Sprint Two Iteration Plan

June 24th 2024

1. Process

Start date: Mon June 24, 2024
End date: Fri June 5, 2024

1.1 Roles & responsibilities

To enhance our team's collaboration, we have assigned each member specific tasks and responsibilities. For the more challenging tasks, such as chat and video call functionalities, we will practice peer programming by assigning two team members to each. This approach aims to foster greater collaboration, creativity, learning, and faster completion of these tasks..

1. Michael Walker: Scrum Master
   - Create final set of user stories
   - Break down user stories to be more granular for each sprint
   - Modify database structure if needed and help teammates communicate with the database when needed
   - Facilitate team meetings
   - Maintain JIRA board and assign tasks on JIRA
   - Feature(s): Video calling (including sharing screen, sending video call link to other user and turning mic on/off) (frontend + connection to database)
2. Aarushi Doshi: Developer + tester
   - Assist in breakdown of user stories to be more granular for each sprint
   - Check/Approve all Pull Requests before merging to sprint2 branch
   - Test application after each merge to ensure code is working
   - Inform teammates of new merges
   - Deploy to main branch
   - Feature(s)/Bug Fixes(s): Update UI for registration pages, bugfix for login redirection, home page tutor search and filter based on rating and hourly rate (frontend + connection to database)
3. Arina Azmi: Developer + Designer
   - Take meeting minutes and keep team updated on new developments
   - Check/Approve all Pull Requests before merging to sprint2 branch
   - Confirm final UI before merging branches with sprint2
   - Assist in breakdown of user stories to be more granular for each sprint
   - Feature(s): Home Page with list of all tutors, filtered by language preferences of user (frontend + connection to database)
4. Miri Huang: Developer
   - Assist in breakdown of user stories to be more granular for each sprint
   - Feature(s): Develop a stream chat where two users can chat, upload images and files, with all chat history and all chats being viewable (connection to database)
5. Anusha Karkhanis: Developer
   - Assist in breakdown of user stories to be more granular for each sprint
   - Feature(s): Develop a stream chat where two users can chat, upload images and files, with all chat history and all chats being viewable (frontend + connection to database)

1.2 Events

1. Daily Standups: These meetings will take place online (Discord or Slack) and occasionally in-person, at 8pm EST every other weekday and after CSCC01 lecture (Tuesday 7 pm) and tutorial (Thursday 3pm). During each standup meeting, each team member will have the opportunity to discuss what they have completed since the last meeting, what they are planning to complete by the next meeting, and if they are having any issues and require assistance. This will also be the time to make any larger team decisions, such as changing UI, using a different API, etc.
2. Weekly Scrum Meeting: Next Saturday at 5pm, the entire team will meet to discuss the final features that have been developed and the complete weekly progress. During this meeting we will go over the code that has been merged so far and ensure that the application is working from end to end and discuss any larger blockers that any teammates are having
3. Final Sprint 2 Meeting: On Thurs June 4th, the entire team will meet online at 1pm to go over the final steps and ensure that all code is working. During this meeting all code on the sprint2 branch will be thoroughly tested.

2. Product: Goals and Tasks

2.1 Home Page

One of our main priorities for the Home page is to display tutors who share the same language preferences as the user. Each tutor will be presented on a separate card featuring their image, name, rating, hourly rate, and the courses they teach. When a user clicks on a particular tutor's card, they will be navigated to the tutor's profile page, which will be initially empty except for a message button.

- Home Page Display
    - Users will be presented with a list of tutors that share the same language preferences as the user.
    - Users should be able to view the home page as soon as they log in to the application with their email and password.
    - Each tutor will be displayed on a separate card on the home page.
    - Each row should have minimum 5 cards and the entire page will be scrollable
    - The cards will be placed right below the navigation bar and will span the entire width of the screen
- Tutor Cards Details
    - Each card will display the tutor's image, name, rating, hourly rate, and the courses they teach.
    - The design should ensure that all these details are clearly visible and well-organized on each card. The tutor’s profile picture should cover the top half of the card and the tutor’s details should be displayed below it and aligned to the left of the card.
- Navigation to Tutor Profile
    - When a user clicks on a tutor's card, they should be navigated to the specific tutor's profile page.

        - The profile page will initially be empty except for a message button, allowing users to send a message to the tutor.
- Technical Specifications
    - The tutor cards should dynamically update based on the user's language preferences.
    - All tutor card should be loaded at once (no pagination)
    - The tutor profile page should be designed to accommodate future additions beyond just the message button.

2.2 Search For Tutors on Home Page

The search feature on the application is really important as it allows users to easily find tutors by course codes, streamlining the process of finding specific educational assistance tailored to their needs

- Search Functionality on Home Page
    - A search bar will be placed at the top of the homepage, centered right below the navbar.
- Search Bar Details
    - The search bar should allow users to type course codes to search for tutors.
    - The search functionality should be case-insensitive, meaning the casing of letters doesn't matter.
- Dynamic Tutor Display
    - Initially, the homepage will display all tutors available on the platform with the same language preferences as the user
    - As users type course codes into the search bar, the displayed tutors will dynamically update to display the tutors who teach the course that is being searched for
    - The tutor cards displayed will update in real-time based on the course codes entered in the search bar.

2.3 Filtering of Tutors on Home Page

Our filter feature enhances user convenience by enabling precise refinement of tutor search results based on criteria such as rating and price, ensuring users find the best match for their learning requirements.

- Filter Functionality on Home Page
    - A filters button will be placed to the right of the search bar.
- Filters Button and Sidebar Details
    - When the filters button is clicked, a sidebar should transition smoothly from the right.
    - The sidebar will have two filter options: filter by rating and filter by price.
- Filter by Rating
    - The rating filter will be a continuous Mui component with marks, ranging from 0 to 5 stars.
    - Users can adjust the rating filter to display tutors based on their rating.
- Filter by Price
    - The price filter will be a continuous Mui component with marks, ranging from $0 to $40.
    - Users can adjust the price filter to display tutors based on their hourly rate.

- Dynamic Tutor Display
- As users adjust the filters, the tutor cards displayed on the homepage will dynamically update to meet the selected filter criteria.
- The filtering should be real-time, providing instant feedback based on the user's adjustments.

2.4 Chat

Chat functionality plays a crucial role in fostering direct communication between users and tutors. It enables seamless exchange of messages, facilitating quick inquiries, scheduling of sessions, and ongoing support, enhancing the overall user experience.

- Navigation to Chat/Messages Section
    - Users can navigate to the chat/messages section by clicking the "Chats" button on the navbar.
    - This action will navigate them to their previous chats with other users on the platform.
    - Alternatively, users can select a specific tutor on the homepage and then click the "Message" button.
    - This will create a new chat channel with that tutor in both the user's and the tutor's chats section.
- Chat UI Design
    - The UI should be similar to Instagram's chat interface.
    - All chats will be displayed in a column on the left side, allowing users to scroll through their chats.
    - When a specific chat is selected, the message box will appear at the bottom of the chat area.
    - The user's messages will be displayed on the right side of the chat, while messages from the other user will appear on the left side.
    - The profile of the person the user is messaging will be shown at the top left corner of the chat.
- Chat Features
    - Users should be able to upload images and files in the chat to send to the other user.
    - The chat will be developed using the Getstream API, which includes comprehensive chat features:
        - Read receipts
        - Emoji reactions
        - Editing and deleting messages
        - File and image uploads
    - Users should be able to scroll through a chat and view entire message history

2.5 Video Calling

Video calling is an extremely important feature in our application, as it enables real-time, face-to-face interactions between users and tutors. This feature supports interactive tutoring sessions, enhances engagement, and allows for more personalized and effective learning experiences.

- Initiating a Video Call
    - At the top of every chat channel, there will be an option for tutors to call students.
    - Students will not have a call button and will not be able to call tutors. This decision was made to prevent students from disturbing tutors or spamming them with calls if a session has not been booked.
- Sending Meeting ID
    - When a tutor initiates a call, a unique meeting ID is generated and sent in the chat to the student.
    - Students can click on the link containing the meeting ID, which opens a separate page for the video call.
- Pre-Meeting Setup
    - Before entering the meeting, users will have the option to turn their camera and mic on or off.
- Using Stream IO API
    - The video calling feature leverages the Stream IO API for easy access to video calling functionalities.
    - The application uses the API key and secret key provided by Stream IO to authenticate all users on the platform with a token.
    - This token allows users to access, join, and create calls, all handled by the API.
- Stream IO Features
    - Stream IO comes with a variety of useful video calling functionalities, including:
        - Built-in call generators
        - Video call layouts
        - Mic and camera settings
        - Screen sharing
        - Recording
- During the Call
    - Once both users have their mic and camera set up, they are free to communicate for up to an hour.
    - The call interface should allow users to easily manage their mic and camera settings, share their screen, and record the call if necessary.
- Ending the Call
    - Once the session is done, the tutor or the student have the option to leave then call and the tutor has the option to end the call for everyone.
    - After ending the call, both users will be returned to the application's home page.
- Interaction Flow

2.6 Bug Fix: Existing Users Redirected to the Registration Pages

In the previous sprint, existing users who tried to log in to the application were incorrectly redirected to the registration pages, requiring them to set up their entire account again.

- Expected Behavior
    - Existing users should be directed to the homepage upon logging in if they have completed their account registration.

        - Account registration is considered complete under either of the follow two conditions:
            - Option 1: The user entered their personal info and selected "Student." They should be directed to the homepage after logging in.
            - Option 2: The user completed the account registration, selected "Tutor," and completed the tutor profile. They should be directed to the homepage after logging in.
            - If a user selected "Tutor" but did not complete the tutor profile, they should be navigated to the personal info page again upon logging in.
- Testing and Validation
    - Test the login process with different user scenarios:
        - Existing students who have completed their personal info.
        - Existing tutors who have completed their profile.
        - Tutors who have not completed their profile.
        - Validate that users are directed to the correct pages based on their account completion status.
        - Ensure no users are incorrectly redirected to the registration pages upon logging in.

3. Product: Artifacts

3.1 Division of Responsibilities

Tasks are assigned based on the similarity between user stories, task difficulty, personal preference, familiarity with technologies, importance of the user story, and team member availability throughout the sprint. The video calling feature was given to a team member experienced with this technology, with another member assigned to assist as needed. Two members are handling the chat functionality due to its complexity and importance, requiring extensive database integration and proper frontend presentation. This arrangement facilitates easier testing, as the pair can log in to different accounts to verify the chat and call features.

The homepage development was assigned to a specific team member because it is a dependency for all other features, and this member had more availability early in the sprint. The search and filter functionalities, being similar and working with the same data and purpose, were assigned to another member who will also handle the bug fix. This decision was made because this team member had previously worked on login and navigation, making them familiar with the required fixes. This division of responsibilities ensures efficient task management and leverages each team member's strengths and experience.

This sprint, we decided to take on the more major features of the application while also keeping in mind the page and interaction flow. Since users should land on the homepage upon registration or login, we decided to take on this feature, and to make it more complete and convenient for users, we decided to take on the task of search and filter. The video calling and chat are two very important features of our application and both features are closely related and located on the same page, so we decided to take those on in this sprint to lay a good foundation for the coming sprints.

3.2 JIRA Management

We will keep track of user stories in our JIRA backlog for the sprint and assign points to each user story to estimate how difficult a task is. We will reassess the points after each week to ensure that we have taken on an appropriate amount of work for the sprint. Once a user story is completed, it will be marked as “Done” on our JIRA board. If time permits, that team member will take on 1 of the user stories that we have kept in our sprint wishlist. Currently, the two user stories that we have put in our wish list are SCRUM-26 and SCRUM-23

JIRA Board

3.3 Team Organization

At the beginning of each sprint, the team will fill out a when-to-meet to indicate the times that they will be available over the following 2 weeks. This way, we can set specific times for our standups and even know when other teammates may be available to help with a blocker.

To ensure that the team is always on the same page, after each standup, Arina will make a short to-do listing the tasks that are yet to be completed for the sprint, alongside who needs to complete each task and which ones require collaborative effort. This will help all team members be aware of which tasks are left, and we will be able to plan out our schedule more effectively. Miri will also make a small list of items that can be improved on from the last sprint, this can include code quality, code structure, file naming conventions, git commit messages, etc. This will ensure that our code quality meets high standards.

3.4 Sprint Retrospective

Once this Sprint is complete, our team will meet to go over all aspects of the sprint and discuss what went well, what could be improved and what didn’t go well for the sprint. This will help us improve for the upcoming sprints.


What could be improved?
More focus on code quality. We should only push clean working code that every team member can understand.
Need more frequent code reviews.
More specific details need to be outlined for the UI.
Code should be more modularized so that components can be reused and so that a single component doesn't have too much responsibility
Maybe having 2 or 3 members review the code before merging to ensure that correct code has been merged
What did you dislike?
Not having enough code review of each pull request. My code got overwritten by the merge multiple time.
I disliked that only 2 people are allowed to approve the PR because if those people were not available the rest of the team members had to wait.
No final review meeting before the demo
I disliked that we ended up having to cancel our final product review meeting before the demo because we couldn't catch the video calling bug.
What did you like?
I liked that the division of roles was based on when a team member was available. This helped me schedule my week better.
I like that 2 member were working on the same tasks. It really helped in developing and testing the feature
I really liked that we were more consistent with merging our branch with sprint2 branch because it made code more accessible to those who needed it.
I liked that all team members documented their code and put comments. It made it easier for us to understand the code without having the ask the member.
Work was completed on time.