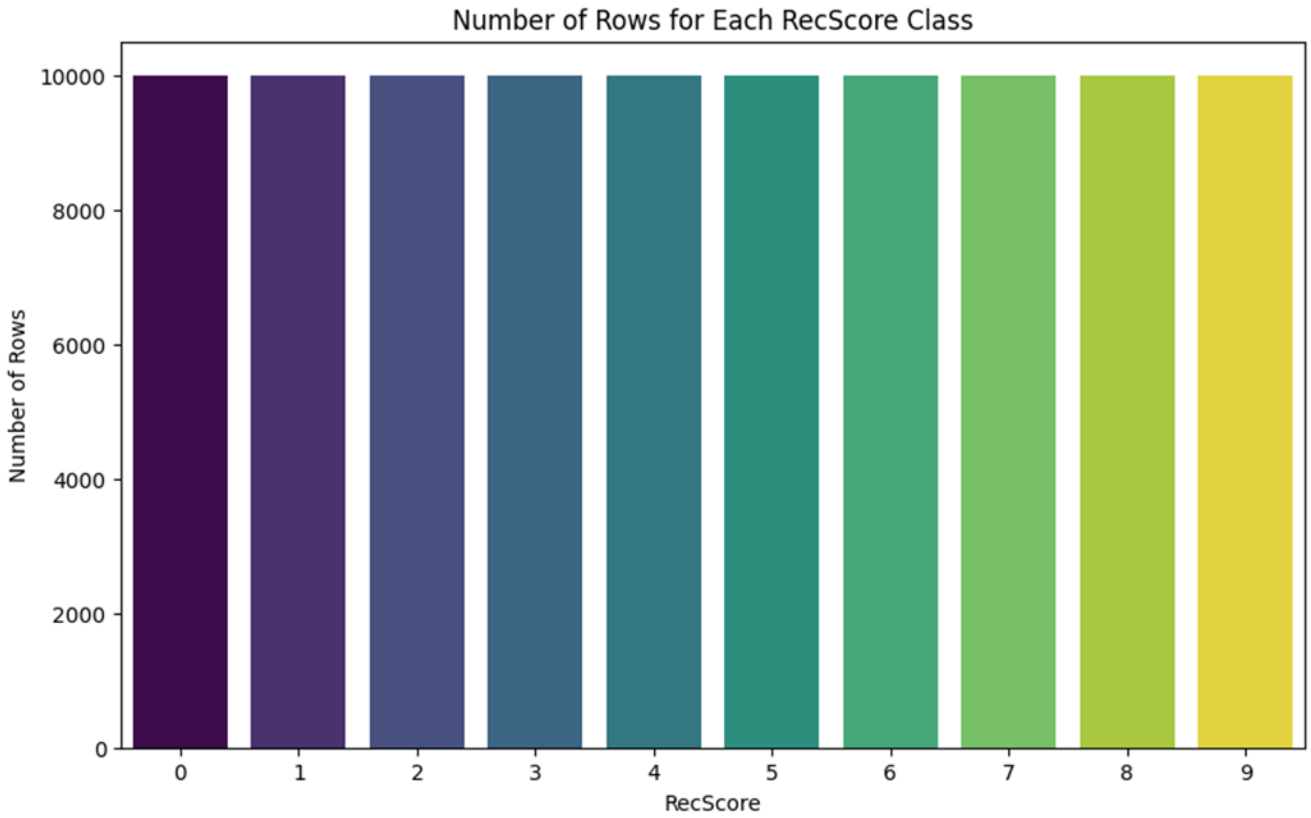
#### *7.1.4.1 Distribution*




This histogram with a density plot shows the distribution of fasting blood sugar (FBS) levels. Most readings cluster around 100 mg/dL, with the majority falling between 80 and 120 mg/dL, indicating the central tendency of FBS levels in the dataset.




This histogram with a density plot depicts the distribution of post-meal sugar (PMS) levels. The data shows a peak around 150-200 mg/dL, with a right-skewed tail, indicating that while most PMS levels are below 200 mg/dL, there are some higher values.




This bar chart shows the number of instances for each RecScore class, demonstrating a balanced distribution across all classes (0-9). This balance ensures that each class is equally represented, which is crucial for training a robust classification model.

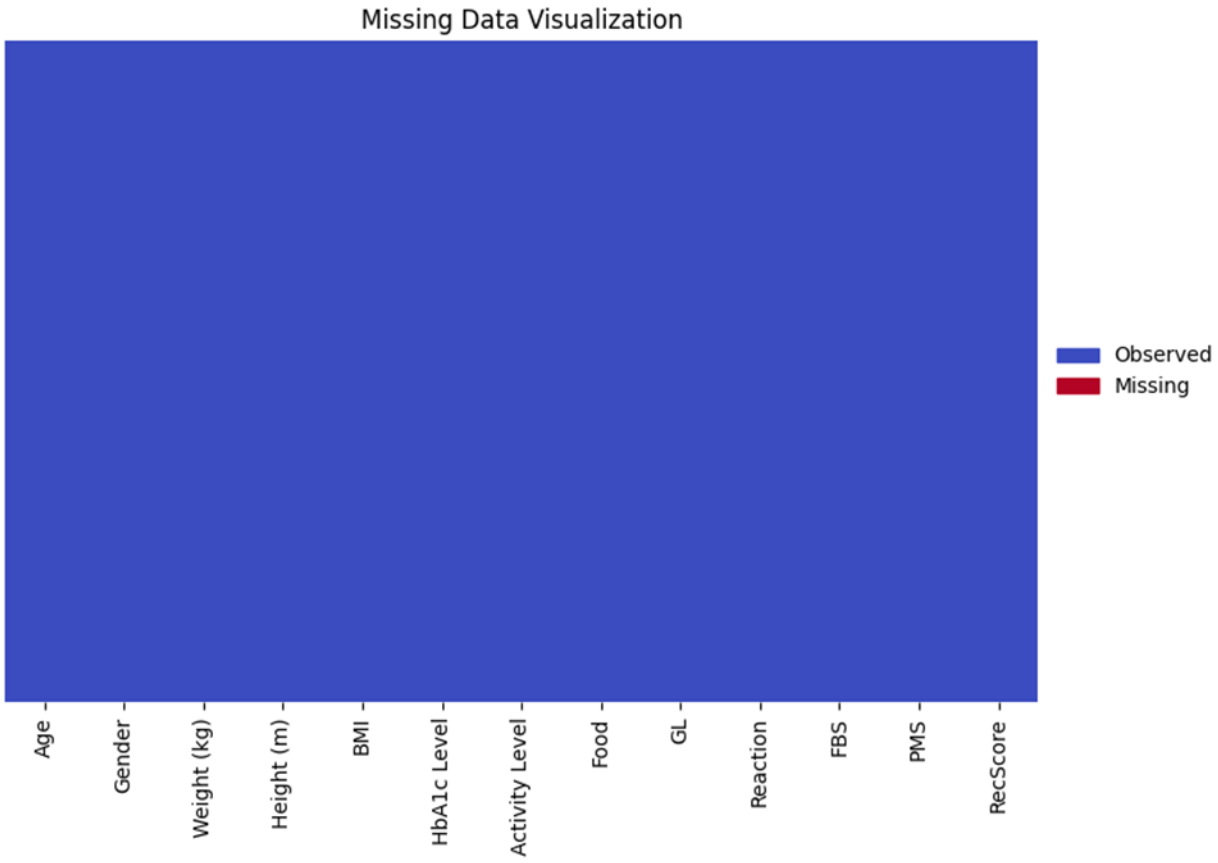
## 7.2 Pre-processing

### 7.2.1 Missing Values

**Overview:** Handling missing values is a crucial step in data preprocessing, as missing data can significantly impact model performance and the reliability of predictions. The presence of missing values can lead to biased estimates and reduce the efficiency of the model.

**Visualization:** To identify missing values in the dataset, a heatmap was generated. The heatmap provides a visual representation of the dataset, highlighting the presence of observed (blue) and missing (red) data.

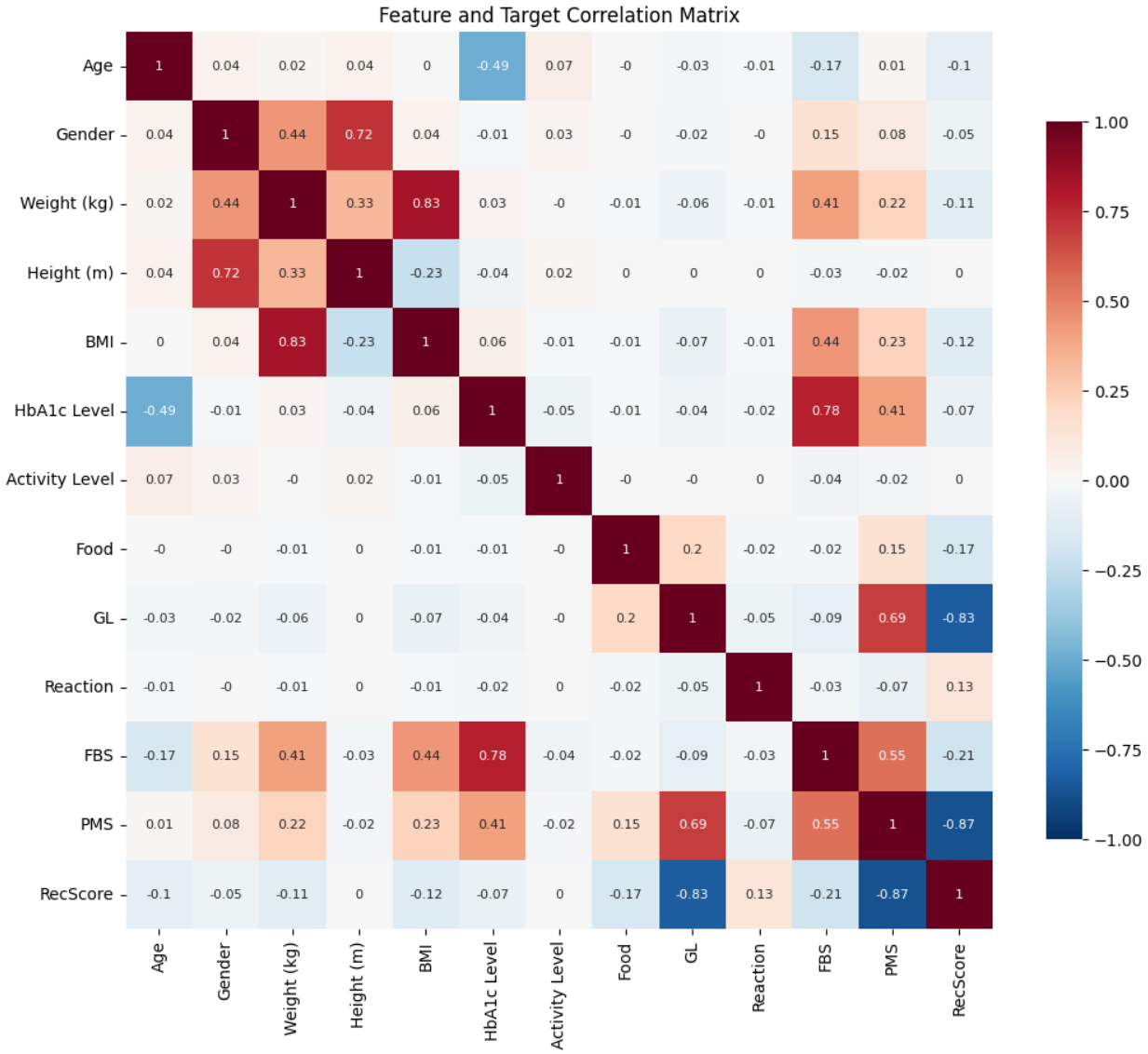
**Findings:** The heatmap clearly shows that there are no missing values in the dataset, as indicated by the uniform blue color representing observed data for all features. This step ensures that all features are complete and ready for further preprocessing without the need for imputation or deletion.




The absence of missing values in the dataset simplifies the preprocessing steps, allowing for a straightforward application of normalization and model training processes. This ensures that the model receives complete data, leading to more accurate and reliable predictions.

### 7.2.2 Correlation

We conducted a correlation analysis to understand the relationships between the features and the target variables in our dataset. The correlation matrix below shows the correlation coefficients between all features and targets. This analysis helps identify which features have significant linear relationships with the target variables, aiding in feature selection and model optimization.




1. **RecScore Correlations**:

   - **GL (Glycemic Load)**: Shows a strong negative correlation with RecScore (-0.83), indicating that higher glycemic load foods are less recommended for diabetic patients.
   - **PMS (Post-Meal Sugar)**: Also shows a strong negative correlation with RecScore (-0.87), suggesting that foods causing higher post-meal blood sugar spikes are less recommended.
   - **FBS (Fasting Blood Sugar)**: Moderate negative correlation with RecScore (-0.21), indicating that higher fasting blood sugar levels are associated with lower recommendation scores.

- **HbA1c Level**: Weak negative correlation with RecScore (-0.07), suggesting a slight trend where higher HbA1c levels correspond to lower recommendation scores.
- **Activity Level**: Very weak negative correlation with RecScore (-0.17), indicating minimal direct influence on the recommendation score.

2. **Feature Interrelationships**:

- **BMI and Weight**: Strong positive correlation (0.83), as expected, since BMI is derived from weight and height.
- **BMI and FBS**: Moderate positive correlation (0.41), suggesting that higher BMI is associated with higher fasting blood sugar levels.
- **FBS and HbA1c Level**: Strong positive correlation (0.78), indicating that higher fasting blood sugar levels correspond to higher HbA1c levels, a common metric for diabetes management.
- **GL and PMS**: Strong positive correlation (0.69), indicating that foods with a higher glycemic load tend to cause higher post-meal sugar levels.
- **Food and GL**: Moderate positive correlation (0.20), showing that certain food items are associated with specific glycemic loads.

3. **Insights for Model Improvement**:

- The strong correlations between GL, PMS, and RecScore suggest that these features are crucial for the model. Ensuring these features are accurately represented and weighted in the model will likely improve its performance.
- The moderate correlations between BMI, FBS, and HbA1c levels indicate that these features also play an important role in determining the recommendation score and should be carefully considered during feature selection and model training.

These findings highlight the importance of specific features in predicting the recommendation score for food items for diabetic patients, guiding both feature selection and model tuning to enhance predictive accuracy.

### 7.2.3 Feature Selection

**Method Used**

The method used to determine feature importance in this model is ElasticNet regression. ElasticNet is a regularized regression technique that combines both L1 (Lasso) and L2 (Ridge) penalties. This method is effective for feature selection as it can both reduce coefficients of irrelevant features to zero (L1 penalty) and handle multicollinearity (L2 penalty). The combination of these penalties makes ElasticNet a robust choice for identifying important features in datasets with many correlated predictors.

**Rationale for Choosing ElasticNet**

The choice of ElasticNet for feature importance analysis is based on several factors:

1. **Feature Selection**: The L1 penalty in ElasticNet can shrink the coefficients of less important features to zero, effectively performing feature selection.
2. **Handling Multicollinearity**: The L2 penalty helps in managing multicollinearity by distributing the penalty across correlated features.
3. **Regularization**: ElasticNet provides a balanced approach to regularization, avoiding the overfitting common in models without regularization.

**Implementation and Results**

The implementation of ElasticNet for feature importance analysis was carried out using the following steps:

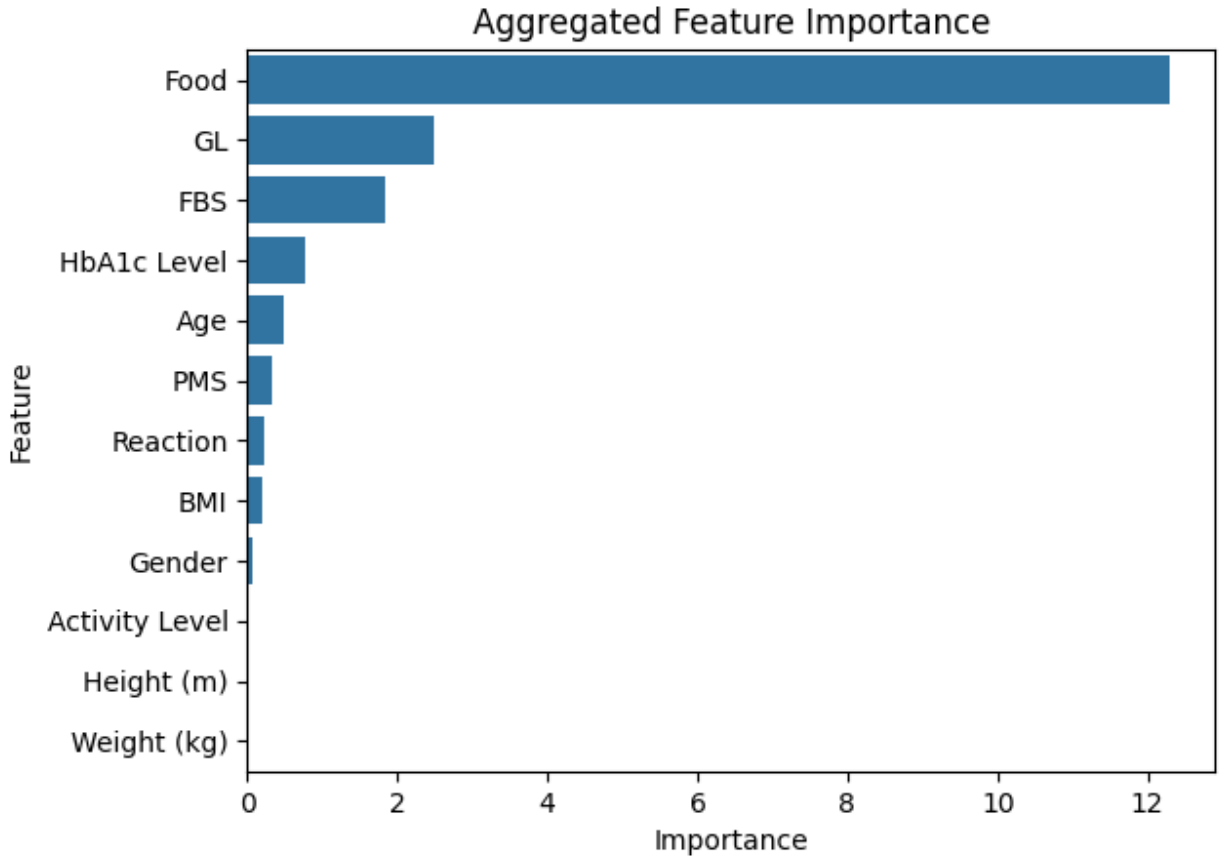
1. **Data Loading and Preprocessing**:
   - The dataset was loaded and any 'Unnamed' columns were dropped.
   - Categorical features ('Gender', 'Activity Level', 'Food') were identified for one-hot encoding.
   - A preprocessing pipeline was created to one-hot encode the categorical features and standardize the numerical features.
2. **Data Splitting**:
   - The dataset was split into training and testing sets with a 80-20 ratio.
3. **Model Training**:
   - ElasticNetCV, which performs cross-validated ElasticNet regression, was used to fit the training data.
   - The coefficients from the trained model were extracted to determine the importance of each feature.
4. **Aggregation of Feature Importances**:
   - The importance scores were aggregated for the original categorical features to get a holistic importance score.
5. **Results**:
   - The aggregated feature importance scores were stored in a DataFrame and sorted in descending order of importance.

Here is the aggregated feature importance as determined by ElasticNet:

| **Feature** | **Importance** |
|---|---|
| Food | 12.289500 |
| GL | 2.484603 |
| FBS | 1.833962 |
| HbA1c Level | 0.770214 |
| Age | 0.481389 |
| PMS | 0.330295 |
| Reaction | 0.215866 |
| BMI | 0.204538 |
| Gender | 0.075869 |
| Activity Level | 0.012357 |
| Height (m) | 0.005102 |
| Weight (kg) | 0.000000 |

These results highlight 'Food' as the most important feature by a significant margin, followed by 'GL' and 'FBS'. The least important features were 'Weight (kg)' and 'Height (m)'.

**Visualizing Feature Importance:**
The bar chart below shows the aggregated importance of each feature. Features with higher bars significantly impact predicting the target variables, while those with lower bars are less influential.


Aggregated Feature Importance
Food
GL
FBS
HbA1c Level
Age
PMS
Reaction
BMI
Gender
Activity Level
Height (m)
Weight (kg)
Feature
0
2
4
6
8
10
12
Importance

### 7.2.4 Normalization

**Normalization Process**

Normalization is a crucial step in data preprocessing, especially when dealing with features that have different scales. Without normalization, features with larger ranges could dominate the model training process, leading to biased results. For our classification model, we chose the Standard Scaler for normalization.

**Choice of Scaling Method: Standard Scaler**

The Standard Scaler was selected to ensure that each feature contributes equally to the model. This scaler standardizes the features by removing the mean and scaling to unit variance, which is particularly useful when the features have different units or scales. The formula used by the Standard Scaler is:

$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean
$\sigma =$ Standard Deviation

**Normalization Implementation**

1. **Loading the Dataset**:
    - The dataset was loaded and cleaned by removing unnamed columns and handling missing values.
2. **Splitting the Data**:
    - The dataset was split into features (X) and target (y). The features include biometric and lifestyle attributes, while the target is the recommendation score (RecScore).
3. **Handling Categorical and Numerical Features**:
    - The features were divided into categorical and numerical categories. Categorical features, such as 'Gender' and 'Food,' were handled separately from numerical features.
4. **Applying Standard Scaler**:
    - Numerical features were normalized using the Standard Scaler, ensuring each numerical feature had a mean of 0 and a standard deviation of 1. This transformation helps make the data suitable for machine learning algorithms by eliminating the bias caused by the different scales of the features.

**Benefits of Using Standard Scaler**

- **Equal Contribution**: Ensures that all features contribute equally to the model training process.
- **Improved Convergence**: Helps in faster and more stable convergence of gradient descent-based algorithms.
- **Enhanced Performance**: Reduces the impact of outliers and improves the overall performance and accuracy of the model.

## 7.3 Machine Learning Model Training

The machine learning model training process is critical to developing an effective predictive model. This section explains the model training steps, including data preprocessing, model selection, hyperparameter tuning, and validation.

### 7.3.1 Model Selection

**Overview**

Choosing the right machine learning model is crucial for achieving accurate predictions. The model selection process involves evaluating different algorithms to determine which performs best on the dataset.

**Chosen Model: XGBoost Classifier**

1. **Reason for Selection**: The XGBoost Classifier was chosen for its ability to handle large datasets, capture complex patterns, and provide robustness against overfitting. XGBoost is known for its efficiency and performance in various machine-learning tasks.

2. **Comparison with Other Models**: While other models, such as Logistic and Decision Trees, were considered, XGBoost provided superior performance and better handling of non-linear relationships in the data.

### 7.3.2 Explanation of XGBoost Classifier

**Overview**

XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm that has gained popularity for its speed and performance. It is an implementation of gradient-boosted decision trees designed for speed and performance.

**How It's Made**

1. **Gradient Boosting Framework**: XGBoost is built on the gradient boosting framework, which combines the predictions of multiple weak learners (usually decision trees) to create a strong predictive model.

2. **Decision Trees**: The core model is a decision tree, and XGBoost builds an ensemble of trees sequentially, where each tree attempts to correct the errors of the previous ones.

**How It Works**

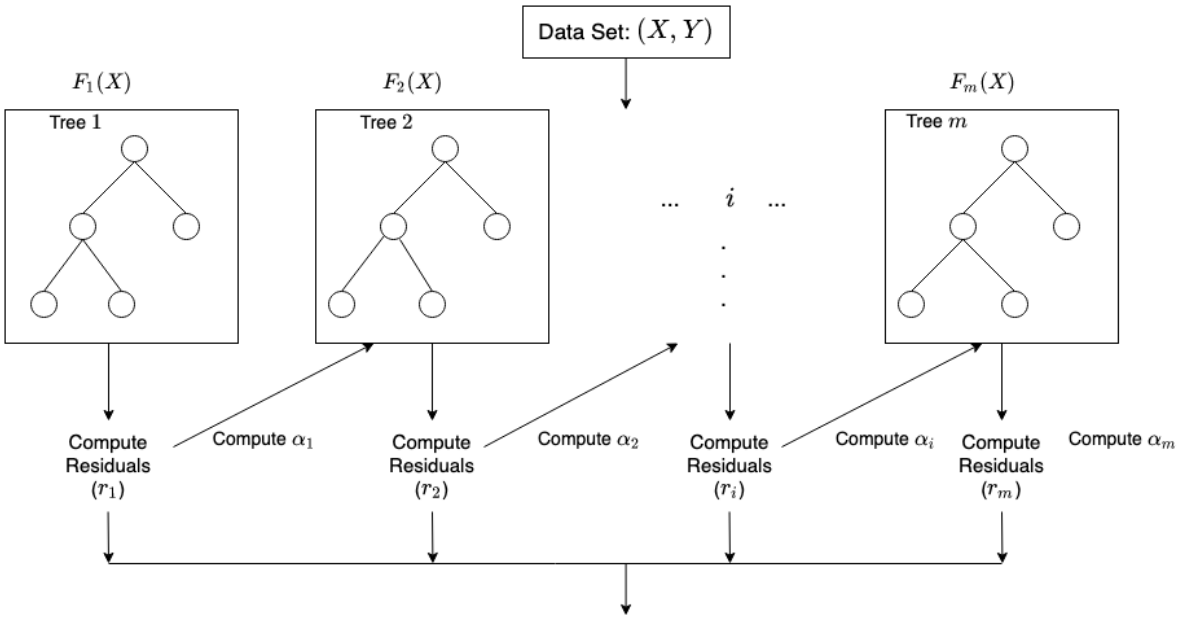
1. **Model Training**:

    - **Initialization**: Start with an initial prediction, often the mean of the target variable.
    - **Tree Building**: Sequentially add decision trees. Each tree is trained on the residual errors of the previous trees.
    - **Gradient Descent**: Use gradient descent to minimize the loss function. The gradients indicate how to adjust the model to reduce errors.
    - **Weight Updates**: Update the weights of the trees to reduce the residuals iteratively.

2. **Regularization**: XGBoost includes regularization parameters to prevent overfitting, making it robust and generalizable.

3. **Parallel Processing**: It supports parallel processing to speed up computation, making it efficient for large datasets.

## Visual Example:




$$F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

where $\alpha_i$, and $r_i$ are the regularization parameters and residuals computed with the $i^{th}$ tree respectfully, and $h_i$ is a function that is trained to predict residuals, $r_i$ using $X$ for the $i^{th}$ tree. To compute $\alpha_i$ we use the residuals computed, $r_i$ and compute the following: $arg \min_{\alpha} = \sum_{i=1}^{m} L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where $L(Y, F(X))$ is a differentiable loss function.

### 7.3.3 Hyperparameter Tuning

**Overview**

Hyperparameter tuning involves optimizing the model's parameters to improve its performance. This step is essential for achieving the best possible results from the chosen model.

**Steps**

1. **Grid Search**: Grid Search was used for hyperparameter tuning, which systematically tests combinations of parameters to find the best settings.
2. **Parameters Tuned**: Key hyperparameters tuned for the XGBoost Classifier included:
    - **n_estimators:** Number of trees (tested values: 100, 200).
    - **max_depth:** Maximum depth of each tree (tested values: 3, 5).
    - **learning_rate:** Step size shrinkage (tested values: 0.1, 0.01).
3. **Results**: The best hyperparameters were found to be:
    - **n_estimators:** 200
    - **max_depth:** 5
    - **learning_rate:** 0.1 These settings improved the model's performance, as indicated by the evaluation metrics.

### 7.3.4 Model Training

**Overview**

Model training feeds the pre-processed data into the chosen algorithm to learn patterns and make predictions.

**Steps**

1. **Training Process**: The training data was used to train the XGBoost Classifier. The model was fitted to the data using the best hyperparameters identified through Grid Search.
2. **Cross-Validation**: Five-fold cross-validation was used to ensure the model generalizes well to unseen data. This involves splitting the training data into five subsets, training on four subsets, and validating on the remaining subset. This process is repeated five times, each time with a different validation set, to ensure robustness.

### 7.3.5 Model Evaluation

#### Overview
Evaluating the trained model is crucial to understanding its performance and identifying any potential issues.

#### Metrics Used

- **Accuracy**: Measures the proportion of correctly classified instances out of the total instances.
- **Precision**: Measures the proportion of true positive instances out of the instances predicted as positive.
- **Recall**: Measures the proportion of true positive instances out of the actual positive instances.
- **F1 Score**: Harmonic mean of precision and recall, providing a single metric that balances both concerns.

#### Evaluation Process
- **Training Set Performance**: The model was evaluated on the training set to check for any signs of overfitting or underfitting.
- **Validation Set Performance**: Cross-validation results ensured the model's generalization ability to new data.
- **Interpretation**: The evaluation metrics showed that the model was well-fitting, with high accuracy, precision, recall, and F1 score on the test set.

#### Evaluation Results
- **Test Accuracy**: 0.9260
- **Test Precision**: 0.9263
- **Test Recall**: 0.9260
- **Test F1 Score**: 0.9261

### 7.3.6 Model Testing

**Overview**

Model testing involves assessing the model's performance on a separate test set not used during training or validation. This step helps understand the model's real-world applicability.

**Steps**

1. **Test Set Evaluation**: The model was evaluated using the same metrics on a separate test set.
2. **Comparison with Training/Validation Performance**: The test set results were compared with the training and validation performance to identify any overfitting or underfitting issues.

**Test Evaluation Results**

- **Test Accuracy**: 0.9260
- **Test Precision**: 0.9263
- **Test Recall**: 0.9260
- **Test F1 Score**: 0.9261

## 7.4 Results

The results section provides a comprehensive overview of the model's performance, presented through tables and figures. This section helps in understanding the trained model's effectiveness and predictive accuracy.

### 7.4.1 Tables

**Overview**
The tables below summarize the model's evaluation metrics on both the training and test datasets. These metrics include Accuracy, Precision, Recall, and F1 Score.

**Training and Test Set Performance:**

| Metric | Training Set | Test Set |
|---|---|---|
| Accuracy | 0.9258 | 0.9260 |
| Precision | 0.9260 | 0.9263 |
| Recall | 0.9258 | 0.9260 |
| F1 Score | 0.9259 | 0.9261 |

**Interpretation:**

- **Accuracy**: Measures the proportion of correctly classified instances out of the total instances. Higher accuracy indicates better model performance.
- **Precision**: Measures the proportion of true positive instances out of the instances predicted as positive. Higher precision means fewer false positives.
- **Recall**: Measures the proportion of true positive instances out of the actual positive instances. Higher recall means fewer false negatives.
- **F1 Score**: The harmonic mean of precision and recall. A higher F1 score indicates a balance between precision and recall, providing a single metric that balances both concerns.

**Comparison of Model Performance:**

The table below compares the accuracies of all the models tried during the project. This comparison helps understand the relative performance of different algorithms.

| Model Name | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Logistic | 0.7403 | 0.7386 | 0.7403 | 0.7389 |
| Naive Bayes | 0.4734 | 0.4938 | 0.4734 | 0.4721 |
| K-Nearest Neighbors | 0.7763 | 0.7814 | 0.7763 | 0.7778 |
| Support Vector Machine | 0.8159 | 0.8171 | 0.8159 | 0.8159 |
| Decision Tree | 0.8652 | 0.8654 | 0.8652 | 0.8653 |
| Random Forest | 0.9245 | 0.9248 | 0.9245 | 0.9246 |

| | | | | |
|---|---|---|---|---|
| Gradient Boosting Machine | 0.4793 | 0.3548 | 0.4793 | 0.3493 |
| AdaBoost | 0.2498 | 0.1273 | 0.2498 | 0.1486 |
| LightGBM | 0.8836 | 0.8840 | 0.8836 | 0.8837 |
| XGBoost Classifier | 0.9260 | 0.9263 | 0.9260 | 0.9261 |

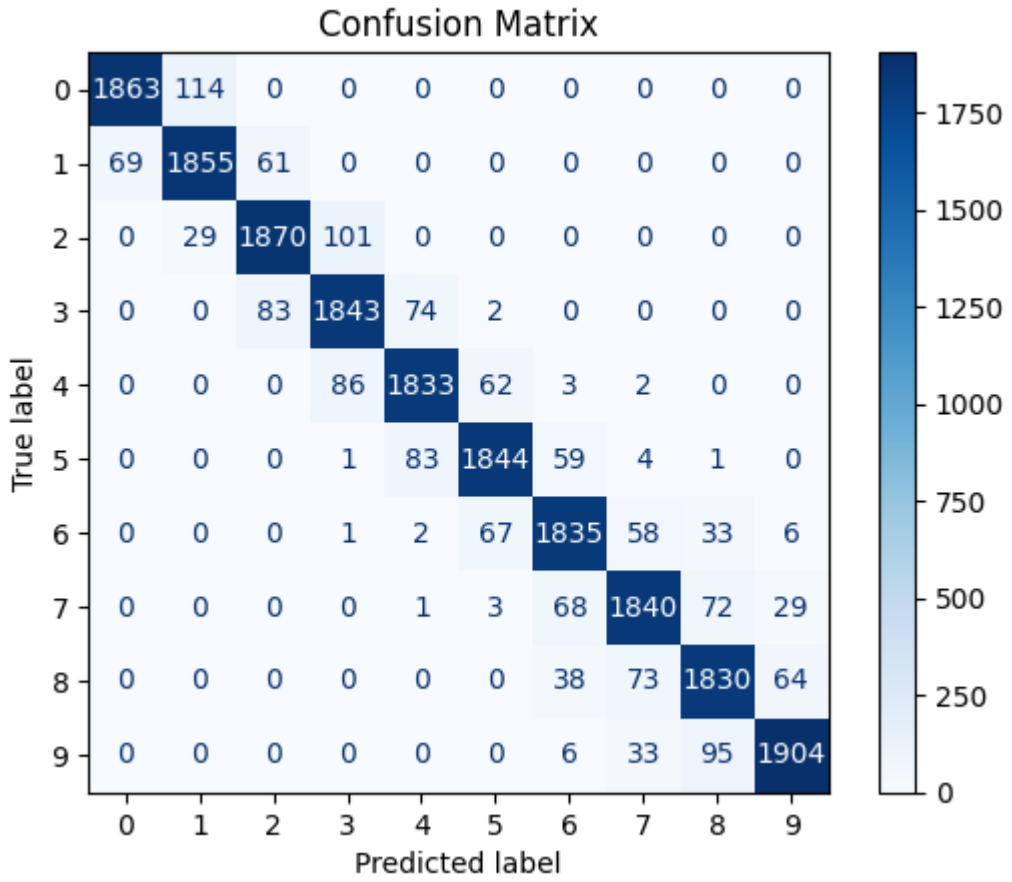
## Interpretation:

- **XGBoost Classifier**: Achieved the highest accuracy (0.9260), precision (0.9263), recall (0.9260), and F1 score (0.9261), making it the best-performing model among those tested.
- **Random Forest and LightGBM**: Also performed well but were slightly less accurate than XGBoost.
- **Naive Bayes and AdaBoost**: The lowest accuracy and error metrics indicated they were the least effective models for this dataset.

### 7.4.2 Figures

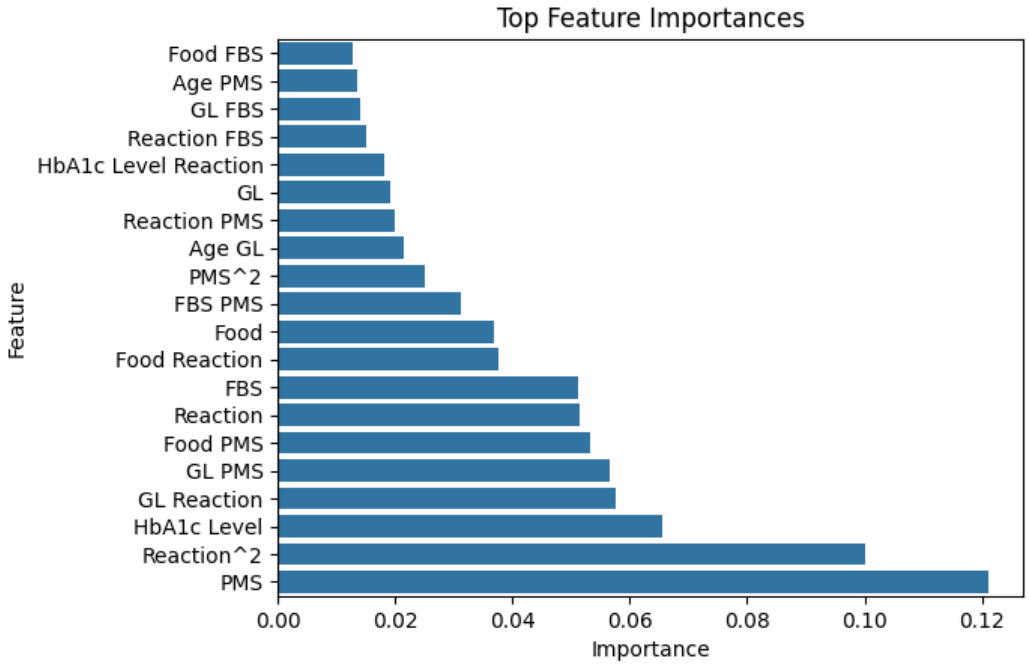
1. **Confusion Matrix**

The confusion matrix provides a visual representation of the model's performance by showing the true positive, true negative, false positive, and false negative predictions.



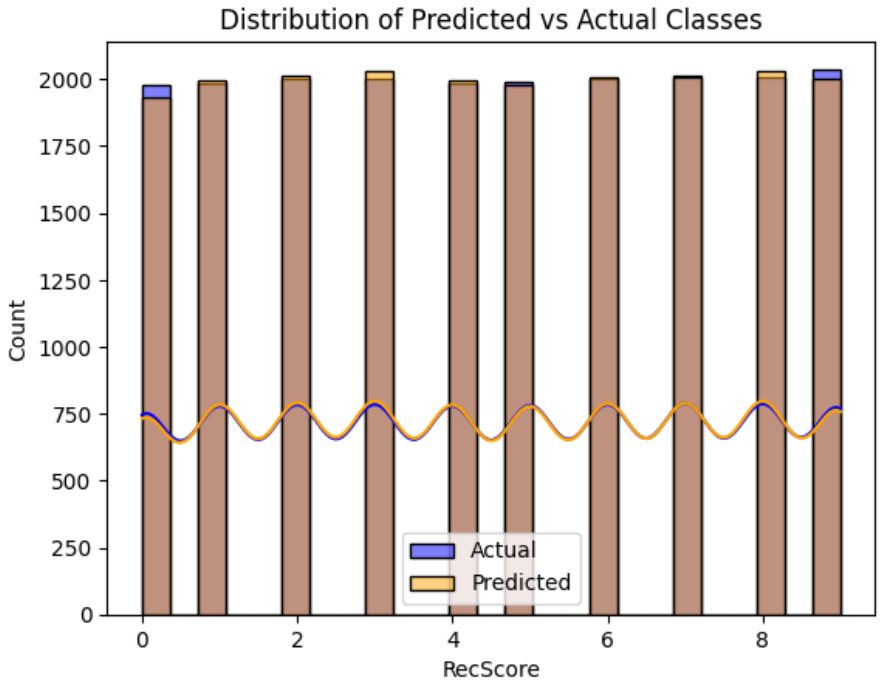

2. **Feature Importance**

The feature importance plot shows the contribution of each feature to the model's predictions, highlighting which features were most influential.




3. **Distribution of Predicted vs. Actual Classes**

This plot compares the distribution of predicted classes to the actual classes, providing insight into how well the model's predictions align with the true labels.



## 7.5 Discussion

The discussion section provides a comprehensive analysis of the model-building process, wrapping up everything done in the model-development process. It highlights the best and worst-performing algorithms, reasons for their performance, and insights for future work. This critical evaluation helps understand the chosen approach's strengths and limitations and identifies improvement opportunities.

**Best Performing Algorithm: XGBoost Classifier**

The XGBoost Classifier emerged as the best-performing model in our project. Its evaluation metrics on the test set were as follows:

- **Accuracy**: 0.9260
- **Precision**: 0.9263
- **Recall**: 0.9260
- **F1 Score**: 0.9261

**Reasons for Superior Performance:**

1. **Complex Pattern Recognition**: XGBoost is highly effective at capturing complex, non-linear relationships within the data. This capability was crucial given our dataset's diverse biometric and lifestyle features.
2. **Regularization**: Including L1 and L2 regularization techniques helped mitigate overfitting, ensuring the model's generalization to new, unseen data.
3. **Gradient Boosting Framework**: The gradient boosting mechanism builds an ensemble of weak learners, each correcting the errors of the previous ones. This iterative improvement contributed significantly to the model's robustness.
4. **Hyperparameter Optimization**: Extensive hyperparameter tuning using Grid Search allowed the identification of optimal parameters, enhancing model performance.
5. **Parallel Processing**: XGBoost's support for parallel processing made it efficient and capable of handling large datasets swiftly.

These features collectively made XGBoost the most reliable and accurate model for predicting the suitability of food items for individuals based on their biometric and lifestyle data.

**Least Performing Algorithm: Naive Bayes**

In contrast, the Naive Bayes model demonstrated the poorest performance with the following metrics:

- **Accuracy**: 0.4734
- **Precision**: 0.4938
- **Recall**: 0.4734
- **F1 Score**: 0.4721

### Reasons for Poor Performance:

1. **Simplistic Assumptions**: Naive Bayes assumes independence between features, which is often unrealistic for complex datasets with interdependent features.
2. **Sensitivity to Feature Scaling**: Despite preprocessing steps, Naive Bayes struggled with variations in feature scales and distributions.
3. **Lack of Complexity Handling**: The simplicity of Naive Bayes in dealing with non-linear relationships and interactions among features was inadequate compared to more sophisticated models like XGBoost.
4. **Over-Simplification**: The model's assumption that all features contribute equally to the outcome often misrepresents the true importance of different features.

These limitations made Naive Bayes unsuitable for our predictive task, highlighting the need for more advanced algorithms for complex datasets.

### Key Insights:

- **Tree-Based Models**: Random Forest and LightGBM also performed well but were slightly less accurate than XGBoost. Their ability to handle non-linear relationships and ensemble nature contributed to their robustness.
- **Linear Models**: Logistic and others performed moderately well but were less capable of capturing non-linear patterns than tree-based models.
- **Support Vector Machine (SVM)**: SVM performed well but was computationally intensive compared to XGBoost.
- **K-Nearest Neighbors (KNN)**: Demonstrated moderate performance, but its simplicity and sensitivity to the high-dimensional space limited its effectiveness.

### Future Work and Improvements

### Potential Improvements:

1. **Feature Engineering**: Creating new features or transforming existing ones could help improve model performance. Incorporating interaction terms or higher-order polynomial features might capture more complex patterns.
2. **Algorithm Exploration**: Exploring advanced algorithms like LightGBM or CatBoost, similar to XGBoost, but may offer improved performance or efficiency in specific scenarios, could be beneficial.
3. **Hyperparameter Tuning**: More sophisticated hyperparameter tuning techniques, such as Bayesian Optimization or Random Search, might yield better results than Grid Search.
4. **Ensemble Methods**: Combining multiple models to form an ensemble could leverage the strengths of different algorithms, potentially improving overall performance.

### Future Directions:

1. **Expand Dataset**: Collecting more data, especially with a broader range of variations, can enhance the model's robustness and generalizability.

2. **Real-Time Predictions**: Implementing the model in a real-time system for live dietary recommendations could provide valuable user feedback and refine the model.

3. **User Personalization**: Enhancing the model to incorporate user preferences and feedback for more personalized dietary recommendations.

## Conclusion

The XGBoost Classifier demonstrated exceptional performance, making it the best choice for predicting the suitability of food items based on biometric and lifestyle data. The Naive Bayes model, while simple, struggled with the high-dimensional data and simplistic assumptions. By leveraging the strengths of advanced algorithms and exploring potential improvements, the predictive model can be further refined to provide even more accurate and personalized dietary recommendations. The insights gained from this project pave the way for future enhancements and practical applications in personalized nutrition planning.

# Chapter 8. Test Specification and Results

## 8.1 Test Case Specification for User

**Table 8.1.1: TC-1 Successful Login**

| | |
|---|---|
| **Identifier** | TC-1 |
| **Related requirements(s)** | UC-2 User Login |
| **Short description** | Verify that the system permits login with correct credentials. |
| **Pre-condition(s)** | 1. User must have an active account.<br>2. Internet connectivity required. |
| **Input data** | Email Address and Password. |
| **Detailed steps** | 1. User launches the mobile application.<br>2. Input valid email and password in the relevant fields.<br>3. Tap the Login button. |
| **Expected result(s)** | The system checks credentials and redirects them to the Home Page upon success. |
| **Post-condition(s)** | 1. The user's session is active.<br>2. User can access personal data and app features. |
| **Actual result(s)** | User is logged in and able to access data and features without any issues. |
| **Test Case Result** | Pass |

**Table 8.1.2: TC-2 Unsuccessful Login with Incorrect Email or Password**

| | |
|---|---|
| **Identifier** | TC-2 |
| **Related requirements(s)** | UC-2 User Login |
| **Short description** | Ensure the system rejects login attempts with an incorrect password. |
| **Pre-condition(s)** | 1. The user must have an active account registered in the system.<br>2. Internet connectivity required. |
| **Input data** | Provide either an incorrect email, password, or both, but never both correct. |
| **Detailed steps** | 1. User launches the mobile application.<br>2. Input an email (valid or invalid) and password (valid or invalid) in the relevant fields. One of them should be invalid for this test case.<br>3. Tap the Login button. |
| **Expected result(s)** | The system checks credentials and identifies the credentials are mismatched and displays an error message “Incorrect credentials. Please try again.”. |
| **Post-condition(s)** | 1. User session not initiated.<br>2. User remains logged out.<br>3. User is prompted to retry entering credentials. |
| **Actual result(s)** | An error message about incorrect credentials was displayed correctly. The user did not gain access to the system. |
| **Test Case Result** | Pass |

**Table 8.1.3: TC-3 Successful Signup**

| | |
|---|---|
| **Identifier** | TC-3 |
| **Related requirements(s)** | UC-1 User Signup |
| **Short description** | To ensure a new user can successfully create an account by providing all required information. |
| **Pre-condition(s)** | 1. The user has no prior registration with the system.<br>2. Internet connectivity required. |
| **Input data** | Valid email, password, and additional information. |
| **Detailed steps** | 1. User launches the mobile application.<br>2. Tap on the 'Register' button besides the "Don't have an account?" message.<br>3. Input all mandatory fields with correct formats in the signup form.<br>4. Tap on the 'Signup' button. |
| **Expected result(s)** | 1. The system performs validation checks to confirm that all inputs meet specified criteria without errors.<br>2. A confirmation alert or page confirms "Registration Successful."<br>3. The system redirects the user to the login interface to enable secure access. |
| **Post-condition(s)** | 1. User credentials are successfully entered within the database.<br>2. User must be able to Login using their registered credentials. |
| **Actual result(s)** | The signup was completed successfully, with a correct redirection to the login interface and user can login with the registered credentials. |
| **Test Case Result** | Pass |

**Table 8.1.4: TC-4 Registration with Existing Email**

| | |
|---|---|
| **Identifier** | TC-4 |
| **Related requirements(s)** | UC-1 User Signup |
| **Short description** | To ensure the system correctly prevents registration with an email address that is already registered. |
| **Pre-condition(s)** | 1. The user must have a registered account in the system.<br>2. Internet connectivity required. |
| **Input data** | Already registered email address. |
| **Detailed steps** | 1. User launches the mobile application.<br>2. Tap on the 'Register' button besides the "Don't have an account?" message.<br>3. Input an already registered email and other information within relevant fields.<br>4. Tap on the 'Signup' button. |
| **Expected result(s)** | 1. The system detects the email is already registered to an account.<br>2. An error message is displayed, "Email address already in use. Please log in or use a different email address." |
| **Post-condition(s)** | 1. No new user account is created.<br>2. The system remains on the registration page, prompting the user to change the input data. |
| **Actual result(s)** | Error message displayed about email already in use. |
| **Test Case Result** | Pass |

**Table 8.1.5: TC-5 View Profile**

| | |
|---|---|
| **Identifier** | TC-5 |
| **Related requirements(s)** | UC-4 User Profile |
| **Short description** | To ensure that the user can successfully view their profile information within the application. |
| **Pre-condition(s)** | 1. User must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Profile page.<br>2. Profile information will be displayed by the system. |
| **Expected result(s)** | User Profile information is displayed correctly by the system. |
| **Post-condition(s)** | Users can view their correct profile information. |
| **Actual result(s)** | The user's profile was accessible and displayed all details as expected. |
| **Test Case Result** | Pass |

**Table 8.1.6: TC-6 Edit Profile**

| | |
|---|---|
| **Identifier** | TC-6 |
| **Related requirements(s)** | UC-4 User Profile |
| **Short description** | To ensure the system's functionality for editing and updating user profile details. |
| **Pre-condition(s)** | 1. User must be logged in.<br>2. Internet connectivity required.<br>3. User must be on the Profile page. |
| **Input data** | Updated profile information (e.g., new phone number, name, dob). |
| **Detailed steps** | 1. Navigate to user's profile page.<br>2. Click on the 'Edit Profile' button.<br>3. Click on 'Save changes' button. |
| **Expected result(s)** | The profile is immediately updated with the new information and the changes are visibly confirmed on the profile page upon saving. |
| **Post-condition(s)** | Updated information is successfully stored in the database. |
| **Actual result(s)** | The profile update process completed successfully, and the updated information is visible on the profile page. |
| **Test Case Result** | Pass |

**Table 8.1.7: TC-7 Logout Functionality**

| | |
|---|---|
| **Identifier** | TC-7 |
| **Related requirements(s)** | UC-8 User Logout |
| **Short description** | To ensure that the user can log out successfully and their session is terminated. |
| **Pre-condition(s)** | User should be logged in. |
| **Input data** | None. |
| **Detailed steps** | Click on the logout button. |
| **Expected result(s)** | User is logged out and redirected to the login page. |
| **Post-condition(s)** | Session is terminated. |
| **Actual result(s)** | User is logged out and redirected to the login page, confirming session termination. |
| **Test Case Result** | Pass |

**Table 8.1.8: TC-8 Update Password**

| | |
|---|---|
| **Identifier** | TC-8 |
| **Related requirements(s)** | Use-case for password update |
| **Short description** | Test updating the password successfully from the settings. |
| **Pre-condition(s)** | 1. User must be logged in.<br>2. Internet connectivity required. |
| **Input data** | Current valid password and new password. |
| **Detailed steps** | 1. Navigate to the Settings page.<br>2. Select the 'Change Password' option.<br>3. Enter current and new passwords in the respective fields.<br>4. Click on the ‘Save’ button. |
| **Expected result(s)** | 1. The system verifies the current password.<br>2. The new password overwrites the old password successfully.<br>3. A confirmation message is displayed indicating the password has been updated. |
| **Post-condition(s)** | User needs to use new password for subsequent logins and previous passwords are invalid. |
| **Actual result(s)** | User is required to use the new password for subsequent logins. |
| **Test Case Result** | Pass |

**Table 8.1.9: TC-9 View Health Profile**

| | |
|---|---|
| **Identifier** | TC-9 |
| **Related requirements(s)** | UC-5 Health Profile |
| **Short description** | Test the functionality that allows users to access and view their health profile information. |
| **Pre-condition(s)** | 1. User must be logged in.<br>2. There must be prior recorded health data of a user. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Health Profile page.<br>2. Observe the display of health information, including health metrics and health status. |
| **Expected result(s)** | The application displays accurate and complete health data without any delay or inconsistently. |
| **Post-condition(s)** | The health profile data is displayed that remains unchanged unless edited in a separate operation. |
| **Actual result(s)** | User's health details were correctly displayed and matched the data previously entered/recorded. |
| **Test Case Result** | Pass |

**Table 8.1.10: TC-10 Update Health Profile**

| | |
|---|---|
| **Identifier** | TC-10 |
| **Related requirements(s)** | UC-5 Health Profile |
| **Short description** | To ensure the application allows users to update and save changes to their health profile data successfully. |
| **Pre-condition(s)** | 1. The user has successfully logged into their account.<br>2. The user has existing health profile data available for editing. |
| **Input data** | Updated health details (e.g. blood sugar, HbA1c, weight, height) |
| **Detailed steps** | 1. Navigate to health profile<br>2. Modify the necessary health details in the appropriate fields.<br>3. Confirm the changes by clicking the 'Save' or 'Update' button. |
| **Expected result(s)** | A confirmation message verifies successful profile updates, and the new details are instantly visible on the health profile page. |
| **Post-condition(s)** | Updated data is stored in the database. |
| **Actual result(s)** | Health profile is updated and displayed accurately. |
| **Test Case Result** | Pass |

**Table 8.1.11: TC-11 Activity Track**

| | |
|---|---|
| **Identifier** | TC-11 |
| **Related requirements(s)** | UC for Activity Track |
| **Short description** | To ensure the ability of the system to display saved daily activity track. |
| **Pre-condition(s)** | User must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Activity Track Page.<br>2. Observe the graphical representation of activities (e.g. daily step counts, sleep patterns). |
| **Expected result(s)** | The system displays the activity profile with all recorded daily physical activities. |
| **Post-condition(s)** | The user can view their activity tracking details without system errors or inconsistently. |
| **Actual result(s)** | The activity profile was successfully accessed and displayed the user's daily physical activities as expected. |
| **Test Case Result** | Pass |

**Table 8.1.12: TC-12 View Meal Plan**

| | |
|---|---|
| **Identifier** | TC-12 |
| **Related requirements(s)** | UC-3 Meal Planning |
| **Short description** | To ensure that user can access and view their previously created meal plans. |
| **Pre-condition(s)** | 1. User must be logged in.<br>2. User has requested a meal plan previously. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Meal Plan page<br>2. Select the "View Meal Plan" option to view the current meal plan. |
| **Expected result(s)** | The application displays the meal plan clearly, showing all details such as meal types, scheduled times, and nutritional information. |
| **Post-condition(s)** | The Meal plan remains viewable and can be followed by the user. |
| **Actual result(s)** | The meal plan was successfully retrieved and displayed as expected, with all details clearly visible. |
| **Test Case Result** | Pass |

**Table 8.1.13: TC-13 Generate Meal Plan**

| | |
|---|---|
| **identifier** | TC-13 |
| **Related requirements(s)** | UC-3 Meal Planning |
| **Short description** | Ensure the application can generate a personalized meal plan tailored to the user's entered health data. |
| **Pre-condition(s)** | 1. User must be logged in.<br>2. User must have entered relevant health information and preferences in their profile. |
| **Input data** | Health metrics and preferences influence meal planning. |
| **Detailed steps** | 1. Navigate to Meal Plan page.<br>2. Select the “Generate New Meal Plan” option to get a new meal plan. |
| **Expected result(s)** | A new meal plan, customized according to the provided health data, is created and displayed to the user. |
| **Post-condition(s)** | Meal plan is viewable and can be followed by the user. |
| **Actual result(s)** | The system successfully generated a meal plan, but it contained inconsistencies or errors relative to the user’s health data inputs. |
| **Test Case Result** | Fail |

**Table 8.1.14: TC-14 View Blogs**

| | |
|---|---|
| **Identifier** | TC-14 |
| **Related requirements(s)** | UC for Blogs |
| **Short description** | To ensure that users can view blogs and navigate to full articles. |
| **Pre-condition(s)** | User must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Blog Page.<br>2. Click on a blog summary to read it.<br>3. Use the provided link or button to access the full article source. |
| **Expected result(s)** | The blog summaries are displayed which a link that redirects to the full article. |
| **Post-condition(s)** | Users have viewed the blog summary and has the option to read the full article via an external link. |
| **Actual result(s)** | The blog summaries are accessible, accurately displayed, and the redirection link to the full articles functions correctly. |
| **Test Case Result** | Pass |

## 8.2 Test Case Specification for Admin

**Table 8.2.1: TC-15 Successful Login**

| | |
|---|---|
| **Identifier** | TC-15 |
| **Related requirements(s)** | UC-7 Admin Login |
| **Short description** | To ensure that the system permits login with correct credentials. |
| **Pre-condition(s)** | 1. Admin must have an active account in the system.<br>2. Internet connectivity required. |
| **Input data** | Email Address and Password. |
| **Detailed steps** | 1. Admin launches the web application on their browser.<br>2. Input valid email and password in the relevant fields.<br>3. Tap the Login button. |
| **Expected result(s)** | The system checks credentials and redirects them to the Dashboard Page upon success. |
| **Post-condition(s)** | 1. The admin's session is active.<br>2. Admin can access personal data and app features. |
| **Actual result(s)** | Admin is logged in and able to access data and features without any issues. |
| **Test Case Result** | Pass |

**Table 8.2.2: TC-16 Unsuccessful Login with Incorrect Email or Password**

| | |
|---|---|
| **Identifier** | TC-16 |
| **Related requirements(s)** | UC-7 Admin Login |
| **Short description** | Ensure the system rejects login attempts with an incorrect password. |
| **Pre-condition(s)** | 1. The admin must have an active account in the system.<br>2. Internet connectivity required. |
| **Input data** | Provide either an incorrect email, password, or both, but never both correct. |
| **Detailed steps** | 1. Admin launches the web application on their browser.<br>2. Input an email (valid or invalid) and password (valid or invalid) in the relevant fields. One of them should be invalid for this test case.<br>3. Tap the Login button. |
| **Expected result(s)** | The system checks credentials and identifies the credentials are mismatched and displays an error message “Incorrect credentials. Please try again.”. |
| **Post-condition(s)** | 1. Admin session not initiated.<br>2. Admin remains logged out.<br>3. Admin is prompted to retry entering credentials. |
| **Actual result(s)** | An error message about incorrect credentials was displayed correctly. The admin did not gain access to the system. |
| **Test Case Result** | Pass |

**Table 8.2.3: TC-17 View Profile**

| | |
|---|---|
| **Identifier** | TC-17 |
| **Related requirements(s)** | UC-15 Admin Profile |
| **Short description** | To ensure that the admin can successfully view their profile information within the application. |
| **Pre-condition(s)** | 1. User must be logged in.<br>2. Internet connectivity required. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Profile page.<br>2. Profile information will be displayed by the system. |
| **Expected result(s)** | Admin Profile information is displayed correctly by the system. |
| **Post-condition(s)** | Admin can view his profile information. |
| **Actual result(s)** | The admin profile was accessible and displayed all details as expected. |
| **Test Case Result** | Pass |

**Table 8.2.4: TC-18 Edit Profile**

| | |
|---|---|
| **Identifier** | TC-18 |
| **Related requirements(s)** | UC-15 Admin Profile |
| **Short description** | To ensure the system's functionality for editing and updating admin profile details. |
| **Pre-condition(s)** | 1. Admin must be logged in.<br>2. Internet connectivity required.<br>3. Admin must be on the Profile page. |
| **Input data** | Updated profile information (e.g., new phone number, name,). |
| **Detailed steps** | 1. Navigate to admin's profile page.<br>2. Click on the 'Edit Profile' button.<br>3. Click on 'Save changes' button. |
| **Expected result(s)** | The profile is immediately updated with the new information and the changes are visibly confirmed on the profile page upon saving. |
| **Post-condition(s)** | Updated information is successfully stored in the database. |
| **Actual result(s)** | The profile update process was completed successfully, and the updated information is visible on the profile page. |
| **Test Case Result** | Pass |

**Table 8.2.5: TC-19 Logout Functionality**

| | |
|---|---|
| **Identifier** | TC-19 |
| **Related requirements(s)** | UC-27 Admin Logout |
| **Short description** | To ensure that admin can log out successfully and his session is terminated. |
| **Pre-condition(s)** | Admin should be logged in. |
| **Input data** | None. |
| **Detailed steps** | Click on the logout button. |
| **Expected result(s)** | Admin is logged out and redirected to the login page. |
| **Post-condition(s)** | Session is terminated. |
| **Actual result(s)** | Admin is logged out and redirected to the login page, confirming session termination. |
| **Test Case Result** | Pass |

**Table 8.2.6: TC-20 Admin Dashboard**

| | |
|---|---|
| **Identifier** | TC-20 |
| **Related requirements(s)** | UC-8 Admin Dashboard |
| **Short description** | To evaluate the system's ability to accurately display essential system statistics and user activity. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. If user logins, they are redirected to the dashboard by default. Otherwise, they can navigate to Dashboard page themselves.<br>2. Observe the graphical representation of system activity (e.g. total users, geographics of users, age distributions, genders). |
| **Expected result(s)** | The dashboard displays up-to-date statistics without any delays or errors. |
| **Post-condition(s)** | The administrator retains access to the dashboard, which consistently displays system metrics. |
| **Actual result(s)** | The dashboard was accessible, and all visual data representations were correct and current as per the latest system data. |
| **Test Case Result** | Pass |

**Table 8.2.7: TC-21 View Users**

| | |
|---|---|
| **Identifier** | TC-21 |
| **Related requirements(s)** | UC-9 User Management |
| **Short description** | To ensure the user management section accurately displays latest user data. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to the Users Table page.<br>2. View the personal and health information of each user in a table format (e.g. name, email, dob, country, gender) |
| **Expected result(s)** | All user information displayed in the 'User Table' is accurate and up to date, reflecting any recent changes. |
| **Post-condition(s)** | The admin continues to have access to the updated 'User Table' with the correct details after any changes are made. |
| **Actual result(s)** | The 'User Table' displayed the latest information for all users accurately during the test. |
| **Test Case Result** | Pass |

**Table 8.2.8: TC-22 Edit User**

| | |
|---|---|
| **Identifier** | TC-22 |
| **Related requirements(s)** | UC-9 User Management |
| **Short description** | To test the functionality that allows an admin to edit user details. |
| **Pre-condition(s)** | 1. Admin must be logged in.<br>2. Admin must be on the Users Table page. |
| **Input data** | None |
| **Detailed steps** | 1. Click on the 'Edit' icon next to the user's row you want to modify.<br>2. Update the desired fields.<br>3. Save the changes by clicking the 'Save' button. |
| **Expected result(s)** | The system allows modifications to the selected user's details and updates the information in the user table upon saving. |
| **Post-condition(s)** | Changes are accurately saved and displayed, and the admin retains editing capabilities. |
| **Actual result(s)** | User details were successfully updated and accurately reflected in the user table. |
| **Test Case Result** | Pass |

**Table 8.2.9: TC-23 Delete User**

| | |
|---|---|
| **Identifier** | TC-23 |
| **Related requirements(s)** | UC-9 User Management |
| **Short description** | To test the admin's ability to delete user profiles. |
| **Pre-condition(s)** | 1. Admin must be logged in.<br>2. Admin must be on the Users Table page. |
| **Input data** | None |
| **Detailed steps** | 1. Click the 'Delete' icon next to the user's row you intend to remove<br>2. Confirm the deletion by clicking on the Confirm' button. |
| **Expected result(s)** | The user profile is removed from the system immediately after confirmation. The user may no longer able to login into the system. |
| **Post-condition(s)** | The user is no longer listed in the user table, and the admin retains the ability to manage remaining profiles. |
| **Actual result(s)** | The selected user was successfully deleted from the system, and the user table was updated accordingly. |
| **Test Case Result** | Pass |

**Table 8.2.10: TC-24 View Foods**

| | |
|---|---|
| **Identifier** | TC-24 |
| **Related requirements(s)** | UC-10 Food Management |
| **Short description** | To test the admin's ability to access and review the list of food items. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to the Foods Table page.<br>2. Can view the list of all available food items. |
| **Expected result(s)** | The food table loads successfully, displaying a comprehensive and accurate list of food items. |
| **Post-condition(s)** | Admin retains access to the system, and the displayed list of food items remains up to date. |
| **Actual result(s)** | The food list was accessed successfully, with all items correctly displayed. |
| **Test Case Result** | Pass |

**Table 8.2.11: TC-25 Add Food**

| | |
|---|---|
| **Identifier** | TC-25 |
| **Related requirements(s)** | UC-10 Food Management |
| **Short description** | Verify that the admin can add new food items into the database. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to the Foods Table page.<br>2. Click on the ‘Add Food’ button.<br>3. Enter the necessary food details in the input fields<br>4. Submit the data by clicking on the 'Add' button. |
| **Expected result(s)** | The system should accept the input, save the new food item to the database, and display it in the Foods Table. |
| **Post-condition(s)** | The Foods Table includes the new food item, which is now available for further actions like edit or delete by the admin. |
| **Actual result(s)** | The new food item was successfully added and displayed in the Foods Table. |
| **Test Case Result** | Pass |

**Table 8.2.12: TC-26 Edit Food**

| | |
|---|---|
| **Identifier** | TC-26 |
| **Related requirements(s)** | UC-10 Food Management |
| **Short description** | To test the admin's ability to edit existing food items in the database. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to the Foods Table page.<br>2. Select an existing food item by clicking the 'Edit' icon in its corresponding row.<br>3. Modify the required fields in the input form.<br>4. Confirm the changes by clicking the 'Save' button. |
| **Expected result(s)** | The system updates and saves changes to food items quickly and shows these changes in the Foods Table immediately. |
| **Post-condition(s)** | The Foods Table shows the updated details, and the system is ready for more edits. |
| **Actual result(s)** | The food item was edited successfully, and the changes were correctly displayed in the Foods Table. |
| **Test Case Result** | Pass |

**Table 8.2.13: TC-27 Delete Food**

| | |
|---|---|
| **Identifier** | TC-27 |
| **Related requirements(s)** | UC-10 Food Management |
| **Short description** | To test the admin's ability to delete food items from the database. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to the Foods Table page.<br>2. Select a food item by clicking the 'Delete' icon next to it.<br>3. Confirm the deletion of a food item by clicking the 'Save' button. |
| **Expected result(s)** | The system should delete the selected food item, remove it from the Foods Table, and not display it anymore. |
| **Post-condition(s)** | The Foods Table no longer lists the deleted item, ensuring the database is updated correctly. |
| **Actual result(s)** | The food item was successfully deleted from the database and no longer appears in the Foods Table. |
| **Test Case Result** | Pass |

**Table 8.2.14: TC-28 View Feedback**

| | |
|---|---|
| **Identifier** | TC-28 |
| **Related requirements(s)** | UC-12 Feedback Management |
| **Short description** | To test the system's ability to display feedback submitted by users. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to the Feedback Table page.<br>2. Review the list of feedback entries. |
| **Expected result(s)** | The system should display all feedback entries in an organized manner, allowing the admin to read each piece of feedback clearly. |
| **Post-condition(s)** | The feedback remains accessible and unchanged, with the admin able to continue viewing additional feedback or exit the section. |
| **Actual result(s)** | All feedback was successfully displayed, with entries being clear and accessible to the admin. |
| **Test Case Result** | Pass |

**Table 8.2.15: TC-29 Delete Feedback**

| | |
|---|---|
| **Identifier** | TC-29 |
| **Related requirements(s)** | UC-12 Feedback Management |
| **Short description** | To test the admin's ability to delete user’s feedback from the database. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Feedback Table page.<br>2. Select certain feedback by clicking the 'Delete' icon next to it.<br>3. Confirm the deletion of feedback by clicking the ‘Confirm' button. |
| **Expected result(s)** | The system should successfully delete the selected feedback, removing it from the list and ensuring it is no longer accessible. |
| **Post-condition(s)** | The deleted feedback should not appear in the list, confirming it has been permanently removed from the database. |
| **Actual result(s)** | The feedback entry was successfully deleted, with the list updated to no longer show the deleted item. |
| **Test Case Result** | Pass |

**Table 8.2.16: TC-30 Update Health Profile**

| | |
|---|---|
| **Identifier** | TC-30 |
| **Related requirements(s)** | UC-14 Blog Management |
| **Short description** | To test the ability of an admin to add new blog posts within the system. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Blog Table page.<br>2. Click the 'Add Blog' button.<br>3. Enter the blog title, body content, link and upload any images if necessary.<br>4. Click the 'Submit' button to add the blog to the system. |
| **Expected result(s)** | The system should accept the new post, adding it to the blog section without issues and making it viewable to user's immediately. |
| **Post-condition(s)** | The blog section updates to include the new post, displaying all details as entered by the admin, who can add more posts or edit existing ones. |
| **Actual result(s)** | The new blog post was successfully created and displayed correctly in the blog section. |
| **Test Case Result** | Pass |

**Table 8.2.17: TC-31 Edit Blog**

| | |
|---|---|
| **Identifier** | TC-31 |
| **Related requirements(s)** | UC-14 Blog Management |
| **Short description** | To test the admin's ability to edit and update blog content. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Blog Table page.<br>2. Select a post to edit, make any necessary changes to the text or images.<br>3. Click the 'Update' button to submit the updates made on the blog. |
| **Expected result(s)** | The system should process and save the changes, immediately reflecting the updates in the blog post. |
| **Post-condition(s)** | The blog post shows all recent edits correctly, and the admin remains able to make further modifications. |
| **Actual result(s)** | The blog post was successfully edited, with all changes accurately shown. |
| **Test Case Result** | Pass |

**Table 8.2.18: TC-32 Delete Blog**

| | |
|---|---|
| **Identifier** | TC-32 |
| **Related requirements(s)** | UC-14 Blog Management |
| **Short description** | To test the admin's ability to delete blog posts within the system. |
| **Pre-condition(s)** | Admin must be logged in. |
| **Input data** | None |
| **Detailed steps** | 1. Navigate to Blog Table page.<br>2. Select a post to edit, make any necessary changes to the text or images.<br>3. Click the 'Update' button to submit the updates made on the blog. |
| **Expected result(s)** | The system should permanently remove the selected blog post, ensuring it no longer appears in the blog section. |
| **Post-condition(s)** | The blog section no longer lists the deleted post, and the admin can continue to manage other posts. |
| **Actual result(s)** | The blog post was successfully deleted and is no longer visible in the blog section. |
| **Test Case Result** | Pass |

## 8.3 Summary of Test Results

**Table 8.2: Summary of All Test Results**

| **Module Name** | Test cases run | Number of defects found | Number of defects corrected so far | Number of defects still need to be corrected |
|---|---|---|---|---|
| **User Authentication** | TC1, TC2, TC3, TC4, TC7 | 3 | 3 | 0 |
| **User Profile Management** | TC5, TC6, TC8, TC9, TC10, TC11, TC14 | 0 | 0 | 0 |
| **Meal Planning** | TC12, TC13 | 0 | 0 | 0 |
| **Admin Authentication** | TC15, TC16, TC19 | 0 | 0 | 0 |
| **Admin Profile Management** | TC17, TC18, TC20, | 0 | 0 | 0 |
| **Users Table Management** | TC21, TC22, TC23 | 2 | 2 | 0 |
| **Foods Table Management** | TC24, TC25, TC26, TC27 | 0 | 0 | 0 |
| **Feedback Table Management** | TC28, TC29 | 0 | 0 | 0 |
| **Blogs Table Management** | TC30, TC31, TC32 | 5 | 5 | 0 |
| **Complete System** | 33 | 10 | 10 | 0 |

# Chapter 9. Project Completion Status/Conclusion

**Table 9.1: Project Completion Status**

| **Module Name** | **Status**<br>(Complete, Partially Implemented, Not Implemented) |
|---|---|
| **Login/Signup** | Completed |
| **Authentication** | Completed |
| **Crud Operations of Manage User, Food, Blog, Feedback for Admin** | Completed |
| **Data Visualization** | Completed |
| **Model 1** | Completed |
| **Model 2** | Completed |
| **Model Pipeline Deployment** | Completed |
| **Web Application Deployment** | Completed |
| **Complete System** | Completed |

**Table 9.2: Objective(s)/Target(s) Status**

| **Target/Objective** | **Status**<br>(Completed,<br>Partially Completed,<br>Not Completed) | **Reason(s)** |
|---|---|---|
| **To provide dietary plans to local users** | Completed | There isn't such application that provide local foods to Pakistani users |
| **To provide user-friendly interface for better diabetes management** | Completed | |
| **Providing blogs and articles to local user regarding diabetes** | Completed | |
| **Number of Targets Completed** | 3 | |

| | | |
|---|---|---|
| **Number of Targets Partially Completed** | 0 | |
| **Number of Targets Not Completed** | 0 | |

# References


[1] N. Tabassum, A. Rehman, M. Hamid, M. Saleem, S. Malik, and T. Alyas, "Intelligent Nutrition Diet Recommender System for Diabetic's Patients," Intelligent Automation & Soft Computing, vol. 30, no. 1, pp. 1–18, Aug. 2021, doi: 10.32604/iasc.2021.018870.

[2] N. M. F. Vieira, "Obesity, Dietary Patterns, and Hormonal Balance Modulation: Gender-Specific Impacts," Nutrients, vol. 16, no. 11, p. 1629, Jun. 2023, [Online]. Available: https://www.mdpi.com/2072-6643/16/11/1629

[3] I. O. El-Badawy, "Macronutrients and Their Roles in Aging," in Nutrition for Aging Populations, 1st ed., vol. 1, Singapore: Springer, 2023, pp. 155–173. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-99-0534-8_8

[4] R. Barouki, "Aging and Nutrition: Theories, Consequences, and Impact of Nutrients," Curr Pharmacol Rep, vol. 5, no. 4, pp. 440–459, 2019, [Online]. Available: https://link.springer.com/article/10.1007/s40495-019-00185-6

[5] J. Slavin, "Impact of dietary macronutrient distribution on BMI and cardiometabolic profile," Nutr Rev, vol. 72, no. 7, pp. 453–462, Jul. 2014, [Online]. Available: https://academic.oup.com/nutritionreviews/article/72/7/453/1825279

[6] E. J. M.-D. et al., "Nutrition Therapy for Adults With Diabetes or Prediabetes: A Consensus Report," Diabetes Care, vol. 42, no. 5, pp. 731–754, May 2019, [Online]. Available: https://diabetesjournals.org/care/article/42/5/731/40480/Nutrition-Therapy-for-Adults-With-Diabetes-or

[7] M. T. St-Onge, "Association of Sleep Quality and Macronutrient Distribution: A Systematic Review and Meta-Regression," Nutrients, vol. 12, no. 1, p. 126, Jan. 2020, [Online]. Available: https://www.mdpi.com/2072-6643/12/1/126

[8] T. R. Silva, "The Impact of Diet and Physical Activity on Fat-to-Lean Mass Ratio," Nutrients, vol. 16, no. 1, p. 19, Jan. 2023, [Online]. Available: https://www.mdpi.com/2072-6643/16/1/19

[9] D. E. R. Johnson, "Macronutrient Intake for Physical Activity," in Nutrition and Enhanced Sports Performance, 2nd ed., Amsterdam: Elsevier, 2019, pp. 55–74. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-18230-8_4

[10] J. K. Westerterp-Plantenga, "Basal Metabolic Rate and Body Composition Predict Habitual Food and Macronutrient Intakes: Gender Differences," Nutrients, vol. 11, no. 11, p. 2653, Nov. 2019, [Online]. Available: https://www.mdpi.com/2072-6643/11/11/2653

[11] B. H. Goodpaster et al., "Age, Obesity, and Sex Effects on Insulin Sensitivity and Skeletal Muscle Lipid Content in Humans," Diabetes, vol. 59, no. 1, pp. 89–97, 2010, [Online]. Available: https://diabetesjournals.org/diabetes/article/59/1/89/16322/Age-Obesity-and-Sex-Effects-on-Insulin-Sensitivity

[12] C. S. Moreira and M. Pires, "Nutrition, Lifestyle Factors and Blood Pressure," in Primary Health Care, A. Santos and C. Machado, Eds., IntechOpen, 2018. [Online]. Available: https://www.intechopen.com/chapters/62232

[13] J. Ma et al., "Predicting Diabetes and Diabetic Complications Using Machine Learning Models: A Comprehensive Review," BMJ Diabetes Research and Care, vol. 12, no. 2, p. e003470, 2024, [Online]. Available: https://drc.bmj.com/content/12/2/e003470

# Appendix A Glossary

1. AI (Artificial Intelligence): Technology that enables machines to simulate intelligent human behavior. In the context of SugarSage, it refers to the algorithm used for generating personalized diet plans.

2. SRS (Software Requirements Specification): A detailed description of the software to be developed, including its functional and non-functional requirements.

3. Personalized Diet Planning: A feature within SugarSage that uses AI to create customized diet plans for diabetics based on their health metrics, medication regimens, and activity levels.

4. Local Food Database: A database within SugarSage that includes various local Pakistani foods and their nutritional information, allowing the system to align dietary recommendations with local dietary habits and preferences.

5. Mobile Application: The end-user platform of SugarSage, designed to provide a user-friendly interface for diabetics to manage their diet and monitor their health.

6. Singleton Pattern: A design pattern that ensures a class has only one instance and provides a global point of access to it. For SugarSage, it will be used for classes like the database connection manager or the configuration manager to manage shared resources effectively.

7. Cloud-Based Environment: A technology infrastructure that utilizes cloud computing to provide scalable and reliable service hosting, including capabilities such as auto-scaling to adjust resources based on demand.

8. Auto-Scaling: A feature of cloud services that automatically adjusts the number of computational resources according to the system's current load, ensuring responsiveness and cost-efficiency.

9. Admin Activity Diagram: A flowchart that illustrates the various actions an administrator can perform within the SugarSage system, including user management, content editing, and feedback processing.

10. User Activity Diagram: A diagram that outlines the flow of user interactions within the SugarSage system, from the welcome page to health profile management and meal plan adjustment.

11. API (Application Programming Interface): A set of protocols for building and interacting with software applications. SugarSage components use APIs for communication and data exchange.

12. User Profile Component: A part of the SugarSage system that stores and manages user-specific information such as health metrics and preferences.

13. Health Tracker: A component of SugarSage that monitors and records health-related data for users.

14. Meal Planner: A feature in SugarSage that generates personalized diet plans based on user health data and dietary preferences.

15. Feedback Module: A system within SugarSage that allows users to communicate feedback directly to system administrators.

16. Blog Reading: A functionality in SugarSage that provides users with access to educational content on diabetes management

17. GI (Glycemic Index) - A number representing the relative ability of a carbohydrate food to increase the level of glucose in the blood.

18. UI (User Interface) - The space where interactions between humans and machines occur.

19. ML (Machine Learning) - A subset of AI that provides systems the ability to automatically learn and improve from experience.

Appendix B IV & V Report

## (Independent verification & validation)

### IV & V Resource

Name Signature

| S# | Defect Description | Origin Stage | Status | Fix Time | |
|---|---|---|---|---|---|
| | | | | Hours | Minutes |
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| … | | | | | |

**Table B.1: List of non-trivial defects**

# Appendix C Deployment/Installation Guide

Our system is a web portal that can be accessed via a live link. To use the web portal, users need a web browser such as Chrome, Safari, or Mozilla Firefox, along with a fast and stable internet connection.

**Steps to Access the Web Portal:**

1. **Open a Web Browser**
   Ensure you have one of the following browsers installed: Chrome, Safari, or Mozilla Firefox.

2. **Navigate to the Live Link**
   Enter the provided live link in the browser's address bar. Go to sugarsage.vercel.app

3. **Login/Signup**
   Use your credentials to login to the portal.

***No installation steps for Mobile Application***

# Appendix D User Manual

## 1. Login Process

1. Open the SugarSage app.
2. Enter your registered email and password.
3. Click on the "Login" button.
4. If you are a new user, click on "Sign Up" and follow the registration steps:
    1. Enter your email and password.
    2. Confirm your password.
    3. Enter your personal details including first name, last name, date of birth, gender, country, and city.
    4. Provide health inputs such as weight, height, etc.
    5. Read and agree to the terms and conditions.
    6. Click on "Register" to create your account.
    7. Login with your new credentials.

## 2. Dashboard Overview

- **View Health Stats:**
    - Navigate to the Dashboard/Home page to view your health stats.
    - Analyse graphs and charts that display your blood sugar trends, nutrition distributions, calories track, and activity levels.
- **Update Health Data:**
    - Navigate to the "Health Profile" section to update your health information.
    - Users can enter/update their weight, height, HbA1c level, and other health metrics.
    - Click "Save" to update your health information.
- **Update Profile Data:**
    - Navigate to the "User Profile" section to update your profile information.
    - Users can enter/update their name, email, password etc.
    - Click “Save” to update your personal information.

- **Meal Plan Generation:**
    - Navigate to the "Meal Plan" section.
    - You can either generate a meal plan or view your current meal plan.
    - The app will generate a list of top food recommendations for the user to create a meal plan.
    - Users must choose from the recommended foods and select their portion sizes to stay within their BMR limit.

3. **Additional Features:**

- **View Blogs:**
    - Navigate to the "Blogs" section to read articles and posts related to diabetes management.
- **Step and Sleep Tracking:**
    - Track your daily steps, nutrition track, and calories consumption.

- **Feedback Form**
    - Users can provide feedback for the administrator to review.
    - Users must select the feedback type, topic, and title, and provide a text explanation of their complaint.

4. **Administrative Features:**

- **Blog Management:**
    - Admins can view, add, update, and delete blog posts.
    - Navigate to the "Blogs" section.
    - Select the blog post to update, make necessary changes, and save the updates.
    - Click on the “Add Blog” button and fill the form to add a new blog.
    - Click on “Delete” icon and give confirmation to delete an existing blog.
- **User Management:**
    - Admins can view, add, and delete user profiles.
    - Navigate to the "Users" section.

- Select a user profile to view detailed information.
- Admins can click on the “Update” icon to edit user information.
- Admins can click on “Delete” icon to delete a user.

- **Feedback Management:**
  - Admins can view or delete user feedback.
  - Navigate to the "Feedback" section.
- **Food Management**
  - Admin can view, add, edit, and delete food items.
  - Navigate to the “Foods” section.
  - Click on the “Add Food” button and fill out the form to add a new food item.
  - Click on the “Update” icon next to a food item to edit its information.
  - Click on the “Delete” icon next to a food item to remove it.