COMP2011 Web Application Development: Coursework 2

BY: Hammad Shahid

Website Deployment Link: https://zeos5eek.herokuapp.com

Purpose of my website

The purpose of this website is to help users improve their health and fitness by providing them with tools and resources to monitor and manage their nutrition, and exercise. It will have a database of foods and workouts the users can add to and make it easy to track food that they have eaten and how much they have exercised, this will allow them to stay motivated and accountable on their fitness journey. It will be able to calculate the calories from the data inputted by the user, such as the protein, calories, and fats. With a formula.

Three tier Architecture

In a three-tier architecture, the application is divided into three layers: The presentation layer/user interface layer, the Application layer, and the Data layer. In this application flask is a three-tier architecture. The presentation layer is known for handling user requests and sending data back to the user, in our application this is done through the templates and the view.py which define the layout and content of the response that is sent to the user.

The Logic layer is known for implementing the core functionality of the application. In this case of this application, this layer is implemented using functions, and modules that define the behaviour of this application by performing tasks such as data validation, processing, and calculation. The data storage layer is responsible for managing and storing data, in our case, it is done using the SQL database.

By dividing the application into three layers, allows us to have a modular and maintainable application that is easy to understand and modify. This architecture also helps with the performance and scalability of this application. Which will allow us to modify it later independently of others.

HTTP features

HTTP is a protocol that is used to exchange data on the world wide web. It is also a client-server protocol, which means that it allows a to send requests to a server, and for the server to send responses back to the client. It is also a stateless protocol meaning that it does not keep any information about the client or the server between requests. This means that my website cannot contain any information about the user or state about the user or their actions across multiple requests since it doesn't know what data was transferred in the last request only the request it's currently running.

One of the key HTTP features this application uses for request handling is the request method which specifies the type of action that the client wants to perform. This application supports all common HTTP methods GET, POST, PUT, and DELETE, which allows the application to define different behaviour for each method.

In this application, I use GET Method to query information from the database about the Food and Exercises that a user has done. And view it on the website. As this method cannot make changes to the data it is safe. The POST method is used to submit data to a web server for processing, such as when submitting a form or uploading a file. Whenever I am submitting data to the form, I am using the POST method, which allows it to get the data and submit it to the database.

Request handling in HTTP also allows this web application to validate and process the data included in a request before responding to it. This will help ensure the integrity and security of the website and to prevent it errors or attacks.

Web Forms

In my application, I have implemented web forms to collect input from users. For example, I have used web forms to gather information about the food the user is eating and the category they want to assign it to. The web forms use flask form elements to create the user interface and allow users to enter their responses. The form data is then processed by the backend server, which is then using python to validate the input and store it in a database.

Database

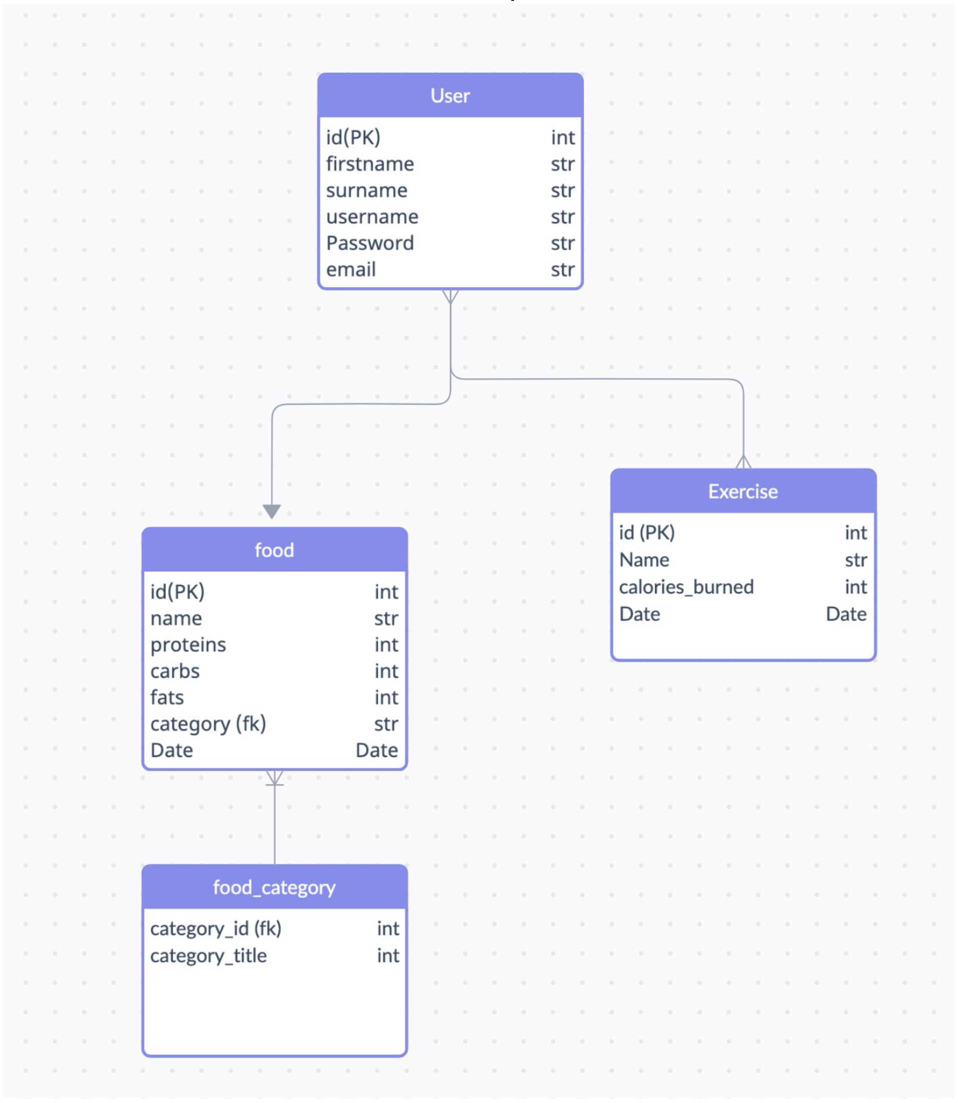
In my database, I have 4 tables: Food, Category, Exercise, and User. These tables are related to one another through relationships. The first relationship is between Food and the User, and the second is between Exercise and the User. These relationships store which User has eaten what Food and has done what Exercise. The third relationship is between food and food category which allows us to know what food goes in what category.

In this database, there are 2 many-to-many relationships. The first is between Food and User, where a Food can have many Users and a User can have many Foods. The second is between Exercise and User, where an Exercise can have many Users and a User can have many Exercises. This allows for flexibility in the data and enables the database to store complex information about which Users have eaten what Foods and have done what Exercises.

These tables can be used to calculate the number of calories each user has burned and consumed. The user table enables the storage of personal information for each user, which can be displayed to them in their account information. This allows for the creation of individual accounts for each User.

The food table can be used to store information about the Food that each user has eaten, while the exercise table can be used to store information about the type of exercise each User has performed and the number of calories they have burned. The category table allows for the categorization of Food items, providing a means of organizing the data in the Food table.

By combining these tables and relationships, the database can provide detailed information about each User's caloric intake and expenditure.



Sessions and authentication

When a user logs into the website, Flask login verifies their login credentials (usually a username and password) and generates a session ID. This session ID is then stored in the user's web browser in the form of a cookie, which is a small piece of data that is sent from the server to the user's browser. When the user makes subsequent requests to the server, this cookie is automatically included in the request, allowing the server to identify the user and retrieve their session information.

The server uses this session ID to look up the user's information in a session store, which is a database that contains the session data for all users. This allows the server to access the user's information and preferences, and to keep track of their session as they navigate the website. When the user logs out or their session expires, the server deletes their session data from the session store, effectively ending the session.

In this way, authentication is managed effectively because a user cannot access the system without entering the correct login details. This prevents unauthorized users from simply typing the URL and gaining access to sensitive user information.

The application also uses cookies to store username, so that when a user logs in and returns to the website, the website can recognize the user and greet them on the login screen. This improves the user experience by making it more personalized and convenient. If a user logs in with a different account, then it would update to greet that user.

Styling and Development

To improve the visual aesthetic of my application, I have implemented the Bootstrap framework for styling. This has allowed me to easily apply consistent styling across all elements, including error messages and the Account information page, which is presented in a Bootstrap card layout.

I have deployed my app to Heroku, a cloud platform-as-a-service provider, which allows for easy and accessible hosting of my web application. This enables anyone with an internet connection to access and interact with my website.

Advanced Features

I have implemented two advanced features in my website using JavaScript. The first feature is a real-time password matching validation, which allows users to check whether the passwords they have entered match while creating an account. This helps save time for the user, as they do not have to submit the form to find out if their passwords are correctly matching.

The second feature is designed to improve accessibility for users with partial vision. This feature allows users to easily navigate through the buttons on all pages using the tab function, and it includes a hover effect to provide visual feedback and indicate which button is currently selected. This feature is also available for use with a mouse, providing additional flexibility for users with partial vision.

Tests and Logging

I have implemented a suite of unit tests for my website using the unit test module in Python. These tests involve creating a disposable test database, running various tests to ensure the proper functioning of the website's core features, such as verifying that links are functioning correctly, and database queries are executing properly. Upon completion of these tests, the test database is destroyed. By running these tests, I can ensure that the website is functioning as expected.

In addition to unit testing, I have integrated the logging library into my application to track the runtime behaviour of the website. This allows me to monitor the functionality of each page and identify any errors that may occur. The logging functionality enables me to generate log messages for events such as page load errors and other exceptions, providing insight into the causes of potential issues with the website. This information can be valuable for troubleshooting and debugging the application. I have implemented two types of logs, namely Info and Warning, to differentiate between the various log entries.

Accessibility

In my application, I have implemented keyboard-based navigation for all buttons on the page. When a user presses the tab key, the focus will automatically move to the next button on the page. Additionally, I have added a hover effect to the currently focused button to provide visual feedback to the user and indicate which button they are currently selecting. For users with partial vision, I have also added a mouse hover effect to the buttons to make them more visible and easier to locate.

To improve the accessibility of my website for users with partial vision, I have implemented high contrast colour schemes using the “colour” and “background-colour” CSS properties. For example, I have set the background colour of the website to white and the colour of the buttons to blue, which provides a high contrast and makes it easier for partially blind users to see and interact with the website.

Security Issues that I had with my website

One potential security issue that I identified with my website was the lack of user authentication and authorization. Before integrating the Flask-Login extension, the website did not have any mechanisms in place to verify the identity of users who logged in. As a result, it was possible for anyone who knew a user's username to access their account by simply entering the username in the URL, which could have allowed unauthorized access to sensitive user information. This issue was addressed by implementing Flask-Login, which provides user authentication and authorization features that help protect against this type of security risk.

There was a user privacy issue where one user was able to view the food items added to the profile of another user. This was resolved by implementing a check on the user identification (ID) to ensure that only the intended user can view their own profile information.