GROUP 5 – CHILL PILL

Anu Bazarragchaa, Elizabeth Dinh, Saima Khaliq, Gina Rubik, Chidimma Umeh, Aleksandra Wojciechowska

DESCRIPTION
Project Documentation for Code First Girls
Specialisation: Software

Table of Contents

Introduction

Project Epic

Our project, Chill Pill, endeavours to create a comprehensive web application centred around wellness journaling and mood tracking. Chill Pill offers a secure and reflective space for users to document their daily thoughts and emotions, facilitating a platform for users to understand wellness, encouraging self-awareness of one's mental health.

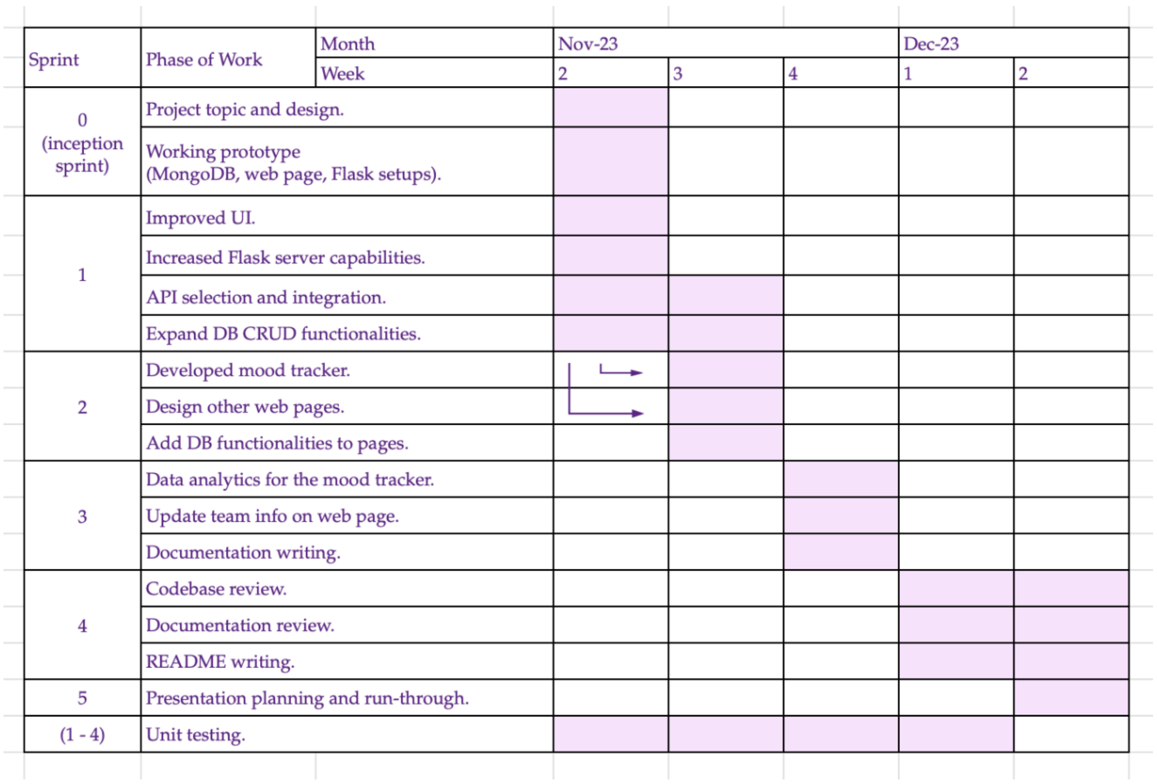
Beyond being just a simple journal, Chill Pill employs analytical tools to uncover insights into the user mood and track trends over time, allowing users to comprehend factors influencing emotional well-being.

Aims and Objectives

- Provide a user-friendly interface users can utilise as a safe space to write emotionally driven entries while being intuitive to use.
- Utilise analytical tools such as a Sentiment API to classify journal entries numerically enabling users to gain insight into their mood trends over time.
- Offer statistics, graphs, and tailored affirmations based on users' moods, fostering positive emotional growth and better comprehension of their own wellbeing.

Project Roadmap

The team followed the Agile Methodology for the software development life cycle of our project. This is illustrated in figure 1, represented as a Gantt chart to outline the sprints we completed and the agenda for each sprint.



Sprint	Phase of Work	Month	Nov-23			Dec-23	
		Week	2	3	4	1	2
0 (inception sprint)	Project topic and design.		▨				
	Working prototype (MongoDB, web page, Flask setups).		▨				
1	Improved UI.		▨				
	Increased Flask server capabilities.		▨				
	API selection and integration.		▨	▨			
	Expand DB CRUD functionalities.		▨	▨			
2	Developed mood tracker.		↳	▨			
	Design other web pages.		↳	▨			
	Add DB functionalities to pages.			▨			
3	Data analytics for the mood tracker.				▨		
	Update team info on web page.				▨		
	Documentation writing.				▨		
4	Codebase review.					▨	▨
	Documentation review.					▨	▨
	README writing.					▨	▨
5	Presentation planning and run-through.						▨
(1 - 4)	Unit testing.		▨	▨	▨	▨	

Figure 1: Gantt chart of project roadmap

Background

Chill Pill is a website designed to help users keep track of their daily moods, store personal insights and visualize their entries' emotional fluctuation over time. This application uses a sentiment analysis API to harvest the users' entries and provide emotional insights and quantify them into numerical values.

The team gained inspiration from one of our teammate's mental health journey. It is not always easy to answer the question "How have you been feeling the past week?" when in therapy, or even when asked by a friend. This made us realise how little time we take to self-reflect on our wellbeing. After testing multiple applications and websites, with the purpose of daily reflection and logging current mood, we found a lot of them were overcomplicated and overwhelming.

Simplicity was the approach for our project: our users can input their daily mood entries, thoughts, or significant life events. The app will in turn log the entries along with its associated timestamps in a MongoDB database. With every successful entry, the user is returned a daily affirmation to encourage daily use. The user may read, update, edit and delete their entries on their own accord. We provide the user with insights such as a plot with their recent sentiment scores and insights such as their lowest and highest sentiment score days and the average sentiment value of the past seven entries. This plot is colour coded, for the user's ease of use and readability.

The ultimate goal of our project is to provide affordable mental health reflection to those who need it the most. With our project Chill Pill, we believe users can reach a higher potential of awareness for their mental health and as of now, the project is currently free to use.

Specifications and Design

Specifications

Chill Pill is designed to provide a safe and intuitive platform for users to document events via journal entries to gain a closer understanding of their wellness. The web app provides functionality for users to analyse their entry where a numerical value for sentiment is returned and visualized graphically to show emotional trends over time.

For the web application to carry out its intended functionality, below are some functional and non-functional requirements we implemented in our project.

Functional Requirements:

1. **Journal Entry Function:**
   - Enable users to create journal entries which are then saved onto the database.
   - Users can read back on previous entries and then update or delete if it feels necessary.
   - Provide a timestamp for each entry to track the date and time of entry.
2. **Sentiment Analysis Integration:**

    - Perform sentiment analysis on journal entries to extract numerical values representing emotional worth (0-10 with 0 being a negative entry and 10 being a positive entry)
    - Classify sentiments to return a personal daily affirmation.
3. **Data Visualization - Mood Tracker:**
    - Display a visual representation of the user's mood over time to track users' wellness.
    - Plot sentiment scores against timestamps using graphical representations.

Non-Functional Requirements:

1. **Security:**
    - We discussed having a user authentication system for the web app to be used globally however we decided to start with creating a remote app that would store data locally to the user's computer due to time constraints of the project.
2. **Scalability:**
    - Choose a database that can handle increasing volumes of journal entries. NoSQL databases such as MongoDB help with scalability in the case that we would want to implement a scale-out architecture in the future.
3. **Performance:**
    - Maintain system responsiveness even with a large number of simultaneous users.
    - Optimize database queries and system processes to reduce load times.

Design

For our program to carry out its desired functionality, we used a client-server architecture where a FLASK framework is used to connect server-side and client-side through HTTP requests (GET, POST, DELETE etc). These requests are sent to specific endpoints that contain functions to handle data, then sending a response to the user accordingly (e.g. affirmations if adding a new entry is successful).

Object oriented programming (OOP) was used to structure the backend where journal entries was our central object. By employing OOP and adhering to SOLID principles we were able to make sound architectural decisions in order to maximise maintainability, scalability, and ease of collaboration across frontend, backend and database teams. Our object for this project were the journal entries; an instance of the 'Journal' class encapsulates functions that carry out CRUD operations such as 'create, 'delete, 'update' as well as abstracting database interactions and creating an organised interface for entry management. A sentiment API by 'text-processing.com' was used to analyse each entry for its sentimental value, rating each entry from 0 to 10 (a scale of negative to positive). To avoid objective entries (entries that are factual rather than being driven by emotion) from being analysed, we implemented a subjectivity tool provided by the TextBlob library to filter out objective entries.

Instead of using SQL as our choice of database, we went out of our depth and used a NoSQL database 'MongoDB' for storing our data. This was the most appropriate choice for our project as we discovered limitations from using SQL that could cause issues if we used it. MongoDB is a flexible schema that allows more freedom when storing unstructured data. In our case, journal entries would have no set length therefore using SQL would be problematic due to having a VARCHAR limit, whereas MongoDB allows for flexibility with the data we want to save. As well as this, MongoDB can also be used as a cloud database (through MongoDB Atlas) and is built to scale-out horizontally. In the event we would want to grow and scale up our web application, MongoDB would be able to handle large influxes of data.

For our frontend design, a pastel pink theme was used for the web application creating a soft and gentle space for users to document their wellness. Our name 'Chill Pill' derives from a phrase that refers to relaxing, focusing on our wellness aspect of the project. This along with the pill opening animation subtly creates a visually stimulating yet aesthetic loop for our page. As a personal touch we decided to add a meet the team page to provide a human element to our web application breaking the wall between the users and developers.

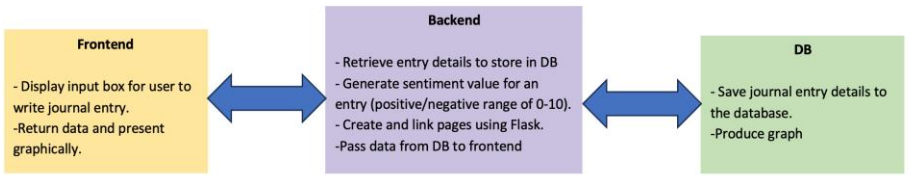

Frontend
- Display input box for user to write journal entry.
-Return data and present graphically.
Backend
- Retrieve entry details to store in DB
- Generate sentiment value for an entry (positive/negative range of 0-10).
- Create and link pages using Flask.
-Pass data from DB to frontend
DB
- Save journal entry details to the database.
-Produce graph

Figure 2: Software Architecture

Implementation and Execution

Our development approach to Chill Pill was cross-functionally collaborative. The team split into three sub-teams: a database team (Alex, Anu, Saima) who were responsible for data queries and data analysis, a frontend team (Chidimma, Gina) who designed and developed the user interface, and a backend team (Lizzie, Chidimma) who handled the backend server and API calls. We decided on the nature and details (such as the main components and tools) of our project democratically and worked in week-long sprints hosted by our project lead (Chidimma) with weekly meetings documented by our kanban manager (Alex).

Our first priority was establishing a simple prototype of the core of our project; in which the user logs a journal entry via a web page and the entry is stored in a database. We built on this by implementing CRUD functionalities that allow the user to manage their journal, and producing a graphical representation of the user's mood with additional interesting insights. During the development process, we faced a few challenges that were discussed as a team and solved practically in our sub-teams. These included considering the subjectivity of a user's entry and the impact a highly objective entry would have on our sentiment analysis. To address this, we introduced a subjectivity API to analyse each entry; this in turn presented its own challenge as we found the API to be stringent in defining subjectivity, requiring us to exercise leniency in setting the minimum subjective threshold. A second challenge we faced was shifting from a terminal-based application to a HTML website. This adjustment necessitated the team to not only rethink the project's design and user interface, but also to work with new technologies and templates such as MongoDB and Jinja. This transition and traversing the accompanying learning curve were a significant achievement for the team, facilitated by efficient communication and collaboration.

The team employed several industry-standard tools for the development of our project, such as Jira for Scrum management, Git and GitHub for version control and hosting our source code respectively, and Zoom and Slack as communication channels. Python libraries such as Flask, Flask-PyMongo, Pandas and Textblob were leveraged for this project as well as text analysis and cat GIF APIs (in addition to the RESTful API we developed for this project). The team adopted an agile way of working, with code reviews and group feedback at the end of each sprint prior to features going live. Tasks were prioritised following group discussion and delegated to the appropriate sub-teams with a completion deadline of the sprint's end. Additionally, continuous integration testing was deployed during the project's lifetime, with unit tests written for new features before their review and subsequent launch. This iterative process allowed for a versatile team who could effectively adapt to the challenges and feedback we encountered, and to changes in the project's direction.

Testing and Evaluation

Testing Strategy

The team adopted testing practices from the Agile methodology, by continuously integrating testing into our sprints. During every sprint, a new feature would be tested by the members who developed it by conducting unit testing and functional testing. If the tests passed, it would be pushed to the respective branch (backend, frontend or DB depending on the team) on our remote repository. Near the end of the sprint, the rest of the team would review the new feature and carry out functional testing; if all members agreed the new feature had been integrated well, it was then merged into the main branch of the remote repository.

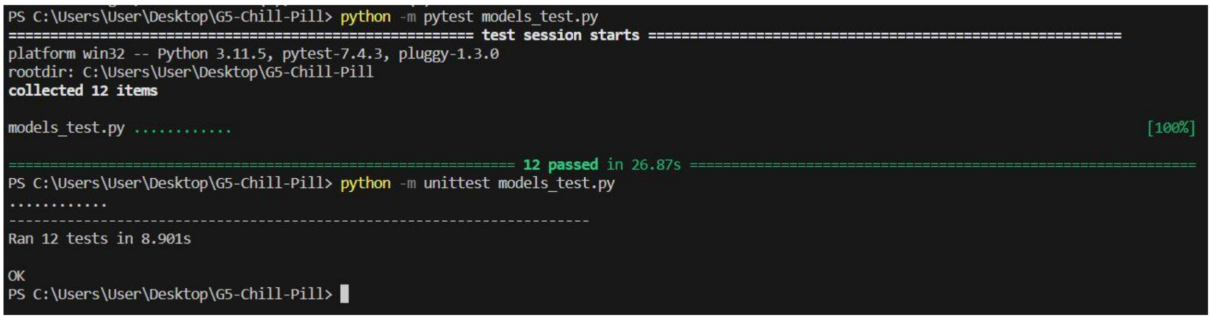
Using this testing strategy, we ensured that the system would function according to the functional and non-functional requirements stated earlier in this report. It allowed for minimal merge conflicts due to the whole team conducting and reviewing functional testing with the new features before it was added to the main branch of our repository. This strategy also reduced the time spent debugging the final version of the system as most issues were detected and solved earlier in the software development life cycle.

Unit Testing

Unit Testing was conducted during sprints by generating unit tests for the newly developed features. This type of testing is essential as it confirms whether a feature can work in an isolated environment before it is integrated with the other features. The database features were individually developed, tested, and moved to their respective classes within the 'models.py' file, to adhere to OOP SOLID principles, and the unit tests were combined into one file named 'models_test.py'.

The testing criteria and justification varies depending on what we were testing for in this project. Within 'models_test.py' file for example, the test criteria for 'TestJournalManager' class includes ensuring the entry creation details are correct and verifying updating and deletion of entries works as intended, by changing entry details and checking if the entry is removed or if it does not exist in the database, respectively. This test class within this file is justified as it validates the CRUD functionalities, which are the core features of our project.

The ‘models_test.py’ file can be run using the testing framework ‘Pytest’ or the in-built module ‘unittest’ and confirms the features in our project pass the unit tests we created. This can be seen below in figure 3.



```
PS C:\Users\User\Desktop\G5-Chill-Pill> python -m pytest models_test.py
========================================================== test session starts ==========================================================
platform win32 -- Python 3.11.5, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\User\Desktop\G5-Chill-Pill
collected 12 items

models_test.py ............                                                                                                        [100%]

============================================================ 12 passed in 26.87s ============================================================
PS C:\Users\User\Desktop\G5-Chill-Pill> python -m unittest models_test.py
............
----------------------------------------------------------------------
Ran 12 tests in 8.901s

OK
PS C:\Users\User\Desktop\G5-Chill-Pill>
```

Figure 3: Screenshot of console log displaying run of models_test.py file.

Functional and User Testing

Members who developed a feature were first to carry out functional testing. This is known as white box testing as these members understand the inner workings of how the feature operates beyond a user interface. Members who were not directly involved in development of the feature carry out functional testing from the user interface once a pull request was made for the new feature to be added to the main branch. This is known as black box testing.

This allowed for us to test the new features from different perspectives and discover bugs that we were not aware of at the time. This strategy also acted as a substitute for user testing as, due to time constraints, the team were not able to recruit users outside of the team to test our project.

System Limitations

The limitations of our system include:

- The system cannot run without an IDE as it has not been containerised (such as through Docker) to include all its dependencies that the system needs to run on multiple infrastructures or platforms.
- The original intention for the display of minimum and maximum sentiment posts was not met as the timestamp dates were to be converted to display the name of the day and date. The system instead displays the full timestamp of an entry on the mood tracker page.
- Using text-processing’s sentiment API and TextBlob to analyse entries means we are relying on outside sources to generate analysis on the text. The sources show to have gaps specifically with negations (eg. ‘not good’), double negatives as well as typically objective words used in a subjective manor.

Conclusion

Chill Pill represents a collaborative effort to create a website that goes beyond traditional journaling. It aims to provide a secure space for self-reflection, offers insights from the input data and in the long term is there to help better one’s mental well-being and self-awareness. In this project we used an Agile methodology and worked in sprints which offered good results and allowed the team to work efficiently and come up with innovative solutions.

Moving forward, there are plans to continuously work on Chill Pill and improve it further as a passion project. There are many directions which this process could take, one of the main ideas is to move from the web and create a Chill Pill app. There are also many features which could be added or developed further. Integration of features such as goal tracking or a personalised pomodoro timer with daily to-do list would further improve the user experience. In a case of ever releasing Chill Pill to be used by the public we would like to focus on data security and privacy, to adhere to the evolving standards. Continuous research and collaboration would also result in advancement in the sentiment analysis and making it more visually appealing.