CS3219 AY22/23 Sem 1
Team 30 Project Report

Name	Matriculation Number
Alfred Koh	A0202065W
Benjamin Koh Wei Jie	A0217739U
Brandon Kheng	A0205819B
Tan Zheng Jie, Dickson	A0217427E

Table of Contents

1. Overview of Project

1.1 Background

Hiring for Software Engineering jobs and internships often involves several technical interview segments to sieve out potential candidates. These can include online coding assessments, phone screen interviews, onsite interviews, or a combination of these. These segments test for both the candidate's problem-solving abilities and communication skills.

Despite the range of resources available online, the hours that must be spent practising technical questions to improve one's interview competency are unavoidable. However, interview practice can be made more effective and enjoyable when done with a companion.

1.2 Purpose

Our team aimed to create a web application that matches candidates preparing for software engineering technical interviews together on a platform that simulates the technical interview.

2. Individual Contributions to Project

Member	Contributions
Alfred	**Technical Contributions:** ● Matching Service (Front-end) ● Collaboration Page (Front-end) ● Selection Page **Non-technical Contributions:** ● Final Report ● UI Mockups
Benjamin	**Technical Contributions:** ● User Service APIs ● User Authentication & Authorization Extension ● Security Extension ● Login & Signup Page (Front-end) ● Dashboard Page (Front-end) **Non-technical Contributions:** ● Final Report ● UI Mockups
Brandon	**Technical Contributions:** ● Matching Service ● Collaboration Service ● Chat Messaging Extension ● Solo Practice Extension **Non-technical Contributions:** ● Preparation for milestone 2 presentation ● Final Report
Dickson	**Technical Contributions:** ● User Model Database Schema ● User Service APIs ● Questions Model Database Schema ● Questions Database ● Question Service APIs **Non-technical Contributions:** ● Final Report

3. Project Requirements

3.1 Functional Requirements

3.1.1 Prioritisation of FRs

Based on the project scope, our team decided to prioritise our Functional Requirements (FRs) according to "Must-Have" and the "Nice-To-Have" features.

Must-Haves

FR Tag	Feature	Implementation Period
FR-1	User Service	Week 3 – Week 6
FR-2	Matching Service	Week 3 – Recess Week
FR-3	Question Service	Recess Week – Week 10
FR-4	Collaboration Service	Recess Week – Week 9

Nice-To-Have

FR Tag	Feature	Implementation Period
FR-5	Communication Feature	Week 10 – Week 11
FR-6	Question History Feature	Week 9 – Week 10
FR-7	Solo Practice Feature	Week 10 – Week 11

3.1.2 User Service

ID	Functional Requirement	Priority	Justification
FR1.1	The system should ensure that every account created has a unique username	High	PeerPrep is a collaborative tool, and we need to be able to uniquely identify each user for better usability of PeerPrep.
FR1.2	The system should allow users to log into their accounts by entering their usernames and password	High	Part of project requirements. Basic login functionalities.
FR1.3	The system should allow users to log out of their account	High	Part of project requirements. Being able to log out allows users to end their session, as well as to allow other users to log in to a single device.
FR1.4	The system should allow users to delete their account	High	Part of project requirements. Allow users to delete their account if they are no longer using PeerPrep.
FR1.5	The system should allow users to change their password	Medium	This will allow users to change their password if they lose their password.

3.1.3 Matching Service

ID	Functional Requirement	Priority	Justification
FR2.1	The system should allow users to select the difficulty level of the questions they wish to attempt	High	Part of project requirements. Peerprep should provide questions with varying difficulty.
FR2.2	The system should be able to match two waiting users with similar difficulty levels and put them in the same room	High	Part of project requirements.
FR2.3	If there is a valid match, the system should match the users within 30s	High	Part of project requirements.
FR2.4	The system should inform the users that no match is available if a match cannot be found within 30 seconds	Medium	Part of project requirements. Helps with User Experience and usability of PeerPrep

3.1.4 Question Service

ID	Functional Requirement	Priority	Justification
FR3.1	The application should have a question bank containing all the questions available for the users to solve	High	Having a question bank allows us to have a variety of questions for users to attempt.
FR3.2	The application should select the question that will be displayed to the 2 users upon match	High	Based on the user's selected difficulty, the question bank will be filtered accordingly.
FR3.3	The application should display the same question on both of the user's selection page	High	For collaboration, the pair of users should only attempt the same questions
FR3.4	The application should display questions users have not attempted before	Medium	This will allow users to attempt new questions instead of repeating old ones.
FR3.5	The application should allow users to redo questions if all questions are attempted/no common question can be found	Medium	When all the questions in the question bank are completed, we will allow users to re-attempt questions.

3.1.5 Collaboration Service

ID	Functional Requirement	Priority	Justification
FR4.1	The application should display an answer textbox to both users	High	Part of project requirements. Part of User-Interface.
FR4.2	The application should allow users to collaborate on the textbox in real-time	High	Part of project requirements. By allowing real-time collaboration, users will be able to simultaneously code out their solutions in real time.
FR4.3	The application should store the content of the textbox and load it if the user refreshes the page	High	In case of any connectivity issues or accidental refresh, this helps the user save the work that has been done.
FR4.4	The application should allow users to leave the collaboration room and join a new collaboration room	High	Allow users to change the difficulty, attempt new questions, or match with a different person.

ID			
FR4.5	The application should allow users to navigate away but rejoin the same collaboration room	Medium	Part of User-Experience. Helps users to rejoin back the room in case of any connectivity issues or accidental exits.

3.1.6 Communication Extension

ID	Functional Requirement	Priority	Justification
FR5.1	The application should allow the 2 users to chat with one another after they have matched	Medium	Part of User-Experience. Allow users to communicate with one another to brainstorm solutions for the question.

3.1.7 History Extension

ID	Functional Requirement	Priority	Justification
FR6.1	The application should update the user question history after the user attempts the question	High	Allows us to keep track of the question statistics and performance of each user. These statistics will be displayed on our dashboard.

3.1.8 Solo Practice Extension

ID	Functional Requirement	Priority	Justification
FR7.1	The application should allow the user to attempt questions alone	Medium	When users want to practice coding alone, they can choose to use the solo function. This helps when there are no matches to be found after multiple queues.

3.2 Non-Functional Requirements

3.2.1 Prioritisation of NFRs

Based on the myriad of quality attributes to consider, our group has narrowed it down to a selection of 7 key quality attributes that we would like to ideally implement in our PeerPrep application.

Attribute	Score	Installability	Performance	Reliability	Availability	Security	Scalability	Usability
Installability	4		^	<	<	^	<	^
Performance	2			<	<	<	<	^
Reliability	7				^	^	^	^
Availability	5					^	<	^
Security	3						<	^
Scalability	6							^
Usability	1							

After comparing each quality attribute with one another, we ranked them based on their relative importance and decided to prioritise the top 3 attributes: **Usability, Performance** and **Security**.

Based on the different milestones which we identified in the previous section we determined at what point we should focus on which quality attributes. For **Security**, we decided to implement it in Milestone 1, along with the **User-Service.** For **Usability,** we determined that it was important to focus on it only in Milestone 2, as we wanted to get the MVP out as soon as possible, and we can focus on User Experience in the latter half of the semester. Finally, we also decided that **Performance** considerations would be prevalent mainly in Milestone 2 when we had to display user information and allow for collaboration.

NFR Tag	Quality Attributes
NFR1	Security
NFR2	Usability
NFR3	Performance

3.2.2 Security Requirements

Security was prioritised in Milestone1, as we believe it is vital to protect the user's personal information, prevent data leaks and ensure authentication and authorization.

ID	Non-Functional Requirement	Priority	Justification
NFR1.1	The application should ensure users utilise a strong password (Longer than 8 characters, including 1 alphabet, number & special character)	High	To protect against a brute-force attack, we decided that a suitable countermeasure would be to enforce a minimum password length.
NFR1.2	The application should hash and salt the user's password before being stored in the database	High	This is to prevent reverse engineering of a ciphertext password and allow for data safety.
NFR1.3	The application should check if the user is authenticated and/or authorized before calling the API	High	This is to ensure that all actions done on Peerprep can be traced back to a single identity, and users will be able to carry actions only if they are authorized.
NFR1.4	The application should blacklist a user's token upon logout/deletion of the account.	Medium	This is to ensure that users have to authenticate themselves every time upon a new session.

NFR1.1 The application should ensure users utilise a strong password (Longer than 8 characters, including 1 alphabet, number & special character)

Based on recommendations by NIST, we have decided to have a minimum password length of 8, and we want to ensure a strong level of password complexity, hence the requirement of at least 1 uppercase letter, 1 lowercase letter, 1 number, and 1 special character. (Jethva, 2022)

To conduct password testing, the team tried signing up for an account with a password that is no longer than 8 characters and includes at least 1 alphabet, number, and special character.
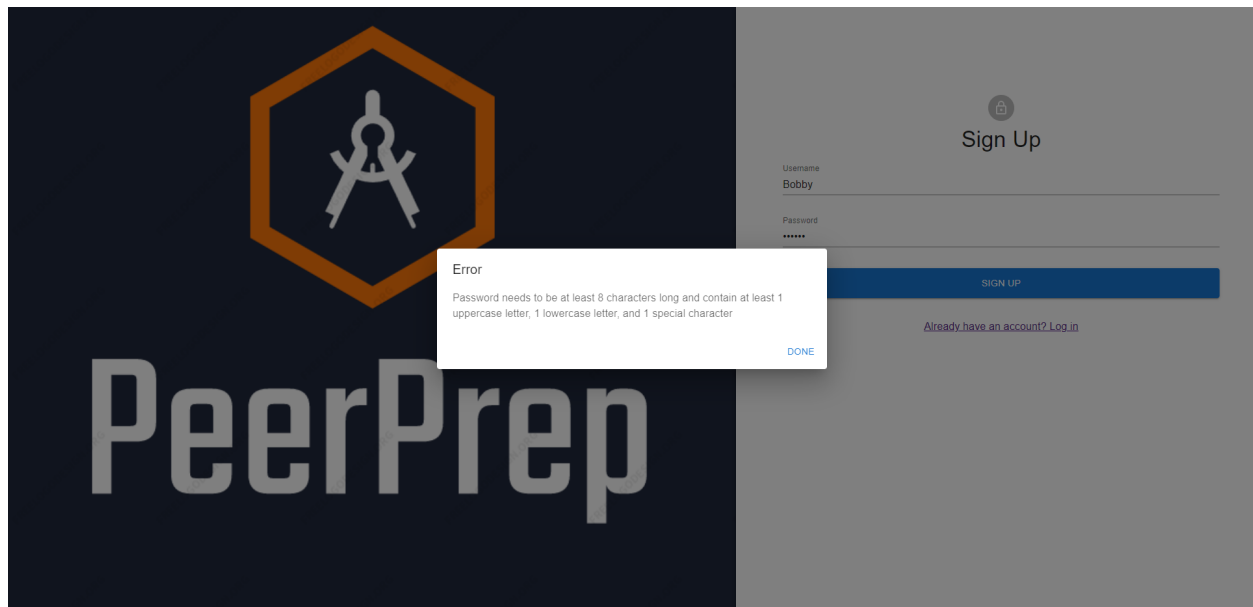
Diagram 1: Password not meeting requirements

NFR1.2 The application should hash and salt the user's password before being stored in the database

Not only that, but we also want to ensure that how the passwords are stored will be secure. (Mindanao, 2022) Thus, we decided to salt the password and hash it before being stored in our database. This makes it impossible for an attacker to reverse engineer the stored ciphertext because the hash function is one-way.

```
  _id: ObjectId('635e000cee0ccb8813e294b9')
  username: "bob"
  password: "$2a$10$i2xDrr7GGwIuQnorKGl2puPVUupv7q5mja9zgjbnHGeKvD5LhPoje"
> easy: Array
> medium: Array
> hard: Array
  __v: 0
```

Diagram 2: Password hashing and salting

NFR1.3 The application should check if the user is authenticated and/or authorized before calling the API

We also added the use of JWT in cookie form to authenticate users upon logging in, and make sure they have the correct authorization before making API calls.

The team tried to fetch a user's dashboard page via Postman without a JWT to conduct authentication and authorization testing.
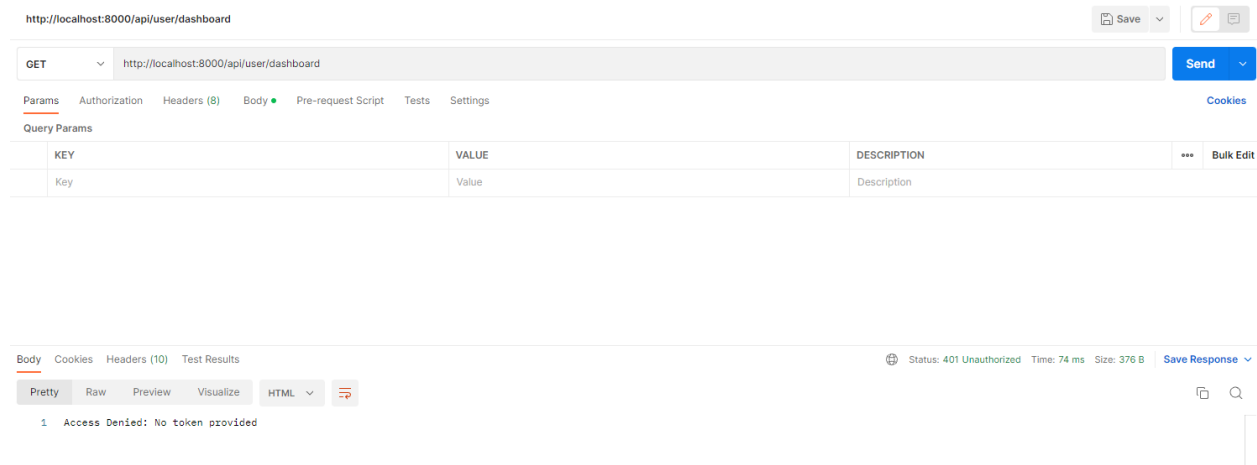

Diagram 3: Unauthenticated/Unauthorised Users cannot access APIs

NFR1.4 The application should blacklist a user's token upon logout/deletion of account, ensuring that the user cannot reuse that token to log in

A user's JWT will be blacklisted upon logout/deletion of an account, to prevent a user from logging in back with an old token. The tokens will also expire after 1hr, and the user would have to log in again.

To conduct blacklist testing, the team logged out of a user account and tried to log in using the same user's old JWT token.

You are not logged in :(Please Log in

GO TO LOG IN PAGE

Diagram 4: User with blacklisted token denied access

3.2.3 Usability Requirements

To ensure a smooth User Experience, we drafted up the following NFRs. We broke down the design of the Frontend into different phases.

1. Integration of User-Service (NFR2.1, NFR2.2)
2. Integration of Matching-Service and Collaboration-Service (NFR2.3, NFR2.4, NFR2.5)
3. Integration of Question-Service and History-Extension (NFR2.6, NFR2.7)

ID	Non-Functional Requirement	Priority	Justification
NFR2.1	The application should have a login and signup page akin to other popular applications	Medium	Following industry standards brings about consistency and familiarity for the user.
NFR2.2	The application's dashboard page should allow users to perform account-related actions intuitively	High	The users should know where to look when they want to perform specific actions like logout or changing their password.
NFR2.3	The application should lead the user from the Dashboard to the Selection page intuitively	High	This is to ensure a proper interaction flow for the users.
NFR2.4	The application should lead the user from the Selection page to the CollabLeet page intuitively	High	This is to ensure a proper interaction flow for the users.
NFR2.5	The application should display a overlay with a timer while waiting for a match	Medium	This provides visibility to the system status, to let the users know that the application is searching for a partner.
NFR2.6	The application's dashboard page should allow users to see their history of questions done	High	This is to allow the users to see their progress, and give them a sense of achievement.
NFR2.7	The application's dashboard page should allow users to see the breakdown of questions done	Medium	This is to allow the users to see their progress, and give them a sense of achievement.

NFR2.1 The application should have a login and signup page akin to other popular applications

Following Nielsen's Usability Heuristics for User Interaction Design, namely "Consistency & Standards" and "Recognition rather than Recall" (Nielsen, 2020), we decided to have a login and signup page that is familiar and simple.
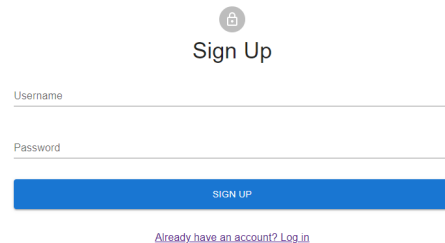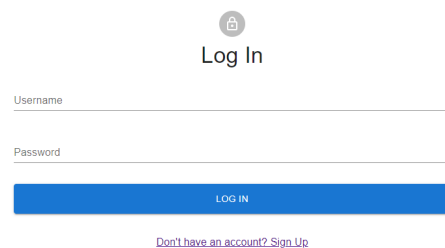

Diagram 5: PeerPrep Signup Page

Diagram 6: PeerPrep Login Page

NFR2.2 The application's dashboard page should allow users to perform account-related actions intuitively

The application's dashboard page allows users to access our Reddit page, change their password, log out or delete their account. We decided to employ "Recognition rather than Recall" and "Help users recognise, diagnose and recover from errors" from Nielsen's Usability Heuristics. (Nielsen, 2020)

We used familiar icons that comply with the actions, and we separated "high-risk" actions such as "Delete Account" with a line and made it red, to provide a warning to the users. Once the user clicks on the "Delete Account", a popup will appear to prompt the user to confirm or cancel their actions.
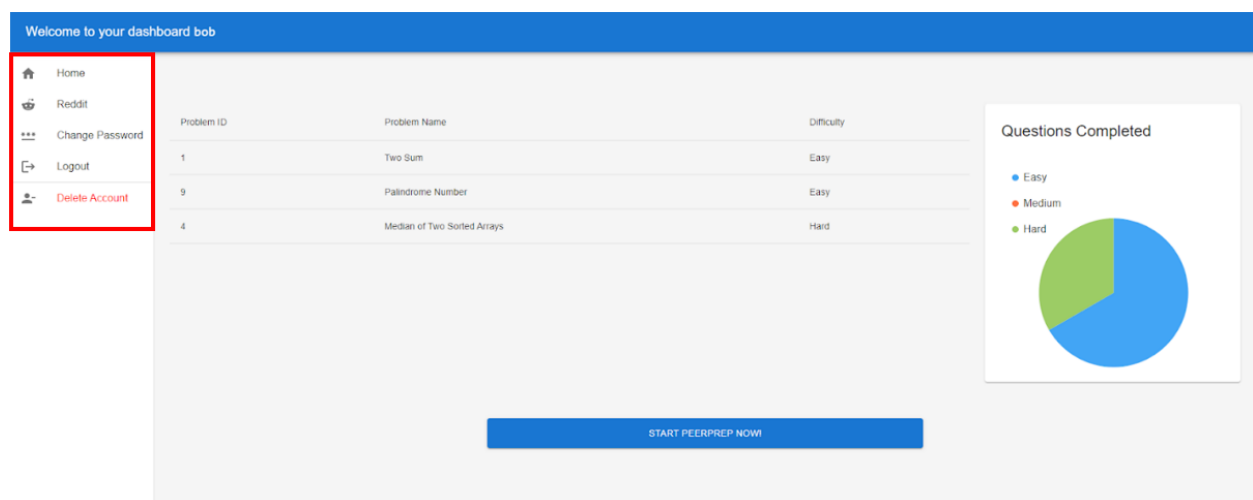

Diagram 7: PeerPrep Dashboard Page

NFR2.3 The application should lead the user from the Dashboard to the Selection page intuitively

We decided to employ a minimalist design, drawing the user to important buttons that will allow them to progress to the next pages

The application has a "Start PeerPrep Now!" button which brings the user from the Dashboard page to the Selection page.
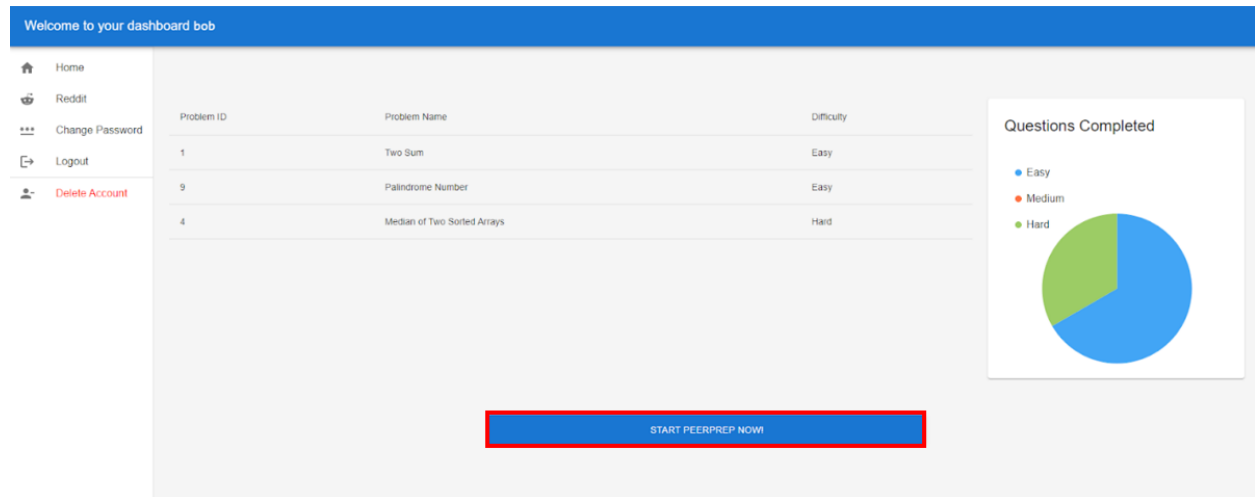

Diagram 8: PeerPrep Dashboard Page

NFR2.4 The application should lead the user from the Selection page to the CollabLeet page intuitively

Employing a similar design to the Dashboard Page, the Selection Page has "Solo" and "Collab" buttons which bring the user to the CollabLeet page, where users will be matched and can start coding.
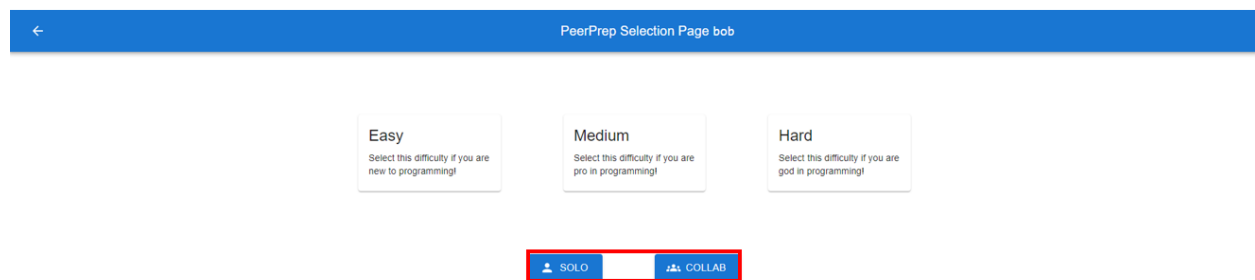

Diagram 9: PeerPrep Selection Page

NFR2.5 The application should display a overlay with a timer while waiting for a match

To allow users to understand the outcome of their interactions and determine the next steps, we decided to ensure that the application displays a timer when finding a match. This provides a user with a "Visibility of System Status", as suggested in Nielsen's Usability Heuristics. (Nielsen, 2020)
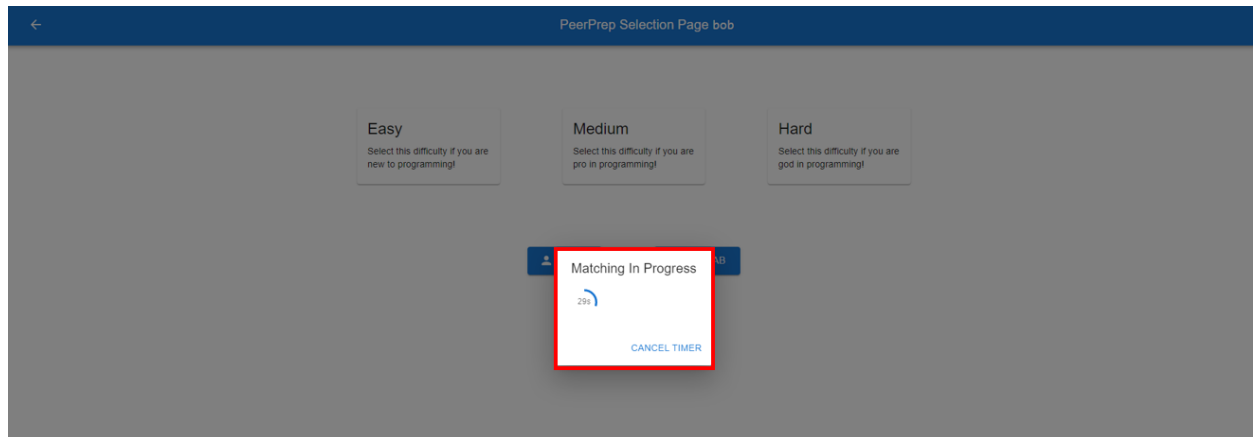

Diagram 10: PeerPrep Selection Page with Timer

NFR2.6 The application's dashboard page should allow users to see their history of questions done

We decided to provide the users with a sense of accomplishment and progress by integrating the History-Extension with the Dashboard Page.

On the application's dashboard page, users can see the questions they have attempted and their corresponding difficulty.
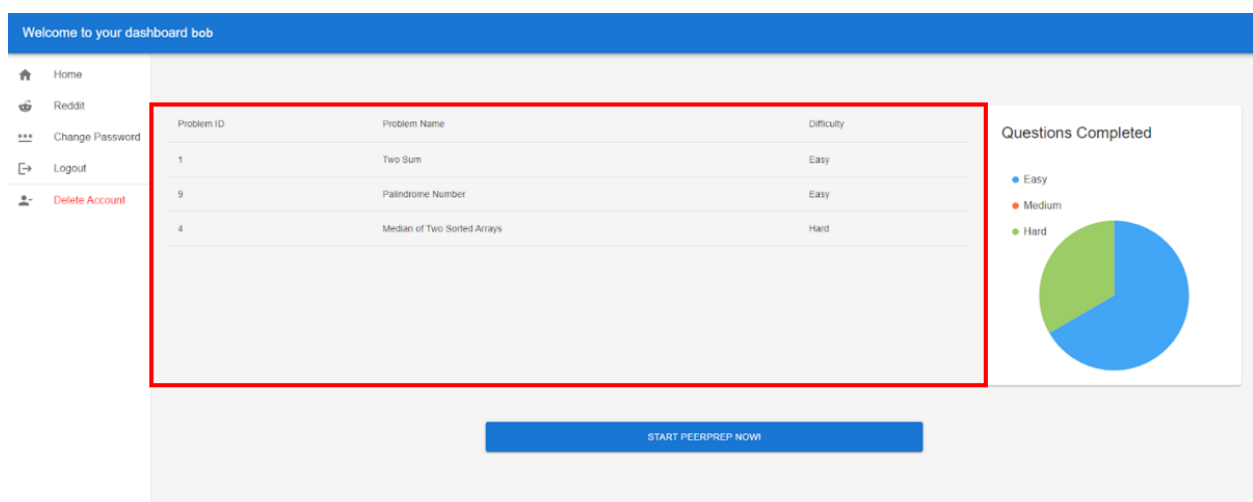

Diagram 11: PeerPrep Dashboard Page

NFR2.7 The application's dashboard page should allow users to see the breakdown of questions done

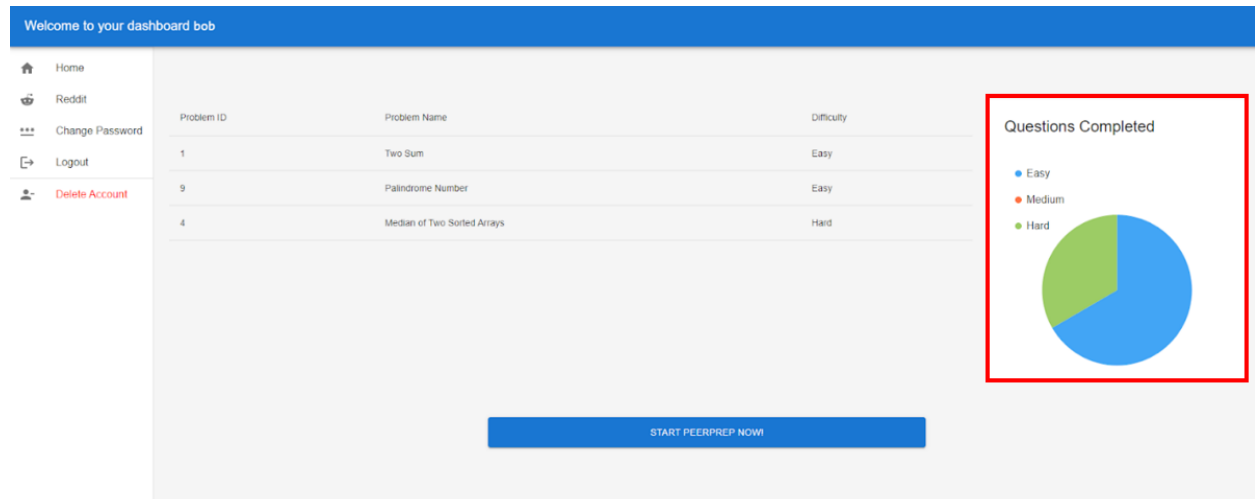On the application's dashboard page, users can see the summary of the difficulty of the questions attempted.

Diagram 12: PeerPrep Dashboard Page

3.2.4 Performance Requirements

ID	Non-Functional Requirement	Priority	Justification
NFR3.1	The user's dashboard should be loaded 3 seconds upon log in	High	Requiring more than 3 seconds to log in and load the landing page will cause a poor user experience.
NFR3.2	The collaborative text editor should have a response delay of no more than 1 second	High	Compared to other collaborative editors, a delay of 1 second will be too slow.
NFR3.3	The communication feature should have a response delay of no more than 1 second	High	Compared to other real-time communication services, a delay of 1 second will be too slow.

Here is a video to demonstrate these 3 NFRs.

4. Development Process

4.1 Timeline (Development Plan)

Milestone	Week	Deliverables
1	6	Creation of MV: User-Service, Matching-Service, Security Extension, UI Mockups, Basic UI
2	9	Completion of all "Must-Haves": Collaboration-Service, Basic Question-Service, Improved UI
3	12	Finalisation of Application: Question-Service, Communication Extension, History Extension, Solo Extension, Finalised UI

Diagram 13: Screenshot of the spreadsheet used for planning our development timeline

4.2 Project Management

The development process was planned during our first meeting, in which we decided to focus on constant improvements to our application. We finalised what development methodology to adopt, and how to manage our tasks well.

4.2.1 Scrum

Our team adopted Scrum, an agile development methodology that focuses on incremental and iterative processes. Based on the 3 milestones given to us by the teaching team, we identified the main goals we want to accomplish with each milestone. To better handle the workload, we further broke the project timeline down into weekly sprints. Hence, we decided to set aside some time every Friday morning, to conduct a weekly sprint.

During each weekly team meeting, we would conduct a stand-up for each of us to share the work that had been completed in the previous week, the work to be completed in the following week, and any obstacles that we were facing. Subsequently, we would demo the features and functionalities that each of us had been working on. Finally, if there were any outstanding issues from previous sprints or milestones, these were discussed and shifted to the next sprint or milestone if needed.

Sprint Outline	
Week	**Deliverables**
3	1. Basic User-Service features 2. Basic Matching-Service features
4	1. Basic User-Service features 2. Basic Matching-Service features 3. Basic Security Extensions 4. Frontend Integration
5	1. User-Service features 2. Matching-Service features 3. Security Extensions 4. UI Mockups
6 (Milestone 1)	1. Finalised User-Service features 2. Finalised Matching-Service features 3. Finalised Security Extensions 4. Demo Preparation
Recess	1. Basic Question-Service features 2. Basic Collaboration-Service features

7	1. Basic Question-Service features 2. Basic Collaboration-Service features 3. Improve UI
8	1. Question-Service features 2. Collaboration-Service features 3. Improve UI
9 (Milestone 2)	1. Question-Service features 2. Basic History-Extension features 3. Finalise Collaboration-Service features 4. Integrate Frontend 5. Demo Preparation
10	1. Finalise Question-Service features 2. Basic History-Extension features 3. Basic Communication-Extension features 4. Basic Solo-Extension features 5. Improve UI
11	1. Finalise History-Extension features 2. Finalise Solo-Extension features 3. Finalise Communication-Extension features 4. Improve UI
12 (Milestone 3)	1. Finalise UI 2. Finalise Application 3. Report 4. Demo Preparation

4.2.2 GitHub Issues

Since our repo is already hosted on Github, we decided to use GitHub Issues to delegate our tasks. At the start of each milestone, our team would discuss the FRs to be implemented for that milestone. These FRs were added to GitHub issues with the appropriate description, tags, and assignees to ensure all the work to be done was being tracked.
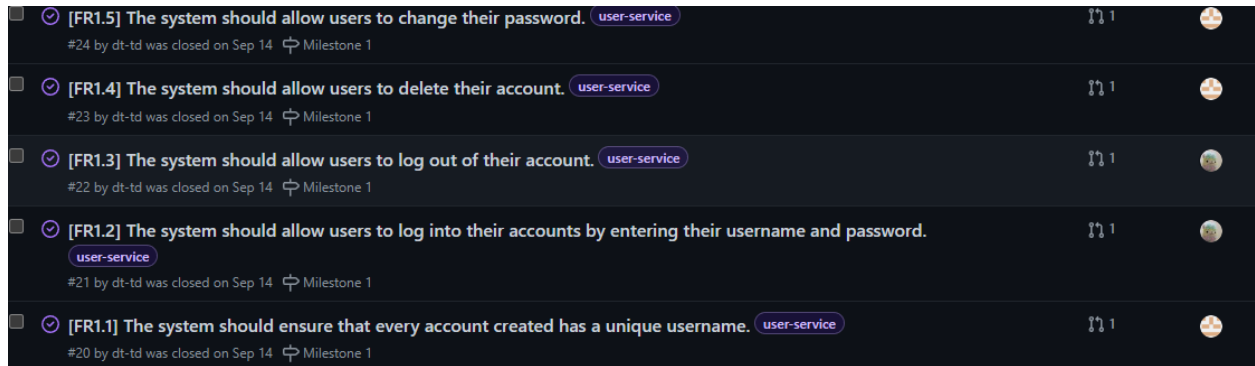
Diagram 14: Github Issues for tracking of tasks

Once a team member was done with their task, we conducted a peer review of their PR to ensure correctness and good code quality. Once completed, the PR was merged into the Master branch and the relevant issue would be closed.

5. Developer Documentation

5.1 Tech Stack

Component	Technology Used
Frontend	React
Backend	Express.js/Node.js
Database	MongoDB
Pub-Sub Messaging	Socket.io
Project Management	Google Drive, GitHub Issues, Zoom

5.2 Architecture and Design Patterns

5.2.1 System Architecture

When deciding on a system architecture to follow, the following considerations were made:

Architecture Design Considerations		
Architecture /Attribute	**Microservices Architecture**	**Monolithic (3-tiered) Architecture**
Maintainability	Since the code is loosely coupled, each microservice can be worked on in isolation, so extending functionality could be easier.	Code base is highly coupled, thus more time must be spent when adding new features and extending the code.
Scalability	Very scalable since each service is developed individually and resources can be allocated separately for each microservice	Not as scalable as the microservices architecture since resources must be allocated for the system as a whole.
Development	More time-consuming since the communication between each microservice must be considered.	Development may be faster as it requires less coordination between microservices and team members working on each service.

As PeerPrep is intended for anyone preparing for technical interviews, it should be able to handle many simultaneous users and we should be able to allocate additional resources to a particular service if needed. Additionally, since our team's development process is agile, new functionality will be added and extensions will be made to existing services in each sprint. Since our team prioritised scalability and maintainability in developing PeerPrep, we decided to follow the microservices architecture.

PeerPrep relies on four main services - the user service, matching service, collaboration service, and question service.
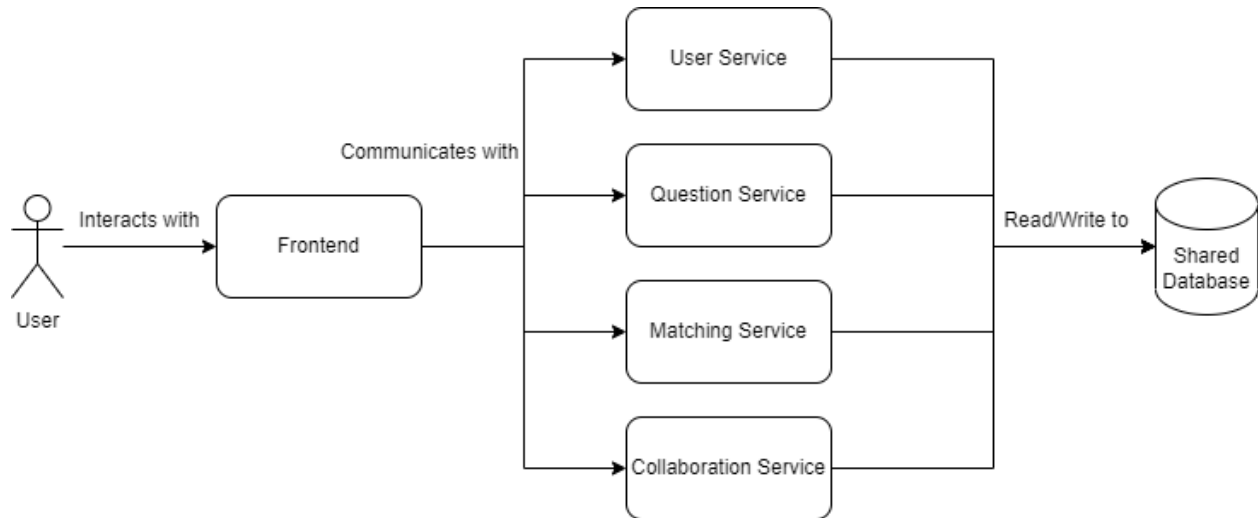
Diagram 15: Simplified architecture diagram for PeerPrep

A more detailed version of this architecture diagram can be found in the appendix.

The Services section will provide a detailed description of each microservice.

5.2.2 Frontend

The frontend consists of the presentation layer. Create React App (CRA) is the framework which we had chosen to develop the frontend while Material UI is the styling library used to enhance the look and feel of our web application.

Elements on our web pages perform specific functionalities and may invoke HTTP requests to fetch or manipulate data. Examples include the selection of question difficulty, the selection of solo/collab mode, etc. These HTTP requests are sent to the application layer to be processed. Upon receiving the response from the application layer, the frontend will display the relevant information to the user.

Frontend Framework Considerations		
Framework /Qualities	**NextJS**	**Create React App (CRA)**
Development	Building a NextJS application requires higher investment from the developers upfront, as ReactJS knowledge is a prerequisite.	It is easier to get started with CRA.
Performance	Applications built with NextJS are faster compared to CRA.	Applications built with React will display slow loading times.
Community	Less developers and support, harder learning curve.	More popular choice and have a lot more support.

Overall, since our team does not have extensive knowledge in frontend, we wanted to use the framework that provides a smoother learning curve, allowing us to efficiently apply what we learned. Hence, we went with Create React App (CRA), the de facto way to bootstrap React applications.

User

Presentation Layer

ReactJS

Material UI

ReactJS Router

Signup Page

Dashboard Page

Login Page

Selection Page

CollabLeet Page

HTTP Requests

Application Layer

......
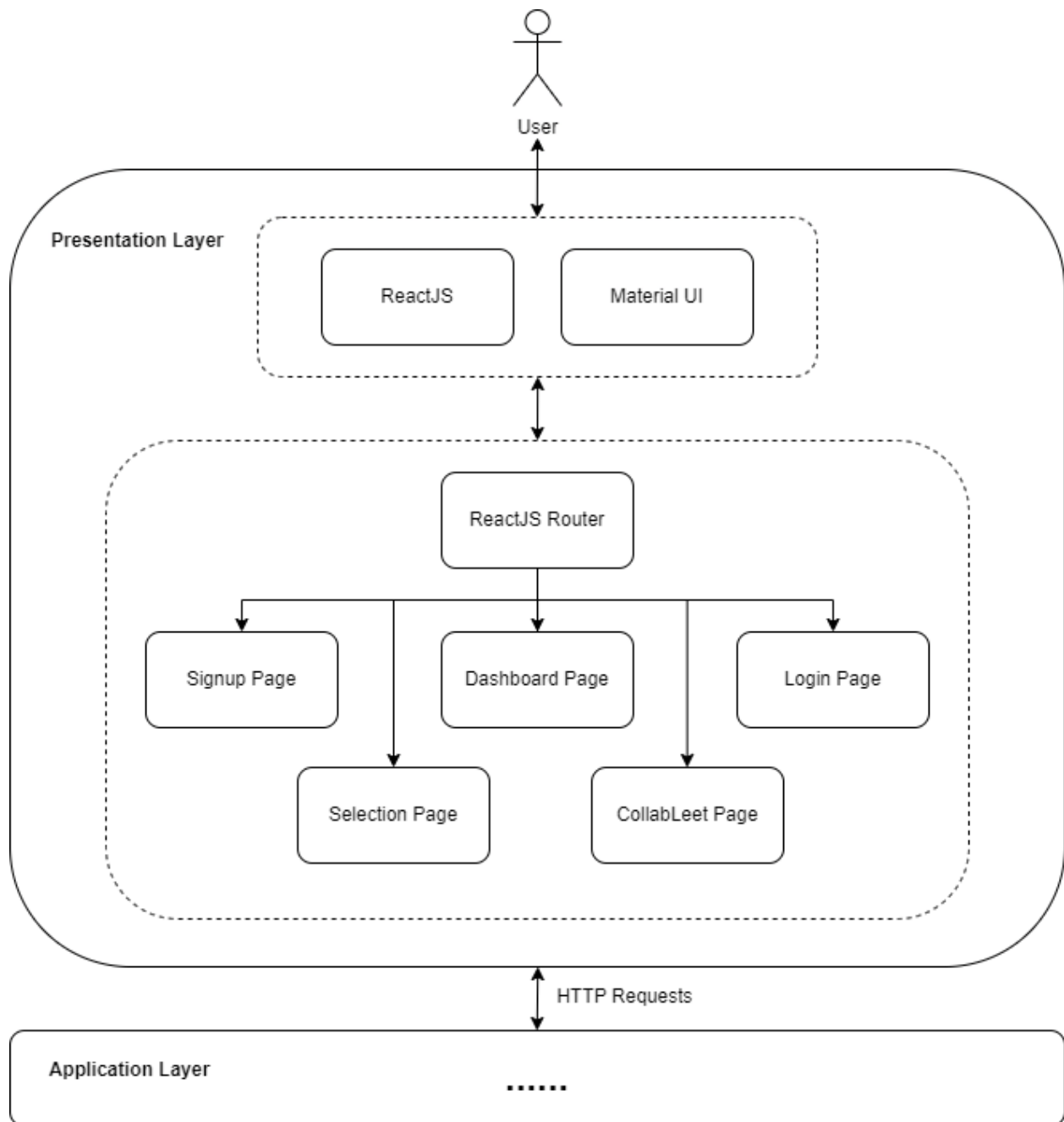
Diagram 16: PeerPrep's Frontend

5.2.3 Backend

The backend consists of the application and database layers. ExpressJS is the framework used to develop the application server. The framework is minimal, flexible, and unopinionated, which allows teams to implement a robust server to handle HTTP requests. Incoming HTTP requests from the presentation layer are received and routed by the in-built ExpressJS router to the relevant controller. The controller will invoke the necessary methods available in its model. These methods include creating, reading, updating, and deleting records in the database.

After read/write operations are performed on the database by the models we have created, the result is returned to the controller which is then encapsulated as an HTTP response and returned to the presentation layer. The database used in the web application is MongoDB, a document database with a reputation for scalability and flexibility for querying and indexing.
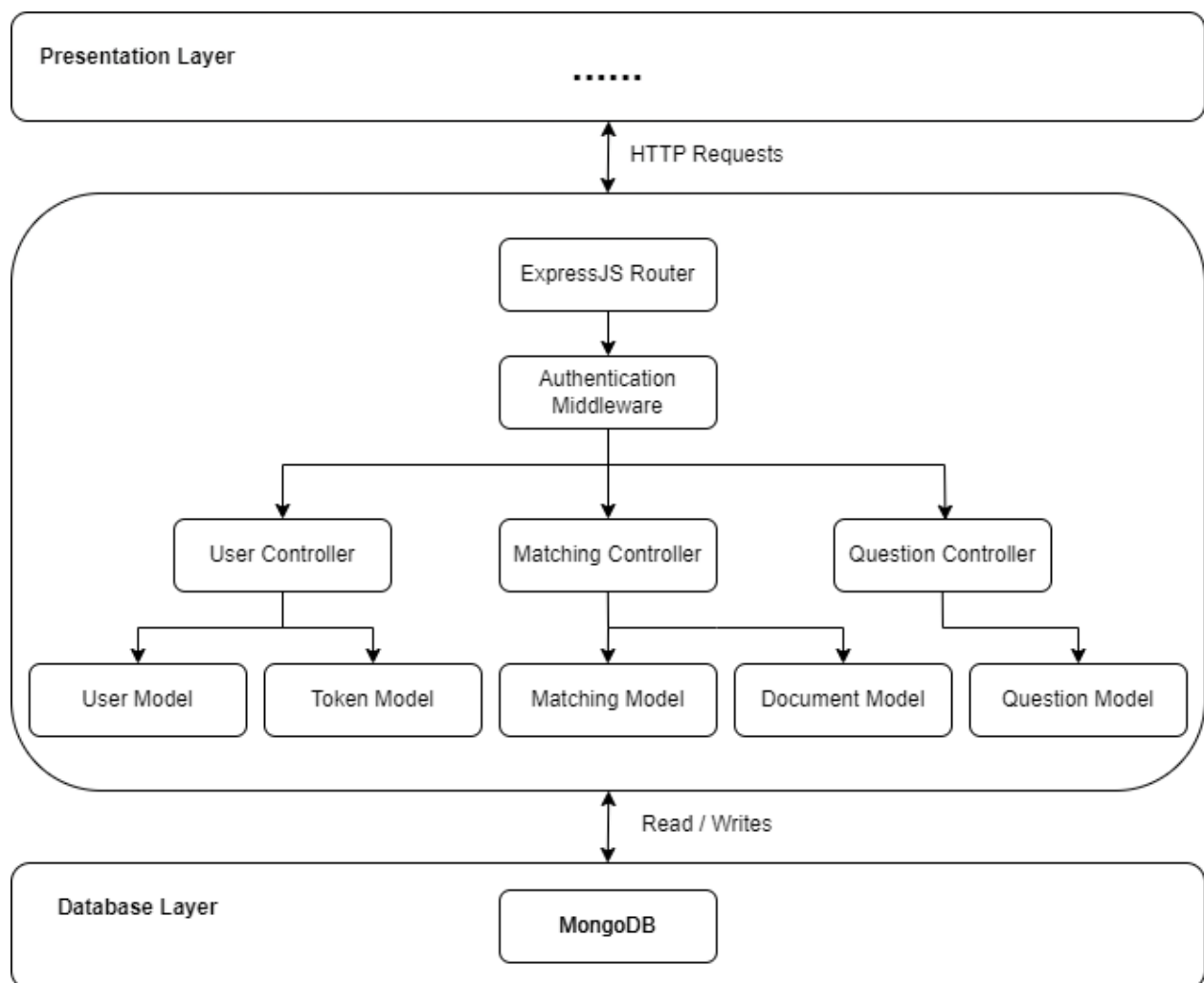
Diagram 17: PeerPrep's Backend

5.2.4 Shared Database

Our team decided to utilise a shared database among the microservices rather than implementing a separate database for each microservice. Given the time and manpower constraints during project development, the shared database pattern was chosen to allow our team to focus on maintaining and deploying just one database. The database could be separated by service and scaled individually in the future should the need arise.

Because it provides efficient lookups, integrates easily into our MERN application, and can be easily deployed on cloud environments, MongoDB was chosen as the database software. Below are the schemas used to represent the data stored for each service:

Model Name	Data stored
User Model	username, password, easy, medium, hard
Token Model	token, TTL
Match Model	isPendingMatch, difficulty, user1Username, user2Username, user1SocketId, user2SocketId, collabRoomSocketId
Question Model	id, title, difficulty, question, examples, constraints
Document Model	id, documentContents

5.2.5 Facade Pattern

Our team utilised the facade pattern within each microservice to expose the APIs of each microservice, yet hide away any complex logic or interactions implemented within it.

Using the user service as an example, despite handling different functionalities like the authentication of users, and storage of user details, user-service/index.js only contains the HTTP request APIs to accomplish all these different functionalities. As such, the interface is isolated from the complexity of the subsystem and other microservices utilising the user-service do not need to work around these complexities.

```
// Controller will contain all the User-defined Routes
router.get("/", (_, res) => res.send("Hello World from user-service"));
router.post("/", createUser);
router.delete("/:username", authProtect, deleteUser);
router.put("/:username", authProtect, updatePassword);
router.get("/login", (_, res) => res.send("Hello World from login"));
router.post("/login", loginUser);
router.get("/dashboard/", authProtect, getProtectedMe);
router.post("/dashboard/", getMe);
router.post("/logout", logoutUser);
router.post("/history", updateHistory);
router.get("/history/:username", getHistory);
```

Diagram 18: Screenshot of the exposed user-service APIs

If the need to change a user-service API arises, code will only have to be changed in one place and the subsystems of user-service will not need to be modified. Thus, the maintainability of the code is also increased.

5.2.6 Publish-Subscribe Pattern

Our team utilised Socket.io in the matching service and collaboration service, as well as the chat messaging extension. Socket.io relies on the pub-sub mechanism to transfer messages between the Socket.io server and clients which, in turn, allows for real-time communication between users.

When a match is created between users, a unique room ID is provided to the users, which functions as the Socket.io room ID. Rather than sending messages directly to/between clients, the server publishes the messages for users of each room to subscribe to. While communication is facilitated by the server, clients can only publish and subscribe to messages in their unique rooms. This allows for private communication between clients in the same room.

Through the publish-subscribe pattern, publishers and subscribers are loosely coupled. Other than emitting and responding to messages appropriately, each can be implemented and run in isolation, which increases code maintainability and scalability.
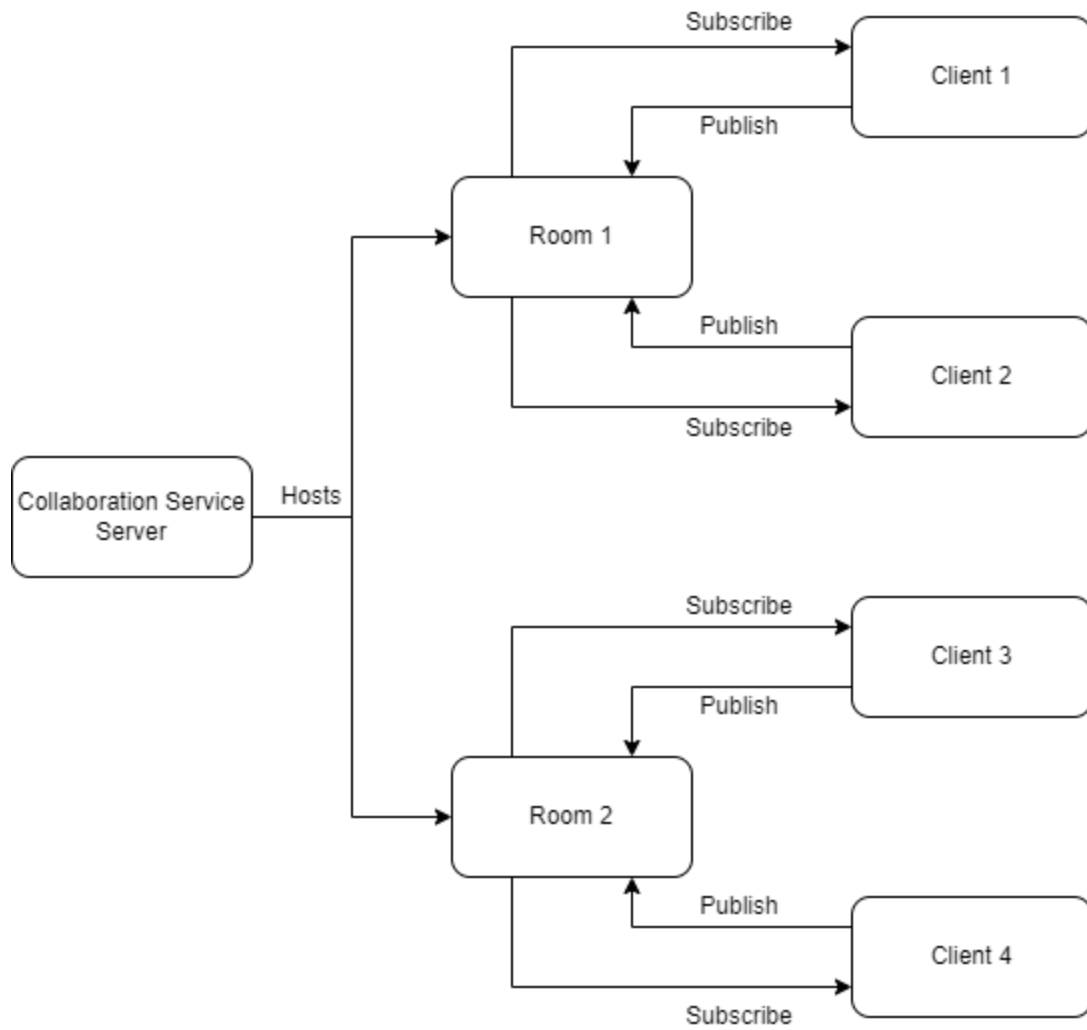
Diagram 19: Illustration of clients communicating via pub-sub messaging in private rooms facilitated by the collaboration service

5.3 Services

5.3.1 User Service

The User Service enables the creation, retrieval, and deletion of users. Users are created with a username and password provided during account registration. It is built with Express and Node.js, and works with nearly every other system to allow users to interact with PeerPrep's API.

Upon successful registration, the user's credentials will be saved in PeerPrep's database. Also, after a question is attempted using PeerPrep, the user's question history will be updated and the user will not see the same question again. To incorporate a well-rounded user service experience into this application, our features include allowing users to change their password as well as delete their accounts.
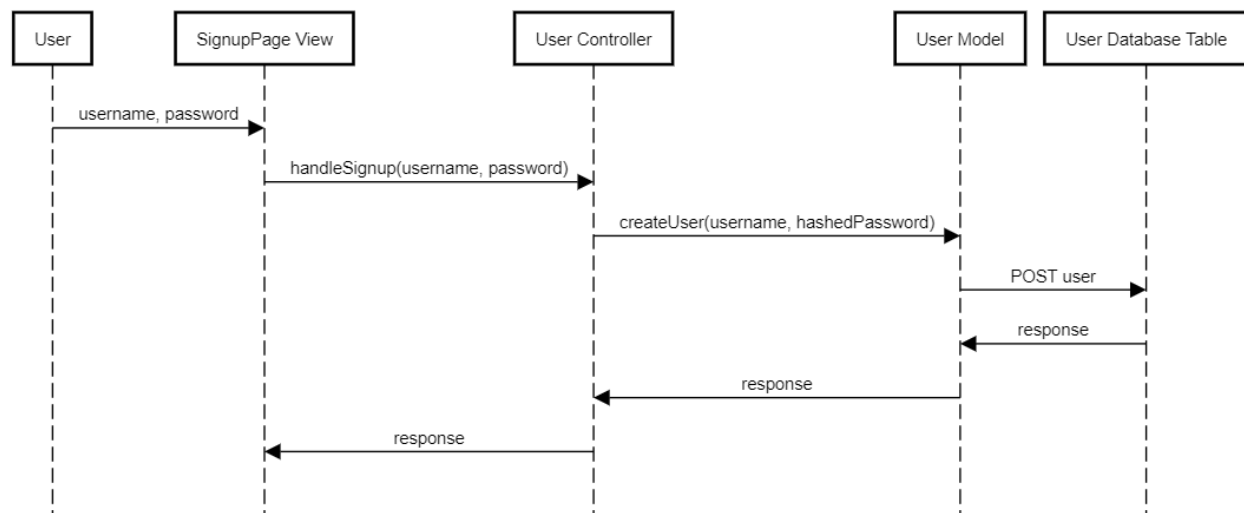
Diagram 20: Sequence Diagram demonstrating user registration

Authentication & Authorization Extension

After the users have been logged in, a JWT will be created and stored as a cookie. That token will be used for authentication and authorization and will remain valid unless the user logs out or deletes their account. Their identity will be authenticated when landing on the Dashboard page, and actions such as changing a user's password will require that token for authorization.

Security Extension

As the user is logging out, the user's JWT will be blacklisted and stored on a database with a TTL of 1 day. After 1 day, the JWT itself will have expired and become invalid.

When deciding on which database to use for the storage of the blacklisted JWTs, the following considerations were made.

Database Considerations		
Database /Attribute	**MongoDB**	**Redis**
Persistence	Data is persistent, and the JWT will be present for the full duration of the TTL, which by then, would have become invalid.	Data can be lost if servers were terminated. JWTs that were intended to be blacklisted can be used again for authentication.
Data Model	Storing each token as an entry means that a user can have multiple blacklisted JWTs	Key-value store. Since a user can have multiple blacklisted JWT in a day, this can be an issue if we store users as keys and JWT as value.
Speed	Since MongoDB uses on-disk storage, it will be slower compared to in-memory storage.	Since Redis uses in-memory storage, it will be faster compared to on-disk storage.

Overall, our team decided that this feature's persistence is more important than speed. We feel that it is acceptable to lose out on speed and performance, rather than forgoing security. The following diagram illustrates the process of a JWT being blacklisted and stored on our database.
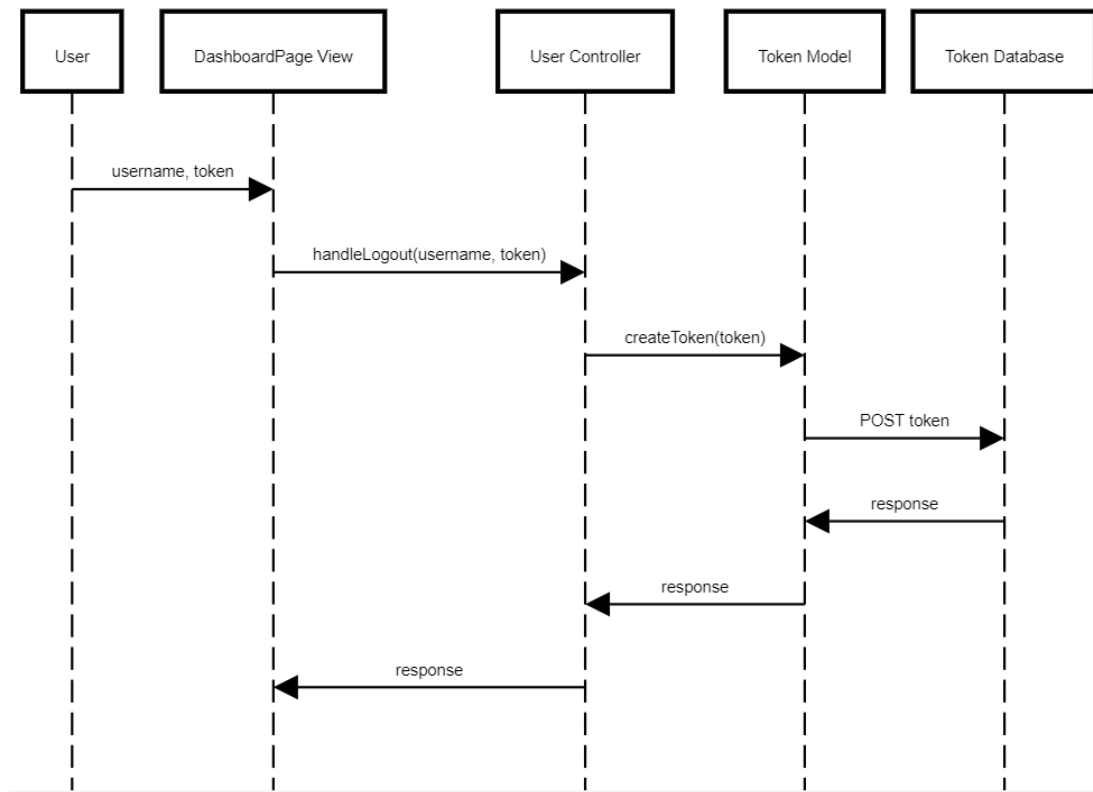
Diagram 21: Sequence Diagram demonstrating token creation

APIs

User Service APIs		
Request	**API Route**	**Description**
GET	/api/user/dashboard	Gets user cookie for authorisation
GET	/api/user/history/:username	Gets user's question history
PUT	/api/user/:username	Change user's password
POST	/api/user	Creates user for PeerPrep
POST	/api/user/login	Login user to PeerPrep
POST	/api/user/logout	Logout user from PeerPrep
POST	/api/user/dashboard	Gets user information
POST	/api/user/history	Updates user's question history
DELETE	/api/user/:username	Delete user by username

5.3.2 Matching Service

The matching service allows users to form matches. Once in a match, a pair of users are then able to collaborate on a PeerPrep question together (via the collaboration service).

To initiate the matching process, a user first selects the difficulty of the question they want to attempt. The user's data is then added to the match database as a pending match. If another user chooses to collaborate on a question of the same difficulty, the pending match is updated to a successful match between both users. Both users are then provided with an empty roomID and sent to the collaboration page.

If a user in a pending match does not find a match within 30 seconds, their pending match is removed from the match database and they can try to find another match again. Additionally, a user may choose to stop waiting for a match by stopping their match timer.
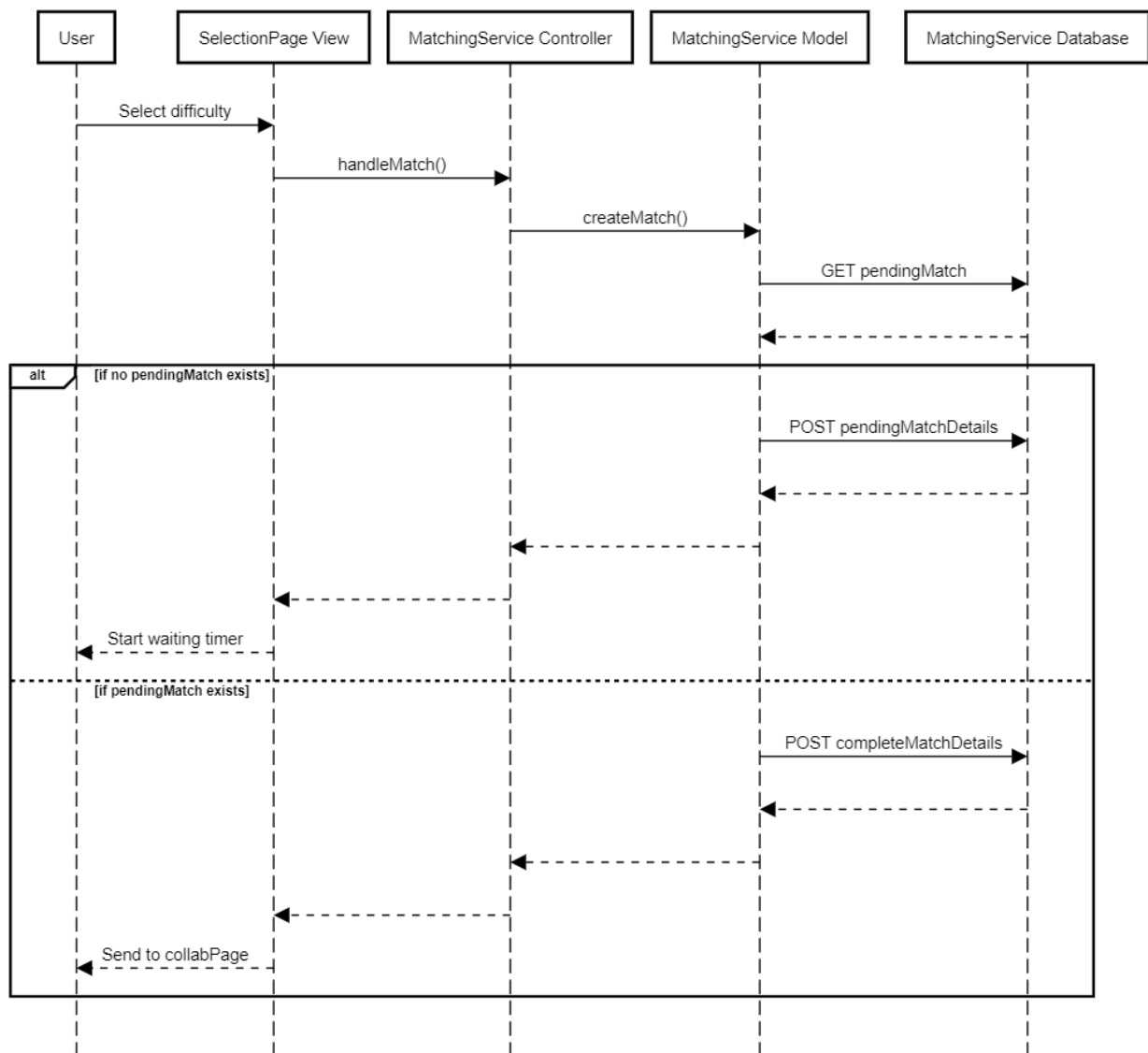
Diagram 22: Sequence diagram for a user getting a match
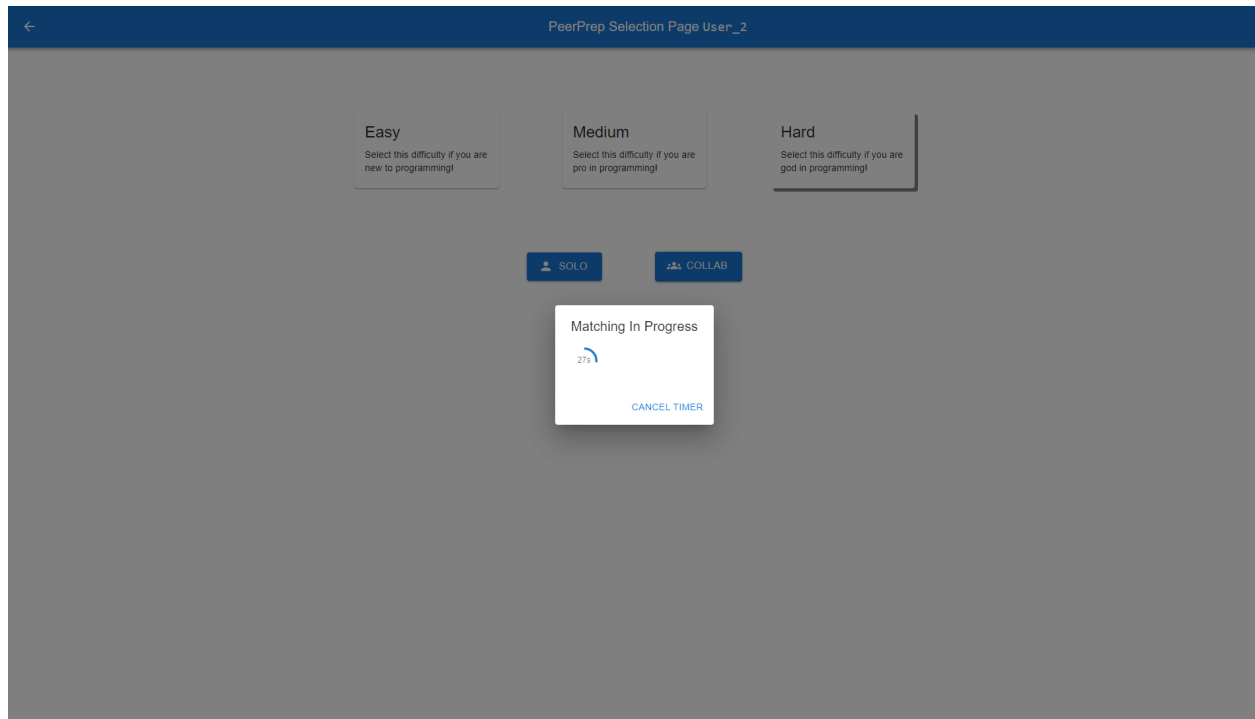
Diagram 23: Screenshot of the selection page timer while user is waiting for a match

Given that scalability is an important factor in the development of PeerPrep, as well as the importance of speed for a client-facing app, we decided to utilise Pub-Sub messaging through Sockets.io. Additionally, Sockets.io supports the delivery of messages to different clients based on the socket room that they are in. This functionality can be relied upon for other services such as the collaboration service and chat messaging discussed later.

APIs

Matching Service APIs		
Request	**API Route**	**Description**
GET	/api/match/:username	Gets the user's roomID
POST	/api/match/	Finds a complete match or creates a pending match for the user
POST	/api/match/solo	Handles a user who wants to practice in solo mode by providing an unused roomID
DELETE	/api/user/:username	Deletes user by username

Solo Practice Mode Extension

If a user is unable to find a match or wishes to practice questions on their own, they can do so with the solo practice mode. This extension is based on the matching service, where users are assigned to a collaboration room. However, a user in solo practice mode is not placed in a match and is sent to a room alone. In the room, most functionality remains the same but there is no other user to communicate or practice with.
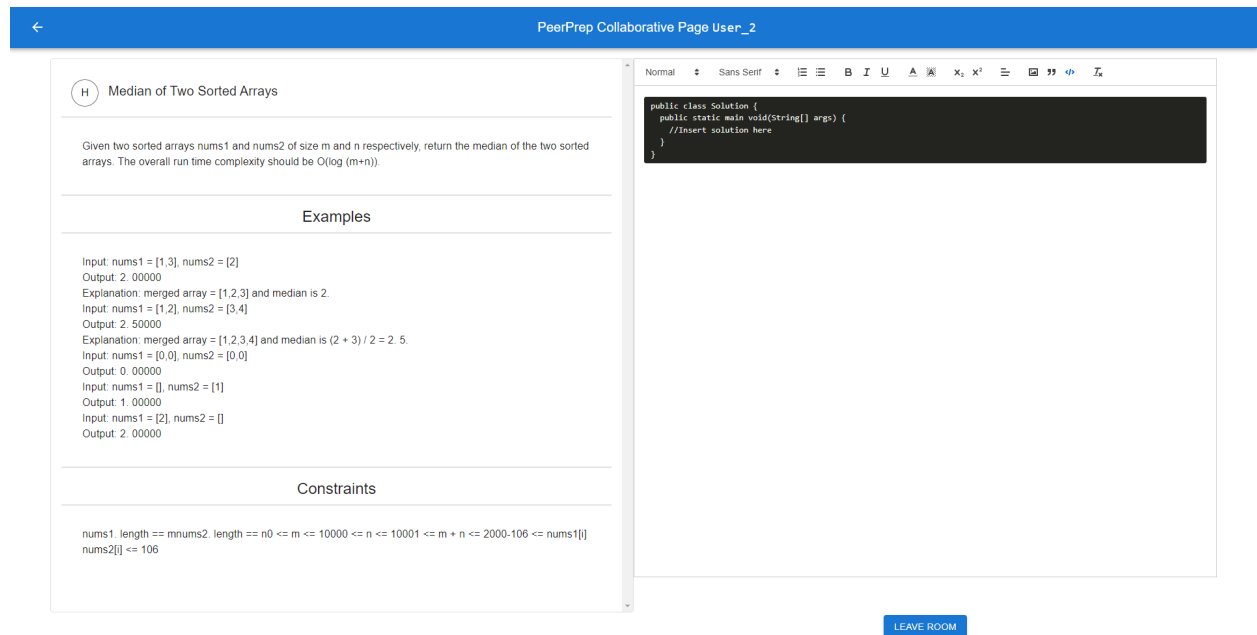
Diagram 24: Screenshot of the solo practice mode which does not have the chat messaging service

5.3.3 Collaboration Service

The collaboration service allows matched users to edit code concurrently in real-time.

A design decision that had to be made for the frontend was deciding on the UI through which users would collaborate with each other. Considering the time constraints of the project, we decided not to build an implementation from scratch, but build the functionality of PeerPrep upon an existing software or library. Our team chose Quill as it allows us to incorporate a text editor on the collaboration page which users can use to collaborate on code. Additionally, changes made to the Quill text editor can be transmitted in the form of deltas (instead of the full contents of the text editor) which allow for smaller data packets in transmission, increasing the speed of the service.
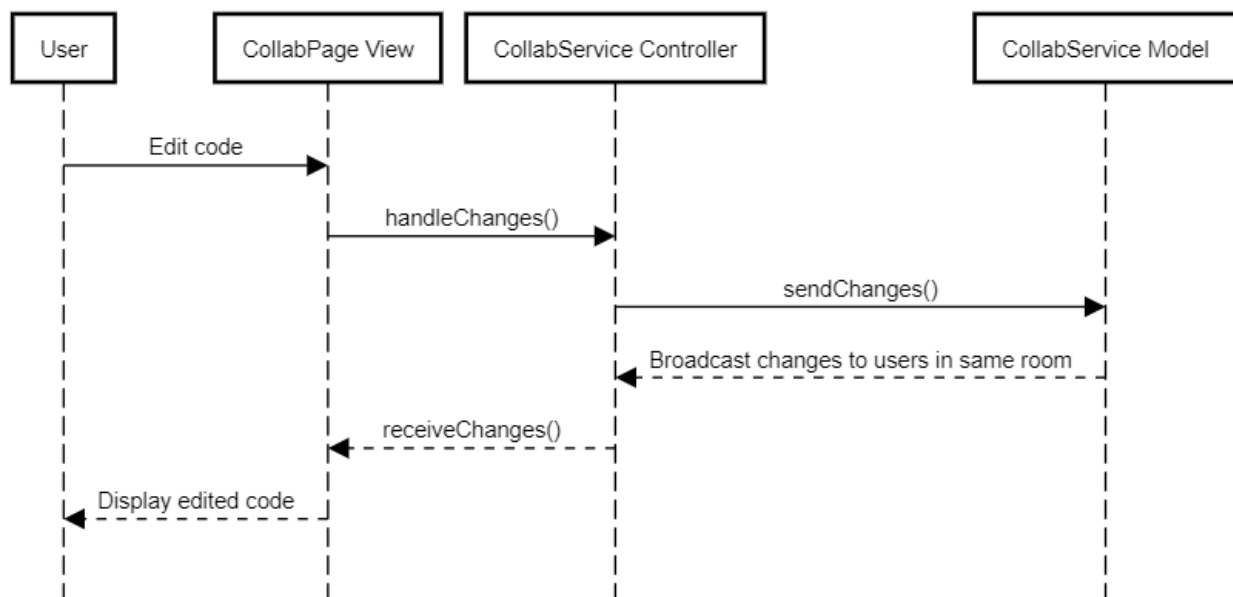
Diagram 25: Sequence diagram for a user editing code in the collaboration service

A document database was also implemented to save the contents of the text editor from each collaboration room at regular intervals. This way, if a user were to refresh their page or leave and return, the document could be retrieved from the database, allowing the user to continue working on their code as they had left it.
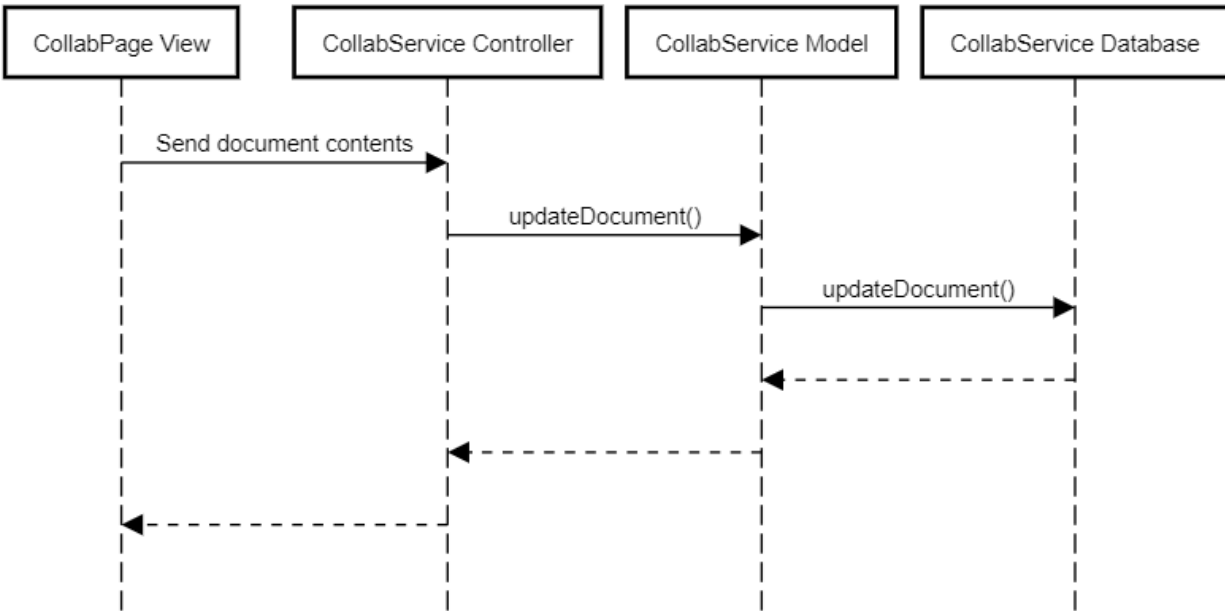
Diagram 26: Sequence diagram for the code editor saving its contents at regular intervals
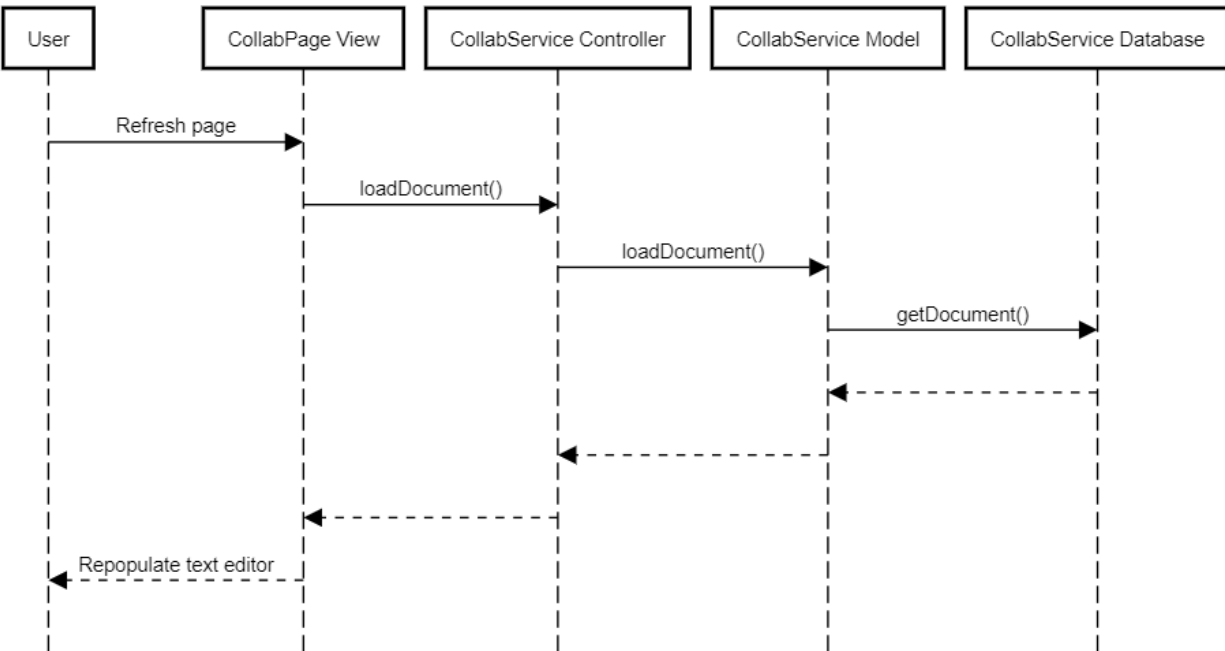
Diagram 27: Sequence diagram for a user refreshing the page and retrieving the code editor document

Chat Messaging Extension

The chat messaging extension also utilises the socket connection established between the server and the client. The chat messaging feature uses the same socket room ID, but sends a different type of message to their partner through the socket connection.

M Maximum Subarray

Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum. A subarray is a contiguous part of an array. Follow up: If you have figured out the O(n) solution, try coding another solution using the divide and conquer approach, which is more subtle.

Examples

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
Input: nums = [1]
Output: 1
Input: nums = [5,4,-1,7,8]
Output: 23

Constraints

1 <= nums. length <= 3 * 104-105 <= nums[i] <= 105

Normal | Sans Serif | B I U A A x₂ x²

Chat 💬

You: How should we solve this?
User_2: Wow, this is a tough one :/

Maybe we could use a brute force approach first?

LEAVE ROOM

Diagram 28: PeerPrep Collaborative Page with Chat Messaging on the right hand side

5.3.4 Question Service

The Question Service enables the retrieval of questions from our question database for users when doing coding questions alone or with another user. We found a repository on GitHub containing a substantial number of leetcode questions with the id, title, question, examples, and constraints provided. We then updated the questions in the question database with their respective difficulty levels before we integrate them into PeerPrep. We built the question service with Express and Node.js, and it works with the Frontend, User Service, Matching Service, and Collaboration Service to retrieve and display questions users have not attempted before from our question database.

Upon successful difficulty selection and user matching, if required, the user's question history is checked and PeerPrep will search for a question in our question database the user(s) has/have not attempted before. When the question has been found, the question will be displayed for the user(s). If the user(s) has/have completed all the questions in our question database/there is no common question not attempted by both users, users will be given a question they have attempted before instead.

Feature Implementation
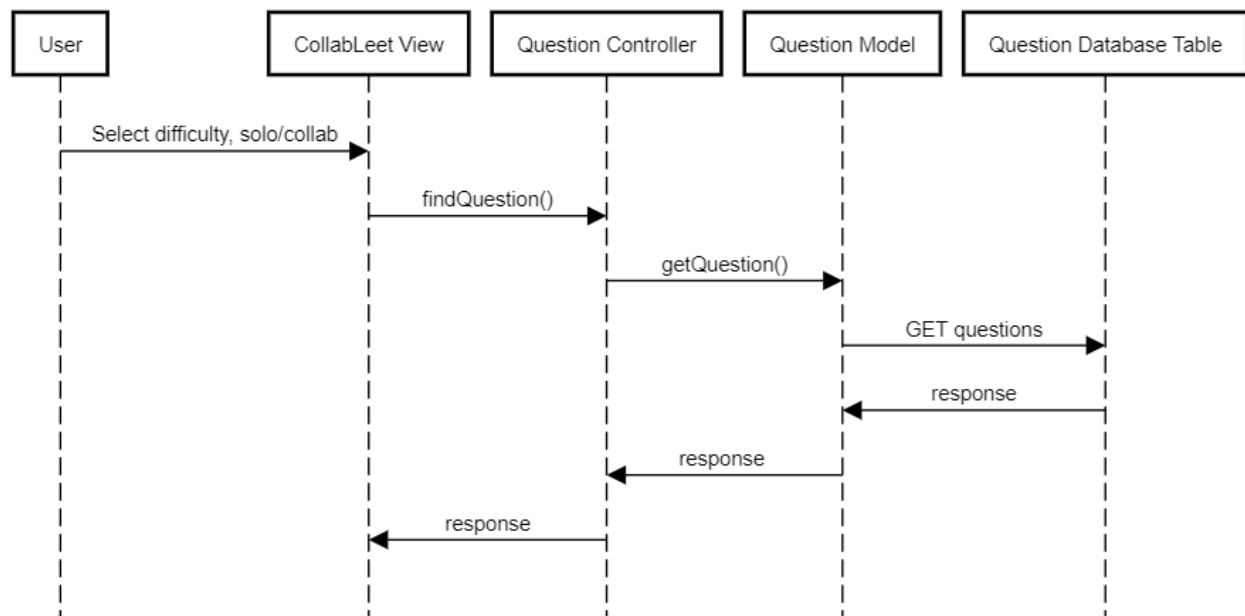
Diagram 29: Sequence Diagram demonstrating retrieving question for user

APIs

Question Service APIs		
Request	**API Route**	**Description**
GET	/api/question	Fetches the question database
POST	/api/question	Maps questions attempted for frontend display

5.3.5 History Extension

The History Extension enables users to see what is the breakdown of the difficulty of all the questions they have attempted as well as the full list of questions attempted. It also allows users to attempt new questions from the question database and ensures that users do not see a question they have attempted again.

However if a user attempts PeerPrep alone or with a partner and they have completed all the questions in the question database, they will see questions they have attempted before in PeerPrep.

Upon leaving the room, it is taken that the user has attempted the question. Hence, the user's question history will be updated, and stored under "easy", "medium", or "hard" in the user database.

Feature Implementation
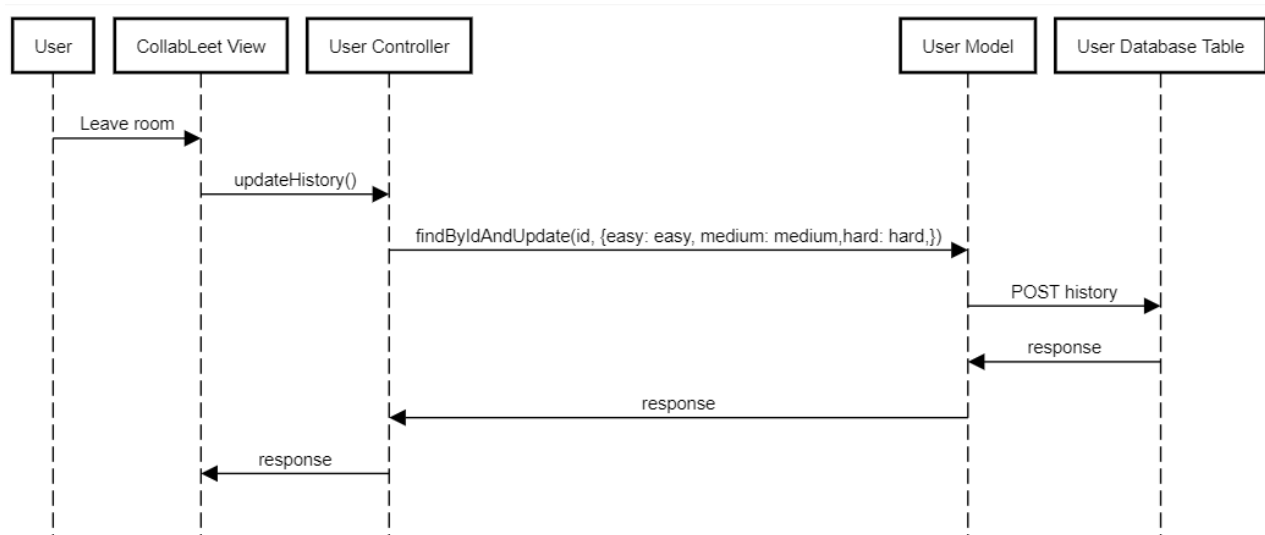
Diagram 30: Sequence Diagram demonstrating updating user question history

Upon the user entering the dashboard, the application will retrieve the questions the users have completed, categorised by their difficulty. The Pie Chart in the user's Dashboard will then be updated to visualise the breakdown of questions attempted by the user.
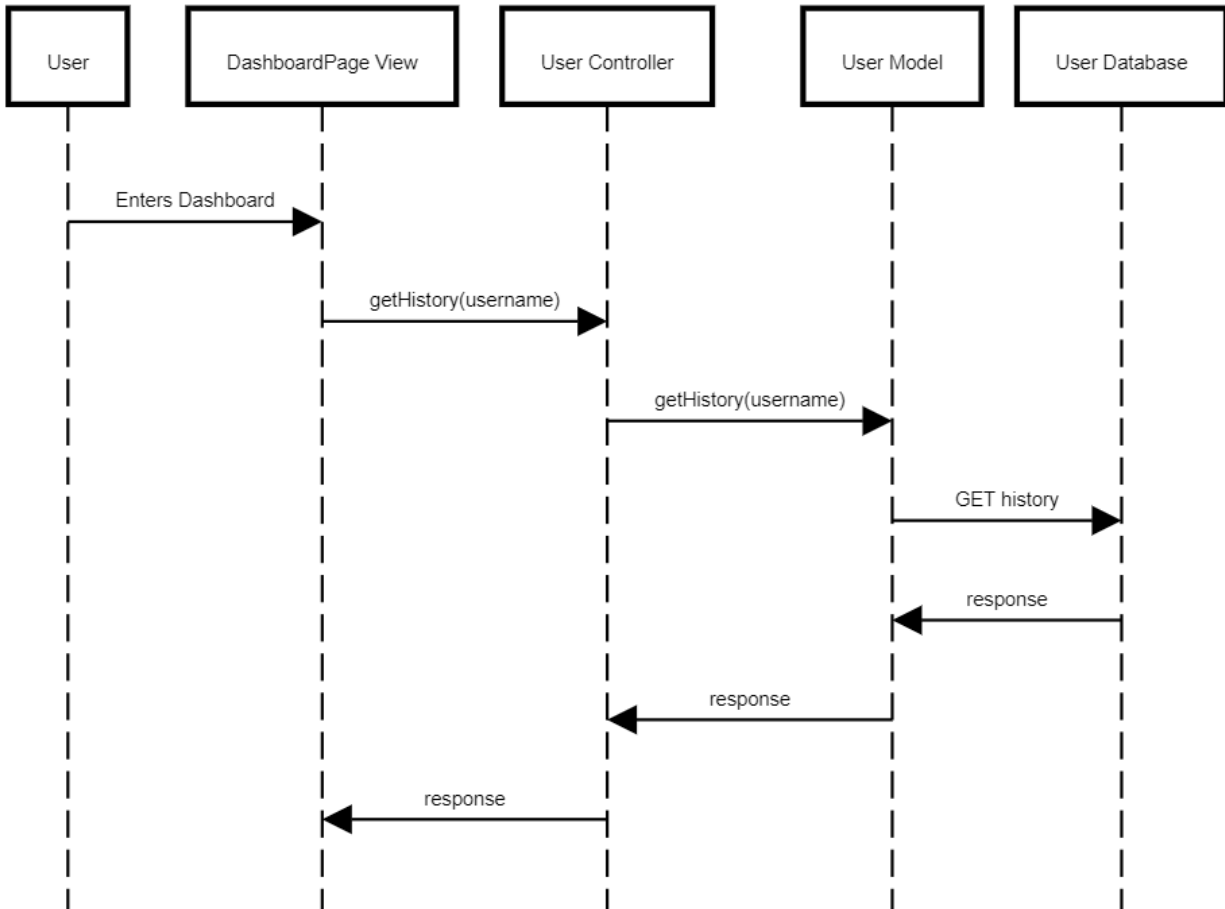
Diagram 31: Sequence Diagram demonstrating getting user question history breakdown
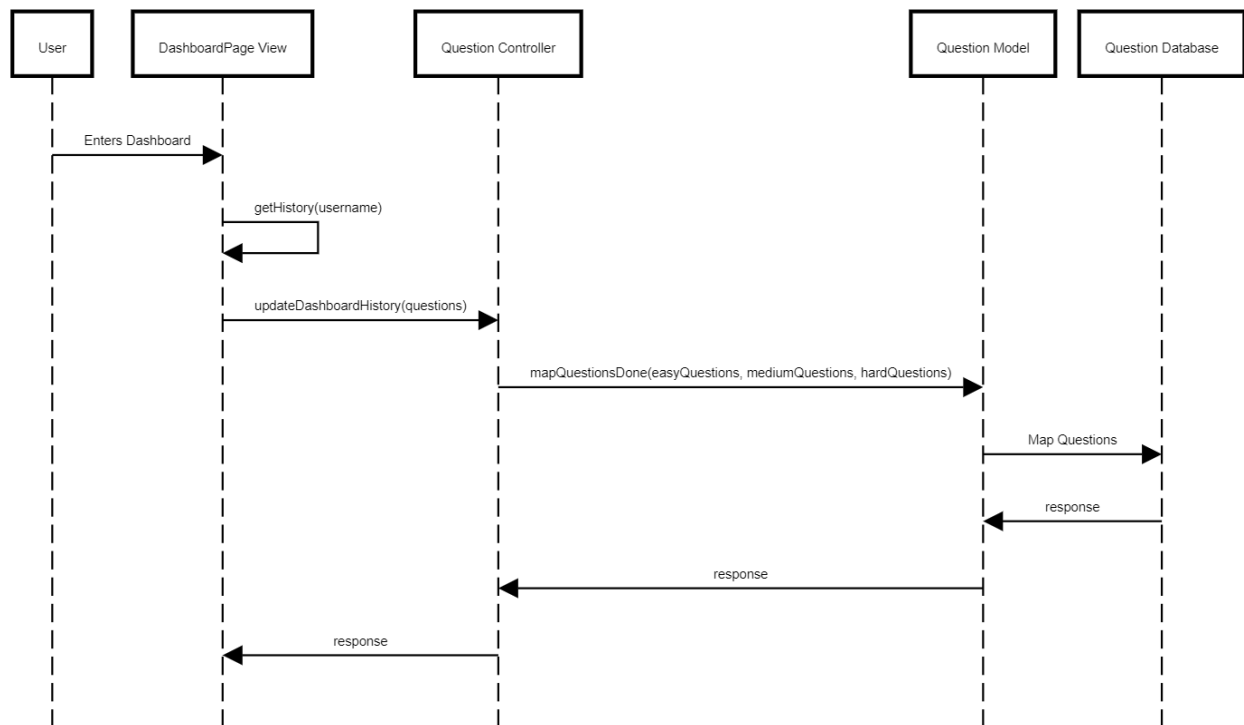
Diagram 32: Sequence Diagram demonstrating getting user question history list

5.3.6 Frontend Service

The Frontend Service provides the foundation for User Interface (UI) and User Experience(UX). The frontend is fully implemented using Material UI v5, as well as libraries such as Quill.

Here are some screenshots of our team's planning:
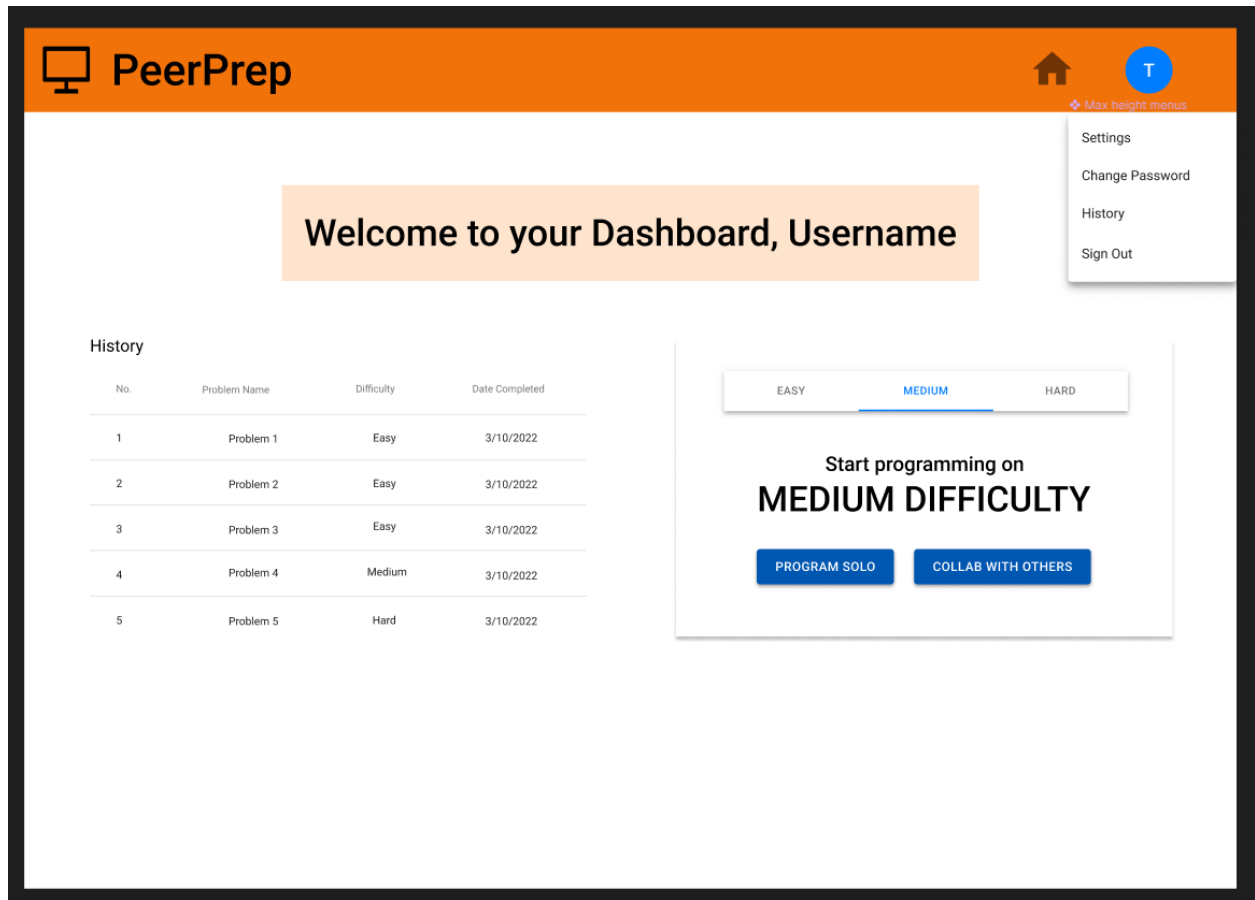
Diagram 33: First draft of Dashboard Page
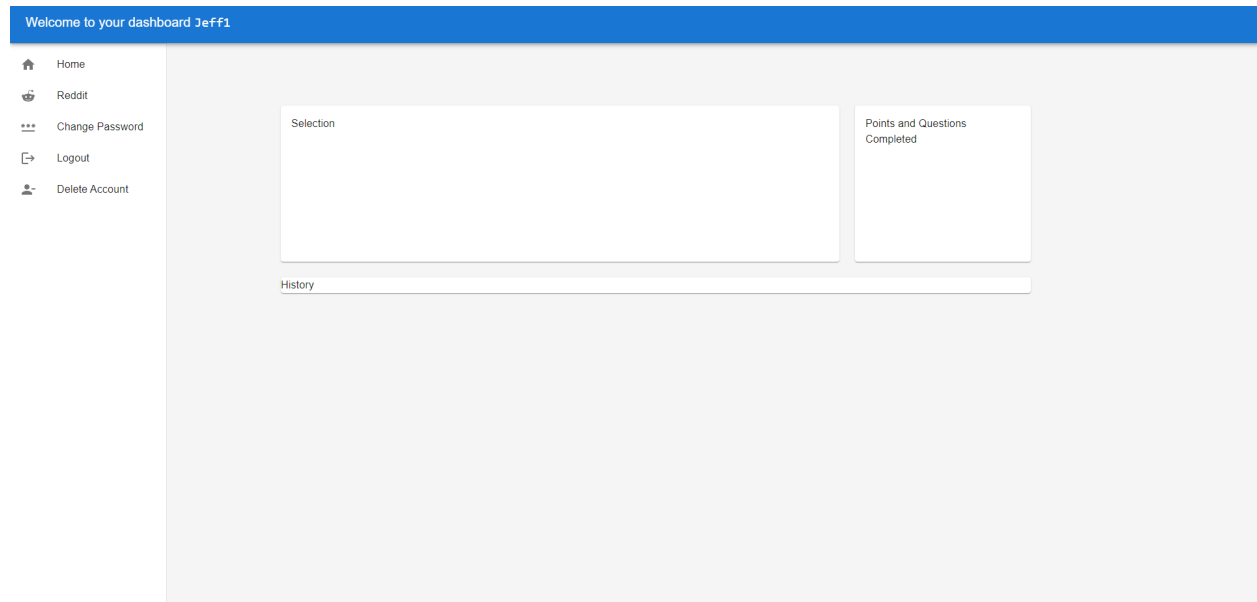
Diagram 34: Updated UI for Dashboard with History and Statistics
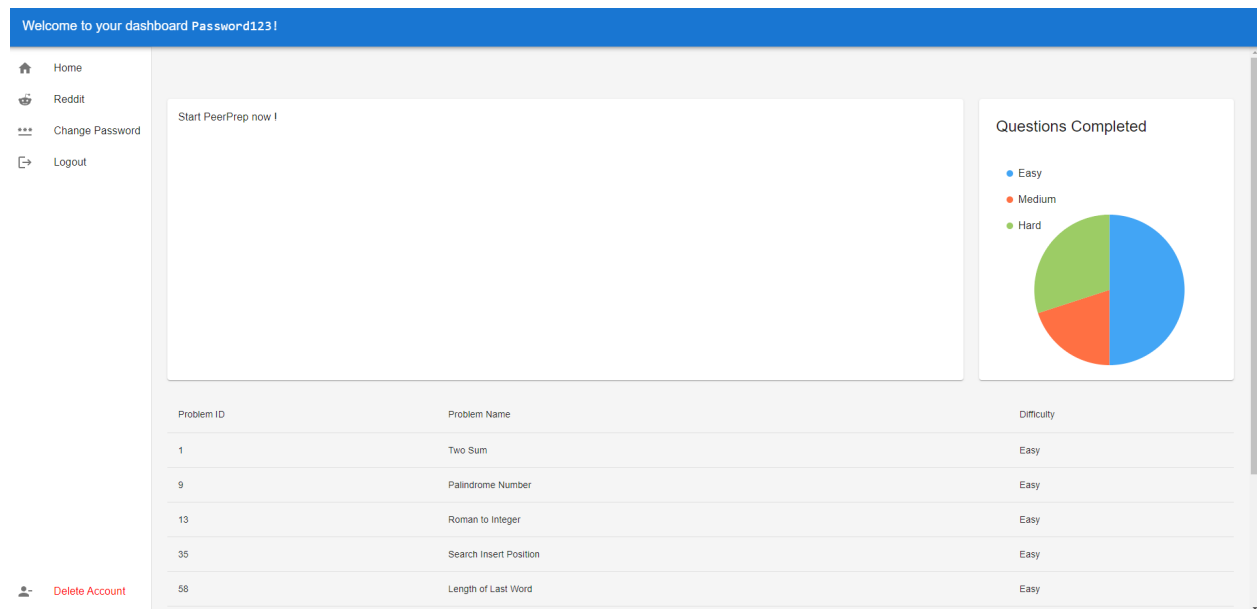
Diagram 35: Dashboard with populated History and Statistics
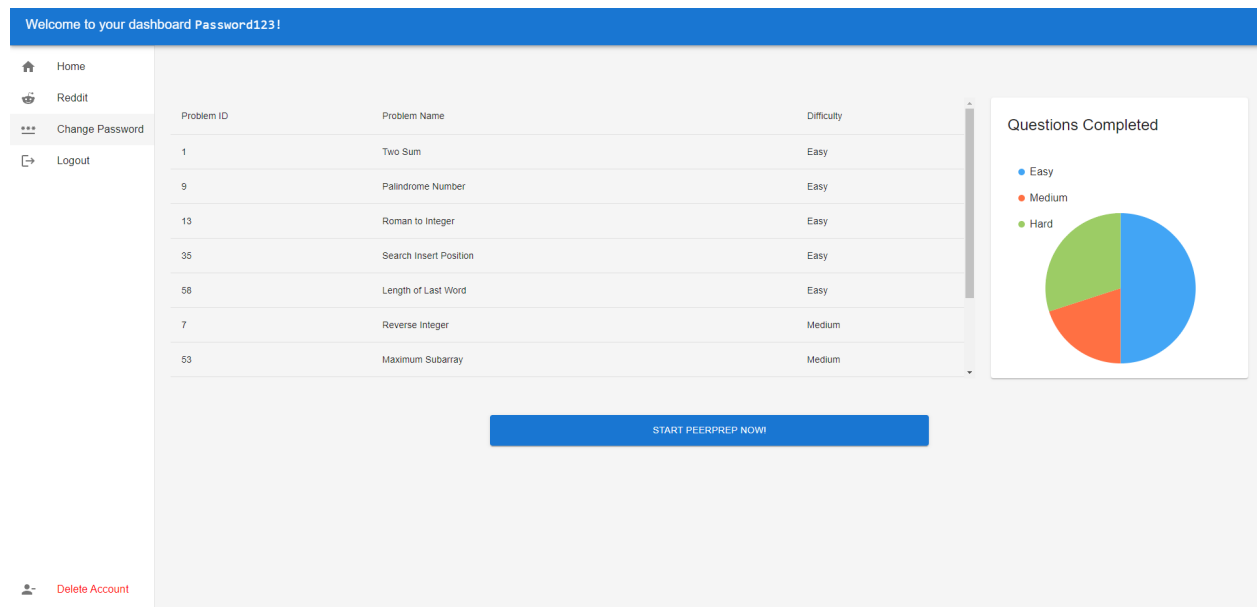
Diagram 36: Finalised Dashboard
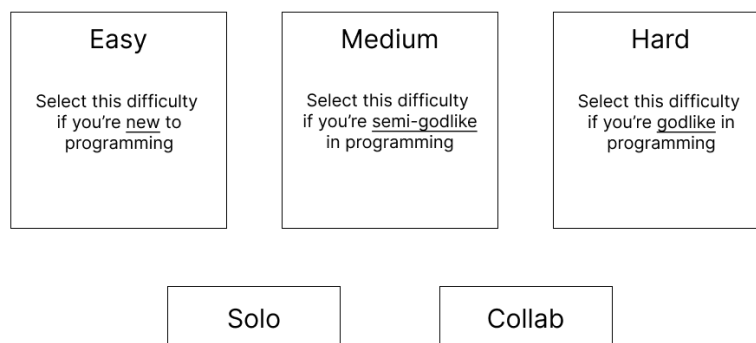
Diagram 37: First draft of Selection Page

Question:

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
You may assume that each input would have exactly one solution, and you may not use the same element twice.
You can return the answer in any order.

Collaboration Area

Back to Selection

Diagram 38: First draft of Collaboration Page, Planning Stages

Two Sums:

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
You may assume that each input would have exactly one solution, and you may not use the same element twice.
You can return the answer in any order.

Example 1:

Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we
return [0, 1].

Collaboration Area

Diagram 39: First Draft of Collaboration Page, Planning Stages

6. Suggestions for Improvements & Enhancements

This section highlights potential enhancements that can be made to our final PeerPrep product:

Improvements to the Code Editor

While Quill was a sound choice for a collaborative text editor between users, it lacks the functionality a user may expect from a code editor. Software like VSCode provides powerful shortcuts and important functionality like compiling and running code. While not in the scope of our project, improving or using a text editor more capable of handling code would have increased the usefulness of PeerPrep immensely, allowing users to run and test their code all within the Collaboration page.

Improvements to Authentication & Authorization

We are currently using an authentication and authorization middleware that we wrote ourselves, to check for the validity of the JWT before calling certain APIs. We can look to integrate OAuth into our application to ensure greater security. This protocol utilises SSL to ensure the data between web servers and browsers remain private. This SSL also uses industry-level cryptography protocols to keep data safe.

Usage of OAuth also allows users to log in using their google account, and this would make it easier for the users to signup and login into our application. Since users can verify their identity using 2FA (Gmail), it would also improve the security of our application.

Improvements to Frontend

The frontend design was implemented using Material UI v5, and although it has a substantial amount of components, enhancing User Experience posed as a challenge for our team. Customization is difficult to be implemented as each time our team decides on a new User Interface decision, our team would have to constantly override the styles of each component. As this issue scales bigger, our team realised that we could be unintentionally breaking Don't Repeat Yourself (DRY) software design principles. One improvement our team has identified would be to use other alternative design frameworks on top of Material UI. Some examples include: KendoUI, Chakra, Ant Design.

One other aspect we could have done better would be to account for different screen sizes. For the Collaboration page, the frontend implemented was 3 grid columns, but the heights of the grid items were not in sync. One way to solve this would be to account for all the paddings, margins and pixels, to divide the screen accordingly.

Additional "Forgot Password" feature

Currently, our application does not allow users to change their password if they forget their password. The user would have to create a new account with a new email since all accounts would require a unique email. All their progress in our application will also be gone. With the adoption of the "Forgot Password" feature, this pain point will be resolved.

7. Reflections and Learning Points

Alfred

Doing up PeerPrep has allowed me to be exposed to the MERN stack. My experience during my internship is only with NodeJS as a backend developer, but this time I'm glad to have taken a more front-end design approach.

Simple research online shows that to implement one feature in our project, we could import many different libraries to get the job done. For example, during my venture into designing a better UI, I wanted to implement a card carousel slider for the selection page. However, the libraries that I sourced out are unable to be integrated within our MUIv5 as the library uses MUIv4. A good practice moving forward would be to research the different ways to implement a feature, and what existing libraries are out there to be used.

Proper planning of the software development design and architecture is so important, our team discussed thoroughly how each service will be structured, and it helped us progress through this project much smoother.

Throughout the project, I am extremely grateful to my teammates. This semester, due to a heavy workload, I've delayed certain deliverables that I should have submitted. Despite that, my team continued to encourage me and it helped me a lot. I've learned more about myself, and areas to improve in terms of time management, and stress management. Thanks, guys!

Benjamin

In this project, I learned about the importance of the planning and designing process. I learned how to weigh the benefits and shortcomings of each technology and ultimately, choose the most suitable ones based on our requirements.

This project and module also exposed me to frameworks and technologies that I had no prior experience with. Hence, the time getting familiar with them was very fruitful as I felt like I finished this project with a lot more theoretical and practical experience.

I learned more about the MERN stack, as well as industry security practices. I felt that it was very fruitful to apply them after learning about the OTOT tasks or through exploring the internet.

After having more experience under my belt, I found myself thinking back to the earlier design phase, and wishing that I learned about that alternative technology before going ahead with the current implementation. However, I see this as a learning experience and I am more equipped to handle future projects

Brandon

Throughout the course of this project, I learned about the importance of spending time learning about and choosing the right technologies for a project. I learned (sometimes after already starting to work with a particular technology) that another technology or framework could make my current task easier to implement and improve the final product. However, considering the time constraints of each project that I work on in the future, this should be balanced with getting work done as there will always be another technology that is better in some way but I could end up spending too much time just choosing a technology to work with.

Through this project, I picked up useful technologies like the MERN stack and Sockets.io. It was helpful to learn to put these into practice after reading about them online or seeing other coursemates create web applications with them during their internships.

Dickson

After going through the guided planning and design of PeerPrep, I am now able to better understand what goes behind the decisions when designing a software application. I learned how the various technologies and frameworks have their pros and cons and it is important to weigh the pros and cons of each technology and framework to determine the most suitable technology and framework to use for the software application.

My experience in a previous internship involved Golang for backend web development and a bit of React for frontend web development. I am glad I was able to do backend web development which I was stronger at while exposing myself more to frontend web development. I also learned about the MERN stack which is a Javascript stack used for easier and faster deployment of a full-stack web application comprising 4 technologies namely, MongoDB, Express, React, and Node.js. Given more time, I would research and explore more on the frontend web development on how the user interface and user experience can be better improved.

After this fruitful learning experience, I am more confident in making better decisions for subsequent software projects. In hindsight, I feel that we could have better weighed all the different technologies we have learned from class and made the decision before starting on the project. Nonetheless, this has been a great learning experience for software engineering principles and patterns.

8. References

Jethva, H. (2022, September 26). *NIST Password Guidelines Requirements for 2022/2023 Best Practices*. Cloud Infrastructure Services. Retrieved November 9, 2022, from https://cloudinfrastructureservices.co.uk/nist-password-guidelines-requirements-best-practices/

Mindanao, K. (2022, October 17). *NIST Password Guidelines 2022: 9 Rules to Follow*. Intelligent Technical Solutions. Retrieved November 9, 2022, from https://www.itsasap.com/blog/nist-password-guidelines

Nielsen, J. (2020, November 15). *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group. Retrieved November 9, 2022, from https://www.nngroup.com/articles/ten-usability-heuristics/

9. Appendix

9.1 Detailed Architecture Diagram
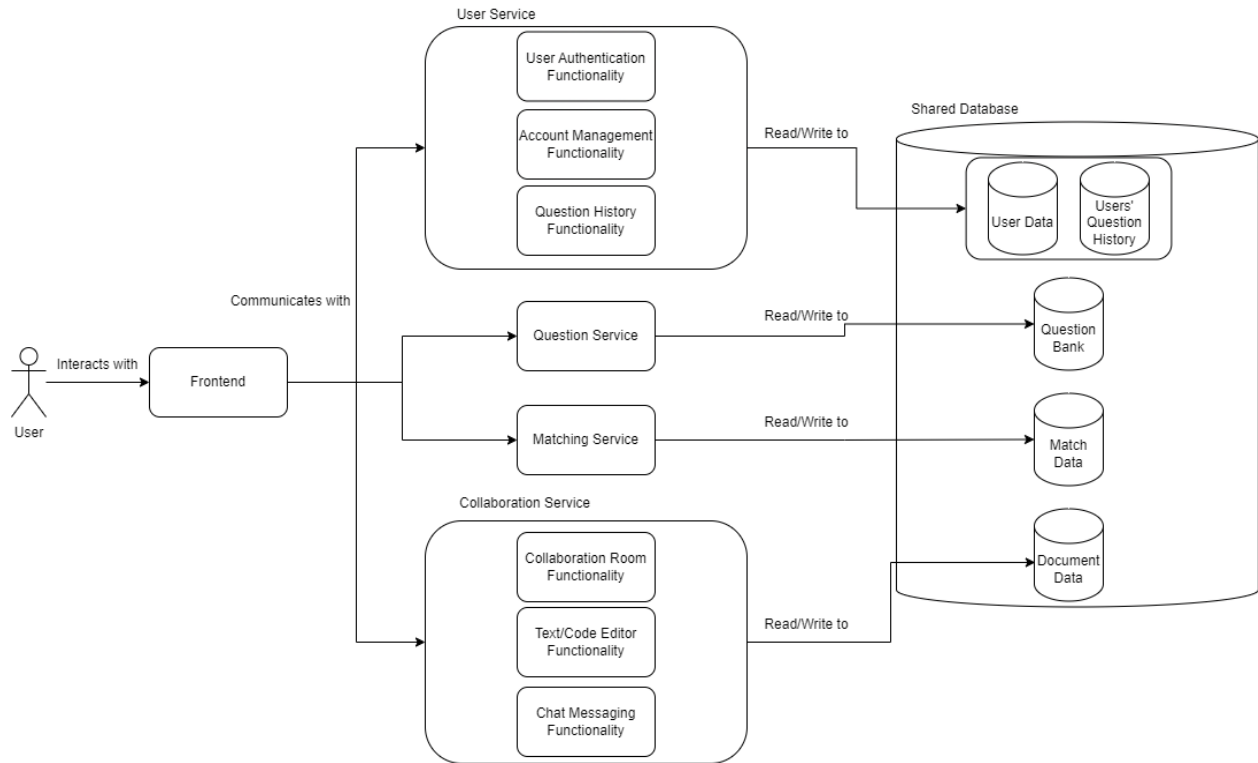
Diagram 40: Detailed Architecture Diagram

9.2 Application Screenshots

Diagram 41: User visits PeerPrep Sign Up page

Diagram 42: User visits PeerPrep Login page

Problem ID	Problem Name	Difficulty
1	Two Sum	Easy
9	Palindrome Number	Easy
53	Maximum Subarray	Medium
7	Reverse Integer	Medium
122	Best Time to Buy and Sell Stock II	Medium
629	K Inverse Pairs Array	Medium
10	Regular Expression Matching	Hard

Home
Reddit
Change Password
Logout
Delete Account

Questions Completed

- Easy
- Medium
- Hard

START PEERPREP NOW!

Diagram 43: On logging in, user is directed to PeerPrep Dashboard page

PeerPrep Selection Page P@ssw0rd1

Easy
Select this difficulty if you are new to programming!

Medium
Select this difficulty if you are pro in programming!

Hard
Select this difficulty if you are god in programming!

SOLO   COLLAB

Diagram 44: User is directed to PeerPrep Selection page

Roman to Integer

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. For example, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II. Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:Given a roman numeral, convert it to an integer.

Examples

Symbol Value
I 1
V 5
X 10
L 50
C 100
D 500
M 1000Input: s = "III"
Output: 3
Input: s = "IV"
Output: 4
Input: s = "IX"
Output: 9
Input: s = "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.
Input: s = "MCMXCIV"
Output: 1994

```
public class Solution {
    public static main void(String[] args) {
        //Insert solution here
    }
}
```

LEAVE ROOM

Diagram 45: User attempts "Solo Practice mode"

Easy
Select this difficulty if you are new to programming!

Medium
Select this difficulty if you are pro in programming!

Hard
Select this difficulty if you are god in programming!

SOLO   COLLAB

Matching In Progress
28s
CANCEL TIMER
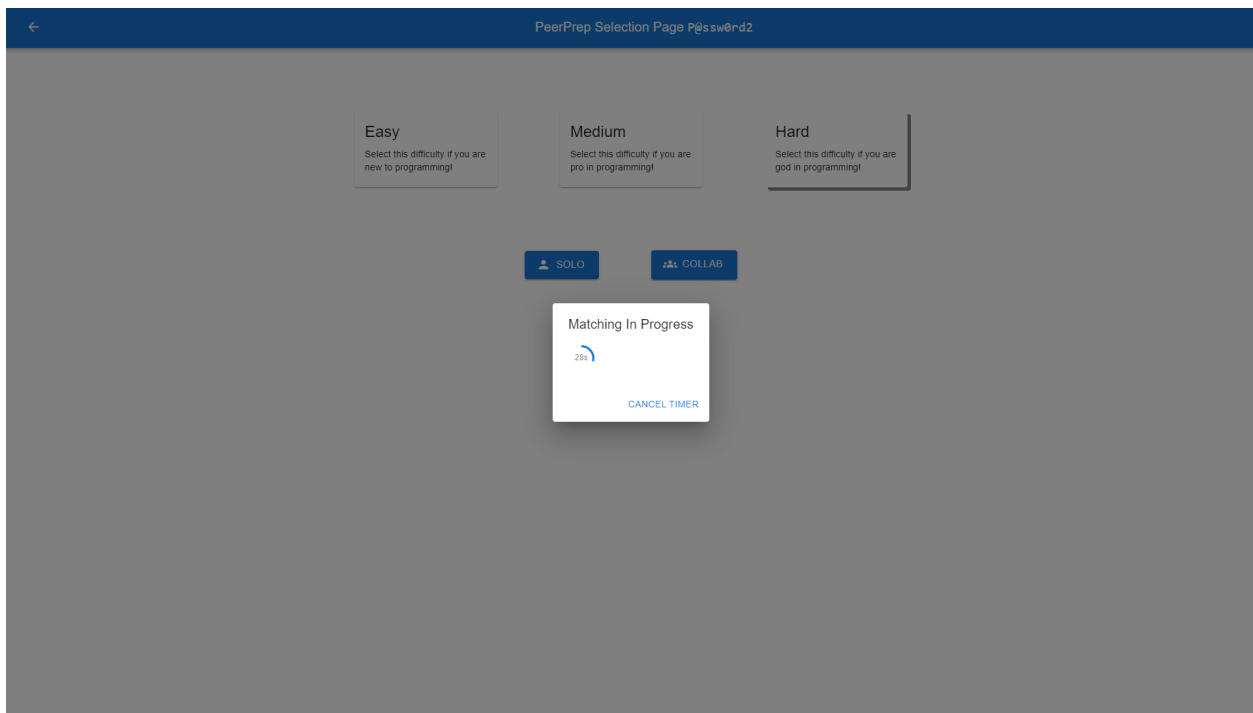
Diagram 46: On selecting the "Collab" option, the user is shown a timer while waiting for a match

Median of Two Sorted Arrays

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).

Examples

Input: nums1 = [1,3], nums2 = [2]
Output: 2. 00000
Explanation: merged array = [1,2,3] and median is 2.
Input: nums1 = [1,2], nums2 = [3,4]
Output: 2. 50000
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2. 5.
Input: nums1 = [0,0], nums2 = [0,0]
Output: 0. 00000
Input: nums1 = [], nums2 = [1]
Output: 1. 00000
Input: nums1 = [2], nums2 = []
Output: 2. 00000

Constraints

nums1. length == mnums2. length == n0 <= m <= 10000 <= n <= 10001 <= m + n <= 2000-106 <= nums1[i]
nums2[i] <= 106

```
public class Solution {
    public static main void(String[] args) {
        //Insert solution here
    }
}
```

Chat

P@ssw0rd1 connected

LEAVE ROOM

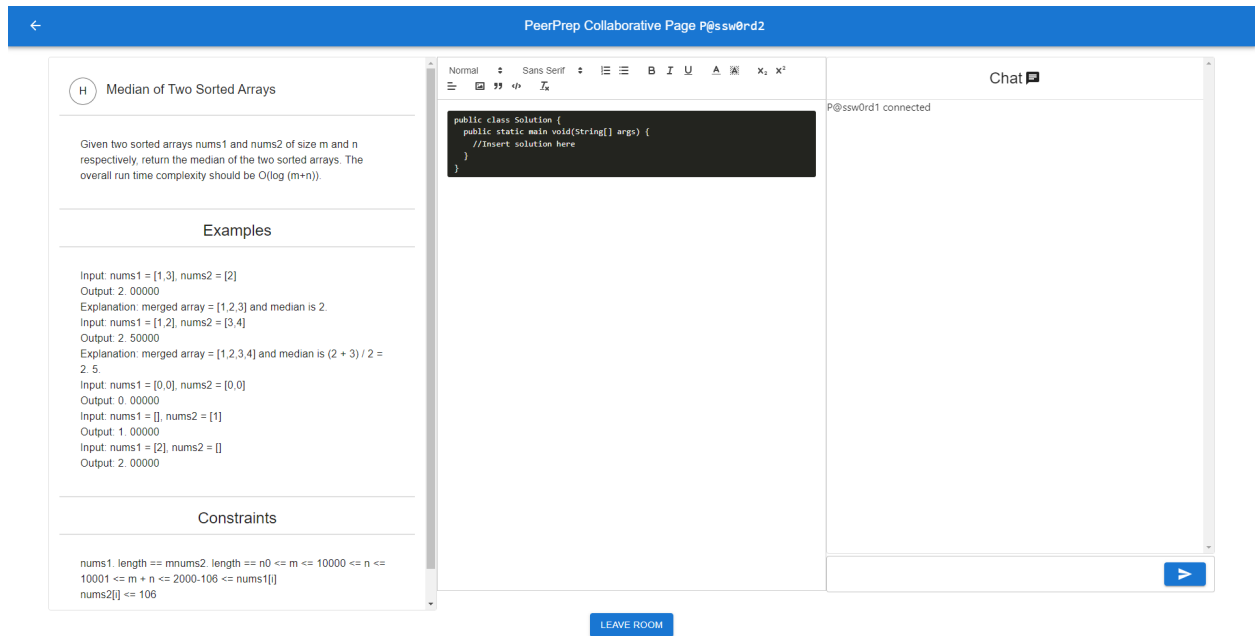*Diagram 47: On a successful match, user is directed to PeerPrep Collaborative page with the question, text editor, and chat messaging service*

Welcome to your dashboard P@ssw0rd1

Home
Reddit
Change Password
Logout
Delete Account

Problem ID	Problem Name	Difficulty
1	Two Sum	Easy
9	Palindrome Number	Easy
13	Roman to Integer	Easy
53	Maximum Subarray	Medium
7	Reverse Integer	Medium
122	Best Time to Buy and Sell Stock II	Medium
629	K Inverse Pairs Array	Medium

Questions Completed

- Easy
- Medium
- Hard

Change your password

New Password

DONE

START PEERPREP NOW!

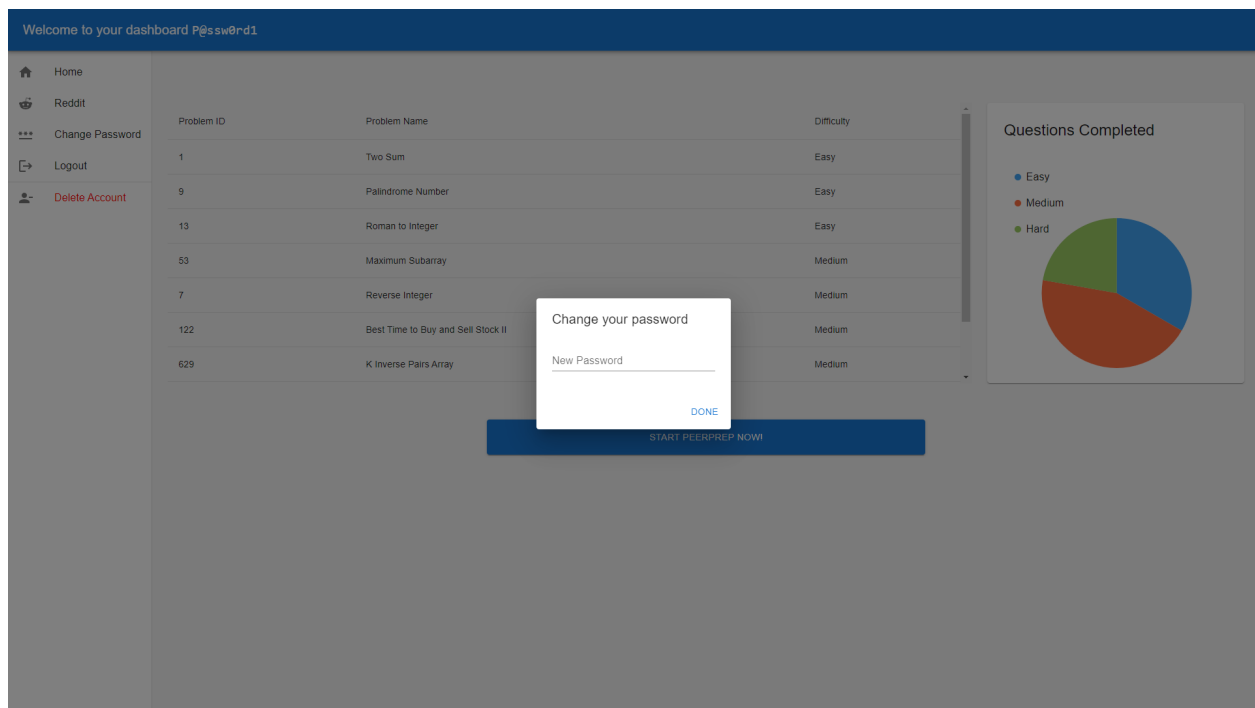*Diagram 48: User changes password*
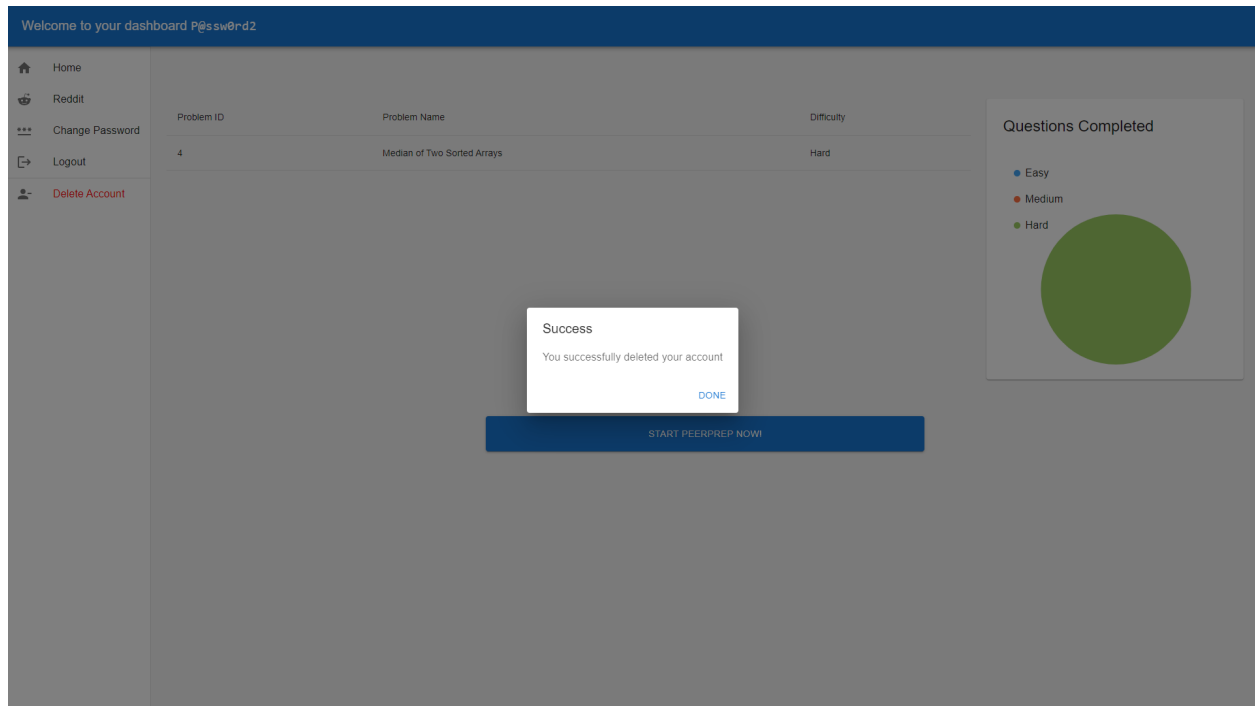
Diagram 49: User deletes account

9.3 Database Schema

User
- username: Unique String
- password: String
- easy: String[]
- medium: String[]
- hard: String[]

Match
- isPendingMatch: Boolean
- difficulty: String
- user1: Unique String
- user2: Unique String
- user1SocketId: Unique String
- user2SocketId: Unique String
- collabRoomSocketId: Unique String

Document
- id: String
- data: Object

Token
- token: Unique String
- expireAfterSeconds: Numbers

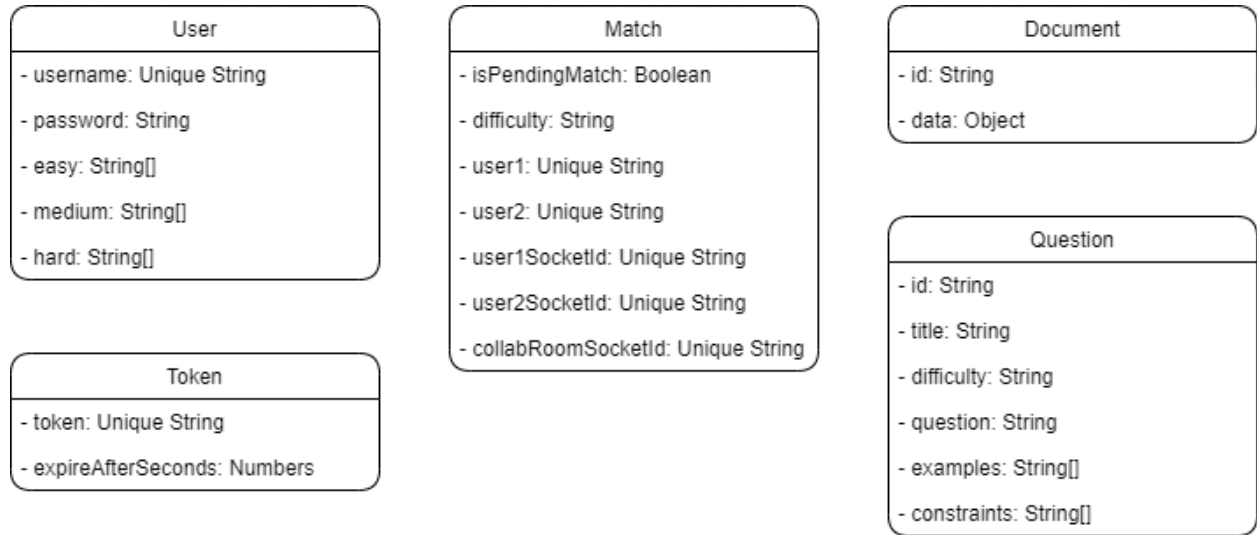| Question |
|---|
| - id: String |
| - title: String |
| - difficulty: String |
| - question: String |
| - examples: String[] |
| - constraints: String[] |

Diagram 50: Database Schema