**CS673 Software Engineering** 

**Team 3 - Project Name**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Zuowen Tang | Security Leader | *Z.T.* | 5/12/23 |
| Zuowen Tang | QA Leader | *Z.T* | 5/22/23 |
| Alekhya Koppineni | Requirement leader | [Alekhya Koppineni](mailto:alekhya.koppineni@gmail.com) | 5/12/23 |
| Edara Umamaheswar | Team Leader | *Mahesh* | 5/15/23 |
| Adithya Prakash | Design and Implementation Leader | *Adithya Prakash* | 5/15/23 |
| Corydon Wood | Configuration Leader | *Corydon Wood* | 5/15/23 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1.0** | **Corydon Wood** | **5/16** | **Added Initial Roles** |
|  |  |  |  |

[Introduction](#_87t9hln2vjz0)

[Software Architecture](#_3ipvmjgn6clp)

[Class Diagram](#_ky60nv8suxxm)

[UI Design (if applicable)](#_7ucksmkf6rzx)

[Database Design (if applicable)](#_tcmuor4nl1kz)

[Security Design](#_x18fj36s1121)

[Business Logic and/or Key Algorithms](#_mtfbusfb0eq3)

[Design Patterns](#_9zvwkmc4luo5)

[Any Additional Topics you would like to include.](#_15tmymhipvdv)

[References](#_50ojo9i46ytq)

[Glossary](#_8n34lvocupub)

# Introduction

The calorie tracker project is a web-based application developed using the Python Django stack. It allows users to track their daily calorie intake by entering consumed food items and provides them with a comprehensive overview of their nutritional intake. It emphasizes data privacy, scalability, and cross-platform accessibility to provide a user-friendly and efficient calorie tracking experience.

The design goals of the software system are as follows:

1. **Calorie Tracking:**

The primary goal of the system is to allow users to track their daily calorie intake. The system should provide an intuitive and user-friendly interface for users to enter their consumed food items and their calorie value, and track their calorie count.

1. **Data Privacy and Security:**

Ensuring the privacy and security of user data is crucial. The system should follow appropriate data protection regulations and implement measures to protect user information, including secure storage and transmission of data.

1. **Modularity and Extensibility:**

The system should be designed with a modular and extensible architecture, allowing for easy maintenance, future enhancements, and the addition of new features. Clear separation of concerns and adherence to coding best practices should be followed.

1. **Authentication and Authorization:**

User authentication and authorization mechanisms should be implemented to ensure that only authorized users can access and modify their own data. Proper user roles and permissions should be enforced.

1. **Social Media:**

Users can create and view posts, follow or unfollow other users, view a directory of all users, explore profiles and posts, and see the followings and followers of other users.

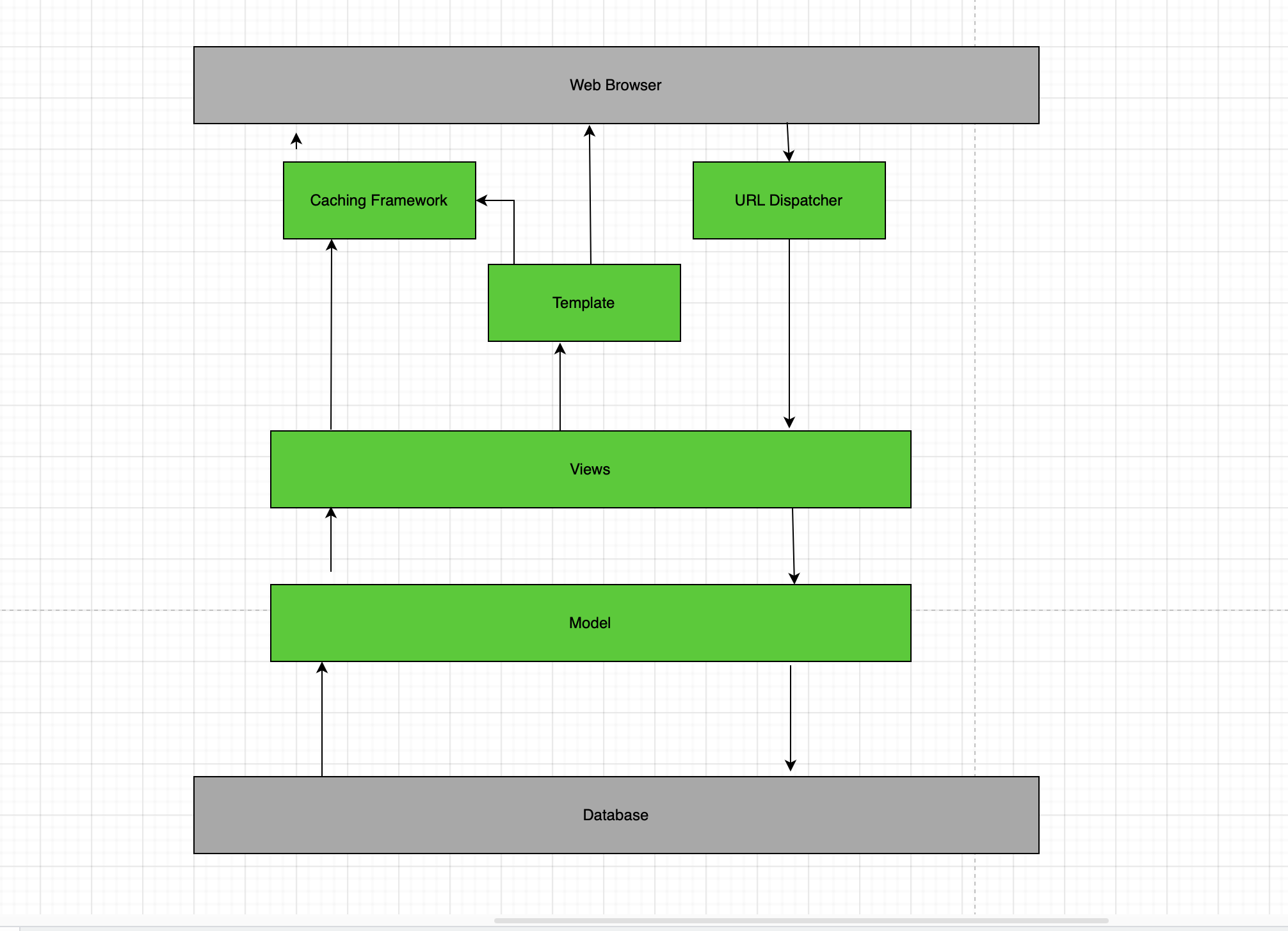
1. **Reliability and Error Handling:**

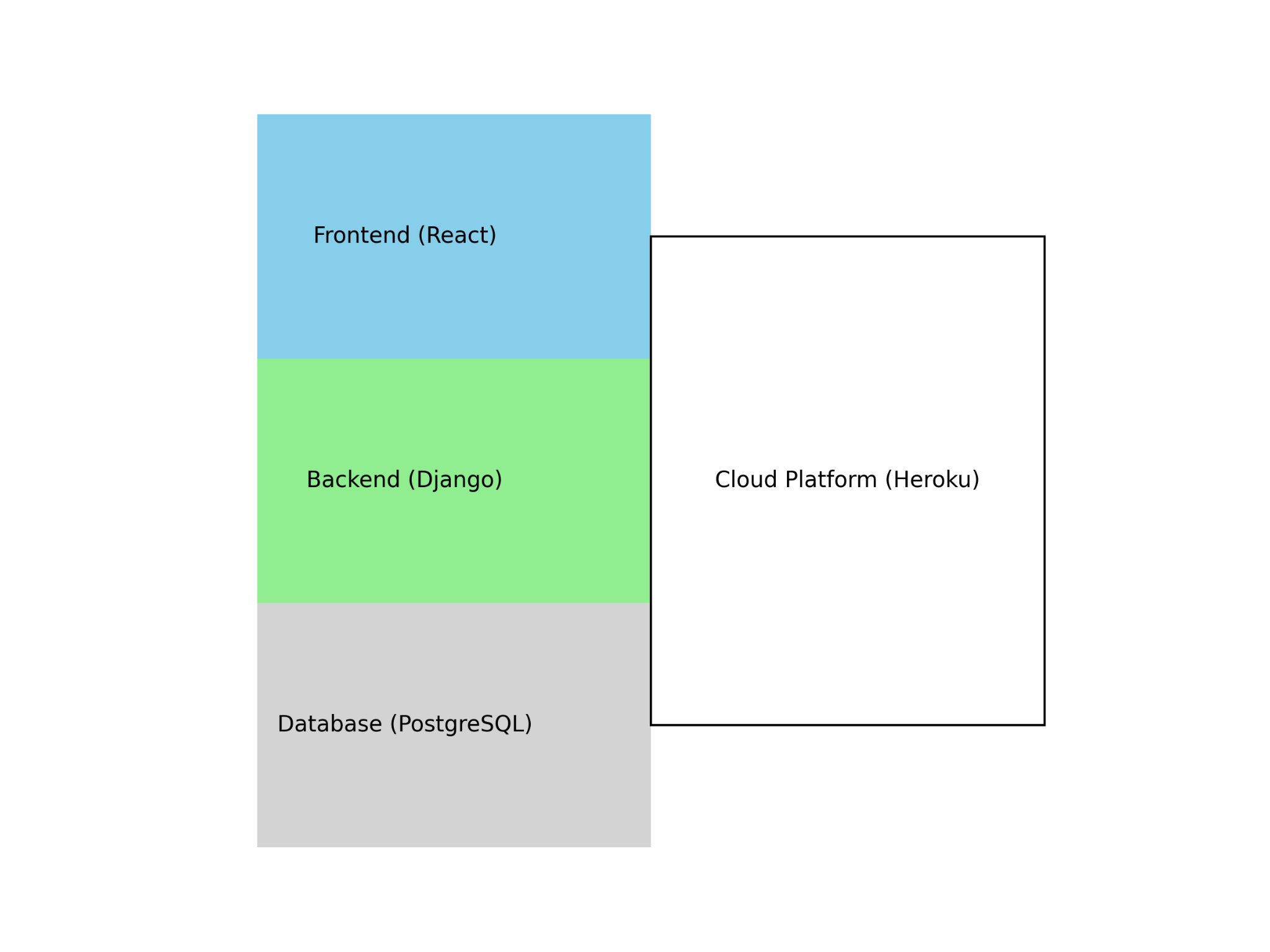
The system should be reliable, resilient, and capable of handling errors and exceptions. Robust error handling mechanisms should be in place to provide meaningful error messages to users( Ex. like popup messages ) and assist in troubleshooting.

# Software Architecture

* User Interface (UI): This component handles the presentation layer of the application and interacts with users. It includes web pages for user registration, login, calorie tracking, and nutritional information display.
* Views and Controllers: These components are part of the Django framework and handle user requests and application logic. Views receive user input from the UI and communicate with the database and APIs to process and retrieve data. Controllers handle the flow of data and control the execution of operations.
* Models: The models represent the data structure of the application and are responsible for data storage and retrieval. They define the entities such as User, FoodItem, and Nutrition that store information related to users, consumed food items, and nutritional details(calories).
* Database: The database component stores and manages the persistent data used by the application. It interacts with the Models component to perform data operations like storing user profiles, food item details, and nutritional information.
* Authentication and Authorization: This component handles user authentication and authorization processes. It ensures that only authorized users can access and modify their own calorie tracking data.

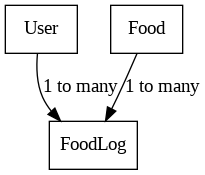
The relationship between these components is depicted in the software architecture diagram. The UI component interacts with the Views and Controllers to handle user input and display information. The Views and Controllers communicate with the Models to fetch and update data from the database. The Authentication and Authorization component interacts with the Views and Controllers to ensure secure user access.

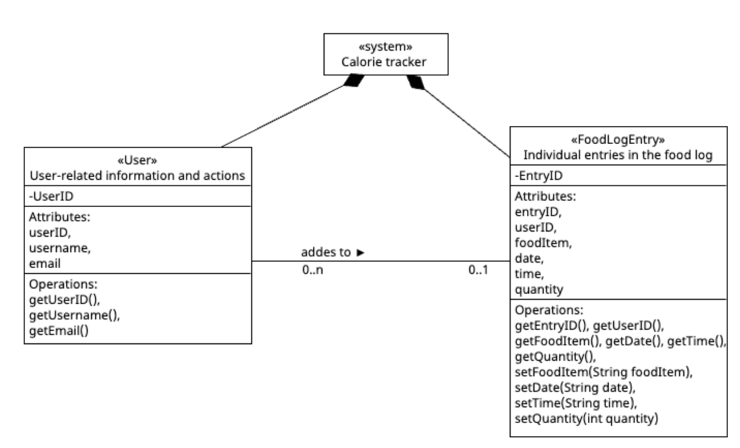


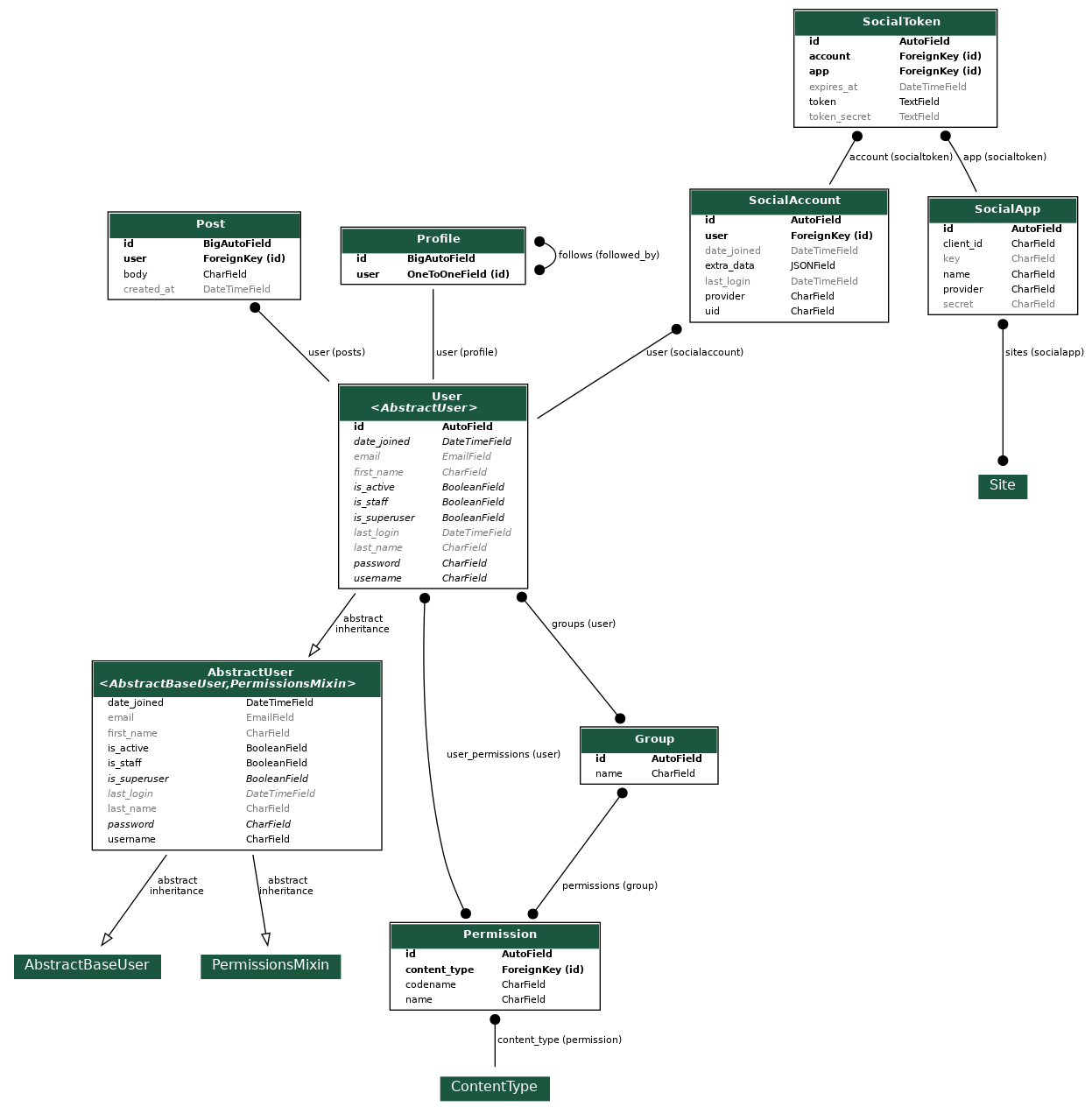


# Class Diagram

In this section, you will provide a detailed description of each component (or package) and use one or multiple class diagrams to show the main classes and their relationships in each component.

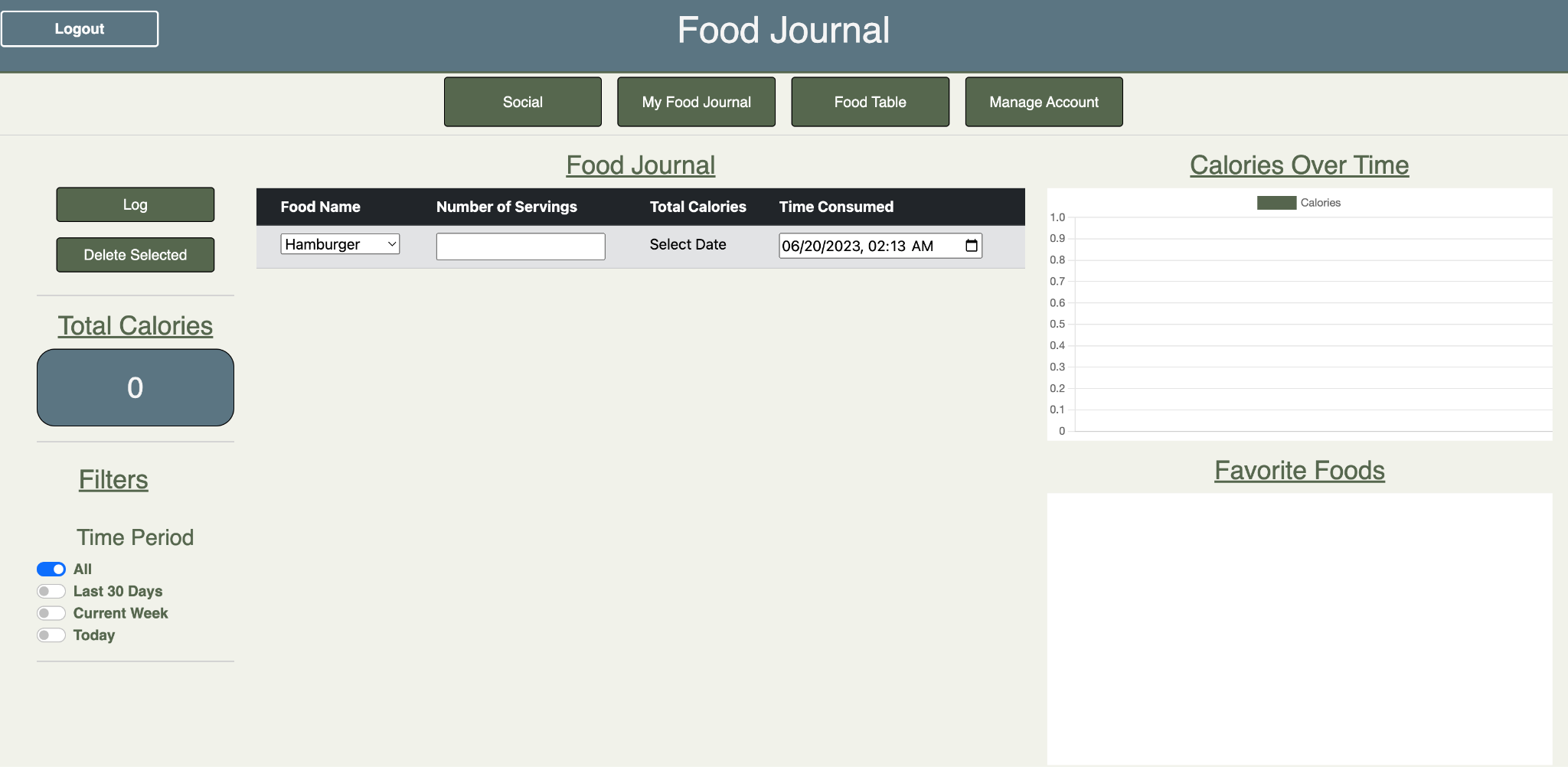


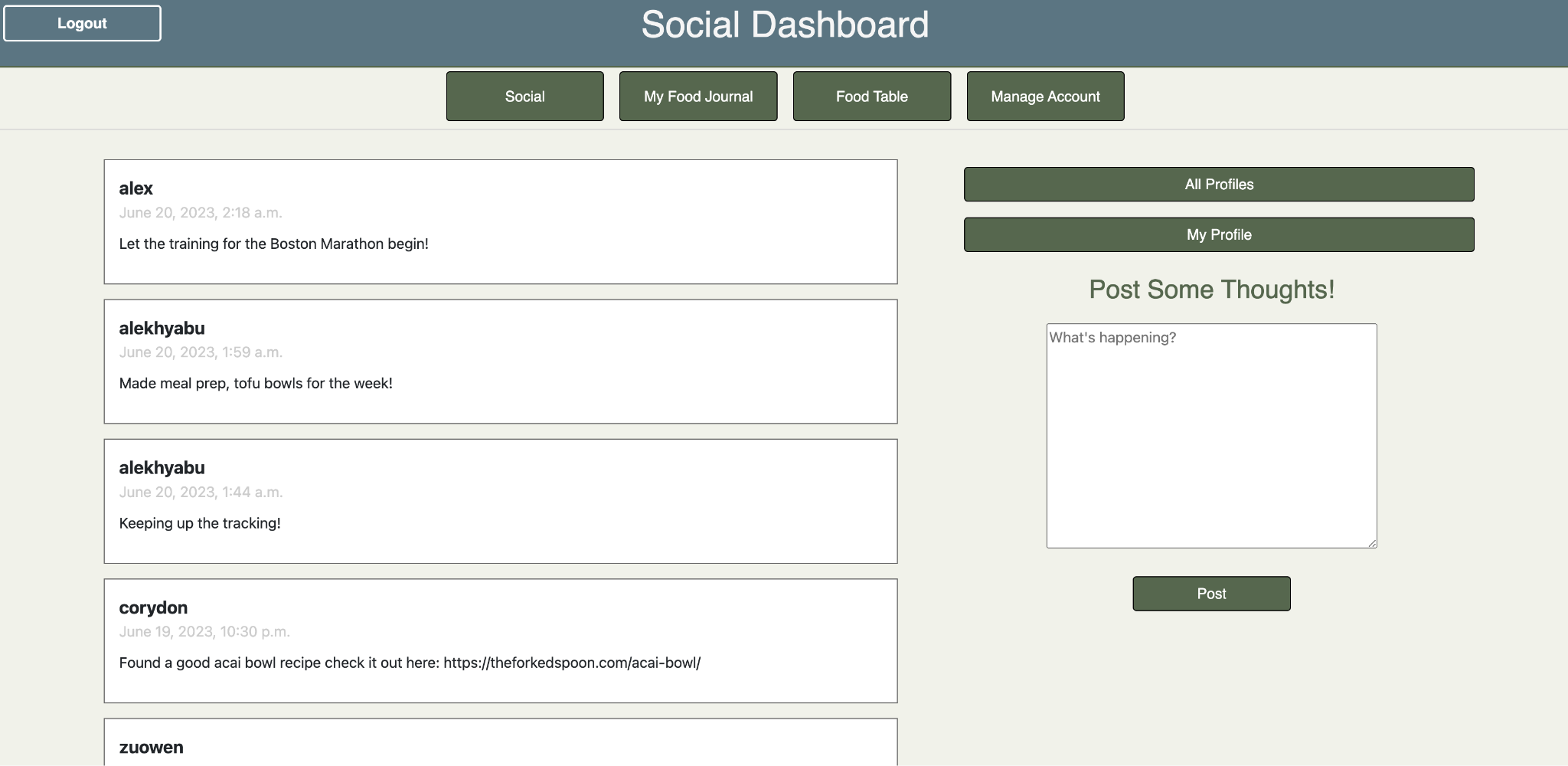
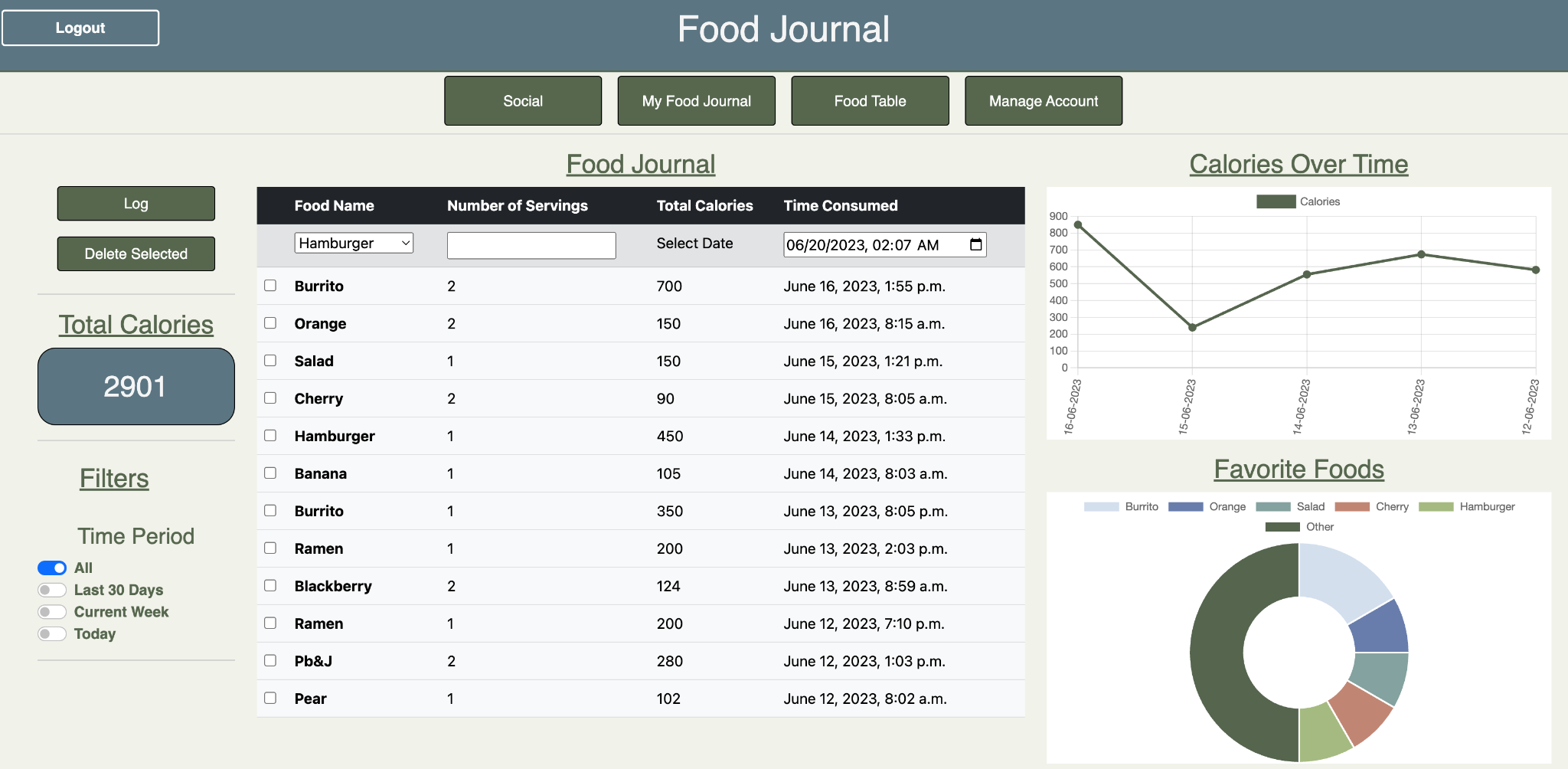




# UI Design (if applicable)

In this section, you can describe your UI design. You can include both your initial design before the implementation and the screenshots of your UI after the implementation.





Sample user pages

# Database Design (if applicable)

In this section, you shall describe any database schema if used in your software system.

User Table:

- id: Primary Key (Integer)

- username: String

- password: String

Food Table:

- id: Primary Key (Integer)

- name: String

- calories: Integer

FoodLog Table:

- id: Primary Key (Integer)

- user\_id: Foreign Key to User Table (Integer)

- food\_id: Foreign Key to Food Table (Integer)

- date: Date

In this design, the User class represents the user entity and has attributes like id, username, and password. The Food class represents food items and has attributes like id, name, and calories. The FoodLog class represents the log entries for each user's food intake and has attributes like id, user\_id (foreign key referencing User), food\_id (foreign key referencing Food), and date.

In the database design, each entity has its own table, and relationships between tables are represented through foreign key constraints.

# Security Design

**Data Encryption:** To protect user data in transit and at rest, all information sent or received will be encrypted using industry-standard encryption protocols (e.g., HTTPS/TLS for data in transit and AES-256 for data at rest, which is supported by Django). Passwords will be stored as hashed values using a strong hashing algorithm.

**User Management:** User management will be designed to allow users to register, log in, and manage their accounts securely. It will include options for password reset, account recovery, and account deletion.

**Network & Web Security:** The application will be designed with security in mind, including coding standards to mitigate common web vulnerabilities. Regular security audits and code reviews will be performed. All network traffic will be monitored for suspicious activities. Firewalls and intrusion detection systems (IDS) will be in place.

# Business Logic and/or Key Algorithms

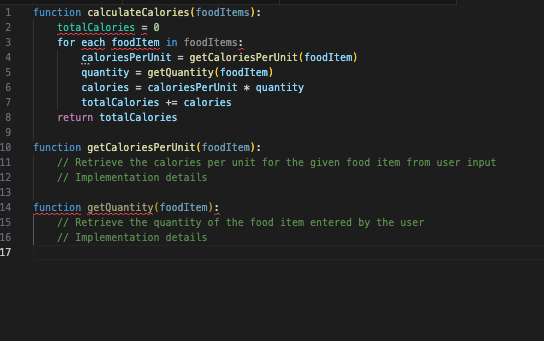
In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart, or sequence diagrams.

* **Calorie Calculation Algorithm:**

**Input**: Food items and their respective quantities

**Output:** Total calorie count

**Pseudocode:**



# Design Patterns

Model-View-Controller (MVC) Pattern:

* Description:

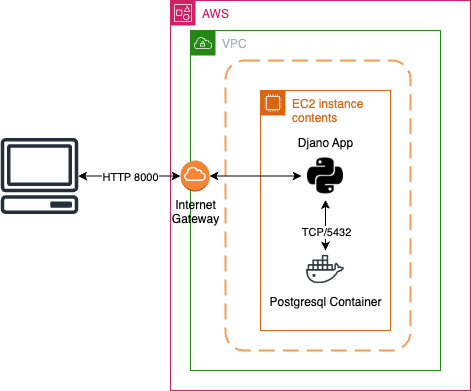
MVC is a structural design pattern that separates the application into three interconnected components: Model, View, and Controller. The Model represents the data and business logic, the View displays the data to the user, and the Controller handles user input and updates the Model and View accordingly.

* Usage:

We can use the MVC pattern to structure our Django application. The Models will represent the calorie data and business logic, the Views will handle user input and interaction, and the Controllers (in Django, they are typically referred to as Views) will coordinate the flow of data between the Models and Views.

# Any Additional Topics you would like to include.

Deployment Architecture



# References

# Glossary