AI-Parsons Problems

Final product report
COMP30022 IT

Team Members

Name
Riley Mitchell
Jiayang Tan
Thomas Lam
Chon Lok Chiang
Nanda Bawana

Contents

[Product overview](#)
[Requirements](#)
[Front-end Design](#)
[Architectural Design](#)
[Coding standards](#)
[Testing](#)
[Deployment](#)
[Handover Documents Checklist](#)

Product overview

AI-Parsons Problems is an AI parsons problem website where students can generate parson problems relating to data science. It leverages the Gemini large language model to assist in generating code-blocks that users can interact with to improve their coding skills. Designed as a learning tool, the platform has an easy to use user interface with useful terminal-like aesthetics inspired by modern IDEs, and allows for signed in users to generate, interact and automatically save generated problems. The website has a built-in IDE-like submission feature where students can check their code to receive interpreter or compiler-like feedback. Students can also choose from a variety of problems on various data science related topics and themes.

Our client is a university professor who wishes for a beginner friendly AI parsons problems learning tool to help undergraduate first and second year students exercise their coding and data science knowledge.

Requirements

For a more intuitive table-style format refer to the artefact called “Artefacts-Testing And Project Setup-011124-001816.pdf” Located in this project’s github repository inside the /docs directory.

The following are high priority functional and non functional requirements

High priority functional requirements

After multiple client interactions the team decided on the following high priority functional requirements.

1. Important pages in the website
    a. Home page
        i. Login and registration forms
        ii. Access to the categories page

    b. Categories/topic selection page
        i. Access to a variety of topics and themes that users can select
        ii. Upon selecting topic or theme users should redirect to the work-space where they can interact with the generated problem

    c. Workspace page
        i. Drag and drop UI

ii. AI generated problem description
iii. AI generated parsons problems code blocks
iv. Regenerate problems button
v. Submit problems button
vi. Highly responsive drag and drop with little to no animation delay

d. Saved problems page
    i. A table of saved problems that users have solved or are yet to solve
    ii. Table should redirect user to the workspace page where they can work on the problem once again

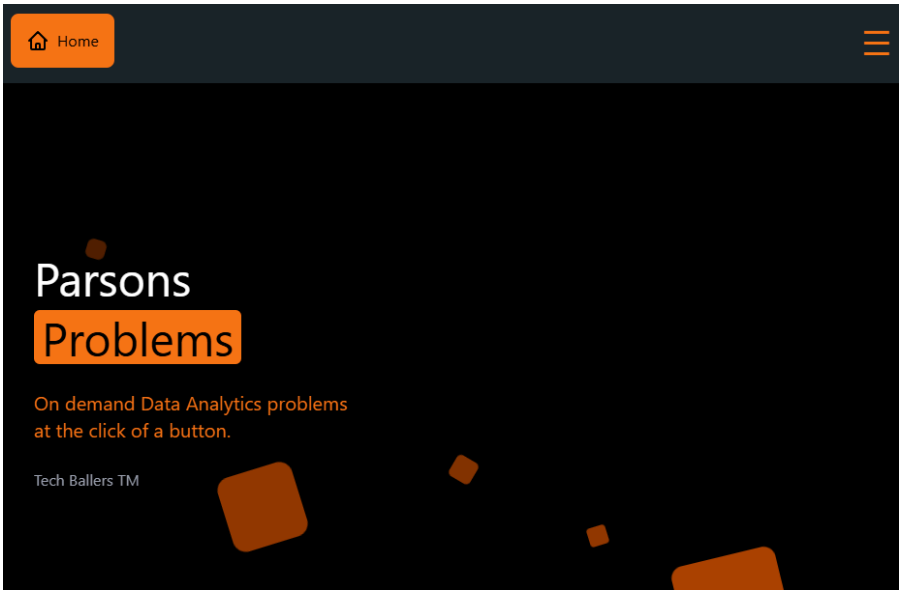
e. Profile page for user related statistics
    i. Users can edit their username
    ii. Displays relevant user statistic like total problems solved

High priority non-functional requirements

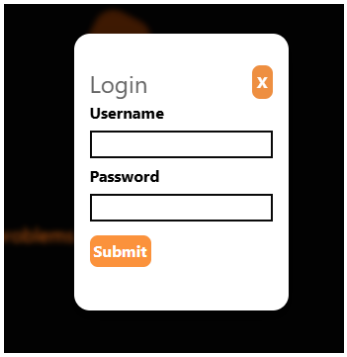
1. Performance
    a. Problems generated should return a result in under a minute.
    b. Problem submission and error checking should occur in under 10 seconds.
    c. Page responsiveness and rendering should take no longer than 2 seconds.
2. Security
    a. The product has authentication for all users
    b. Route protection is implemented on the frontend
    c. API endpoints are protected
3. Useability
    a. Design and colours are uniform, and consistent across all pages
    b. UI is simple and highlights the core features of the website
    c. Easy to follow page layout

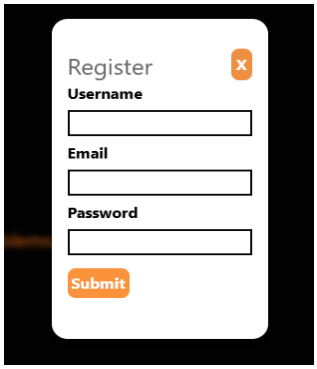
Front-end Design

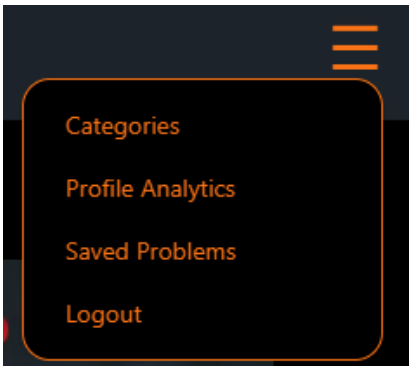
Home page

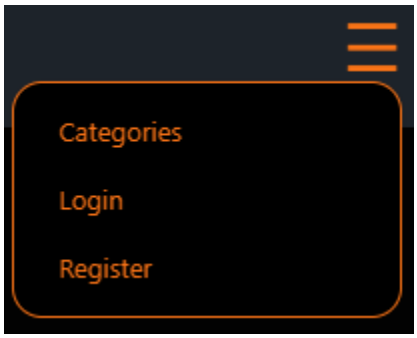


Login, registration and navigation
Users can create an account, register and navigate between pages using the menu.

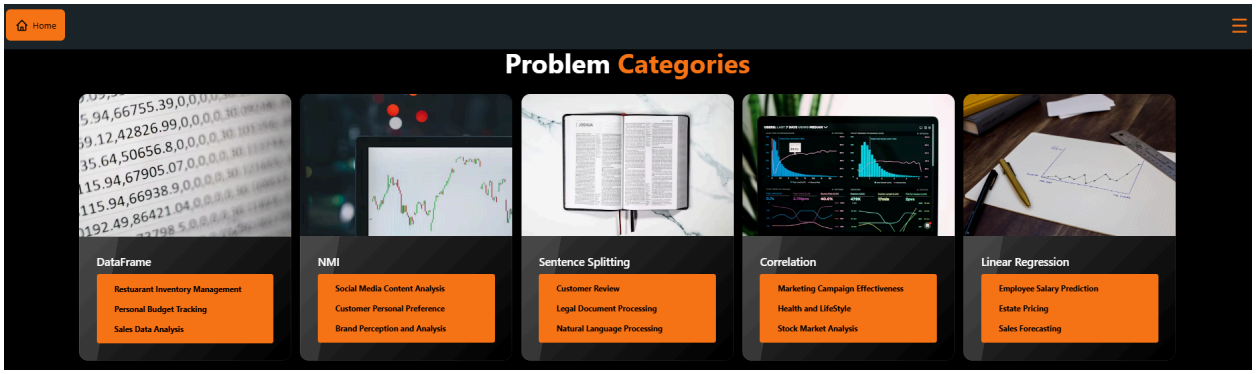








Categories page
A variety of predefined topics and themes for users to choose from. Clicking any topic will navigate to the workspace page and generate a new problem.

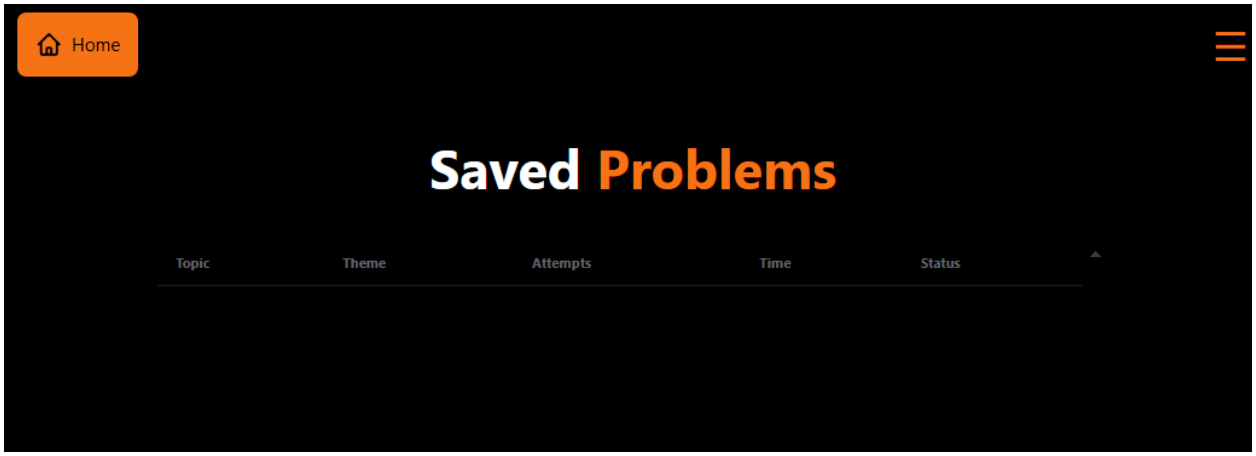


Workspace page

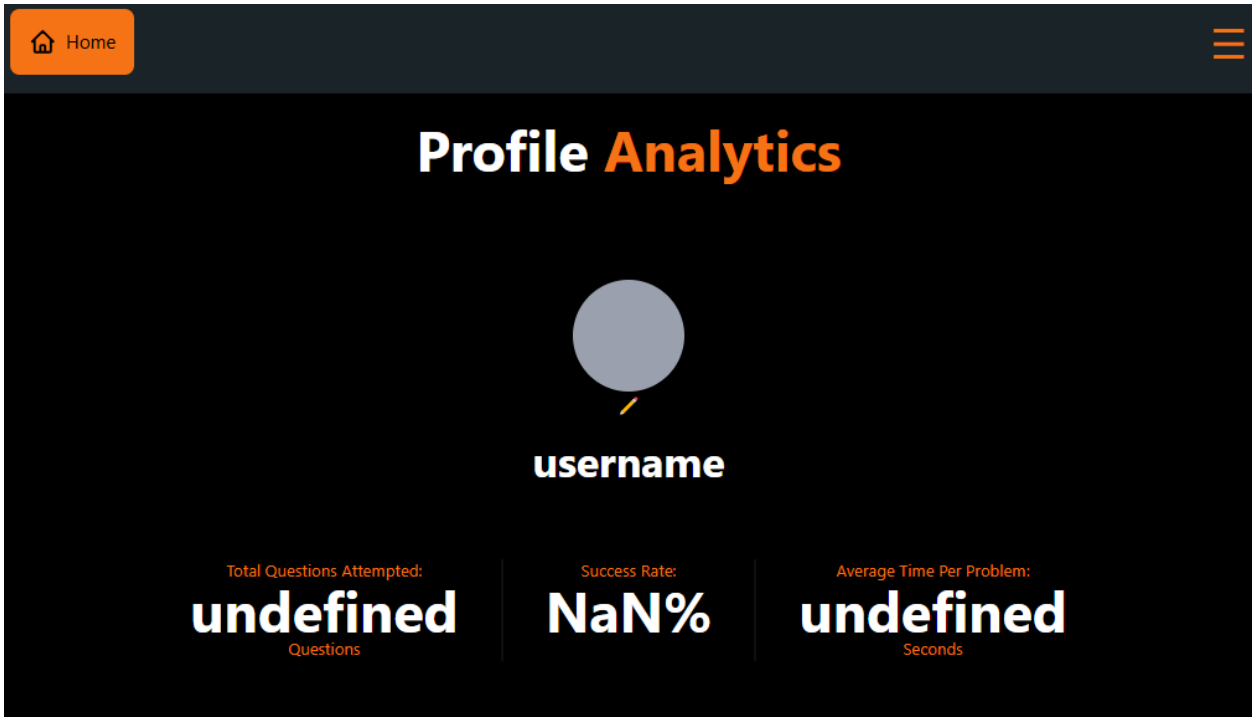
The main page for users to interact with. Users can re-generate a problem with the restart button and type in “drop.py” in the terminal to submit their code.

Saved problems page

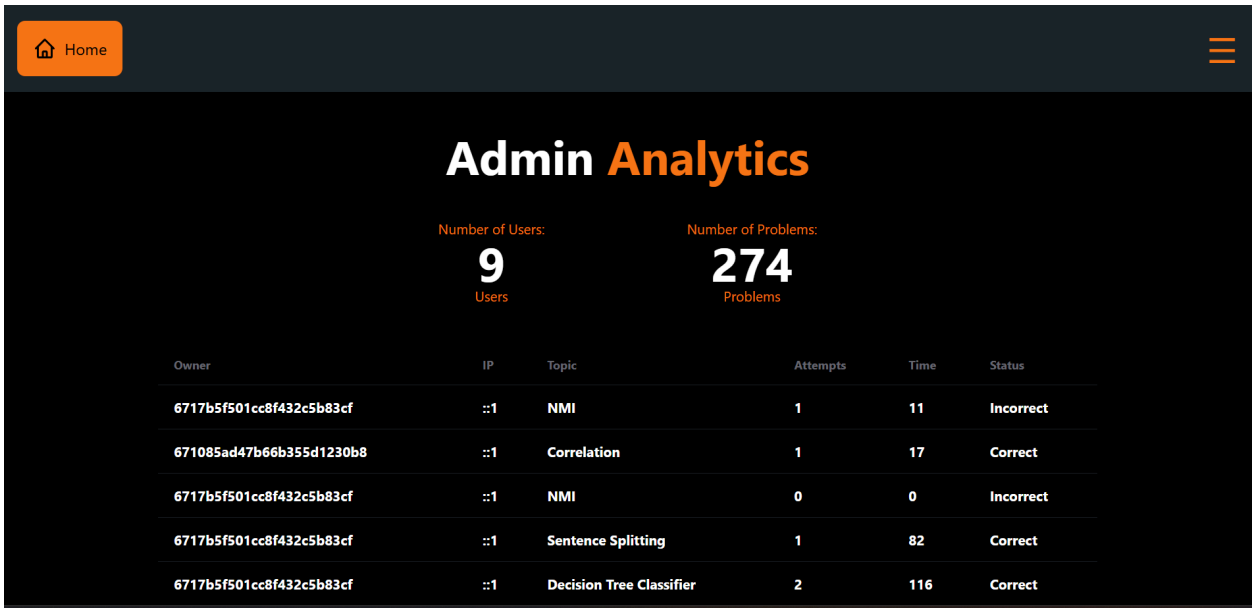
This page will display saved problems. Problems when generated will automatically save for signed in users.



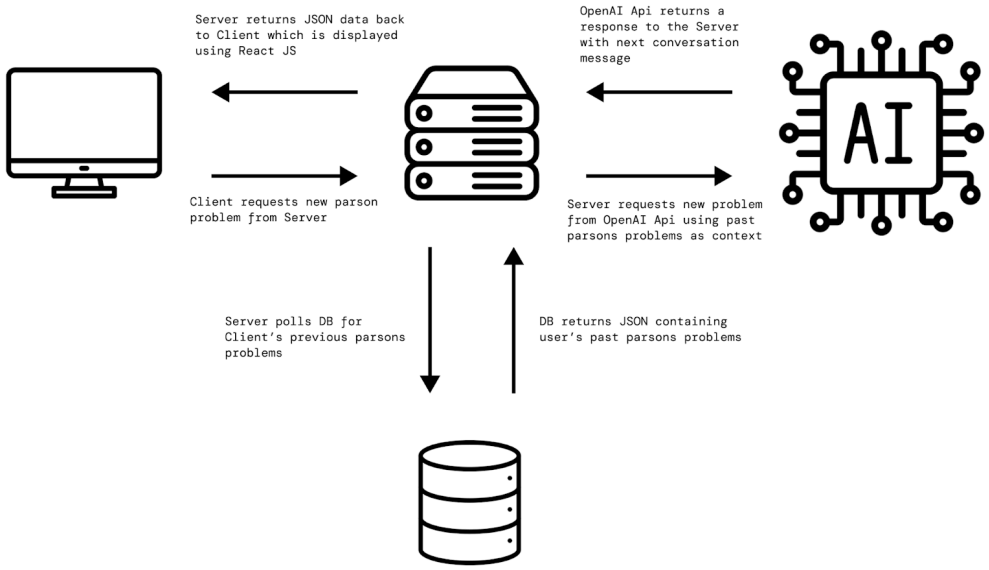
Profile Analytics
Displays relevant user statistics.



Admin Analytics
Displays relevant admin statistics.

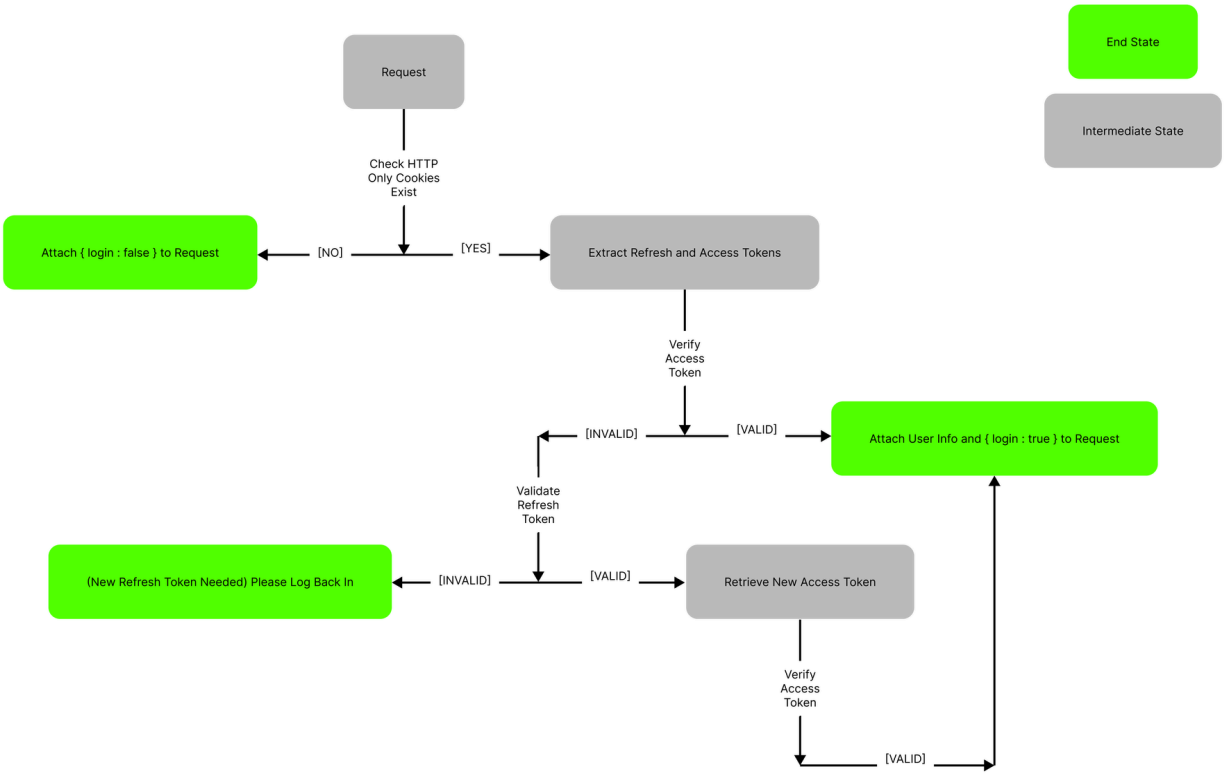

Home
Admin Analytics
Number of Users:
9
Users
Number of Problems:
274
Problems
Owner IP Topic Attempts Time Status
6717b5f501cc8f432c5b83cf ::1 NMI 1 11 Incorrect
671085ad47b66b355d1230b8 ::1 Correlation 1 17 Correct
6717b5f501cc8f432c5b83cf ::1 NMI 0 0 Incorrect
6717b5f501cc8f432c5b83cf ::1 Sentence Splitting 1 82 Correct
6717b5f501cc8f432c5b83cf ::1 Decision Tree Classifier 2 116 Correct

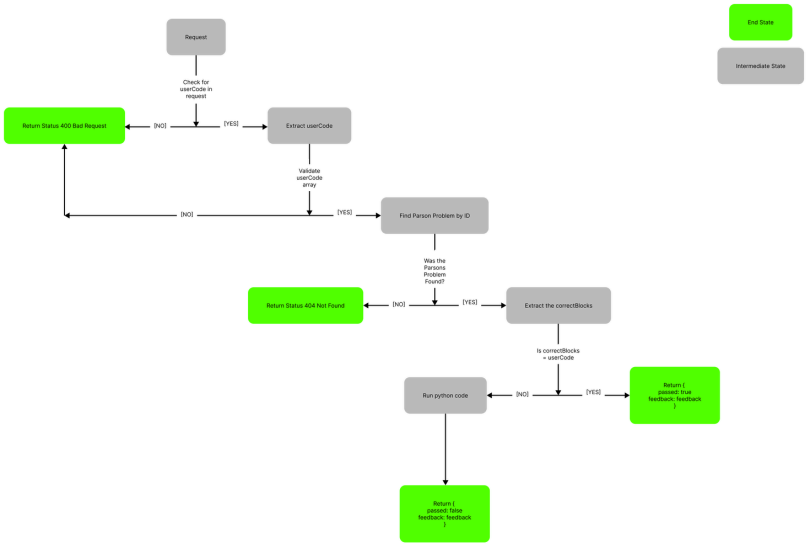
Architectural Design



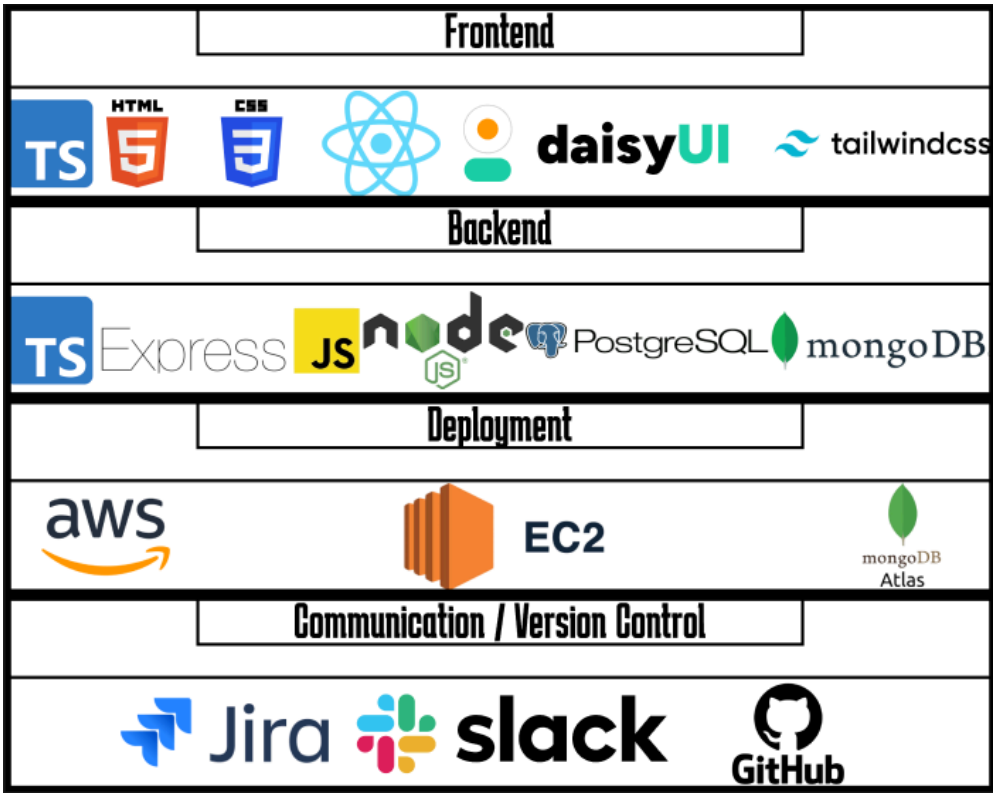
Authentication and submission handling

For clearer diagrams refer to “Artefacts-Model Artefacts-011124-001609.pdf” in the /docs directory in the project github repository.





The team decided to use the MERN stack for this project. This decision came about due to the easy setup, and the abundance of resources available to learn about the stack. It also supports front end tools like tailwind and daisyUI to help speed up the front end design process.



Coding standards

The coding standards for this project are straightforward. Use error handling statements throughout the code to ensure that when errors occur they can be identified in the console.

As for project structure, the team decided to split the frontend and backend code into two directories namely client and server. This is to ensure that the frontend and backend teams do not accidentally modify files that they are not supposed to during development.

Testing

The front end testing procedure involves viewport testing across major web-browsers to test for responsive design. We tested on chrome, and firefox to ensure that all features work as expected across three different screen/viewport sizes.

The backend utilises thunder client to test each API route. We simply enter valid and invalid inputs to ensure that the backend responds to these inputs correctly.
For more detailed information refer to the README.md file in the team's github repository.

Deployment

As per client approval, we were not required to deploy the project onto the cloud for remote access. After various discussions, the client insists that it is sufficient for the team to hand over the repository and the documentation such that they can set up the project on their local machine.

Handover Documents Checklist

For the project's handover, the following have been transferred to the client.

1. Ownership and access to the github repository.
2. Access to the team's database so the client does not have to set up MongoDB on their local machine.

   (Note, the .env file only has the API keys and TOKENs for a temporary database account that the team has setup specifically to allow easier time for those assessing the project. These keys will be invalid after the end of the semester for security reasons. So, if the client wants to use the project in a similar structure for purposes outside of marking, they must configure their own MongoDB account and generate the relevant keys and tokens. Information on how to do this is abundant on the official MongoDB website.)

3. All relevant confluence documentation for the project. This includes a project setup document. These can be found in the /docs folder in the git repository.
4. A document with all the AI prompts that the project uses to generate parsons problems. This is also found in the /docs folder in the git repository.