SW Engineering CSC 648-848 Fall 2024

<u>GatorTutor</u>

Team 6

Team Lead, Alex Hoff - (ahoff2@mail.sfsu.edu)
Backend Lead 1, Dylan Faulder - (dfaulder@mail.sfsu.edu)
Backend Lead 2, Austin Ng - (Ang@sfsu.edu)
FrontEnd Lead, Dalan Choy - (dchoy3@mail.sfsu.edu)
Github Master, Jack Richards - (jrichards7@sfsu.edu)

Milestone 1

History

Date Submitted:	October 20th, 2024
Date Revised:	N/A

1. Executive Summary:

GatorTutor is a tutoring service created by San Francisco State University students for SFSU students. We aim to provide a place that facilitates the matching between all SFSU students with their best fit SFSU tutor. We feel that the dividing factor between a successful student and a struggling student is gaps in their knowledge and our goal is to bridge that gap. Our service helps match tutors with experience in those subjects and the students seeking assistance. Since our application is institution specific, our tutors are able to provide a level of service unique to SFSU that foreign services can not compete with.

Since our team is composed of SFSU students, we are able to target institution specific problem areas that's stopping students from achieving their potential through tutoring. Our application allows students to search for tutors by university specific classes. Professors don't always teach the same curriculum as others, so tutors are able to specify which ones they have experience with, allowing students to select them based on their issues. Tutors are able to create profiles unique between individuals through multimedia to customize their impressions, introduce themselves through text descriptions, show interested students their rates along with their availability, and appear on searches by course identifiers. Our platform also facilitates communication between interested students and tutors with our built-in messaging service.

2. List of main data items and entities:

Guest User - Mandatory (1)

　　Entity Name: Guest User

　　Description: A general term for anyone who interacts with the system.

　　　　Usage: All users must register with SFSU credentials to use the platform. They can create accounts, search for tutors, and communicate with others. Some users may create tutor posts.

Registered User - Mandatory (1)

　　Entity Name: Registered User

　　Description: A user who has successfully registered on the platform.

　　Usage: Registered users can create posts, search for tutoring services, send messages, and interact with other users. They can also associate themselves with tutor posts.

Tutor Posts - Mandatory

　　Entity Name: Tutor Posts

　　Description: Posts created by registered users who offer tutoring services.

　　　　Usage: These posts contain details such as subjects offered, availability, and hourly rate. Registered users can create these posts and students can search for and contact tutors through them. Tutor posts are visible to other users, and interactions are initiated through them.

Admin - Mandatory (1)

    Entity Name: Admin

        Description: An individual designated to manage the platform and handle database operations.

        Usage: Admins do not interact with the platform as regular users do. They only use tools such as an MQL workbench to manage the system's data. They handle tasks such as profile approvals, database queries, and maintenance. Admins do not have standard user profiles.

Message - Mandatory (1)

    Entity Name: Message

    Description: A communication between registered users, typically between students and tutors.

    Usage: Messages are exchanged through the platform for inquiries about tutoring services and general communication.

Subject - Optional (2)

    Entity Name: Subject

    Description: A specific academic subject at SFSU.

    Usage: Subjects are associated with tutor posts. Students search for tutors based on these subjects.

Reviews - Optional (3)

    Entity Name: Reviews

    Description: Reviews for Tutors

    Usage: Stores reviews for tutors including details about the reviewer and the content of the review. It helps provide feedback and ratings for tutors.

3. Functional Requirements - Prioritized:

Priority 1:

    1.1 Registered Users - Creating Posts: Users shall be able to create a tutor profile and advertise their skills on the website.

    1.2 Registered Users - Dashboard: Users shall have a functioning dashboard that displays their postings and messages received.

    1.3 Unregistered/Registered Users - Browsing: Users shall be able to browse and easily navigate the website and most of what it has to offer.

Priority 2:

2.1 Admin - Deleting Inappropriate Content: Site Admin shall be able to delete inappropriate items or users based on the criteria of the post.

2.2 Unregistered/Registered Users - Searching: Users shall be able to use the search bar in order to search for SFSU-specific classes and professors to find hyper-relevant material.

2.3 Admin - Approval of Tutor Posts: Site Admin shall be able to approve or deny any tutor post through the MySQL Workbench.

2.4 Registered User - Login: User should be able to login into their account to see their dashboard.

2.5 Registered User - Messaging: One way messaging system from student to tutor.

Priority 3:

3.1 Registered User - Interactive Chat: A two way messaging between tutor and student for real-time contact.

3.2 Student Reviews for Tutors: Students can leave detailed feedback and ratings for tutors, providing additional transparency and accountability.

Priority 4:

4.1 Tutor Availability Calendar: A feature that allows tutors to share a live calendar of their availability, enabling students to schedule sessions directly.

4.2 Meeting Scheduler: An integrated scheduling system that allows students to book tutoring sessions directly within the platform, syncing with the tutor's availability and sending reminders.

4.3 Personalized Tutor Suggestions (For You Page): A feature that provides students with personalized tutor recommendations based on their search history, previous interactions, and subjects of interest, similar to a "For You" page.

4.4 Smarter Search with Filters: An enhanced search system that allows users to apply more specific filters (e.g., by tutor ratings, price range, availability, or specific experience), making it easier to find the perfect match.

4.5 Group Tutoring Sessions: A feature allowing tutors to offer group sessions where multiple students can attend a single tutoring session.

Clarification for Priority 4:
We recognize that these features may not be included due to time constraints or specific guidance, but we've added them simply for feedback sake to better understand what is definitely not allowed and what might be feasible if more time or resources were available. Some of these are

presumably technically allowed but considered extremely low priority by our team due to the project's timeline.

4. UI Storyboards For Each Main Use Case:

The search page includes all of the tutors' contact-cards on the page along with a filter box to the top left of all of the cards. In this box you can filter things like "Subjects Taught", and "Price per Hour". Once you find a tutor you want you can click on their contact card and be brought to the **Tutors page**.
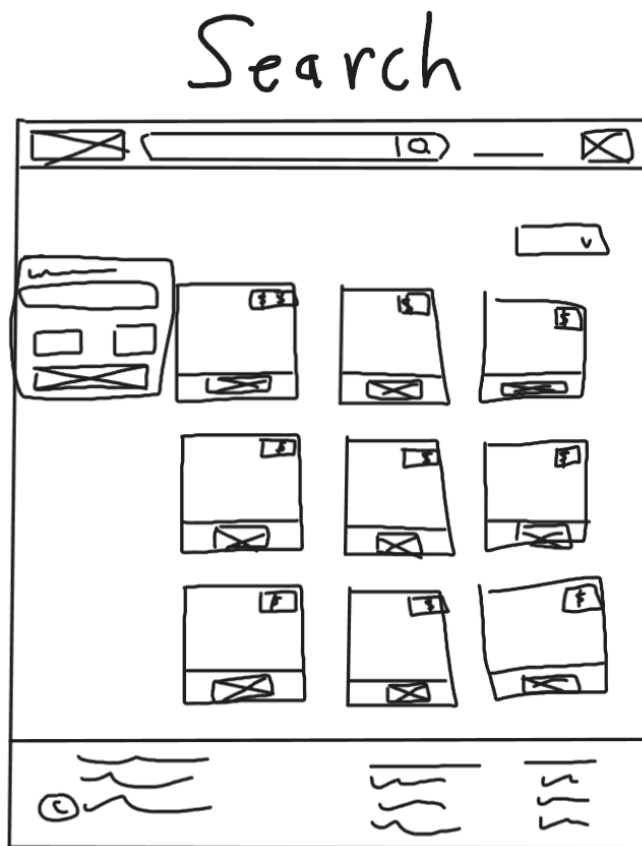
The tutor page includes information about the specific tutor that you would like to contact such as "Price Per Hour", "Subjects Taught", along with more personal information such as a description of themselves, a profile picture, and a video used as an example of the material they might teach. Before you can contact a tutor you must **register** for an account.

Tutor Page

The register page includes a table with boxes where you can create your Username and Password, once you create your password you will need to confirm it by typing it in again. Once you have filled in these three fields you then can agree to the terms of service and create your account. If you already have an account you can use the "Already have an account?" link in order to navigate to the **login page**.

The login page is much like the registration page except for the fact that you don't have to type your password in twice like when you are creating an account. You just type in your username, password, and click "Login". There is also a link labeled "Don't have an account yet?" that will redirect you to the registration page. Once you login you will either be redirected to the page you were previously trying to reach or, by default you will be redirected to the **landing page**.

Login
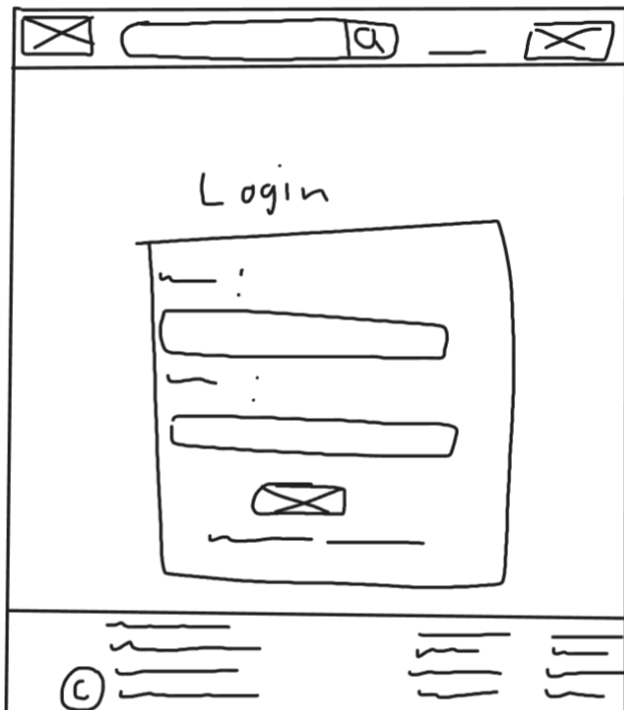
The Landing Page is the first thing every site-visitor will see upon directing to GatorTutor. It will welcome the user at the top and then list relevant information that might lure the user into using the website such as "Best Rated Tutors", "Most popular Subjects", "About the Team", etc. This will ensure our website will not just get visitors, but turn visitors into customers.
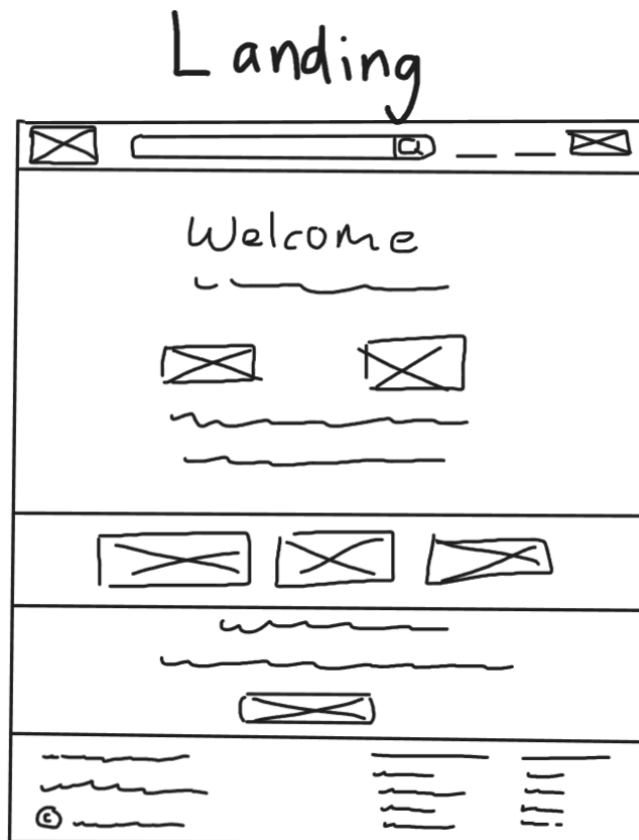
The Dashboard will display the students profile information as well as any messages they have been sent if they have created a tutor post.

Dashboard
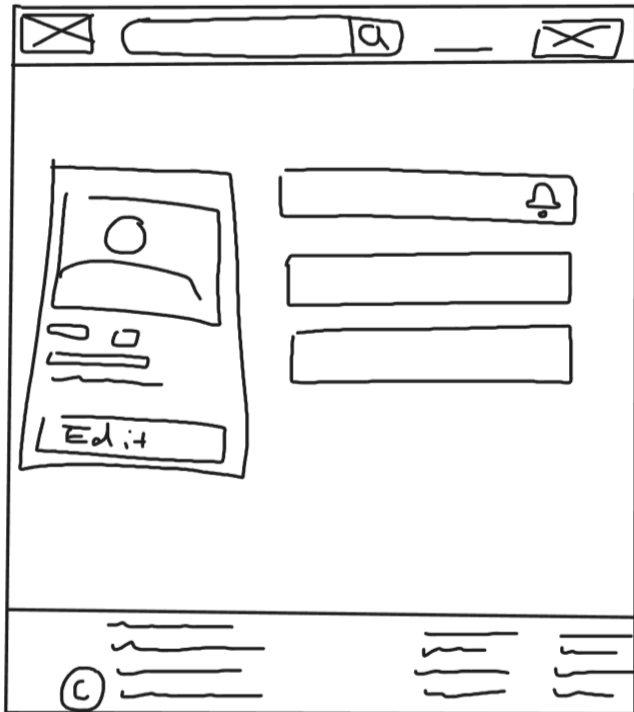
The About Us Page displays our creators of the website proudly, along with a description of each student and their role in the project.
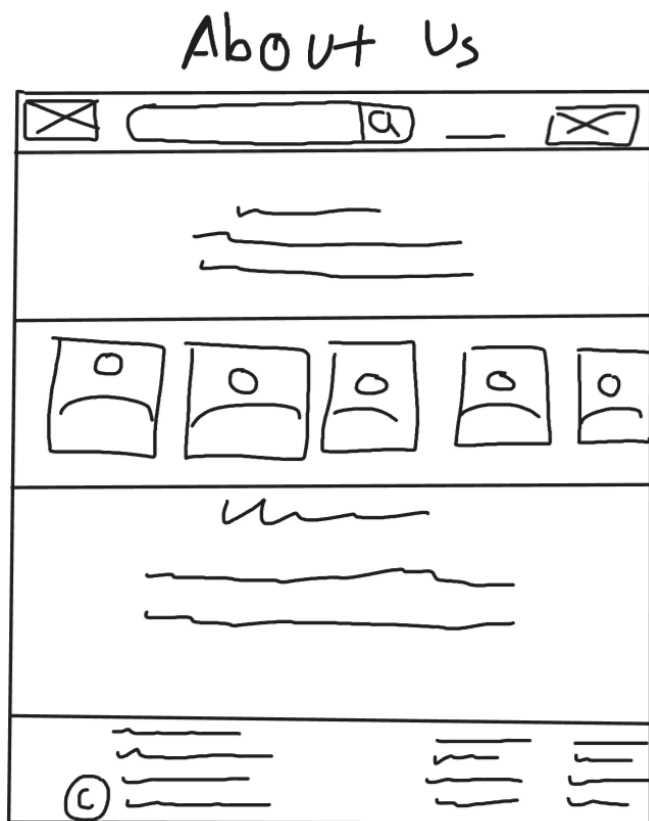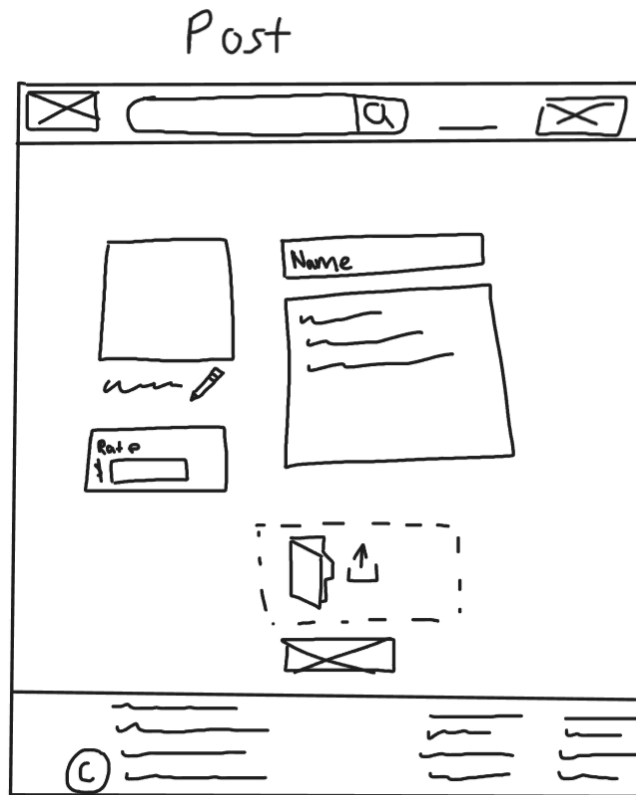
The Post Page will include features that need to be filled in by the user in order to create a tutor post such as "Profile Picture", "Name", "Description", "Price per Hour", and "Video of Example Material".

UI Storyboards Descriptions

Search: The search page allows users to explore available tutors based on specific filters such as subject, rate, and availability. Users can browse profiles in a grid format with an intuitive layout that supports quick navigation. Each tutor profile includes basic details with the option to expand for more information.

Register: The registration page provides a straightforward form for new users to sign up for the platform. Users are required to fill in necessary information such as name, email, and password. Once completed, they can create an account and gain access to platform features like posting or messaging. (Lazy registration still applies however).

Tutor Page: Each tutor page is a detailed profile showcasing the tutor's skills, subject expertise, and availability. It also includes contact details, hourly rates, and an optional bio or introduction video. Students can interact with tutors by sending messages directly from this page to inquire about services.

Dashboard: The dashboard provides a central hub for users to manage their profile, view posts, and track incoming messages. Users can also make updates to their tutoring posts or respond to inquiries from students. Notifications and alerts keep the user informed of any new activity on their account.

Landing: The landing page serves as the homepage, welcoming visitors with a clean design and quick access to key sections like browsing tutors or signing up. It provides a brief overview of the platform's mission and features to engage new users. Visitors can navigate to other areas such as About Us or Register from this page.

Login: The login page allows registered users to access their accounts by entering their email and password. If the credentials are correct, users are directed to their personal dashboard. Forgotten password recovery options are also provided for users facing login issues.

About Us: This page offers information about the GatorTutor team, highlighting their mission and values. The page showcases team members with their photos and roles, creating a personal connection with users. It serves as a way to build trust and transparency with students and tutors alike.
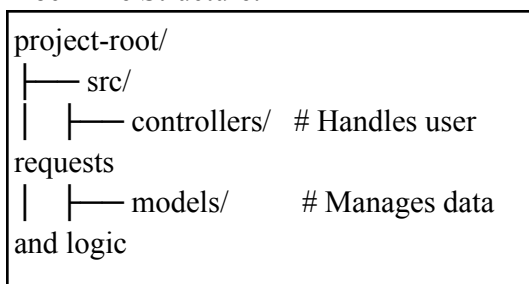
Post: Tutors can create and manage their tutoring posts from this page, uploading relevant details such as subjects offered, rates, and availability. Multimedia options such as profile photos, videos, and attachments can also be included to enhance the post. The simple layout makes it easy for tutors to update their listings and keep them current.

5. High level Architecture, Database Organization Summary:

MVC design pattern:
The core logic resides in the src directory, which includes all the MVC components and routes. The **Models** handle data and business logic, **Controllers** manage user requests. **Views** define the user interface. **Routes** link user actions to the appropriate controllers. Additionally, we have a **public** folder for static assets, including a uploads directory for media files. This setup allows us to serve media efficiently from the file system.

Mock File Structure:

```
project-root/
├── src/
│   ├── controllers/   # Handles user
requests
│   ├── models/        # Manages data
and logic
```

```
|     ├──── views/          # Defines user
interface
|         └──── routes/     # Maps URLs to
controllers
    ├──── public/
    |     ├──── css/        # Stylesheets
    |     ├──── images/     # Static images
    |     └──── uploads/    # Media files
    └──── app.js            # Main
application setup
```

DB organization:

Users	Messages	Tutor Posts
Stores user information like username, email, and role. It's essential for managing user accounts and authentication. id: Unique identifier for each user username: User's chosen name email: User's email address password: Encrypted password role: User's role (e.g., student, tutor) is_tutor: Boolean indicating if the user is a tutor created_at: Timestamp of account creation last_login: Timestamp of last login updated_at: Timestamp of last update	Contains messages between users, including sender and recipient details. It enables communication on the platform. Messages will only be one directional as specified, however this approach allows for easy expansion of the messages feature in the future. id: Unique identifier for each message sender_id: ID of the user sending the message recipient_id: ID of the user receiving the message message: Content of the message created_at: Timestamp of when the message was sent read_at: Timestamp of when the message was read	Holds tutor profile details such as bio, availability, and rates. It helps tutors showcase their services. id: Unique identifier for each post user_id: ID of the tutor creating the post bio: Short biography of the tutor availability: Tutor's available times hourly_rate: Rate charged per hour contact_info: Tutor's contact information created_at: Timestamp of post creation updated_at: Timestamp of last update profile_photo: File for the tutor's profile photo profile_video: File for the tutor's profile video ~~experience: Tutor's experience details~~ additional_images: Files for multiple images uploaded to the tutor's profile resume_pdf: File for the uploaded PDF
Subjects Lists all available subjects. It organizes the subjects that tutors can teach and students can learn. id: Unique identifier for each subject subject_name: Name of the subject	**Classes** Represents different classes linked to subjects. It helps categorize educational content. id: Unique identifier for each class class_name: Name of the class	**Tutor Subjects** Links tutors to the subjects they can teach. It ensures students find the right tutor for their needs. tutor_id: ID of the tutor subject_id: ID of the subject
Class Subjects Connects classes to their relevant subjects. It maintains an organized educational framework. class_id: ID of the class subject_id: ID of the subject	**Reviews** Stores reviews for tutors, including details about the reviewer and the content of the review. It helps provide feedback and ratings for tutors. id: Unique identifier for each review tutor_id: ID of the tutor being reviewed user_id: ID of the user writing the review rating: Numeric rating given by the user comment: Content of the review created_at: Timestamp of when the review was created	An **Appointments** table and its corresponding linker table could be useful if we decide to implement scheduling features, but it will not be a priority right now. It would help manage session bookings between tutors and students, storing details like date, time, and participants. If user demand for organized scheduling grows, this feature could become more relevant. (we know it is not in the scope of this project)

Search/filter architecture and implementation:
We will implement a search feature that allows users to find tutors based on specific criteria such as name, subject expertise, and hourly rate. This will be achieved by using a method that involves querying our database with filters applied to these criteria. The search will utilize SQL's **%LIKE%** operator to match partial text entries, ensuring that users can find relevant tutors even if they only remember part of a name or subject. Additionally, we will allow sorting of results by different parameters like price or recency, providing a flexible and user-friendly experience. This approach will ensure that users can easily navigate and find the right tutor for their needs.

The **%LIKE%** operator will be applied to text-based fields such as <u>username</u>, <u>subject_name</u>, <u>class_name</u>, <u>short_bio</u>, <u>experience</u>, <u>availability</u> allowing users to search for partial matches in names, descriptions, ect.

Non-trivial algorithms:
Dynamic Query Building for Tutor Search: We will have a system that dynamically constructs SQL queries based on user-provided filters and sorting options. This allows users to search for tutors using various criteria such as subjects, price range, and availability, providing a tailored search experience.

Tutor Profile Management: We will manage tutor profiles by handling complex data inputs, including availability, subjects, and multimedia uploads. This ensures that tutors can maintain comprehensive and up-to-date profiles, enhancing their visibility to potential students.

Subject and Tutor Association: We will implement a mechanism to associate tutors with specific subjects, allowing for efficient retrieval and display of tutors based on their expertise. This involves managing relationships between tutors and subjects in the database.

Recommendation System: We <u>might</u> develop a recommendation engine to suggest tutors to students based on their past interactions, preferences, and similar user behaviors. This would enhance user engagement by providing personalized tutor suggestions.

Dynamic Pricing Model: We <u>could</u> implement a dynamic pricing algorithm that adjusts tutor rates based on demand, time of day, or tutor experience. This would optimize both tutor earnings and student affordability, ensuring a balanced marketplace.

Changed or added SW tools and frameworks

- **Multer:** Our choice for handling file uploads, essential for multipart/form-data.
- **Express-Validator:** Keeps our input clean and secure, preventing SQL injection and XSS.
- **Express-Session:** Manages user sessions seamlessly, crucial for authentication.
- **Bcrypt:** Ensures password security, trusted by LinkedIn and Dropbox.
- **EJS:** Our templating engine for dynamic HTML, simple and effective.
- **TailwindCSS**: We love its utility-first approach for quick, responsive UI building.
- **Autoprefixer:** Saves us time by adding vendor prefixes for cross-browser CSS.
- **PostCSS:** Automates CSS tasks like linting and minifying, streamlining our workflow.

Development Tools

Nodemon + Browser-Sync: Nodemon automatically restarts our server on file changes, while Browser-Sync refreshes the browser, making development faster and more efficient.

Docker

Docker allows us to containerize our application, making it easy to run consistently on AWS servers. It simplifies sharing our environment across the team, ensuring everyone works with the same setup, reducing "it works on my machine" issues.

6. Key Risks:

Skills Risks

Q: What if a team member lacks the knowledge or skills to accomplish their task?

A: Our team agreed on a "no shame" environment where everyone is encouraged to ask each other questions, no matter how obvious they may be. There are no bad questions.

Q: What if a team member is not happy with the task they are assigned with?

A: The team allows for flexibility. Team members are allowed to switch tasks with another person if they are uncomfortable. Everyone should be competent in the task they were assigned.

Schedule Risks

Q: What if a team member is unable to finish their task on time?

A: Our team will help accommodate another member by having others help work on that specific task. This will be discussed through our Discord server or even through direct messaging in more urgent situations.

Q: What if an idea is too ambitious to finish?

A: The team has to be reminded that the goal is to push out a project that meets the minimum requirements. If the idea does not directly contribute to completing the main goal, it shouldn't be included.

Technical Risks

Q: Are there any unknown technical challenges that could affect progress?
A: Unknown technical challenges like integration issues, unexpected bugs, or tool limitations (e.g., with Trello or Discord) could impact progress.

Q: What if a team member makes changes or additions to the code that others are confused about?

A: Code should be well-documented using comments and a member should explain their code if others need explanation.

Teamwork Risks

Q: What if a specific team member is not communicating or being active on the project?

A: Our team will first reach out to the team member directly. Continued silence will lead the team to address the professor directly about the situation.

Q: What if the entire team lacks communication with each other?

A: Weekly online and in-person meetings are required every week to ensure everyone stays updated on the current progress of the project.

Legal/Content Risks

Q: What if copyrighted images are used?

A: All material needs to be reviewed by the team to ensure that every image is original or royalty-free.

7. Project Management:

This workflow generally represents the team's process, from milestone discussions to task assignments, progress check-ins, and tracking using Trello.

[Discussion] --> [Assign Tasks] --> [Task Details (Discord)] --> [Check-ins] --> [Trello Updates]

For each milestone, our team will discuss the various tasks each milestone contains to ensure that everyone understands what needs to be done and how we can accomplish it.

After discussion, our team lead will assign us the task that he believes will best fit our roles and skills. The tasks will be divided up in such a way where everyone will have a similar workload. Each team member is allowed to give input on which task they would prefer before and after the roles have been assigned.

Once everyone has agreed on the task they have been assigned, our team lead will then create documents for each member highlighting each specific task in more detail. These documents are then sent through discord with a deadline to get them in by.

Each week, our team will decide on a day to have a <u>weekly discord meeting</u>. This meeting acts as a mid-point check in for all of us to discuss our current progress and what else needs to be done to complete the assignment.

However, constant communication about the milestone progress is also maintained in between meetings through the <u>discord text channel</u>. The team lead will occasionally bring up progress checks and any other updates relating to the completion of the current milestone. More meetings are added if required for the milestone.

This system has allowed our front-end and back-end teams to maintain a solid form of communication. In order to further improve the efficiency of managing the project, we are incorporating the usage of Trello for this and future milestones.

With the use of <u>Trello</u>, we will have an easier time keeping track of task deadlines and updates. Instead of asking or looking back at the discord server for status updates, we are able to view these updates directly on the platform.

8. Use of genAI tools like ChatGPT and copilot:

Our team used ChatGPT 4o, the latest free model from openAI as our main Generative AI tool.

Our team used GenAI in order to:

- Figure out how to format our main data items and prioritize them based on the System the class CEO gave to us - LOW, you will find a trend in this section regarding AI and its abilities when it comes to workloads that are highly-tailored to a specific party. Since GenAI takes advantage of so many works across the internet, the answers are very general and well-balanced which can be a good thing but is not necessarily helpful when we are working in an environment that only WE know. For example, WE know that a search feature, although useful, is not the #1 priority when it comes to developing a tutoring website but chatGPT may see it as a high priority because it is trained on data all around the internet that might see a search feature as a #1 priority. For this reason, genAI is not great with prioritizing features.
- Divide up the workload of milestone 2 evenly between group members as well as assign relevant topics to the same group members who might have worked with them before - HIGH, as an experiment, I fed chatGPT the entire Milestone 2 instruction document that was provided to us by the class CEO. I prompted it to read all requirements and then divide the workload into even sections. It did this surprisingly well and even remembered which sections I gave my group members in Milestone 1 and matched some of the same

topics up to the same students. This really impressed me and went beyond what I expected.

- Give us an idea of how to prioritize our high-functional requirements - LOW, once again, having genAI prioritize information that only WE really know how to prioritize is futile and can give you some idea of how to structure the section, but ultimately will not be accurate to what is expected when we turn in our document.

- Help us understand what "key risks" in a working environment might look like as well as what they would look like unique to our project - MEDIUM, I labeled the efficiency and accuracy of this task as medium because although it gave some good examples of what some key risks might look like in a working environment, it wasn't able to go the extra step and customize the potential risks to our situation of being students, emulating a working environment, working on a group project together,.

- Structure our project management section - HIGH, Our group as a whole didn't really have a solidified idea of what we wanted for our "Project Management" section. Using GenAI and re-reading the original document helped us understand what we were expected to do. This was one of the more simple requirements but it was very important that we include good, relevant information of how we are maintaining our group's organization and lines of communication, which will in turn result in a great end-product for our team.

9. Team Lead Checklist:

- So far all team members are fully engaged and attending team sessions when required **DONE/OK**
- Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing **DONE/OK**
- Team reviewed suggested resources before drafting Milestone 2 **DONE/OK**
- Team lead checked Milestone 2 document for quality, completeness, formatting and compliance with instructions before the submission **DONE/OK**
- Team lead ensured that all team members read the final Milestone 2 document and agree/understand it before submission **DONE/OK**
- Team shared and discussed experience with genAI tools among themselves **DONE/OK**