

MEAL PLAN APP

For project submission towards CFG Nanodegree (Software Specialization)

Abstract

Meal Prep is a nutrition and recipe app to monitor user calorie intake, keep track of their favourite recipes and determine their health

Contributors:

Bora Kim, Charu Gera, Najma Hersi, Ola Ajibola and Whitney Ikenwe

Introduction

This is a nutrition and recipe application for anyone who is keen to monitor their calorie intake, keep track of their favourite recipes and determine their health through an integrated BMI calculator. It could also serve as a tool for Nutritionists and Dietitians to create meal plans for their clients according to their dietary requirements.

Simply enter your details on the user page and you are ready to start using all the features on offer. These include a recipe finder (specific to your food preferences and/ or any allergies you may have) a BMI calculator (which allows you to compare your weight against the NHS guidelines) and a tracking page where you are able to view everything you have eaten (giving you easy access to recipes you have enjoyed – and allowing you to delete any you did not!)

As a group we deliberated over whether to make a command line app or whether the features we had in mind for our app were more suited to a Flask framework-based app in Python. We ultimately decided to use Flask as we felt that this would allow us to utilise everything we had learned on the nanodegree, implement the list of objectives we had for our project, and provide a better user experience overall.

In order to ensure that we completed all of our desired tasks efficiently and in time, we adopted the Agile methodology during this project. This allowed us to delegate tasks and continually review our achievements and remaining work. It also meant that we were meeting regularly, and this created a really supportive network which we have all enjoyed being a part of. We adopted the use of Trello board to hold ourselves to account with regards to the tasks at hand. And we used GitHub for version control allowing us to update each other on any changes to our application's code, test each of these changes and leave comments for each other.

This report will outline the rationale behind our project idea and choices, the specifications and design, how we implemented and executed our code and how we tested it. We also consider what we would have added if we had more time as we are firm in our belief that everything can be improved.

Background

Our project is a Flask based app. We have both a frontend and backend (which we have implemented using HTML, CSS and Jinja2). The app is incredibly intuitive to use. The user is first directed to the 'Home' page where they are given the instructions as to how to navigate the app for the best possible experience.

The application is intended for a varied use. This is from independent users wanting to just get meal ideas based on their needs, users who would like to check their health and then search meals with the appropriate calories, to nutritionists who may want to keep track of their clients' meals, check their progress and advise them best on their diet.

We wanted to ensure that our application catered not only to those who are allergy-free but to the most common dietary requirements, including peanut allergy, shellfish and gluten. We do however understand that this is not all the possible allergies and so see room for future

development here. The diabetes filter was also an important implementation for our application so we could branch out of just dietary restrictions but also include health cautions in the meal choices too.

Users having the choice to save and then track the meals they have saved, and presumably made to eat was another important feature of the application. We are hoping that when combined with the other features of the app, that over time our users would be able to refer back to their eating habits and make choices (in line with the NHS guidelines) regarding their daily calorie consumption depending on what their BMI status is. Users being able to view their calorie intake and their recipe choices will provide the user with a sense of accountability as to what health choices they have been making (and what health choices they should be making promoting the improvements and maintenance of healthier lifestyles over time.

Specification and Design

There are many requirements both technical and non-technical that were agreed in this application. The technical requirements include: site functionality, so ensuring that users can input their information where appropriate. That all the submission buttons work and all the pages can be used and navigated easily without any errors. One major feature in our project is that users need to input their information in order to make full use of the site due to the application tailoring meal options to a specific user. Therefore, having technical requirements that will ensure that the user does complete these steps was crucial and we were able to implement this.

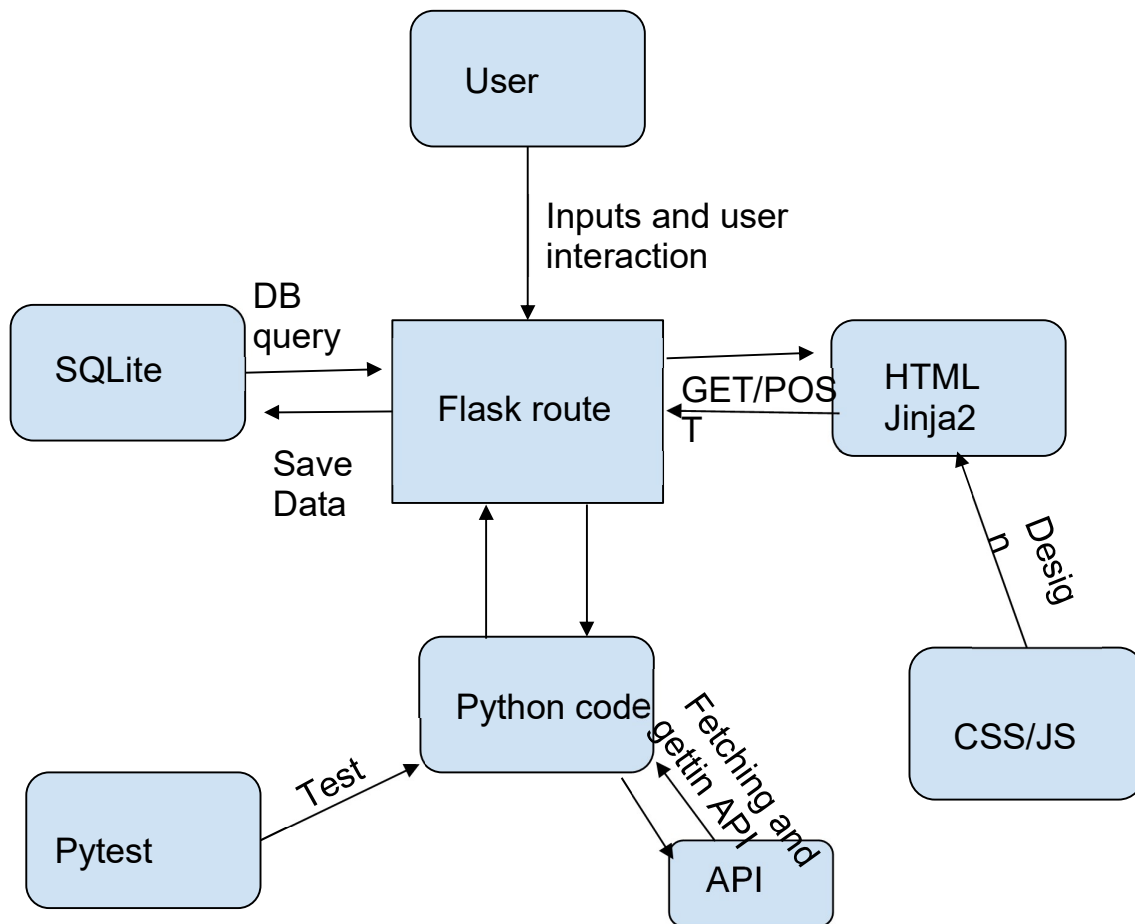
Another functionality technical requirement was users being able to save their food. Using SQLite allowed our team to achieve this.

Non-technical requirements laid mainly in the interface of the application. These can be described as the surface layer on the functionality requirements. In order for a user to be able to have their information saved and track their food as well as receive health guidance after their BMI calculations the most important non-technical requirement was that all of these features were clear and visible for the user to do. For example, having a clear place to input your name and age was a non-technical requirement which then in turn aids the technical requirement of having that information saved to a database.

A more obvious non-technical requirement is content of the application, particularly with its relevance. Once we had decided on an idea and focus for our application, which is a place for people to plan their meals, have health guidance (via BMI calculator) and save meals, we wanted to make sure what is being portrayed on our application was based on this; making the content as relevant as possible.

The ideas for the design and architecture for this project was to keep the site simple and easy to navigate. The order of the pages in our site navigation bar have also been put in order for which the user should use the site. Providing a simple and clear site is important to allow the user to have a good user experience. The tones used on the site are neutral to keep users focused and not distracted by any unnecessary features.

In regards to the architecture of the application, this can be best detailed in a diagram:



Implementation and execution

As mentioned in the document earlier, the group debated using a command line vs flask app-based approach for the project. Ultimately the group decided to use the latter, as it allowed us to incorporate other coding languages that we know such as html, css and js. Additionally, creating a flask based app would allow several users to be added to the application without terminating the programme, which was something that we all wanted to make sure of.

Together as a team we did a lot of pair-programming where each person had a chance to take the lead. This allowed all of us to implement code into our program while being able to see and understand the code of one other. Working in pairs also allowed us to have ideas and help from other members of the group, especially when needing to debug.

We used various tools and libraries such as the following:

Libraries

- Flask
- SQLAlchemy
- Jinja2

- Requests
- Werkzeug
- Pytest
- Ast

Tools

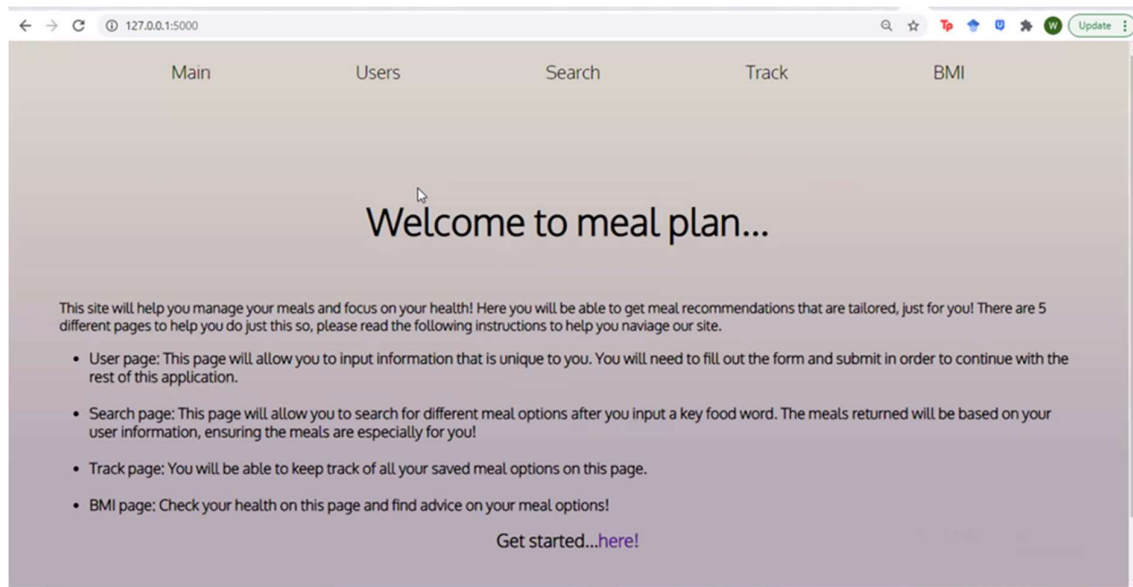
- Python
- SQLite DB
- HTML & Jinja2
- Postman
- Trello Board
- Edamam API
- Slack
- Zoom

The group used flask to build endpoints and HTML & Jinja2 to make HTTP requests so it can send/receive data to backend python code. SQLite and SQLAlchemy were used for DB and query it out on backend python code as well. TrelloBoard for project planning and agile implementation of the project, edamam API (<https://developer.edamam.com/>) for data, Postman and Pytest for testing, Slack and zoom for team communications and meetings, PowerPoint for presentation and MS Word for documentation.

Implementation Process: After brainstorming ideas and coming to a final narrative for the application we began the project by deciding the objectives. For these objectives to be met, we began with the design of the app. It was decided that the app should have the following endpoints:

- Main
- Users
- Search
- Track
- BMI

Main page served as the welcome page which gave information to users about what the app does and how it executes it.

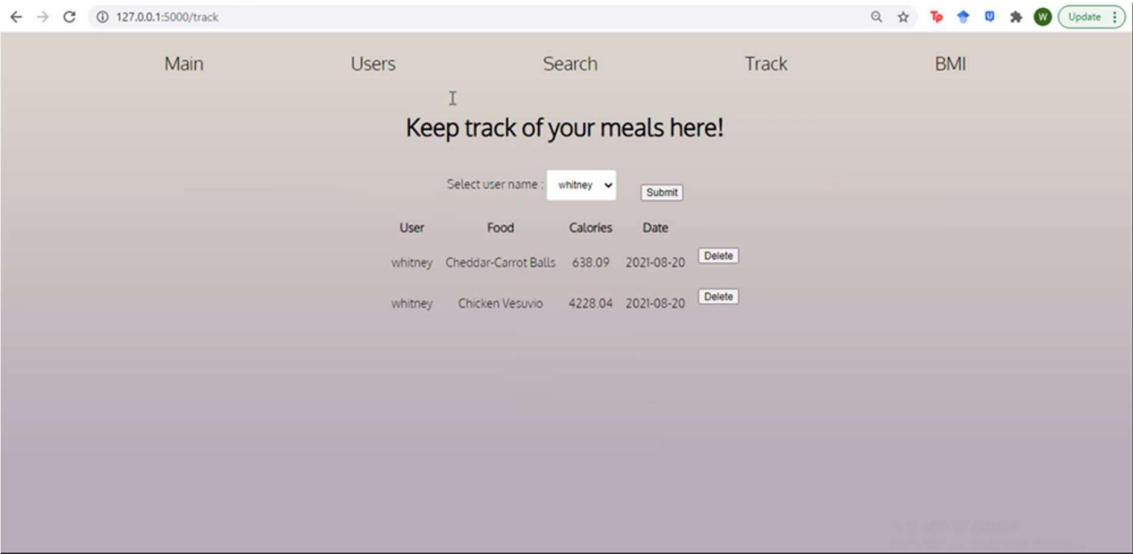


Users promoted users to input their information under the following headers:

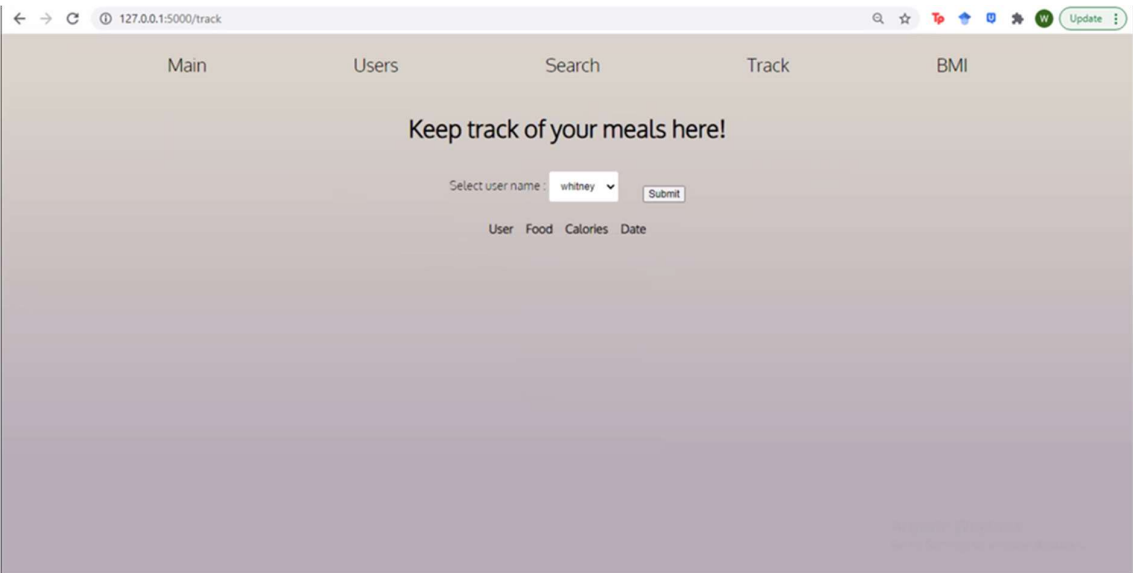
- UserName- Text input
- Age -Numeric Input
- Gender - Text input(Male, Female)
- Diabetes- Text input (Yes, No)
- Allergies-Text input(Peanut, Shellfish, No allergy)
- Activeness-Numeric input (1-10) 1 was the least active and 10 was the most active

Search

This endpoint was used to keep a track of the users meals based upon the parameters added by them on the users page. For example, if a user had diabetes, this page showed them only low-calorie options. If they had peanut allergy, the users were shown only peanut free recipes, so on and so forth.



Track



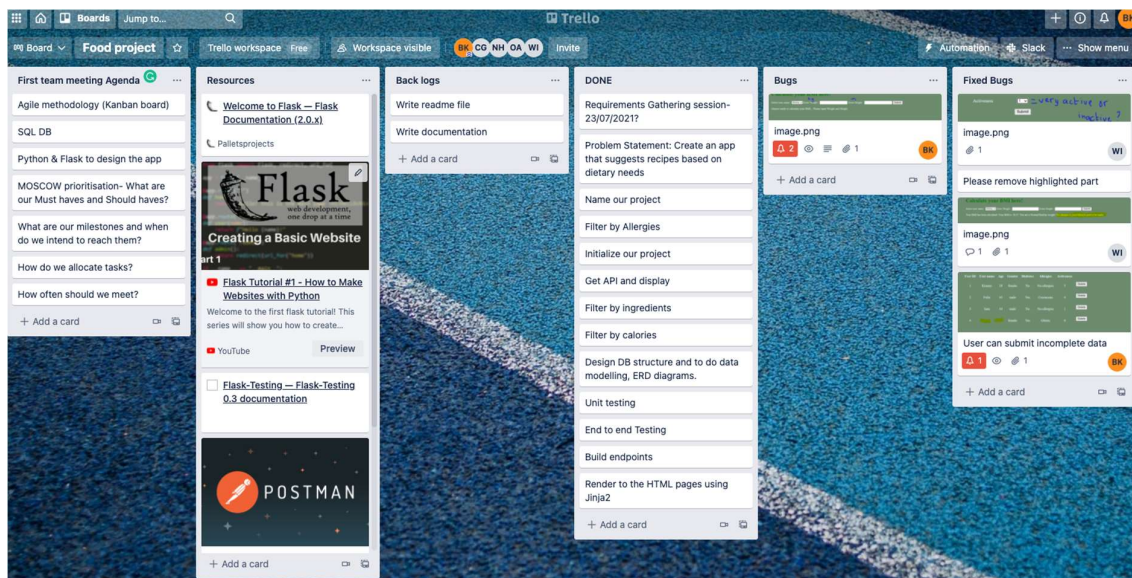
BMI

This endpoint asked the user to input their weight in kilograms and height in meters. Based on this information their BMI was calculated. The screenshot of this endpoint is shown below.

The screenshot shows a web application with a navigation bar at the top containing links for 'Main', 'Users', 'Search', 'Track', and 'BMI'. The main heading is 'Calculate your BMI here!'. Below this, there is a form with the following fields: 'Select user name:' with a dropdown menu showing 'whitney', 'Enter Weight (KG):' with an empty text input, and 'Enter Height (Meters):' with an empty text input. A 'Submit' button is located to the right of the height input. Below the form, a message reads: 'Almost ready to calculate your BMI... Please input Weight in KG and Height in Meters.'

Agile Development

We had regular stand-ups to get up to speed about any progress that was made from the previous day and to discuss the plan for the day. We reviewed our Trello board and sprint burning-down chart frequently to ensure we maximised productivity. When any of the team were feeling stressed on completing their tasks, we often worked together to resolve the problem which ensured we were able to meet our daily milestones. The Trello board and agile methodology were both vital in the competition of the successful project.



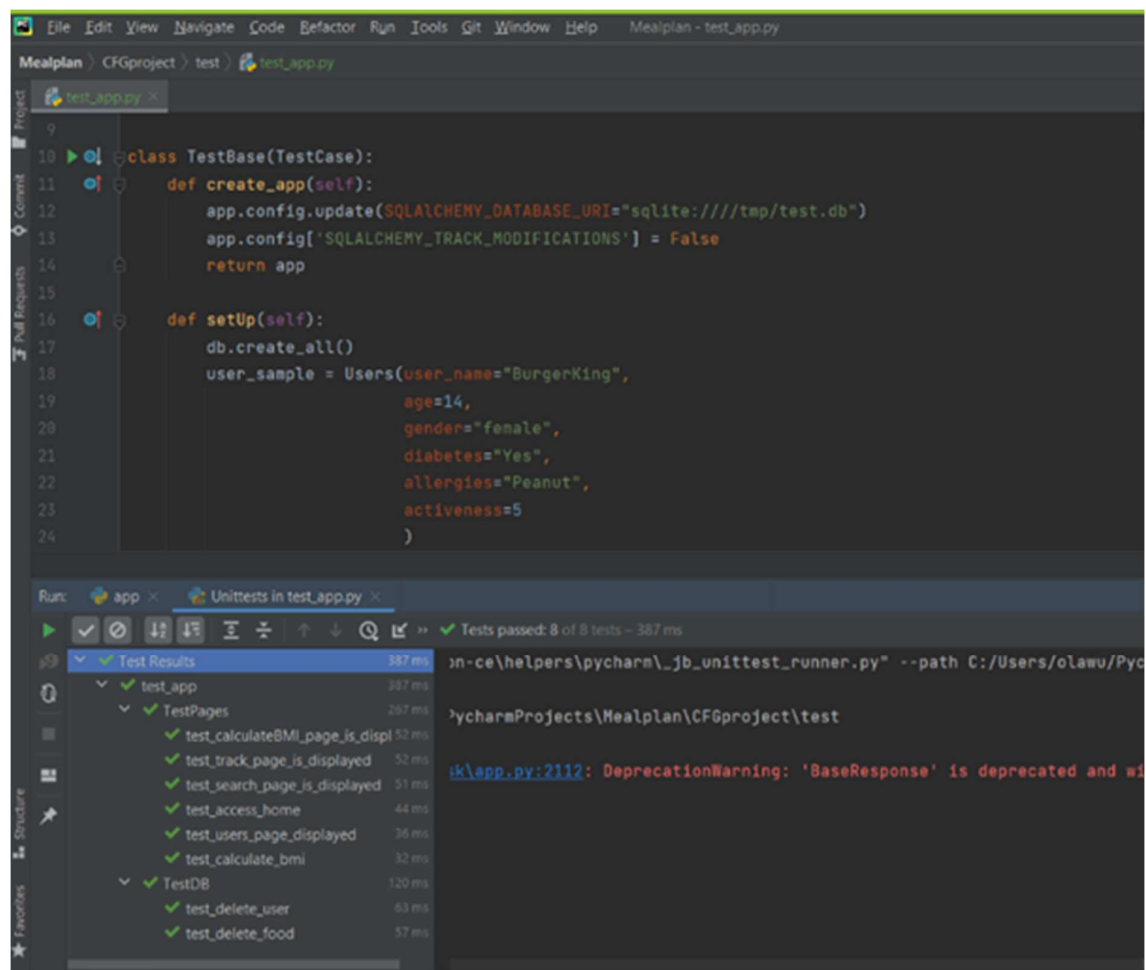
In regards to Implementation Challenges, when it came to implementing the code for our program one of the prominent challenges was having to learn and adapt to new techniques and languages such as HTML, SQLite, SQLAlchemy and Jinja2. All of these were essential for the running of our program and therefore adapting to this was a challenge, however not one we couldn't conquer.

Testing

We tested our application to ensure that it runs as it should. Below is a summary report of the testing activities completed for the MealPlan Application. Development followed the Agile methodology therefore the team was able to start testing the application early.

Unit Testing

Unit testing was carried out using the Flask-Testing extension. This extension provides unit testing utilities for Flask applications. All endpoints and functions of the application were successfully tested.



The screenshot displays the PyCharm IDE interface. The top pane shows the source code for `test_app.py`, featuring a `TestBase` class with `create_app` and `setUp` methods. The bottom pane shows the 'Run' output, which includes a 'Test Results' table and a command prompt window.

Test Results	387 ms
test_app	387 ms
TestPages	287 ms
test_calculateBMI_page_is_displ	52 ms
test_track_page_is_displayed	52 ms
test_search_page_is_displayed	51 ms
test_access_home	44 ms
test_users_page_displayed	36 ms
test_calculate_bmi	32 ms
TestDB	120 ms
test_delete_user	63 ms
test_delete_food	57 ms

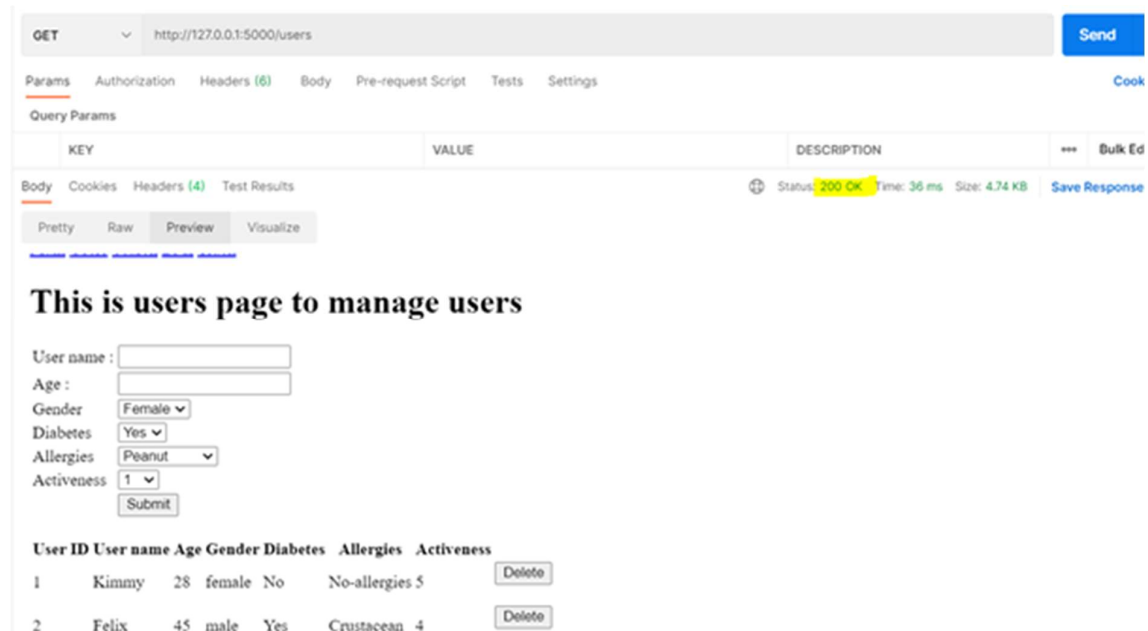
Tests passed: 8 of 8 tests - 387 ms

```
python -ce\helpers\pycharm\jb_unittest_runner.py" --path C:/Users/olawu/PycharmProjects/Mealplan/CFG6project/test
```

`ik/app.py:2112: DeprecationWarning: 'BaseResponse' is deprecated and will`

Integration testing

The Postman tool was used to test and verify our API requests and endpoints and to ensure that responses were as expected.



System Testing:

No performance issues were detected during this testing phase. In addition to testing new features as they were added, regression Tests were carried out manually after every successful merge on Github to ensure that code changes did not break the application.

Defects found during this stage were raised and assigned to the relevant member of the team and tracked. In total, 4 bugs were raised and these have been fixed. Retests were carried out after each bug fix.

User Acceptance Testing:

All group members tested the application as end users would and the general consensus was that the application behaves as expected and is easy to use however this could be a biased view as all the testers in question were involved in the development of the app.

Evaluation/Conclusion:

Overall, the MealPlan application is one that helps users make better food choices. Although development and testing were carried out to the best of our abilities, we are aware that our application can be improved upon. The following are work that we could have completed if we had more time:

- Inclusion of more allergies
- More lifestyle questions added.
- Carry out more UAT involving Nutritionists

Future Enhancements:

We have considered some future enhancements that would advance the application if we had more time and these include:

- Itemising portion size to measure caloric intake more accurately
- Include ratings for recipes and include suggestions based on these ratings
- Weight chart: This can be used with the calculateBMI feature to monitor weight loss/weight gain progress.
- A feature for users to set a target for themselves (weight loss/gain or nutrition) which they can work towards.

Conclusion

Overall, we were able to achieve a fully working app in the set time which we believe is both attractive and informative to the user. We hope you enjoy using it as much as we have enjoyed working on it!