CFG Project Document

PAUSE

Study Break
Timer App

Group 8 - Ivy Kalegi, Joanna Waller & Nifesimi Komolafe

INTRODUCTION

- Aims and objectives of the project

The aim of our project was to create a user-friendly web application that will help students to study effectively by providing timed study sessions and timed break sessions. This will allow the student to focus during the study time and then undertake a suggested activity in the break time. The activities are aimed at providing Brain Gym movements to stimulate memory and focus, and well-being activities to help maintain a healthy mind and body.

The app will provide help to all students, particularly between the ages of 5 and 18, but students with additional learning needs such as ADHD may especially benefit from using the timer to study with regular short break sessions.

- Roadmap of the report

This report will cover the background and concept of our project idea; the technical and non-technical specifications and the design of our software; details of our team approach, its implementation and execution; how we tested our software and evaluated the final product; and our project conclusion.

BACKGROUND

With the challenges of prioritisation, concentration and time management, students often find self-studying, online learning and/or revision a difficult task and poor time management can even lead to academic failure. It has been found that studying for longer than 60 minutes is not recommended as our brains become less efficient at processing information, therefore, regular scheduled breaks in study time are important. This is the theory behind the Pomodoro Technique, developed by Francesco Cirillo to achieve a focussed, time-work session to accomplish a series of tasks that can be done in short intervals with short breaks in between sessions (Gobbo & Vaccari, 2008). Traditionally, using the Pomodoro Technique the study time is 25 minutes then a 5 minute break, which repeats until you have completed 4 sprints (2 hours of work) then longer breaks are implemented.

We have based the Pause study break timer on this idea. Our website application will allow students to divide up their study time into small timed sessions, selecting the duration of their study time, break time and total study duration. We have suggested starting with 25 minutes study and then when the timer runs out, a study break of 5 minutes begins. This repeats throughout the total study session and will help students manage time and stay focussed. During the study break timer, an activity of approximately 5 minutes will be suggested - this may be a Brain Gym type activity with a YouTube video link to follow or a wellbeing activity with instructions.

The aim of Pause is to provide a website application that can be accessed on a computer, tablet or phone device to help students manage their study time into smaller sessions, have a timed and focussed break session and to ultimately improve their concentration and productivity. Poor concentration is especially prevalent for students with a learning difficulty such as ADHD, so using Pause can help them with any focusing problems they have during study time for homework or revision.

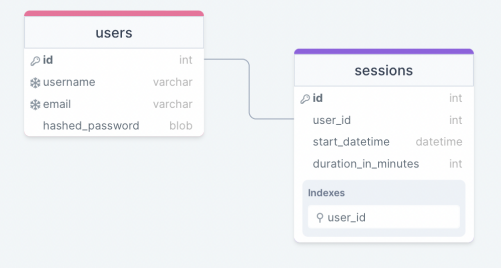
SPECIFICATIONS AND DESIGN

- Requirements technical and non-technical

We used the MoSCoW technique on our Trello board to help us prioritise the requirements in our website application for students:
MUST HAVE - study timer, break timer, database
SHOULD HAVE - user account set up, sign in, break activity suggestions
COULD HAVE - music player, progress tracker, account page
WON'T HAVE - to-do list, study subject tracker

Our main criteria and specifications were therefore defined in the MoS block.

- Sign In page
  The program requires users to have an account before they can begin their study session. The sign in page is therefore the landing page of our app. Users would input their username and password in the respective text fields to gain access to their account. If the username or password is incorrect, an error message would be returned. Furthermore, if any of the fields are left blank, the user would be notified via the error message.
  Once the “sign in” button is pressed, the SQL query is initiated and authenticates the user by checking whether their details match against the **users** table in the database.
- Sign Up page
  If a user does not have an account, they would be required to create one. When signing up, the user is prompted to input their username, email and password.
  Once the “sign up” button is pressed, the SQL query is initiated, and inserts the new users details to the database pending the username and email are unique and the password is more than 8 characters long, otherwise an error message would be returned.
- Database
  MySQL was the database management system of choice to store user information and user study sessions in the **users** table and **sessions** table respectively.
  The users table consists of the user id, their username, email and password stored as a binary large object (BLOB) to allow the storing of the hashed password.
  The sessions table consists of the id of the session, the user id which is a foreign key referenced from the users table, the start datetime and the duration in minutes. The id column in both tables was selected as the primary key and the auto-increment function was implemented to allow for easy entry of a new user and session and to prevent duplicate entries.



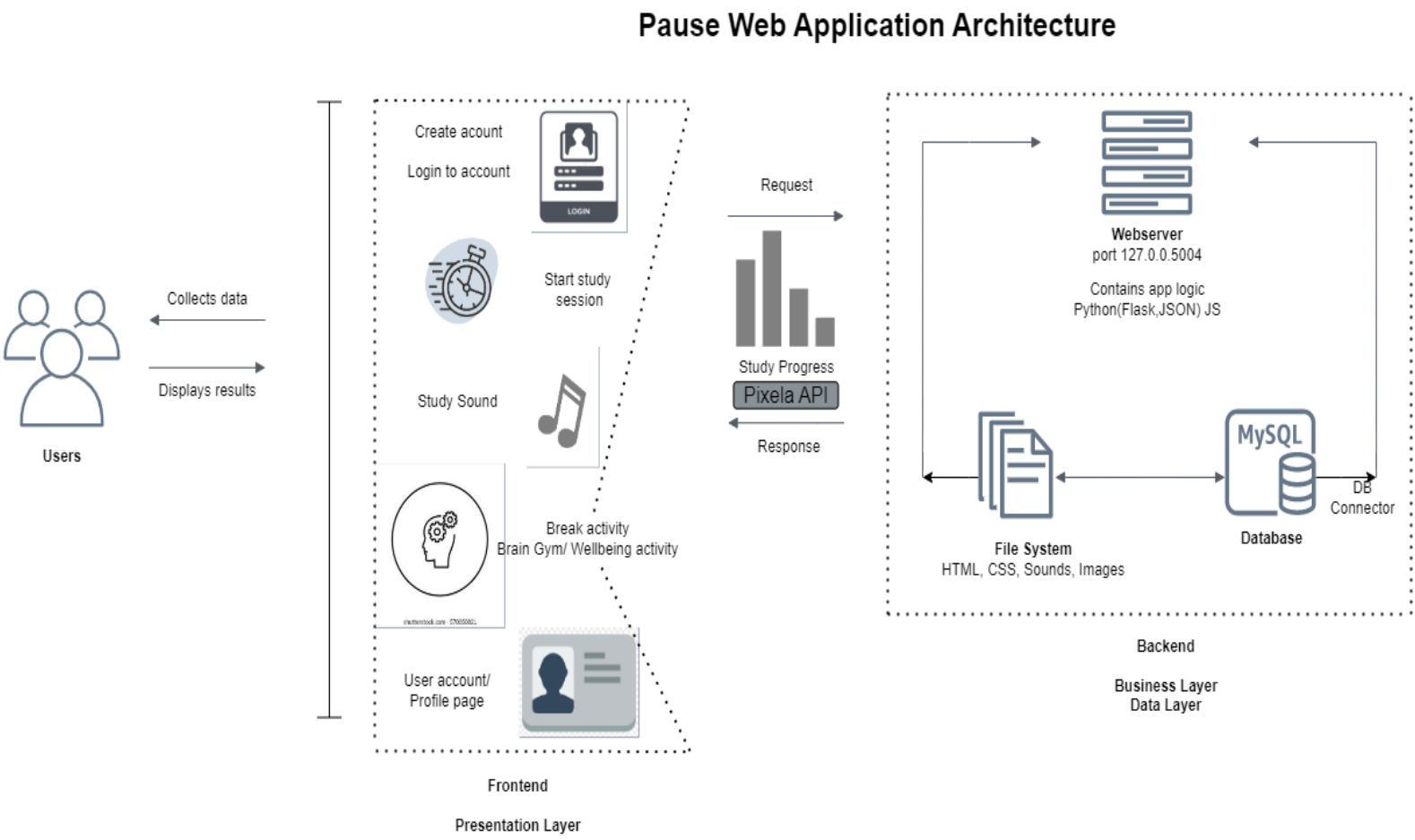
- Timer
  The timer includes a study period (in minutes), break period (in minutes) and the total study session (in hours). The timer has buttons that allows the user to start, pause and reset the timer at any point in time.
  Once the “start” button is pressed, the SQL query is once again initiated, the new session is inserted into the database in the **sessions** table.
- Break Activity
  Throughout the break period, different activities (such as well-being and Brain Gym type activity) would come on screen. The Brain Gym type activity may include a YouTube video link and the Wellbeing activity will have instructions and its benefits. The data for this is stored in two Python files in the computer memory.
- External API
  An external API to display the daily total study time of the user in pixels. It is quantified in minutes
- My Account/ Profile page
  Simple account page showing the user details- their name and password
- About

Tells the user how to get started and an overview of the web app.

❖ Target Audience
We had to tailor the application to students aged between 5 and 18. We kept the features simple, language clear and made sure no inappropriate API's were used.

- Design and architecture

System Architecture Diagram:



The first page that appears on our website is the Sign-in page. The user enters their correct credentials which allow them access to the dashboard. Once on the dashboard there are a variety of options they can do- Start a study session, View their progress, View their account page, View the about page and pick a study sound.

If the user does not have an account, they can create one using the link provided.

As an account is created, a new graph for the user simultaneously, so that their study progress can be updated in parallel. We made use of Pixela API for data visualisation, to display the user progress graph as it is visually appealing and suitable for our target audience. The study time per day was stored as pixels, the longer the user studies, the darker the pixels become.

When the "start study session" is selected, they are directed to another page to commence the session. Here the user can input how long they want each session to be (in minutes), how long the break timer should be and the total hours they are going to study. We recommend a study block of 25 minutes and a break period of about 5 minutes. Once a study block ends, the database is updated using JSON. get

requests in the program. It gets the current date and end time, inserts it to the database and at the same time updates the session to the graph

During the break period, random activities are pulled from the program to engage the user. These are stored in the activities folder in our program.

Once the break period is up, the study session continues again. If the user stops the session, the progress so far would be updated in the database and the progress graph.

There is also an option for the user to choose a chilled study sound to play audio in a separate window during their study session. This is done using Mp3 audio files.

The study timer originally written in Python had to be re-written in JS to allow for functionality in the frontend.

For the design, a design template was made using Canva to visualise how we wanted the frontend to look and then incorporated it with Bootstrap and HTML.

IMPLEMENTATION AND EXECUTION

- Development approach and team member roles

We first had an initial meeting where we developed ideas and planned the project. A Trello board was used to plan tasks and create a project backlog. The Trello board was used to keep track of the tasks. We divided the tasks between each group member:
Timer and database: Ivy
User sign in and database: Nifesimi
Activity data and API: Joanna
Progress tracker chart (external API): all
Frontend development: all
Testing: all

- Tools and libraries used to build our web application

- Programming Language(s): **Python**- wide selection of libraries and frameworks and used IDE's Pycharm and VS Code; **JavaScript**- for the client-side; better choice for making our web-app interactive and adding features. Additionally, it blends well with HTML and CSS.
- Backend Framework(s): **Flask**- We chose Flask because it provides flexibility to expand the application quickly and as we go.
- Database: **MySQL**- for the server-side; support for specialised web functions, considered to be highly rated in the area of data security also for its scalability and reliability.
- Code management tool(s): **GitHub** - Allows for collaboration, compatible with multiple platforms and supports many programming languages.
- External API(s): **Pixela** - A free API that can be used for data visualisation. We set up a main user account where we can then build multiple graphs for each Pause user. This is done by using the POST request to send the USER_TOKEN and USERNAME. A new user graph is created using the GET request to get an empty graph and then the data is added from the timer duration and date using the PUT request to send the data to Pixela. The result is a Pixela chart showing the study time in minutes for that date and the user progress with their studies.

- ❖ Logo and Website Page Design: **canva.com** - with a wide choice of templates and modern designs, we used canva.com to design the overall look for Pause and the frontend elements. These were then converted into html to develop the frontend website application using **Bootstrap**.

- Implementation process

Our achievements include:
1. building a timer and an API for activities during a break;
2. created a MySQL database to store users information;
3. and building a progress tracker using an external API (Pixela).

- Agile development

In our team we used an agile development approach to develop the application, which allowed for flexibility and adaptability to any changes throughout the process. As a group we made a SWOT analysis which helped us decide what we needed to focus on the most and how to delegate the tasks.

A Trello board was used for sprint planning. We had daily standups on Google Meet to discuss every team member's progress and any changes we felt relevant to the project development. We used a Slack channel for project communication. We also had a weekly one hour meeting where we worked together to review and integrate the code, using GitHub to share code updates and as version control. This helped to successfully maintain constant collaboration and communication.

- Implementation challenges

- ❖ Using external APIs: using an external API to generate random quotes for our activity section was not possible. This is because the structure and theme of the quotes did not consistently suit our target audience (children) for the timer application. We therefore decided to use YouTube links to videos for inspirational quotes, number facts and Brain Gym activities instead.
- ❖ The initial timer was created in Python. However, to implement the timer in the front-end part of the web application, the timer app had to be written in Javascript.
- ❖ Creating a progress tracker for our timer application: Various methods were explored so that progress could be shown in the application such as imagechart, matplotlib, quickchart, seaborn and Pixela. After several trial and error attempts, we were able to implement the progress tracker using Pixela.
- ❖ Building the frontend user interface was a challenge due to our inexperience and knowledge about frontend development.
- ❖ We lost a team member part way through our project so this meant we had to share the workload between fewer people. This caused an additional challenge to our time management and ability to implement all of the elements in our initial design plans, on top of the challenges of working remotely with full time jobs and other commitments.
- ❖ The study sounds option caused a development challenge The idea was to allow the user to choose from three background study sounds (Mp3 files) to play during their study session. The Mp3 player we originally used in Python (Pygame module) was not very reliable, cannot be run in a web application and it sourced the files from our computer desktop so this was abandoned. In the website application, the sounds need to play continuously whilst the user navigates to the timer page so the audio player opens in a separate window that has to stay open in the background for the sounds to play. A pop up audio player may be a further solution to develop.

TESTING AND EVALUATION

- Testing strategy

We decided to use unit testing on our functions with the Unittest module in Python.

We ran unit tests on the activity_manager.py functions using mock.patch() to test if a specified activity in the data is returned by the random activity function and whether the functions return wellbeing activities and Brain Gym activities as intended. All tests passed.

We also tested the user_mgmt.py functions for the following categories; invalid and valid login details, available and unavailable email addresses and finally getting a user by a specified id.

Using the unittest.mock module we had a get_mock_db_connection and a get_mock_ db_cursor named mock_db_connection and mock_db_cursor respectively. This helped to set up a ‘fake’ connection and generate a ‘fake’ cursor. The cursor executes and fetches data but reading from our actual database.

For testing for valid and invalid login details if someone types in the incorrect password, the return value of the get_mock_db_cursor should be None otherwise return the mock_user.

In testing the available and unavailable emails, we patched it to the mock_db_connection. It should work as the function in database.user.mgmt file works. The test email to be used for unavailable email has to be an actual email that is in the database because we are asserting that a user is already registered to that email and therefore it should be self.assertFalse. The opposite would be the case for testing for an available email.

All 5 tests passed.

- Functional and user testing

We constantly debugged our coding and visited the website application as a user to test all the functions and website performed as intended. We tested the sign in function by creating various usernames and passwords to access the website and to find any problems with the user input. We also tested the timer function by selecting different numbers of minutes and letting the timer run down and switch over. This is also how we tested the activities function for the user by setting the timers to check the suggestions appeared and were easy to read and the links to YouTube worked. We also user tested the study progress page by setting up new users and started the timer to check the Pixela graph showed the time correctly and displayed the relevant date. If errors were discovered, we revised our code and programming to handle the errors, e.g. using try and except statements in the Pixela programming to avoid the failure of a PUT request due to the API’s 25% failure rate when adding to a graph.

- System limitations

Data storage became an issue for the activity data, as an external API could not provide the activity selection as we had originally planned. Our next idea was to build our own API by creating our own data, storing it and accessing it as an external API. Twenty Brain Gym activities and wellbeing activities were researched and collated in a list of dictionaries in Python, which were then accessed and returned using Flask. This however, did not seem like an efficient way to store the data by creating an API from a list of dictionaries to then retrieve the data from those files via the API. Therefore, we decided to store the two Python files in the computer memory for easy access. This system has limitations as the data cannot easily be updated so the activity selections may become repetitive for the user, and it may cause memory issues if the data grows too large. Ideally, the data needs to be stored in a SQL database table and accessed by Python MySQL-connector.

If we had unlimited time for development, the following are some ideas for future features in Pause:

- Weekly email on study tips and information sent to all users using an automated email delivery provider and based on user email address data.
- Add in a study subject tracker into the progress tracker to see how much studying time is spent on each subject.
- Add a to-do list option for students to visually ‘tick off’ when they achieve a study session or subject homework.
- Add stickers, rewards and positive messages once study session time is fulfilled to promote a sense of achievement in students.
- Set up a network of students across Pause who can message each other and discuss school/ study topics and build a study community.

CONCLUSION

In conclusion, we set out with a highly ambitious project idea to create a user-friendly web application as a tool to help students study. With good planning, an organised approach and effective team-work, we have achieved our aim with Pause and have made an application that can provide users with timed study and break sessions, suggesting fun activities to do and tracking their progress in a visually-pleasing chart. We achieved all of our must have, should have and could have targets in our original plan, with many developments and adaptations along the way. We utilised our team skill set to program in Python and create a SQL database, but also learnt new skills whilst attempting to program the timer in Javascript, use html code for our frontend development, use the Pixela external API to display data, and build our own API. We hope Pause users will enjoy using the application as much as we enjoyed developing it.