**CFG PROJECT REPORT**

**WELLNESS Web Application**

*By Hafsah Ali, Amanda Mandiki, Vilte Noreikaite,*

*Nivetha Kamallathurai, Nina Krasteva*

**CONTENTS**

1. Introduction. ………………………………………………………………………………………… 2
   1. Aims and objectives of the project……………………………………………… 2
   2. Roadmap of the report ……………………………………………………………… 2
2. Project background. ………………………………………………………………………………4
3. Specifications and Design. ………………………………………………………………………6
   1. Design and architecture ……………………………………………………………7
   2. Requirements technical and non-technical …………………………………7
4. Implementation and Execution. ………………………………………………………………7
   1. Development approach and team member roles …………………………7
   2. Tools and libraries ……………………………………………………………………… 8
   3. Implementation process ………………………………………………………………8
   4. Agile development ………………………………………………………………………9
5. Testing and Evaluation. ……………………………………………………………………………10
   1. Testing strategy ……………………………………………………………………………10
   2. Functional and user testing …………………………………………………………10
   3. System limitations ……………………………………………………………………… 12
   4. Future Improvements …………………………………………………………………13
6. Conclusion. ………………………………………………………………………………………………13

**WELLNESS PROJECT**

1. Introduction.

The project we undertook was designed to enable users to manage their overall wellbeing, by providing them with a flexible alternative to the conventional way of journaling. This was done in three ways: we created a journaling page, where users can log in and with the help of prompts, they can set goals and objectives for the day, which should bring them clarity and a sense of purpose. We also added a page for Mood tracking, as we recognise the importance of being mindful of your current emotional state and intended this page to help users recognise why they may be experiencing negative emotions and to also be able to remedy this with the help of some of the external resources. The external resources have been to encourage the user to seek help if they are feeling unwell or seek to improve their mental wellbeing overall. This report will detail how we tackled the project.

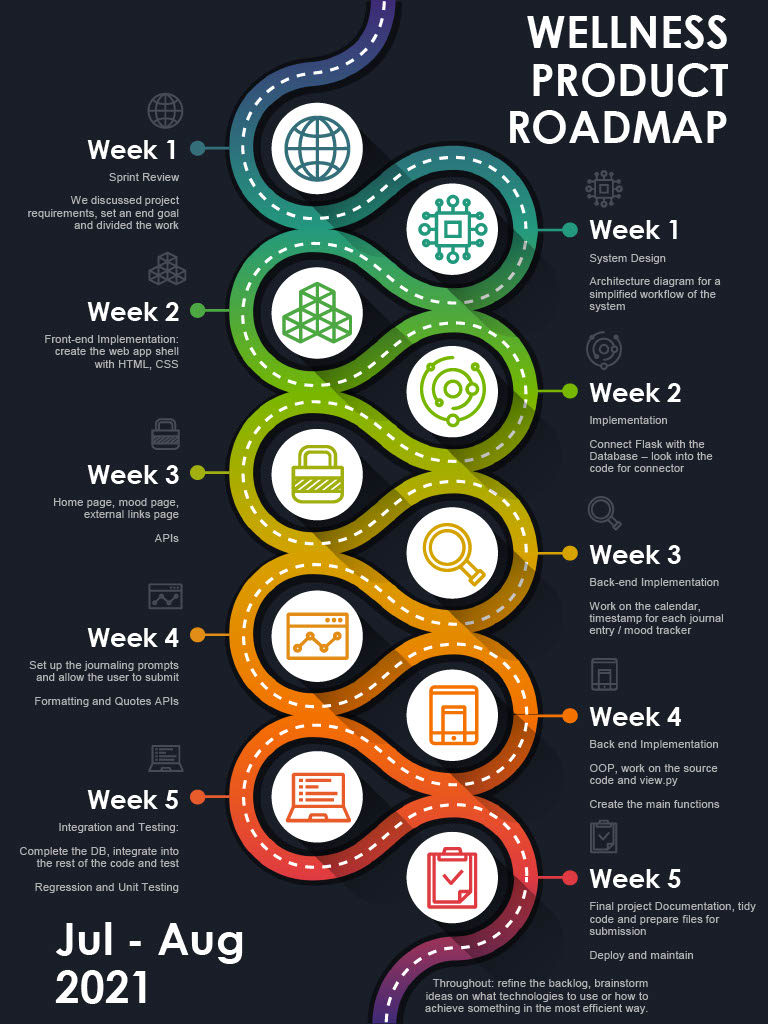
* 1. Aims and objectives of the project

The project is intended to present an alternative to the conventional journaling, as this way the user can access the website on the go and have access to all resources in one place. The mental wellbeing topic has recently been on the rise, with many organisations prompting their employees or students to actively take care of their mental state in whatever way is convenient for the individual. Moreover, following the Covid pandemic a lot of people have been living in isolation, separated from friends and family and unable to do activities or hobbies that were in their daily routines prior to the pandemic. It is for those reasons that we have chosen this topic, with the intention to satisfy the gap on the market and provide a safe space for people to be able to express and track their feelings, bring clarity and direction to their day and access useful resources on wellbeing.

* 1. Roadmap of the report

As the diagram below demonstrated, we divided our team meetings to a minimum of two per week, however we also had follow-up meetings whenever necessary. We aimed to follow the SDLC process of planning out our work, starting with Planning, Requirements Gathering and Design, moving on to Implementation and Testing and finally Deployment and Maintenance.

We structured our workload in a way that would make it possible to return an MVP by the project deadline. Starting with research on how to build the front-end, as none of us had previously worked with HTML and CSS. We then proceeded to building the APIs, core functions (source code) and connecting those with the database. Finally, we performed the necessary tests, both functional and non-functional to ensure that our MVP product is user ready.

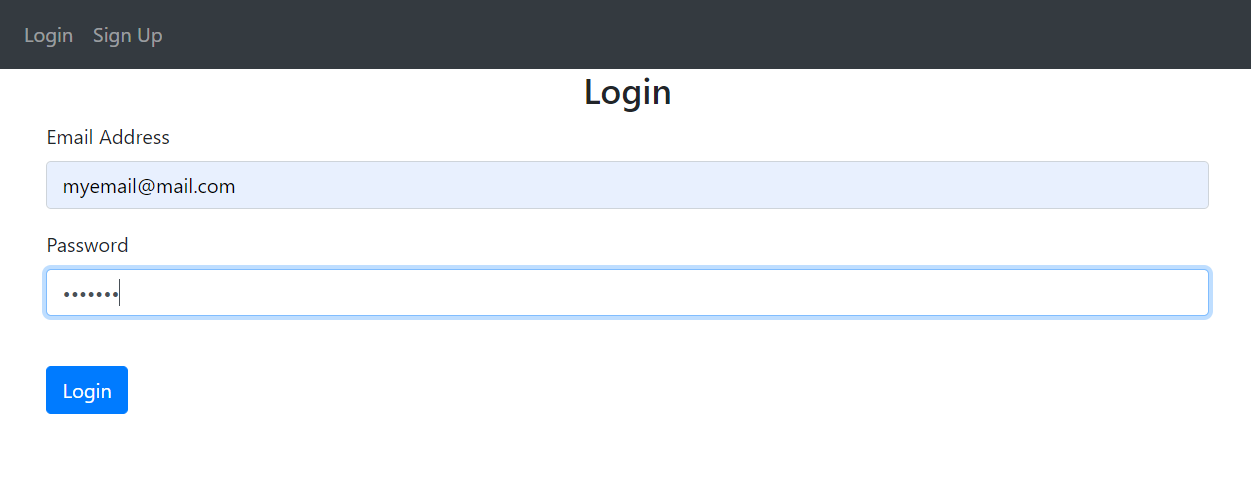


1. Project background.

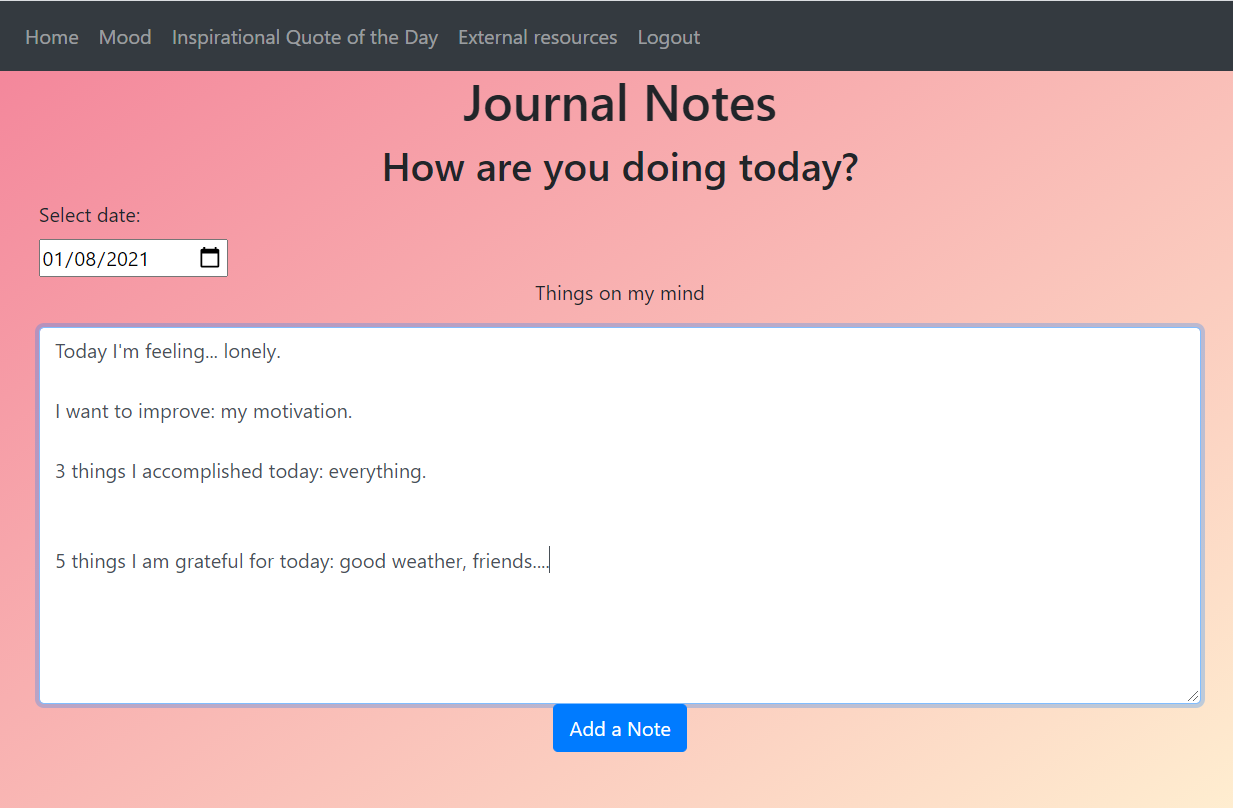
The web app for our project was designed in a way that would allow individual users to access their accounts and have a secure place for journaling and mood tracking. The link between the web app and the database allows users to “write” information, that is saved in the database and also “read” or retrieve information that they have previously submitted.

The Wellness Web app intends to make the user’s journaling experience more comfortable and easier, with the option for daily mood tracking and a link to external resources to help users manage their wellbeing.

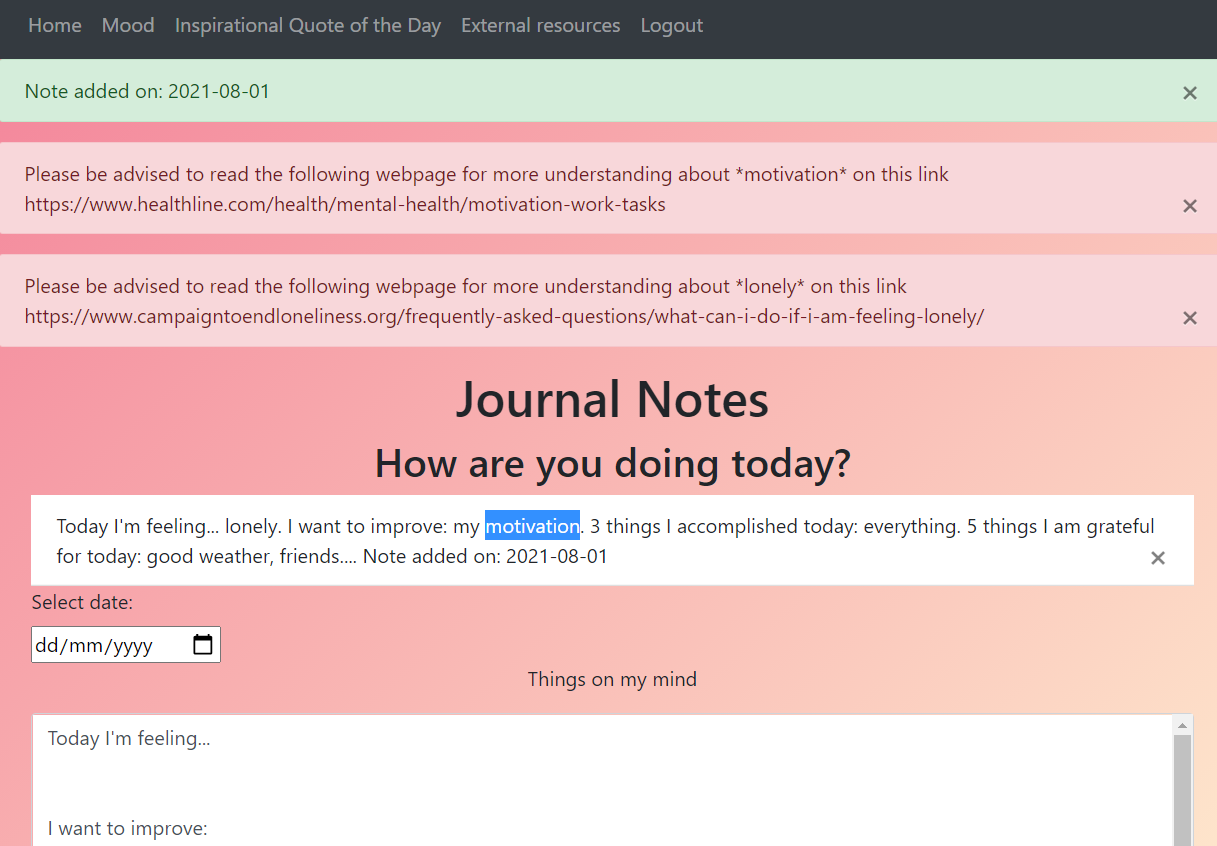
The web app is designed to store user log in details, so that the journal inputs would be kept secure and visible only to the account holder.



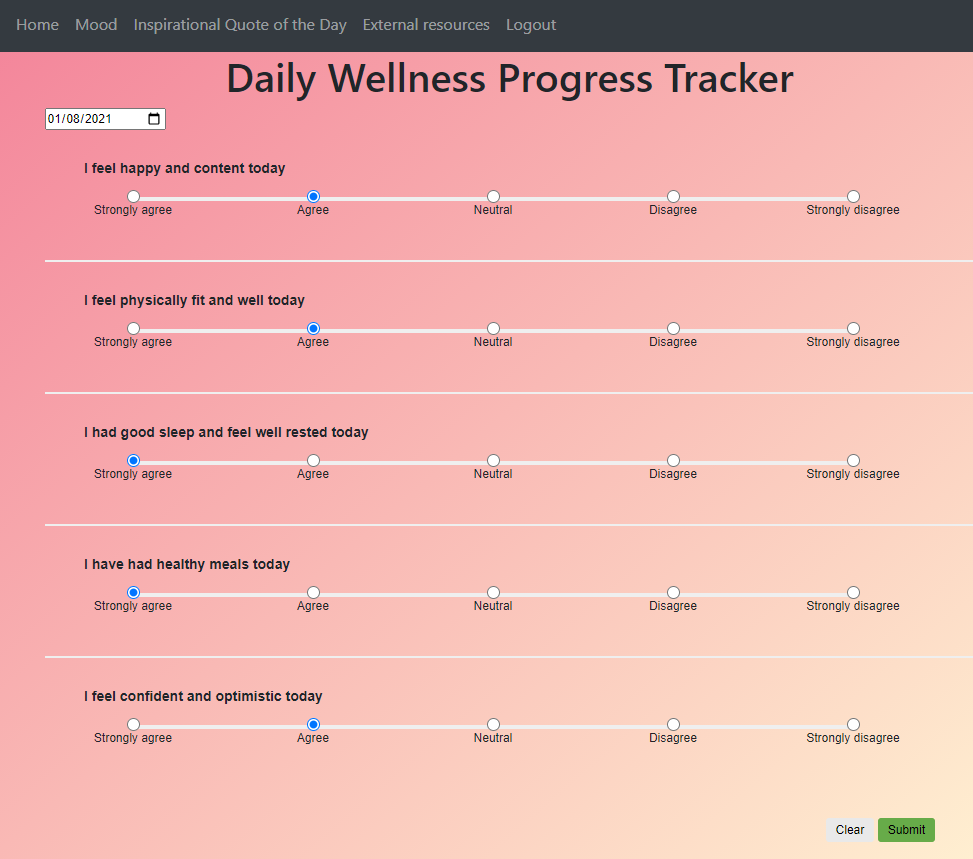
The users are given several journaling prompts, which encourage the user to dwell on their day, practice gratitude, bring clarity and awareness to their mental wellbeing state.

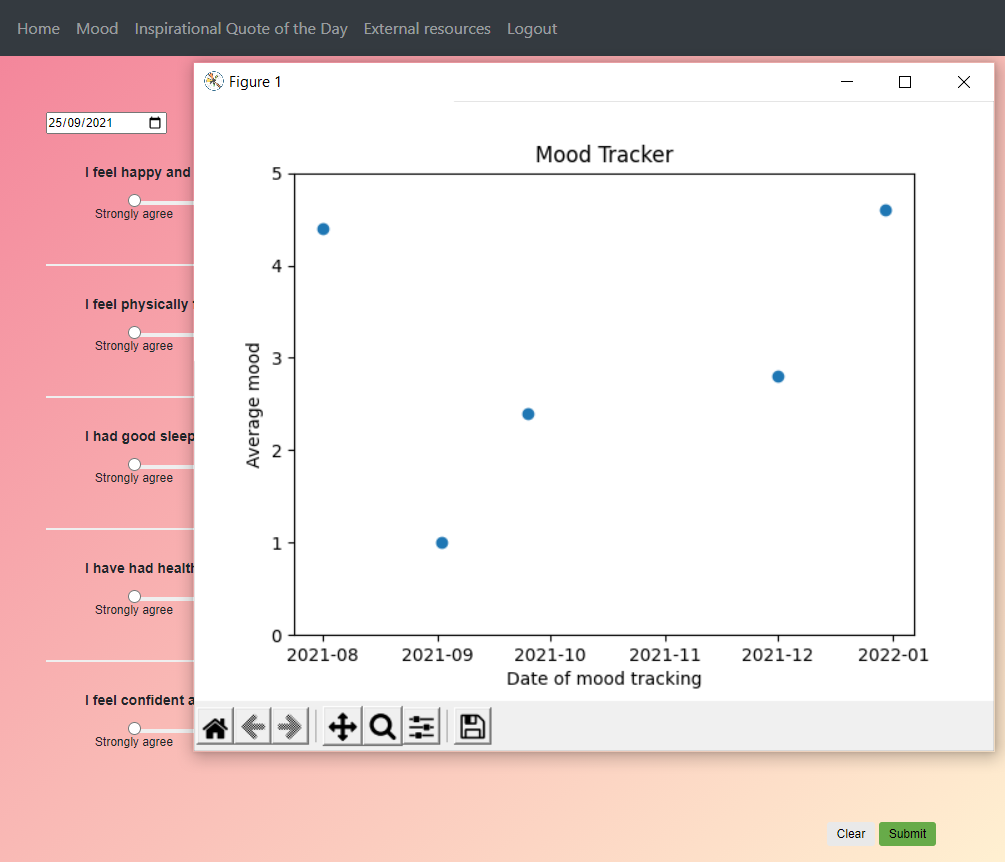


Once the user has completed the journal, they can submit their answers. And the keywords from their journaling input will be used to flash messages with advice.

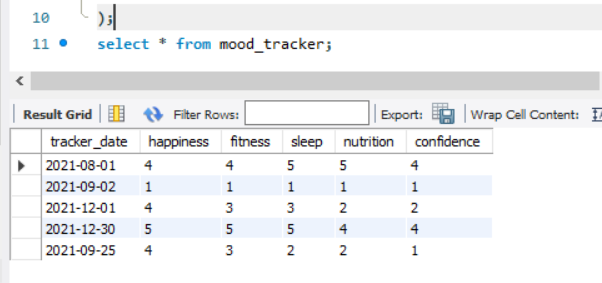


The Daily Wellness Progress Tracker can be found under the “Mood” page and it is designed for the user to answer several questions and once submitted, this will return the user’s overall mood for the day with a score, and it will return a graph of mood scores over a period of time.

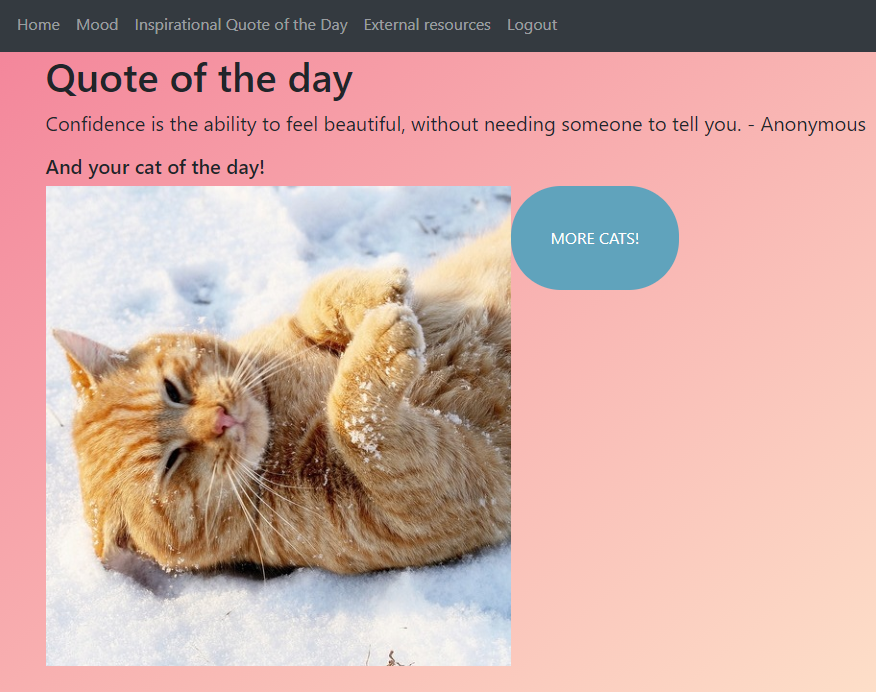




The input from the Mood tracker is stored in the MySQL database.

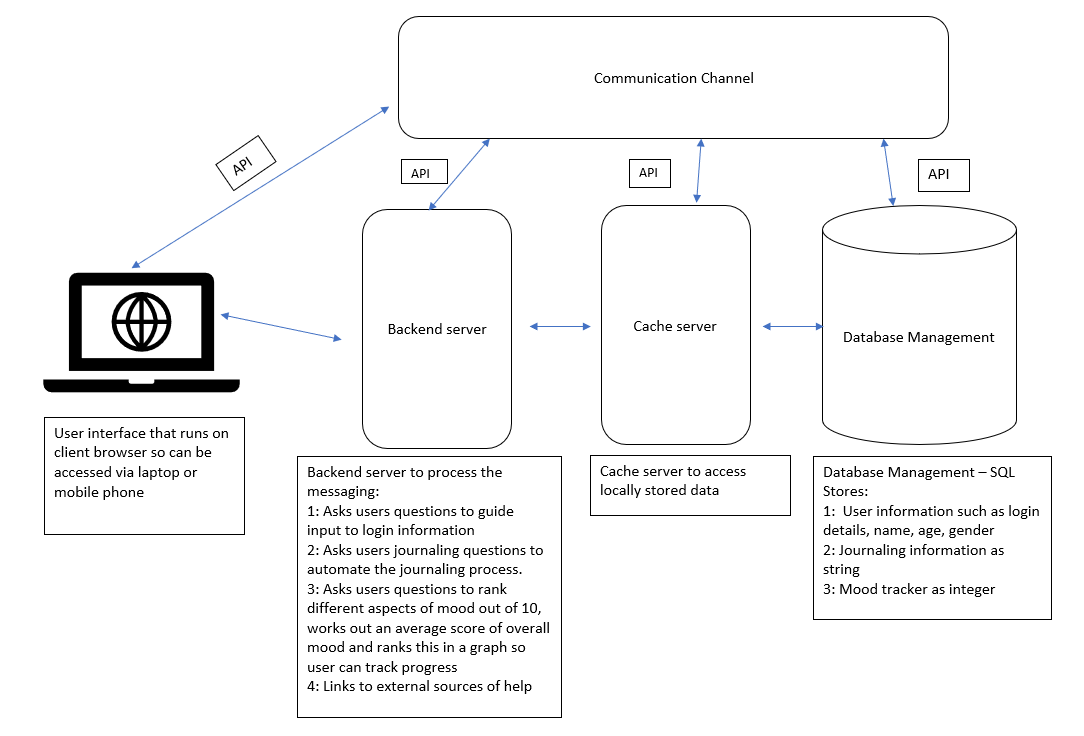


The inspirational Quote of the Day returns a motivational quote and cat pictures from APIs.



1. Specifications and Design.

Key system features:

* Personal account with sign / up login functionality which stores information for that account
* Input information regarding their day, such as mood, what they want to achieve, how they are feeling, etc., as well as relevant links to external sources in order to tackle their mood goals.
* Mood tracker - graph, graph to show progress (weekly / monthly)
* External help / advice links
  1. Design and architecture.
  2. Requirements technical and non-technical
* The system will maintain availability of at least 95%.
* The system will have an average page load time of less than 5 seconds.
* The system will comply with our architectural and security requirements.
* Privacy: the user interface will not allow any third parties to access any private user information such as DOB, email, password.
* System errors will result in an error code that will be communicated to the user and this error code will also be documented, in order to resolve the issue.
* The system will continue to function if an API is down with the exception of any functions that depend on uncashed data from this API.

1. Implementation and Execution.
   1. Development approach and team member roles

We undertook an Agile development approach, which was dynamic, flexible and we were open-minded when it came to altering our plans, how to divide the tasks or using a different technology/tools.  
  
With regards to team member roles, most often than not we collaborated by having brainstorming sessions/code reviews, etc, however as a generalisation we completed the following:

* Amanda - Creating the base of the website (API and endpoints for each webpage such as mood, login, logout etc.), journal notes, authorisation of user to sign-up, login, logout, as well as unit testing.
* Nivi – Creating the mood tracker form. Working on the mood tracker part of the database. Connecting the mood tracker page so user inputs are stored on the database and retrieved to show graphs.
* Hafsah - Creating the external resource webpage and creating a dataset of external resources. Quote generator, API webpage (with Amanda) Formatting and colour scheme for all webpages, also set up a google document for the PDF and started working on bullet points.
* Nina – Working on the code for Python MySQL connection and the database tables regarding the journal (user details, login). Completing the Report documentation and ReadMe files.
* Vilte – implementing a calendar to get user input and display it on journaling and mood tracker pages, adding flash notes on mood tracker page to prevent user from submitting a form with missing values. Together with Nivi connecting the user input from mood tracker page to the MySQL table, and displaying a graph for mood ratings. Added cat picture API to generate random cats on a webpage.
  1. Tools and libraries

Packages:

-mysql-connector-python

-numpy

-matplotlib

-pip install flask

-pip install flask-sqlalchemy

-pip install flask-login

* 1. Implementation process

One of our initial challenges and achievements later down the line was learning how to create the front-end of our web application. After doing the research we created a working web page and gained new knowledge in HTML and CSS.

Because of the time constraint we made the decision not to save the users’ journal inputs, but rather just have them reflect on their goals and set a daily agenda. We focused on creating the Mood Tracker page which instead of a calendar view, would display a daily score of their mood, which would be displayed with a corresponding emotion picture. We also added a pop-up graph that shows average mood scores from different points in time.

Another challenge throughout the implementation process was to do with completing the unit tests for return functions with render\_templates, as well as understanding how the methods=[POST, GET] are implemented in practice. We made a decision to change the way the users input their details, as we were unable to create a table for this, however we learned how to connect frontend (HTML), backend and the database, more specifically understanding how to convert user input from buttons in html, to variables that can be used in python. Having HTML, Python and SQL interact with each other was a challenge, which we approached with perseverance and worked on it to the best of our abilities given the time constraint.

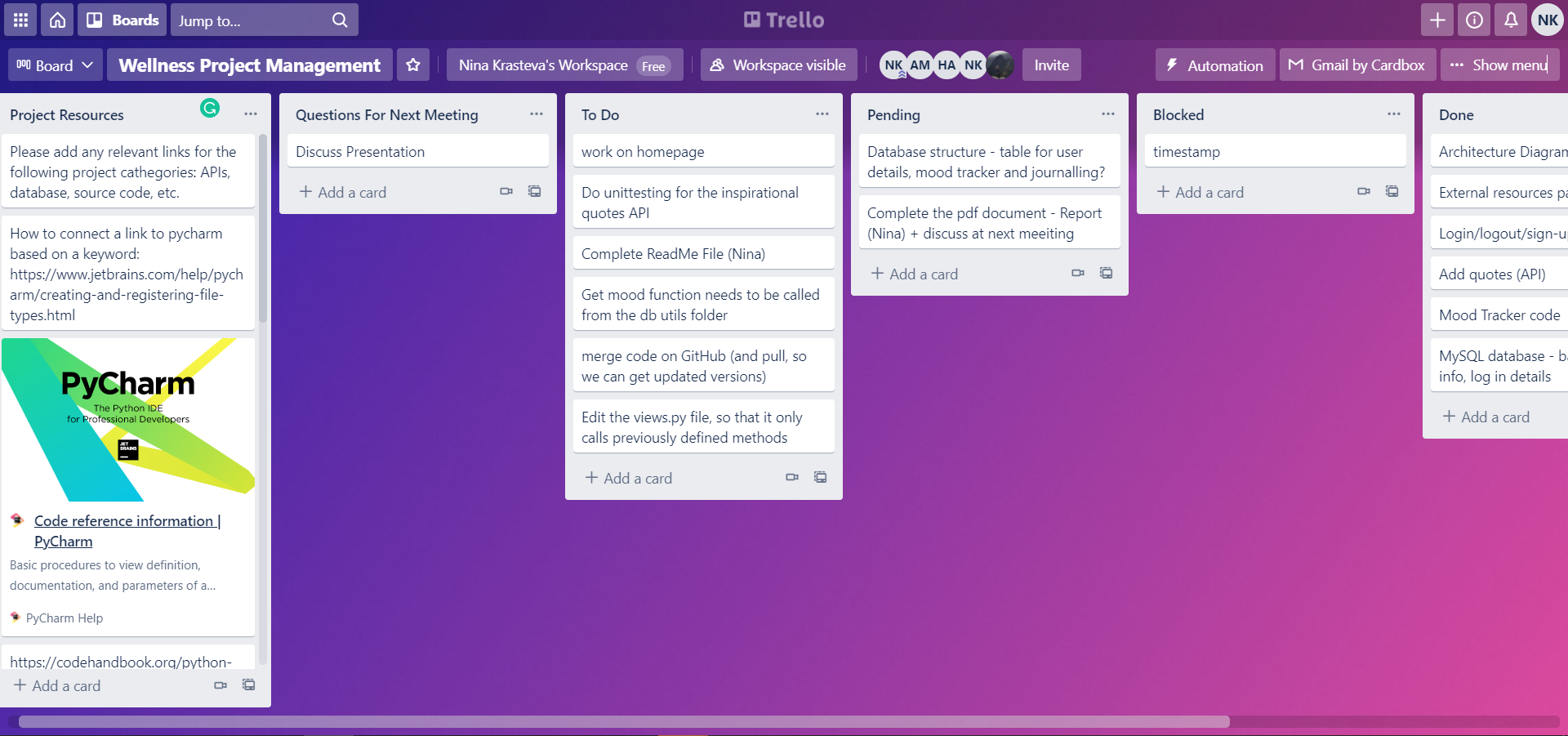
Overall, we learned how to work as a team, delegate tasks and set achievable goals. Learning how GitHub works was a challenge that we all undertook, by creating a team repository and shared files, created multiple branches and merged them. One of our greatest achievements as a team was sticking together, being supportive and collaborative and following the Agile methodology. The process of working on this project was a learning experience that was extremely useful for everyone on our team.

* 1. Agile development

We undertook an Agile development approach, which was centred around the idea of iterative development. As a remote team, we found ways to communicate our intentions and code alterations with the use of platforms such as GitHub (where we would store the code and utilize Version Control), Slack (for everyday communication and exchange of ideas), Trello (To manage our projects and tasks). This allowed us to keep track of our group goals, as well as individual goals and to bring clarity on what needs to be done, as well as accountability.

Moreover, we aimed to work in a SCRUM methodology, so we would divide each work into a separate Sprint. At the beginning of each week, we hold a team briefing, which equates to the first step: Sprint Planning. We would clarify what needs to be achieved that week, divide the work and collaborate during the planning stage. Although we were not able to hold a Daily Scrum as such, we were continuously in conversation over Slack regarding any questions or setbacks we were facing. Often, we would find time prior to the lessons and hold a 10-15min conversation that we would dedicate to evaluating what has been done and adjust or adapt our approach when necessary. Near the end of the week, we would have a meeting intended to represent a Sprint Review, which we utilised to showcase the progress we have made thus far, refine the product backlog on Trello and also contemplate our goals for the next week. We would often combine this with a retrospective of how this week has gone, as most of us have busy schedules and cannot make separate meetings.

A key concept we held into account was flexibility and adaptability - ensuring that we find a way to work though some challenges, adjusting the product backlog items as necessary and use different technologies to what was originally planned. We learned to work as a team in a flexible and supportive way and we often did code reviews, as we understood that a second or third perspective on the problem at hand comes in useful. This was instrumental in completing the project in an Agile environment.

The picture below is from our Trello board, where we would set and keep track of the tasks and refine the backlog.

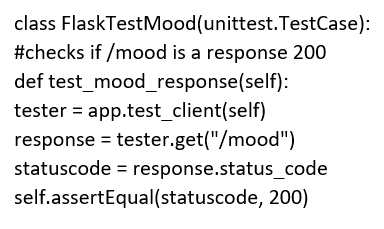
1. Testing and Evaluation.
   1. Testing strategy

Initially, within unit testing we had to first determine if our HTTP response status code was a 200. The HTTP 200 OK success status response code indicates that the request has succeeded.

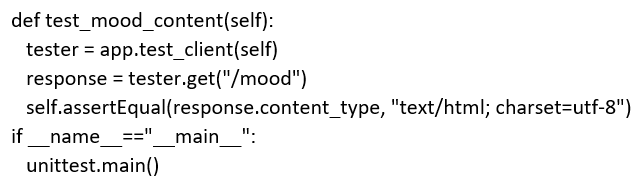
All testcases were 200 apart from the main ‘/’ which was a 302. But this does not count as a 200 because the ‘/’ page is only for redirecting the user. This is similar to google.co.uk as it is the home domain which only directs rather than gives any information.

Thus, this was not considered an error. However, for future projects maybe creating the journal page into a separate endpoint like ‘/journal’ would have better as the ‘/’ can only be redirecting the user and not causing any conflicts.

* 1. Functional and user testing

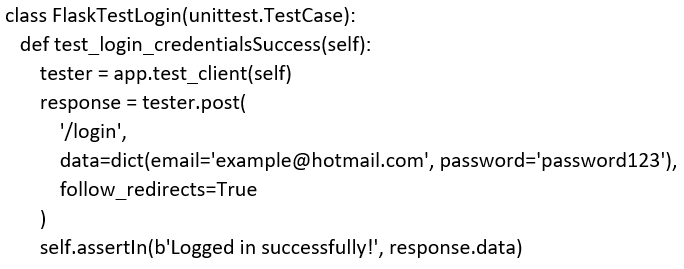
Example of a unit test case for mood in response checking:

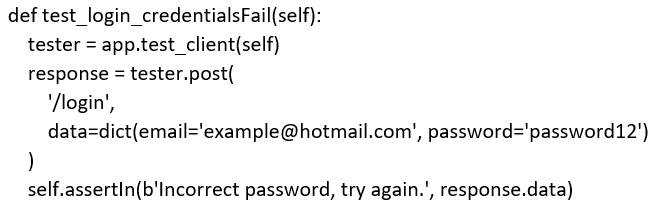
Then another unit test was completed to check if the function of the @apps.route returned a html template for each webpage. All of them had succeeded in this part as each endpoint returned a render\_template()

Example of a unittest case for mood in content checking:

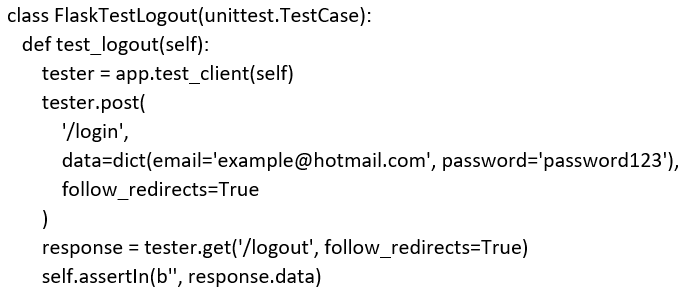
Once checking if the response and content had passed. We moved onto checking if certain functions were returning what was expected. The first function to be tested was the Login aspect. This was the most challenging aspect as it required understanding what was occurring behind the scenes to be logged in successfully. In this case it was specific to the user’s computer as this line:

Required to input the correct email and password from what was created on the website. So if no credentials were found in the database this output would be unsuccessful. Thus, an account must be created before testing this function.

Unit testing to see if correct login details would let us login and send us a ‘Logged in successfully!’ message:

Conversely, the opposite was also tested to see if wrong credentials would flash an error for ‘Incorrect password, try again’:

Finally, the function ‘logout’ was also tested:



This tested whether the user would be redirected to another endpoint when logging out and also that they received no flashing message. This test also passed.

Further testing:

* We performed non-functional tests on the web application, from a user point of view, i.e. quality control (we tested it all ourselves).
* We performed non-functional testing, including performance testing (although this was limited, due to lack of user), compatibility testing (different computing environments) and usability testing.
* We had checked manually if the user could post journal notes, if the mood points would be entered into the database, if python could GET/retrieve the data from MySQL and if a graph was presented.

Difficulties and limitations during testing: There were not many functions created that return a simple datatype instead it was all "text/html; charset=utf-8" so we were limited in unit testing, as all the information was based on HTML/CSS templates.

* 1. System limitations
* Limited user capacity at any given time. We tested the system using a small amount of data and a few user inputs, however if multiple users were to use it over an extended period of time, the system capacity would be limited.
* When it comes to time space time complexity, if we were to simplify the code or alter our algorithms in the future, we could improve performance and loading time, however this is also limited for now.
* As we currently do not have one local server which we can all access, the database is stored in a personal device, so there is an access and storage space limitation.
* From a user point of view there would be a limitation if the users are not tech-savvy.
* Limited Mobile Device Feature Accessibility.
* No Offline Availability - users would need to have access to the internet in order to use the system.
* No App Store or Google Play Access — Users won’t find the app on marketplaces, where they are used to finding them. Hence, they may not be aware of availability.
  1. Future improvements

During the next development cycle:

* Currently the mood graph (that is displayed as a pop up) will show the progress of only one user, as the table in the database is not set up to connect this to unique users. In the next development cycle, we will extend this to store all user graphs by linking this with their unique IDs.
* We would tidy the code and have separate files. One for client side - with only methods required for the information clients need, one for controller side i.e. methods we need. Also for better code readability: we’ll put all the def functions in a separate utils page, and then just call the methods in the view page.
* We would like to have used a more OOP-cantered approach for our code development, so at the next stage, we will refine the code and build out classes for the different mood categories, as they can be a part of one superclass. This would also make the code easier to read.
* Given more time, we would have made it possible for the user to store their journal inputs in the database and then to be able to go back and choose on the calendar view which day they would like to review.
* Concerning the mood page, we would like to present a calendar view, as well as a graph of the user’s mood logs though time, so they can go back and cross-reference their mood and their journal at specific days and see what factors contributed to this.
* We would also change the mood results (currently bad/good mood) to different traffic light colours.
* At the moment we are reading the APIs as an HTML response, which is why it cannot be tested. In the future we can store the response in the database and then test it. We will do this by:

1) Using a method to get the value from the API and store it in the database (write function)

2) Read from the DB and link it on our website (read)

3) Test the outcome (the output) if what is actually being read by the DB is correct

1. Conclusion.

In conclusion, this project was developed with the purpose of addressing a current need in society and a gap in the market for a well-structured, eco and user-friendly web app that would encourage users to take care of their mental wellbeing though journaling, mood tracking and access to external resources all convenient to access on the go.

The project was built with the use of Python (and Flask framework) for the back end, HTML and CSS for the front end, as well as MySQL where the main database is stored. With the use of existing libraries and modules a functioning MVP product was built within less than 5 weeks.

We employed the Agile development approach throughout, with the use of SCRUM boards, and weekly Sprints. The Product roadmap and architectural diagram played an important role with keeping us on track with the SDLC process and by refining the product backlog our team was able to keep clear communication and task division. We utilised GitHub for Version Control and tested the final product by means of both functional and non-functional tests.