**CS673 Software Engineering**

**Team X - Project Name**

**Software Design Document**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Ray Clark | Team Lead | *Ray Clark* | June 1, 2021 |
|  |  |  |  |
| Chen Zhao | Requirement leader | *Chen Zhao* | May, 31, 2021 |
| Huimin Diao | Configuration leader | *Huimin Diao* | May, 31, 2021 |
| Danny Graziano | Design & Implementation Leader | *Danny Graziano* | June 1, 2021 |
| Josephine But | QA Leader | *Josephine But* | June 1, 2021 |
| Zixia Zhou | Security Leader | *Zixia Zhou* | May 27, 2021 |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| 0 | Group | May 18, 2021 |  |
| 1 | Group | June 1, 2021 |  |

**Table of Contents**

[Introduction](#_87t9hln2vjz0)

[Software Architecture](#_buttcq9i221r)

[Design Patterns](#_x18fj36s1121)

[Key Algorithms](#_mtfbusfb0eq3)

[Classes and Methods](#_7ucksmkf6rzx)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# **Introduction**

This document provides the details of the software design for the **MyDietHub** software application. **MyDietHub** is a full stack application built on modern frameworks that allows a user to quickly compute the calories of a given food type and for an entire meal. The purpose of the application is to allow a user to track their caloric intake and thereby track and regulate their diet.

**Application Overview**

According to data from the National Health and Nutrition Examination Survey (NHANES, 2013–2014 Survey), approximately 2 in 3 adults in the U.S. are either overwieght or obese (BMI > 30). Since diet control (calorie intake control) is one of the most important ways to fight obesity, our group will build an app that helps people calculate the calories of the food they eat and will provide tips, encouragement and warnings to help people fight obesity. Any person hoping to regulate their caloric intake is a potential user.

**Software System Design Goals**

In developing our software system design, we will focus on an architectural structure that will satisfy the stated system requirements. Second, we will attempt to implement a design that will be flexible and enable us to make changes as necessary. Lastly, we will integrate various software qualities such as usability, reliability, scalability and security.

More specifically, the ‘External Quality Characteristics’ that we will integrate include the following:

1) Efficiency – Measured by response time and throughput

2) Reliability - Measured in terms of mean time failure, i.e., up-time

3) Robustness - System error handling

4) Usability – UI/UX, ease of use

5) Security – Integrating measure to protect against malicious attack

The ‘Internal Quality Characteristics’ will include the following:

1) Maintainability – Ease of revising and/or enhancing the system

2) Re-usability – Use OOD to ensure reuse of code

3) Readability – Integrate clear design structure

4) Portability – System must work across different platforms

**Class Relationships**

Cohesion and coupling can help measure the dependency and complexity of the software architecture. Cohesion measures dependency within the subsystem, meaning classes with similar functionality should be grouped together, while Coupling measures dependency among subsystems. A low level of coupling implies that a change in one subsystem does not affect any other subsystem.

We will strive to obtain a high level of cohesion and a low level of coupling.

The subsystem can be further decomposed into finer-grain sub-subsystems. Usually, the number of top-level subsystems is 3-5 with each sub-system focusing on a single concern.

# **Software Architecture**

In this section, we describe the decomposition of our software system, which includes 1) components (packages), and 2) the relationship between components.

Listed below is also a diagram to show the whole architecture, and class diagrams for each component. The interface of each component and dependency between components are also described. Lastly, we have provided a short description of the Flask framework that is used. Database design is also described.

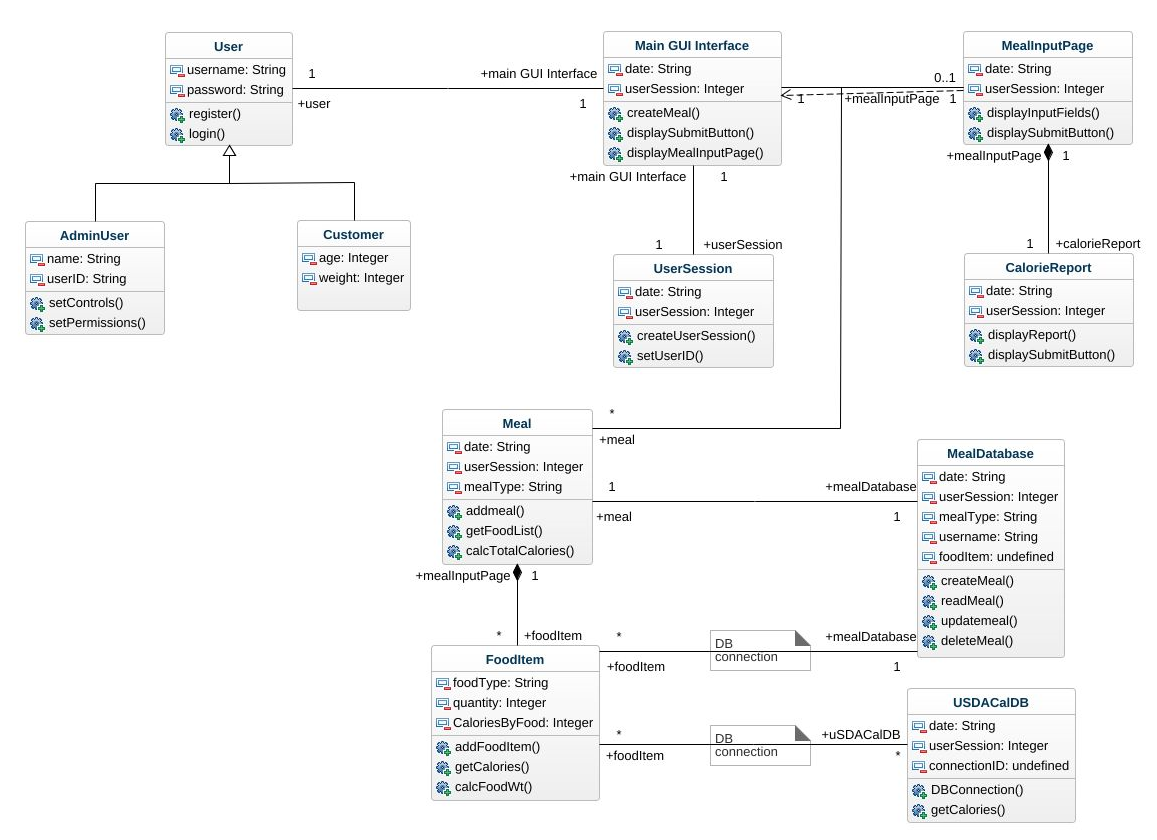
**Architecture Overview**

Our overriding goal with regard to software architecture is to decompose the software system into a number of smaller subsystems (components) to reduce system complexity while allowing for change.

The subsystems or components will be structured or decomposed based on functions and data flow. For example, as our system tracks meal calories, the system will feature a “Meal” class that will maintain the meal’s key attributes and functions such as adding a new meal. There will also be an external data class to access the USDA data through an API and a view class that includes the various UIs.

More broadly, our architecture can be divided into “frontend” and “backend” components. The frontend components include the user login and registration portal, calorie display page, calorie history page, and anything that generally displays or retrieves information to/from the user. This data that’s collected is used by the backend components of our app to perform the processing and data retrieval tasks, at which point the data will be returned to the frontend for display to the user. Callback functions will allow backend functionality to be accessed via frontend elements (such as a button click). Generally, the classes that will comprise the backend are *Meal, MealDatabase, FoodItem, USDACalDB*. The frontend classes are *User, Main User GUI, MealInputPage (and their respective subclasses)*.

# **MyDietHub Class Diagram**

The class diagram below outlines the key classes, attributes and functions for the **MyDietHub** application. As discussed, the components are decomposed based on functions and data flow. In general, the classes can be grouped together into four (4) distinct packages, including: 1) User related classes, 2) View or GUI related user interfaces, 3) Model, or key object classes that store the food and meal data, and 4) Database (control) classes that are responsible for retrieving data from the user and 3rd party data sources and storing that data in a persistent structure. By employing this structure, we can achieve a high degree of cohesion with a low degree of coupling. Please note that, whereas, we have listed key fields and methods, not all are listed.

# 

# 

# 

# 

# 

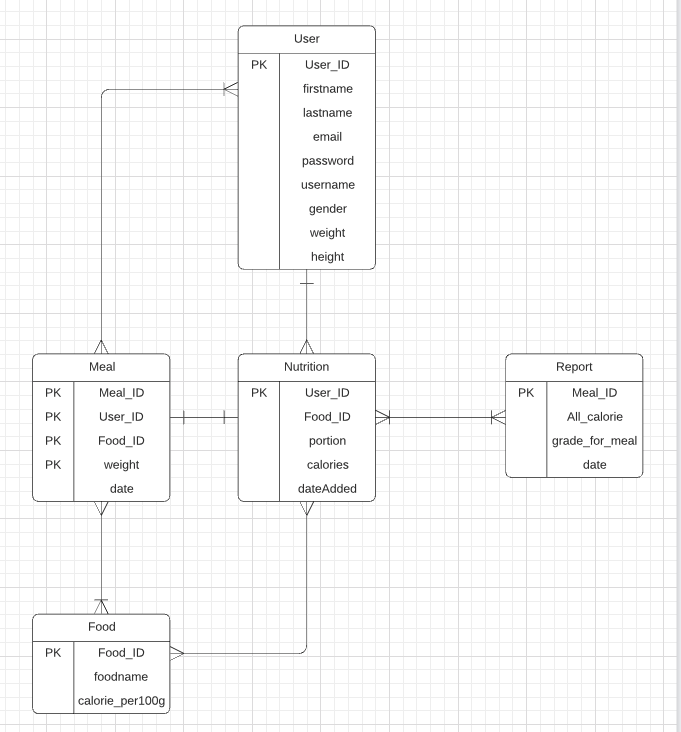
# **Database Design**

We will implement a sqlite3 database to store both user and retrieved nutritional data . This is an extremely simple database design that works well with python3 and flask. Due to the time-constraints on this project, we elected to go with the most simple solution possible to form a minimum-viable product. Sqllite3 is very similar to other sql-like databases, and can be replaced later on with a more advanced, but compatible database solution like MySQL or Postgres.

Our “**User**” database has 9 keys: id, firstname, lastname, email, username, password, gender, weight and height. So that the application could calculate BMI and the appropriate calorie intake for each user.

Because the MyDietHubapp needs to calculate the calorie for each meal, so that “**Food**” can make sure the MyDietHub could find the calorie for each kind of food. And “**Counter**” can calculate the calorie for each meal with given data.

Our “nutrition” database has the following keys: userID, food, portion, calories, dateAdded



“**Report**” will retrieve relevant slices of the databases based on meal, calorie info and date range for eventual rendering and display to the user.

# **Security Design**

We have to take care of the user password before storing it into the database for security purposes. We use Flask-login, Werkzeug and we are going to use the method PBKDF2 (*method='pbkdf2:sha256')* to do the password hashing. **PBKDF2** is a simple cryptographic key derivation function, which is resistant to [dictionary attacks](https://en.wikipedia.org/wiki/Dictionary_attack)  [and rainbow table attack](https://en.wikipedia.org/wiki/Dictionary_attack)s. It is based on iteratively deriving **HMAC** many times with some padding. The **PBKDF2** algorithm is described in the Internet standard [RFC 2898 (PKCS#5).](http://ietf.org/rfc/rfc2898.txt)

We also take care to ensure that API keys we used to fetch the data from USDA are not exposed in our public code repository. They will be held in a separate configuration file that the user will populate with their own key.

**Design Patterns**

Design patterns provide a shared vocabulary to solve common problems. The famous design patterns book “Design Patterns: Elements of Reusable Object-Oriented Software” published in 1994 by the “Gang of Four” outlines 23 design patterns in three categories:

1) Creational patterns: design patterns about class or object creation, including abstract factory, builder, factory method, prototype, singleton, etc.

2) Structural patterns: design patterns about class and object composition, including adapter, bridge, composite, decorator, facade, proxy, etc.

3) Behavior patterns: design patterns about class or object communication, including command, interpreter, observer, state, strategy, template method, etc.

**Factory Pattern** – The ‘Factory’ pattern is one of the most used design patterns in software. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object. With the Factory pattern, we create an object without exposing the creation logic to the client and refer to the newly created object using a common interface.

Given that a particular user of the calorie tracking app will most likely be creating meals of different types such as breakfast meal, lunch meal and dinner meal, and, perhaps, a mid-day or late-night snack meal, the Factory pattern may be particularly useful as all of these types of meals share common attributes and functions that can be implemented from as abstract class within the concrete sub-classes.

**Singleton** – The ‘Singleton’ pattern is one of the simplest design patterns. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object. This pattern involves a single class which is responsible for creating an object while making sure that only a single object gets created. This pattern is particularly useful when creating a single user session with an associated ID. Hence, we may utilize this pattern for user session instantiation.

# **Key Algorithms**

Key algorithm

[Essential]

1. BMI calculator

We have user data so that we can get the user’s BMI,

USER BMI = weight(kilogram) / height^2 (meter)

2. Calorie calculator

We have the type of food and the weight of food which is input by user, so that we can get the calorie of each kind of food:

Each food calorie = [weight(gram)/100] \* calorie\_per\_100gram(for each kind of food)

3. Meal calorie

We have each food calorie in the second algorithm, so that we can get the sum calorie for one meal:

Meal calorie = sum(each food calorie)

# **UI Design**

This section presents the graphical user interface of the MyDietHub Web application, where all functionalities are grouped visually and logically according to the types of entities they are associated with.

The design foresees the development of the web application using the following HTML templates:

* HTML5;
* css;
* use of web standards

The section that follow contain mock views of a selection of the most important aspects

of the GUI, along with textual descriptions of their purpose and contents.

1. **Home (Landing Page)**

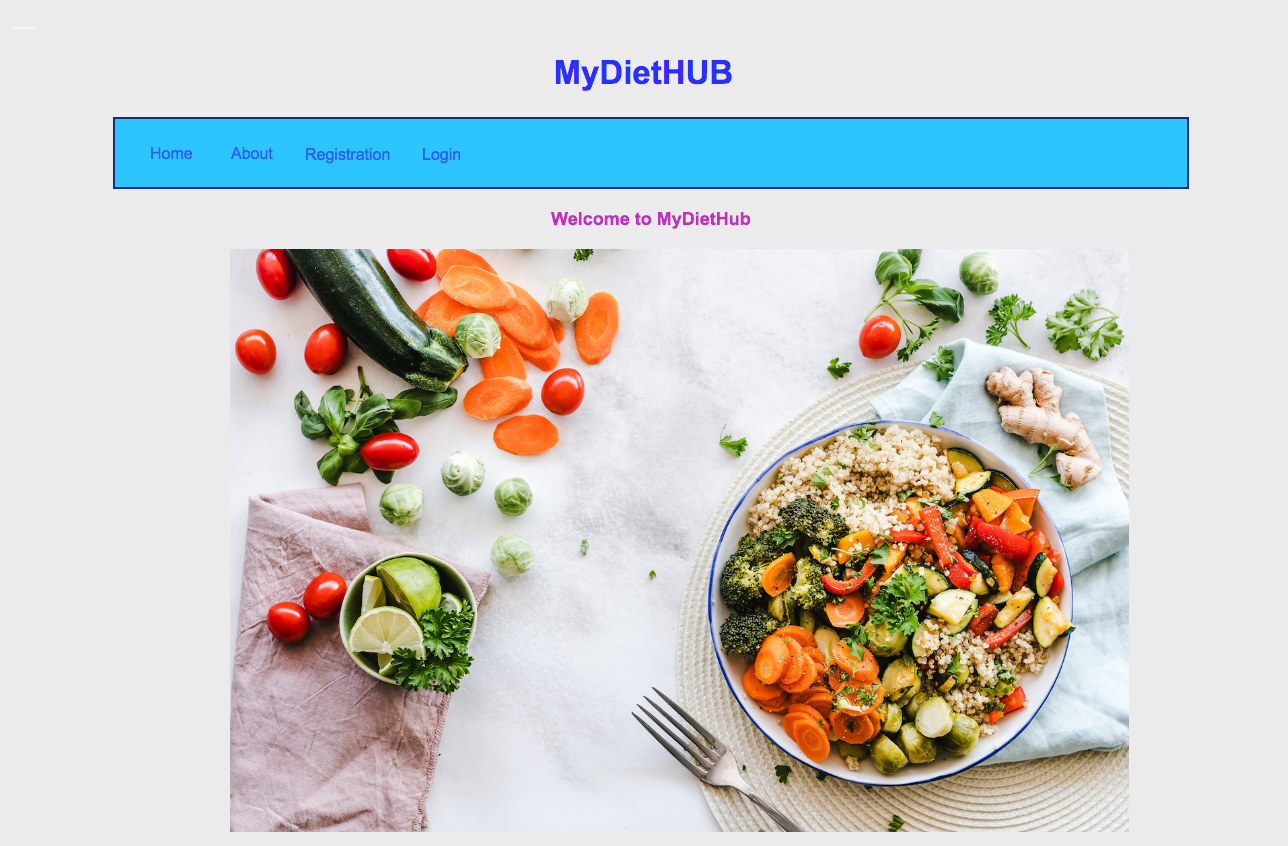
This section describes the “Home” page. It is a gateway to a site’s content. The main component is that the user can take a look on the “About” page, do the registration and login into the system.

Purpose :

A gateway to MyDietHub's content.

Navigation & User Interaction :

* User can click on “About” link to advance to the about page.
* User can click on “Login” link to advance to the login page.
* User can click on “Registration” link to advance to the Registration page.



1. **About**

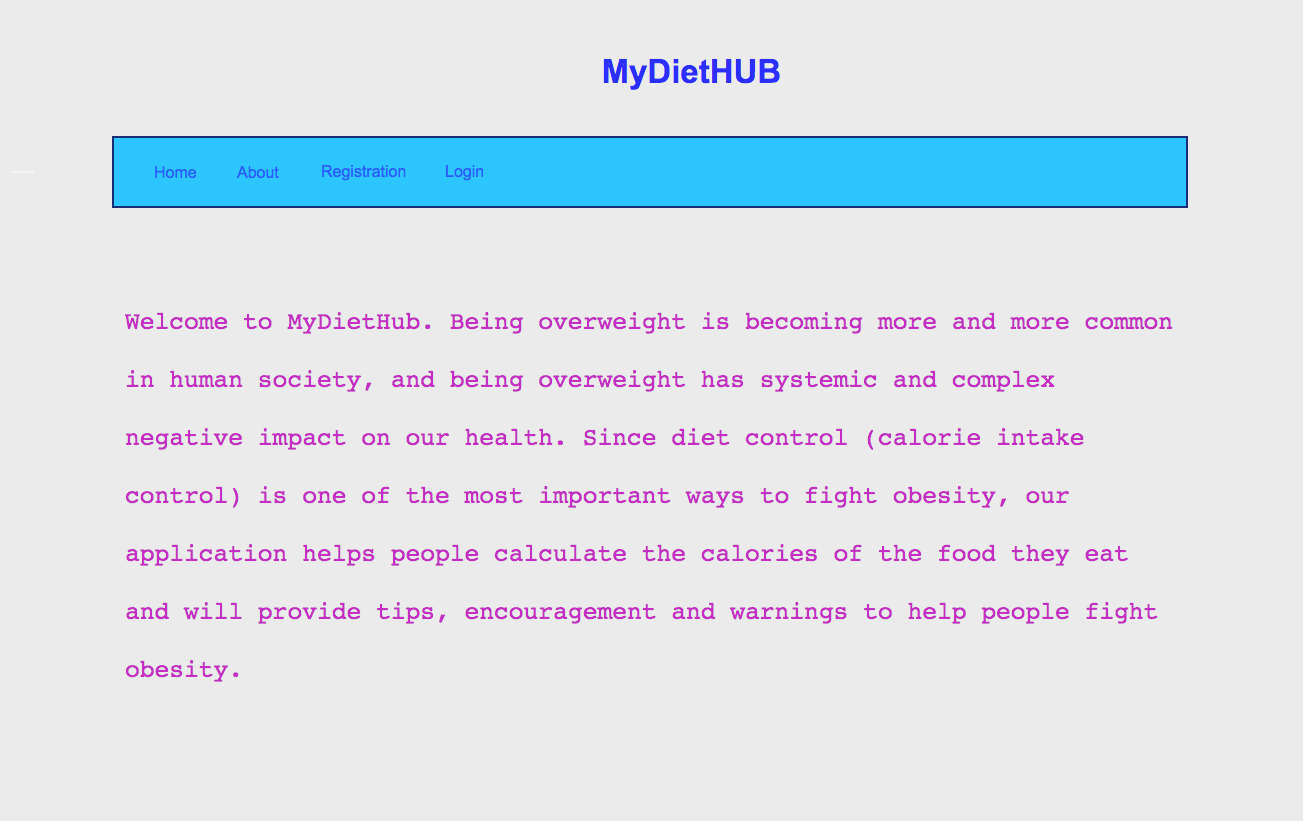
This section describes who we are for a visitor to learn more about us.

Purpose :

For the user to know more about us.

Navigation & User Interaction:

* User can select “Home” to go back to the landing page
* User can select “Registration” to register as a user.
* User can select “Login” to login to go to the user’s dashboard.

****

1. **Registration**

This section describes the “Registration” page. The main component is the

registration form, where the user needs to provide the firstname, lastname, username, password and the email.

Purpose :

For the user to do the registration.

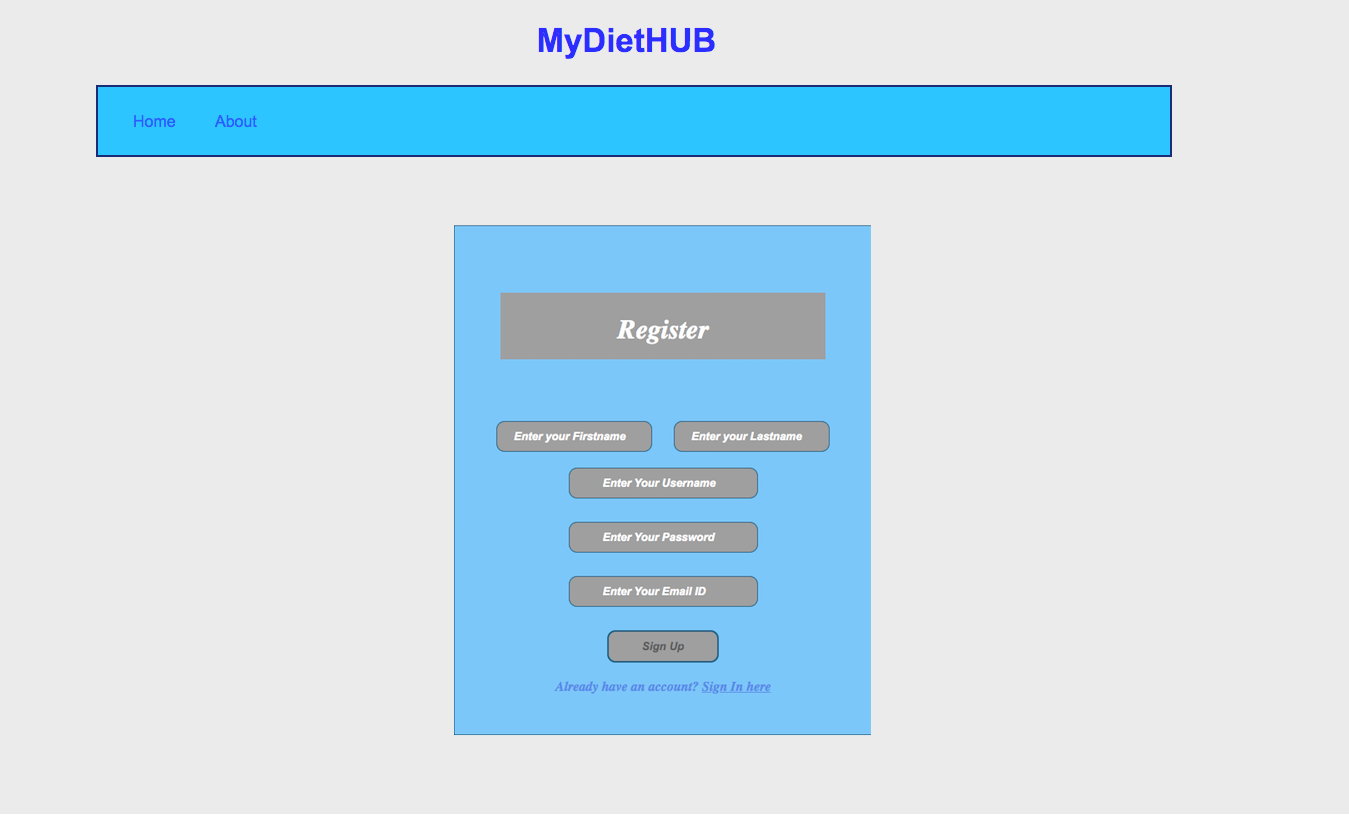
Navigation & User Interaction :

* To register the user is required to enter the user's username, password and email and click the “Sign up” button.
* The successful registration will advance the user to the login page.

1. **Login**

This section describes the “Login” page. The main component is the

login form, where the user needs to provide the username and password.

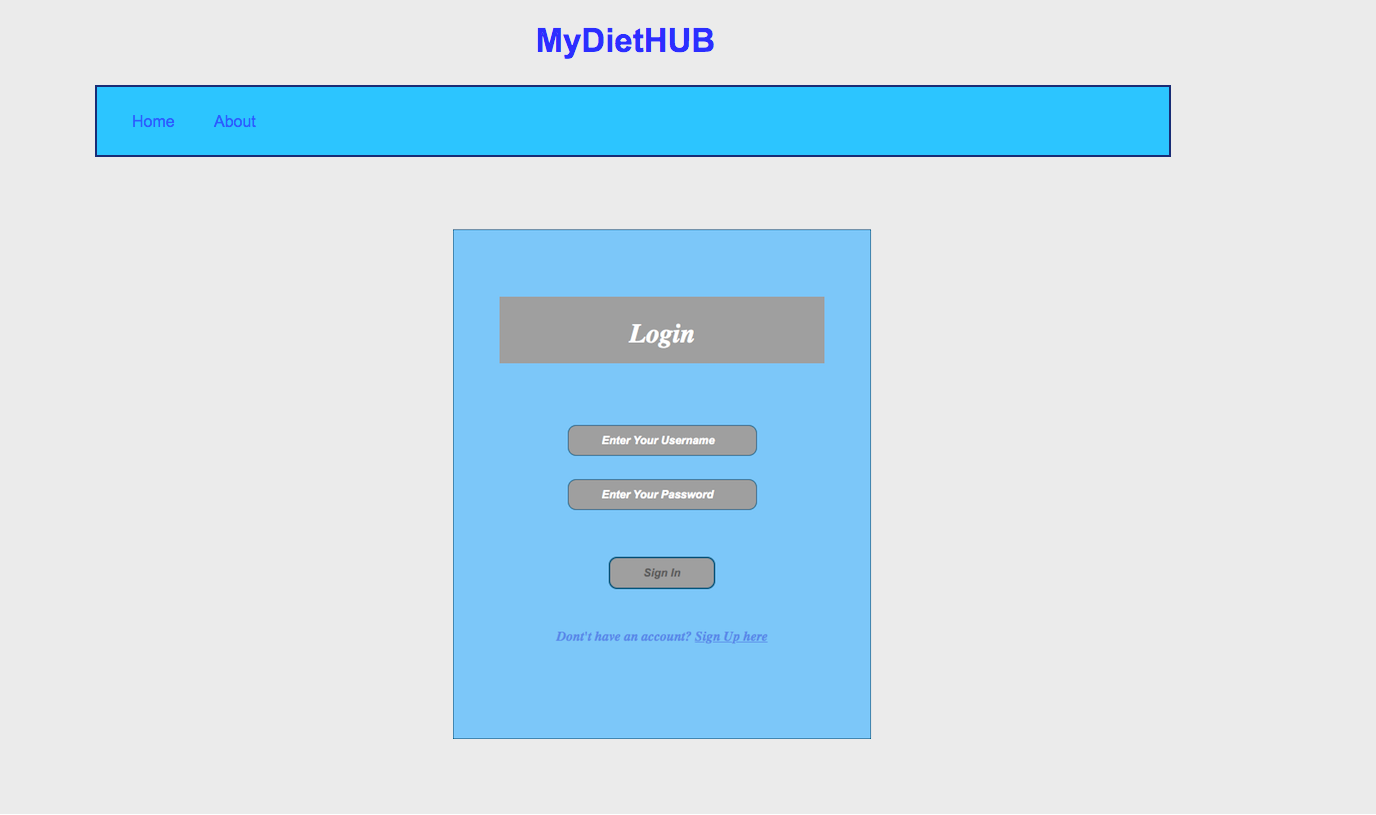


Purpose :

For the user to do the login and go to the user’s dashboard.

Navigation & User Interaction :

* To login the user is required to enter the user's username and password and click the “Sign In” button.
* The successful login will advance the user to the user’s dashboard.

****

1. **User’s dashboard**

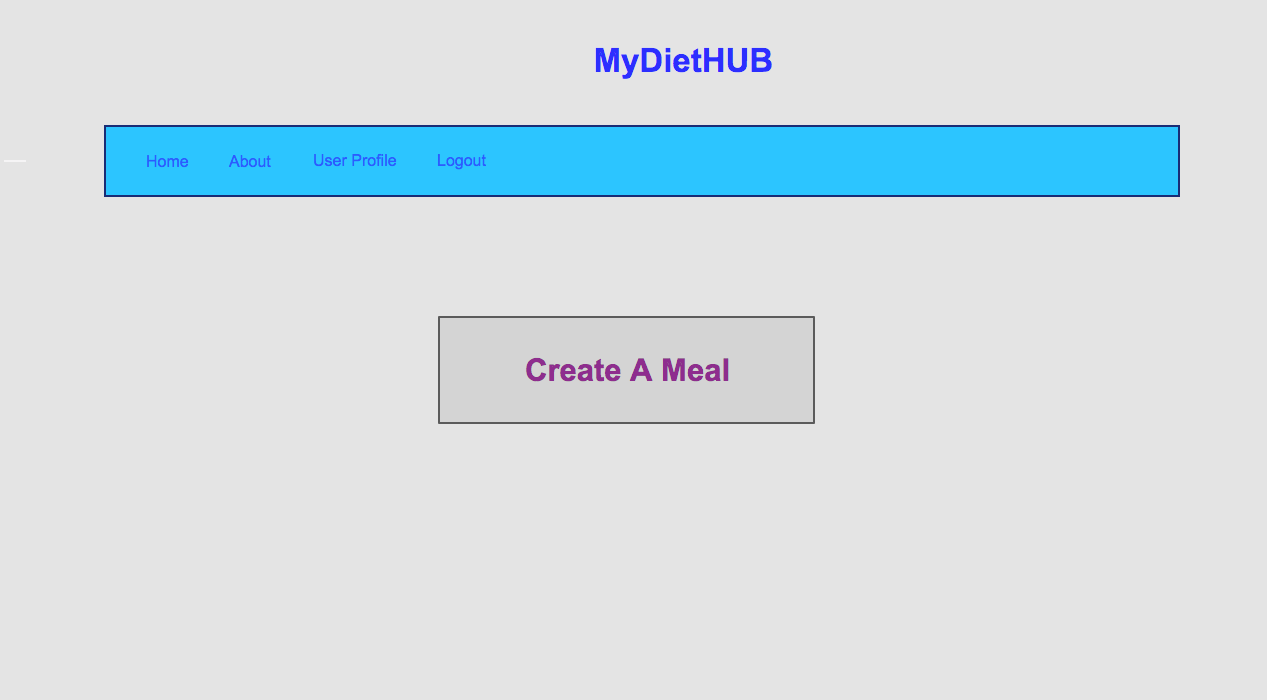
This section describes the user’s choice page. The main component is the “Create a Meal” button which lets the user navigate to the “Create a Meal” page.

Purpose :

For the user to go to “Create a Meal”

Navigation & User Interaction :

* User can click on the “Create a Meal” button to advance the user to the “Create a Meal/Calculate Calories” page.
* User can select “Home” to go back to the Home page.
* User can select “About” to go to the About page.
* User can select “User Profile” to go to the User Profile page.
* User can select “Logout” to logout from the system and go back to the landing page.



1. **Create a Meal / Calculate Calories**

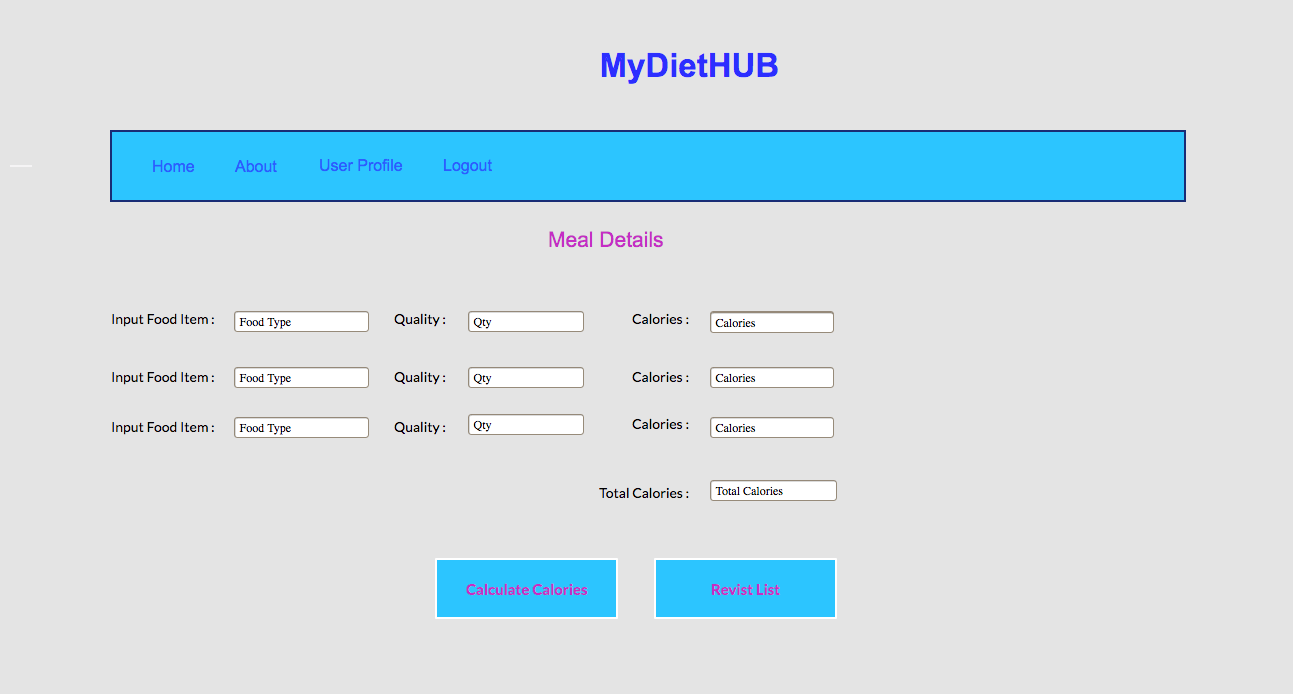
This section describes the “Create a Meal/Calculate Calories” page. On the “Create a Meal” page, where the user needs to provide food types and quantities of the food types. After all the inputs, user can click the “Calculate Calories” button to obtain the calories for each food type and the total calories of the meal.

Purpose :

For the user to calculate the calories of a meal.

Navigation & User Interaction:

* User can click the “Calculate Calories” button to obtain the calories of each food type and the total calories of the meal.
* User can click the “Revised Button” to go to the page for a revised list of the foods that the user has entered.
* User can click the “Home” link to advance to the Home page.
* User can click the “About” link to advance to the About page.
* User can click the “User Profile” link to advance to the User Profile page.
* User can click the “logout” link to advance to the landing page.



1. **Revised list of foods (TBD)**
2. **User Profile**

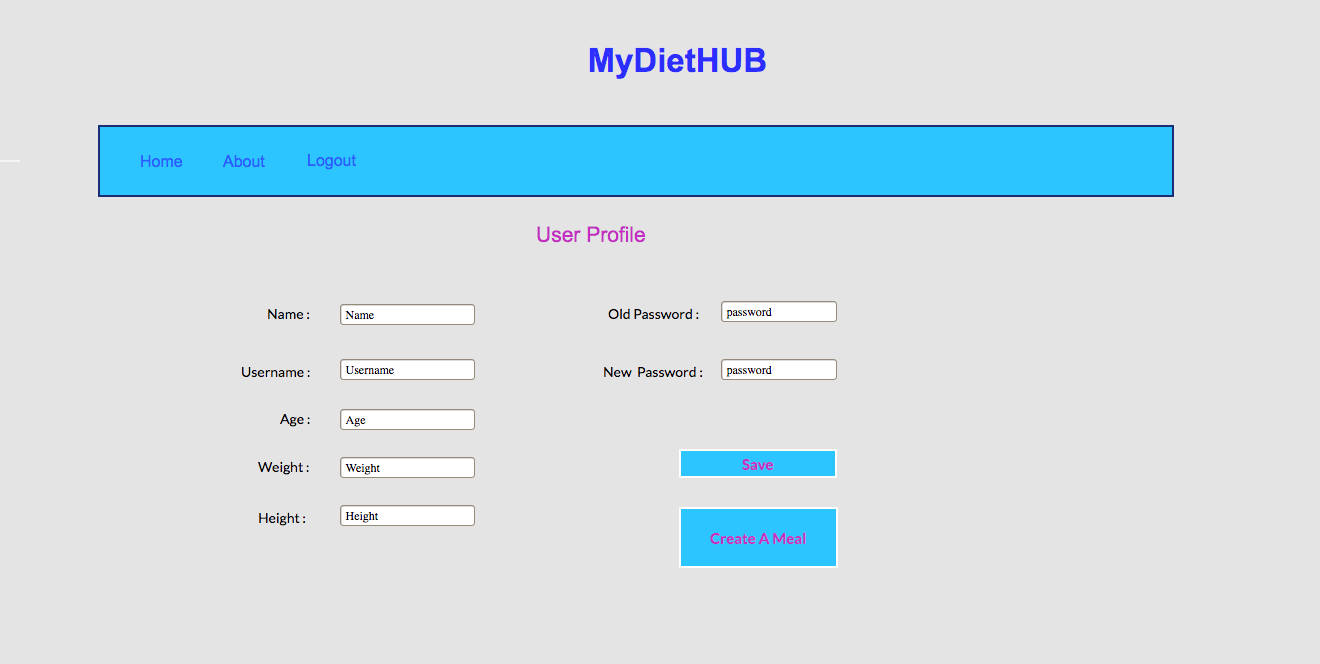
This section describes the “User Profile” page. The main component is a form for the user to edit the Name, age, weight, height and username and let the user change the password.

Purpose :

To edit the user’s name, age, weight, height and username and let the user change the password by entering the new password.

Navigation & User Interaction:

* User can edit the name, age, weight, height, username and new password fields and then click “Save” button to save the new information for the user.
* User can click “Create a Meal” button to advance to the “Create a Meal/ Calculate Calories” page.
* User can click the “Home” link to advance to the Home page.
* User can click the “About” link to advance to the About page.
* User can click the “Logout” link to advance to the landing page.



# Classes and Methods

Please provide a link to your application API document which should be automatically generated.

# References

# Glossary