



CS3219 Project Report

Group 54

Name	Student No	Contact
Ang Yong Liang	A0217430R	yongliangang@u.nus.edu
Jerome Wong Jia Jin	A0217861B	e0543897@u.nus.edu
Lai Yu Heem	A0201317X	yu.heem@u.nus.edu
Tan De Yi	A0200403H	e0407384@u.nus.edu

Table of Contents

1 Introduction	3
1.1 Background	3
1.2 Purpose	3
1.3 User Journey	3
2 Contributions	4
2.1 Contributions Table	4
3 Requirements Specification	5
3.1 Functional Requirements	5
3.1.1 Auth Service	5
3.1.2 User Service	5
3.1.3 Matching Service	6
3.1.4 Question Service	6
3.1.5 History Service	6
3.1.6 Collaboration Service	6
3.1.7 Communication Service	7
3.2 Non-functional Requirements	8
3.2.1 Availability Attribute	8
3.2.2 Reliability Attribute	8
3.2.3 Security Attribute	8
3.2.4 Usability Attribute	9
3.2.5 Scalability Attribute	9
4 System Architecture	10
4.1 Architecture Diagram	10
4.2 Tech Stack	10
4.3 Architecture Decisions	11
4.3.1 Microservices over Monolithic Design	11
4.3.2 Database per Service	11
4.3.3 Redis Caching	12
4.4 Design Principles & Patterns	12
4.4.1 Facade Pattern	12
4.4.2 Publisher-Subscriber Pattern	12
5 Microservices Implementation	13
5.1 Gateway Service	14

5.2 Auth Service	15
5.3 User Service	17
5.4 Matching Service	20
5.5 Question Service	21
5.6 Collaboration Service	23
5.7 Communication Service	24
5.8 History Service	24
5.9 Judge Service	27
6 User Interface	28
6.1 Landing Page	28
6.2 Signup Page	28
6.3 Login Page	30
6.4 Main Page	31
6.5 Collaboration Page	32
6.6 Question Search	35
7 Milestones	36
7.1 Gantt Chart	36
7.2 CI/CD	40
8 Future Enhancements	41
8.1 Enhance matching criteria	41
8.2 Allowing match to be more than 2 users	41
8.3 Providing solution code	41
8.4 Adding review to each session	41
8.5 End-to-end and Load testing	42
8.6 Persistency of collaboration data	42
9 Reflections	43

1 Introduction

1.1 Background

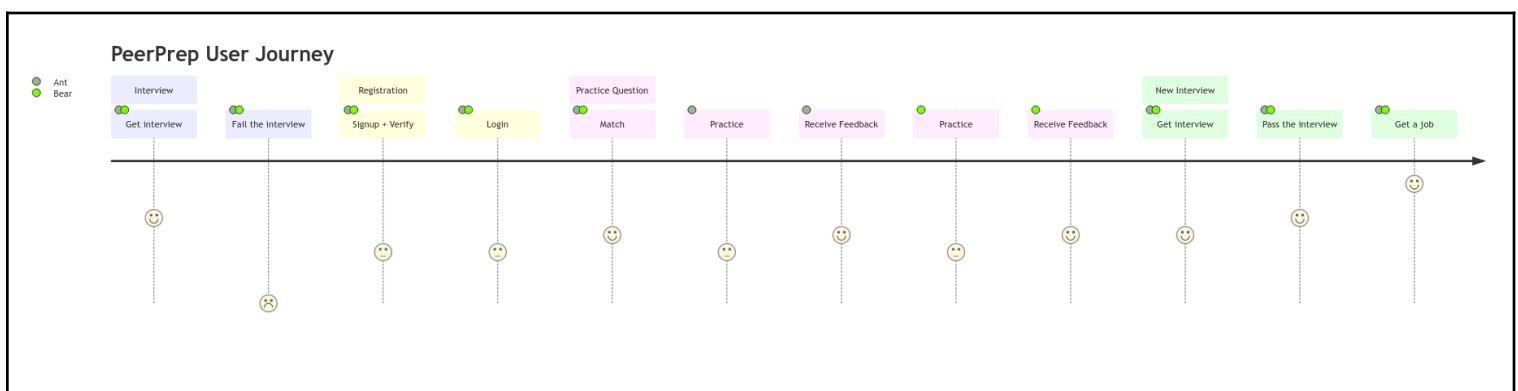
Increasingly, students face challenging technical interviews when applying for internships or jobs which many have difficulty dealing with. Common issues range from a lack of communication skills to articulate their thought process out loud to an outright inability to understand and solve the given problems under time constraints. Moreover, grinding practice questions alone may become tedious and monotonous.

1.2 Purpose

PeerPrep targets NUS students who want to improve their technical interviewing skills under conditions that emulate an actual interview. Our app provides students with a platform to practice communicating their thought process as they work through a technically challenging problem with a peer. Students get paired based on a chosen question difficulty, where they will alternate roles of the interviewer and interviewee. Our goals are to help students land their dream internship or job and build their confidence for interviews.

1.3 User Journey

The diagram below depicts the user journey for two students Ant and Bear. Ant and Bear are students that have failed a technical interview before stumbling on PeerPrep. Both of them use the platform, where they match each other. Each one of them takes turns being the interviewer and interviewee with two separate questions and provide each other with feedback on how to improve. They get better at their interviews and finally get hired. Yay!



2 Contributions

2.1 Contributions Table

Members	Contributions
Ang Yong Liang	Frontend, Gateway, Auth-service, User-service, Matching-service, Question-service, Collaboration-service, Communication-service, History-service, Deployment, Tests, Report
Jerome Wong Jia Jin	Frontend, Gateway, Auth-service, User-service, Matching-service, Question-service, Collaboration-service, Communication-service, History-service, Deployment, Tests, Report
Lai Yu Heem	Frontend, Gateway, Auth-service, User-service, Matching-service, Question-service, Collaboration-service, Communication-service, History-service, Deployment, Tests, Report
Tan De Yi	Frontend, Gateway, Auth-service, User-service, Matching-service, Question-service, Collaboration-service, Communication-service, History-service, Deployment, Tests, Report

3 Requirements Specification

3.1 Functional Requirements

Functional requirements gathered during the ideation phase have been grouped by microservice and recorded in a table below. The priority was determined by how critical each requirement would be to the system's features of achieving our purpose.

3.1.1 Auth Service

S/N	Requirements	Priority
F-A-1	The auth service will allow a user to log into their accounts by entering the email and password they signed up with	High
F-A-2	The auth service will allow a user to log out of their account	High
F-A-3	The auth service will verify a user's access token before accessing protected resources	High
F-A-4	The auth service will allow a user to renew their access token using a refresh token	High
F-A-5	The auth service will allow a user to choose whether to persist their login session	High

3.1.2 User Service

S/N	Requirements	Priority
F-U-1	The user service will allow a user to create an account with an email address and a password	High
F-U-2	The user service will send a confirmation email when a new user signs up and creates an account	High
F-U-3	The user service should ensure that every account created has a unique email address	High
F-U-4	The user service will allow a user to reset their password via their email address	High
F-U-5	The user service will allow a user to delete their account	Medium
F-U-6	The user service will allow a user to update their current password to a new password if they remember their current password	Medium

F-U-7	The user service will allow a user to view their own settings	Low
-------	---	-----

3.1.3 Matching Service

S/N	Requirements	Priority
F-M-1	The matching service will allow a user to select a difficulty level (Easy, Medium, Hard, Random) of the question they wish to attempt	High
F-M-2	The matching service will match two users waiting for a question of the same level of difficulty and allocate them to the same room	High
F-M-3	The matching service will notify a user that no match is available if a match cannot be found	Medium
F-M-4	The matching service should allow a user to cancel the match search at any time during the search	Medium
F-M-5	The matching service should allow a user to leave a room once matched	Medium

3.1.4 Question Service

S/N	Requirements	Priority
F-Q-1	The question service will generate a random question from a question bank for the user based on their selected level of difficulty	High
F-Q-2	The question service should provide an interface for users to view questions in the question pool	High

3.1.5 History Service

S/N	Functional Requirements	Priority
F-H-1	The history service will show a user their past submissions for a question	High
F-H-2	The history service will provide a user interface for users to view the details of their submissions	High

3.1.6 Collaboration Service

S/N	Requirements	Priority
-----	--------------	----------

F-CL-1	The collaboration service will allow a user to type code in the provided editor	High
F-CL-2	The collaboration service will allow a user to view shared updates in the provided code editor during the coding session	High
F-CL-3	The collaboration service will allow a user to leave the coding session	Medium
F-CL-4	The collaboration service will allow a user to sketch on a collaborative whiteboard	High
F-CL-5	The system should show user what the other user is sketching	High
F-CL-6	The system should allow users to compile and run their own solution	High
F-CL-7	The system should allow user to specify input before their run their code	Medium
F-CL-8	The system should allow the roles of interviewer/interviewee to be swapped the first interviewee submit their solution	High

3.1.7 Communication Service

S/N	Requirements	Priority
F-CM-1	The communication service will allow a user to view other user messages during the coding session	High
F-CM-2	The communication service will allow a user to chat over video and voice call during the coding session	Medium
F-CM-3	The communication service will allow a user to toggle turning on and off their video	Medium
F-CM-4	The communication service will allow a user to toggle turning on and off their audio	Medium

3.2 Non-functional Requirements

Non-functional requirements have been grouped according to quality attributes covered in the lectures.

3.2.1 Availability Attribute

S/N	Requirements	Priority
NF-A-1	The system should not take more than 10 seconds to load when user send a request	High
NF-A-2	The system should bring the user back to main page after failing to find a match for the user within 30 seconds	High

3.2.2 Reliability Attribute

S/N	Requirements	Priority
NF-R-1	The system should ensure the data is being saved properly when a network error occurred	High

3.2.3 Security Attribute

S/N	Requirements	Priority
NF-SE-1	The system should revoke access tokens that were expired	High
NF-SE-2	The system should only generate access tokens with a lifespan of 15 minutes	High
NF-SE-3	The system should not persistently store any user's access token/refresh token	High
NF-SE-4	The system should only provide the client with a refresh token using httpOnly secure cookie	High
NF-SE-5	Users' passwords should be hashed and salted before storing in the DB	High
NF-SE-6	User's email should be restricted to only contain "u.nus.edu" within the domain.	High
NF-SE-7	Users' password length should be within 8 to 16 characters	High

NF-SE-8	The user should not be able to login if they are not email verified	High
NF-SE-9	Backend microservices are only accessible via the gateway by authenticated users through the frontend	High

3.2.4 Usability Attribute

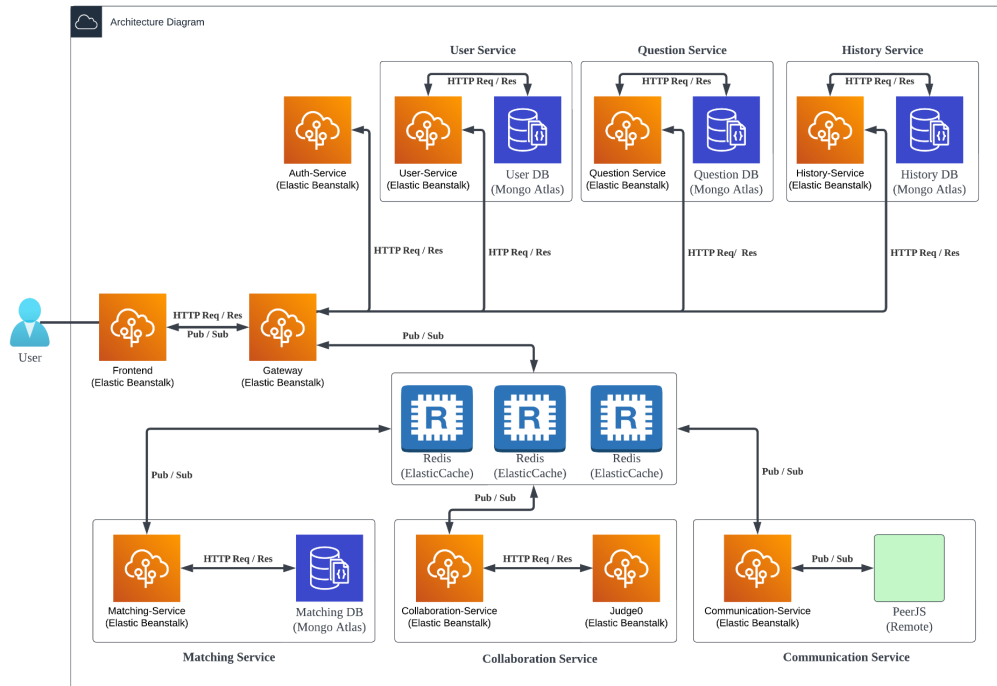
S/N	Requirements	Priority
NF-U-1	The system should be able to persist the login session after page refreshes.	High
NF-U-2	The system should only record compilable submissions	Medium

3.2.5 Scalability Attribute

S/N	Requirements	Priority
NF-SC-1	The system should be able to scale up or down to match demands	High

4 System Architecture

4.1 Architecture Diagram



4.2 Tech Stack

Component	Technologies
Frontend	ReactJS
Backend	Express.js, Node.js
Database	MongoDB
Deployment	AWS Elastic Beanstalk, AWS ElasticCache
Pub-Sub Messaging	Socket.IO, Redis
Cloud Providers	AWS, MongoDB Atlas, PeerJS
CI/CD	GitHub Actions
Orchestration Service	Docker Compose
Project Management	GitHub Issues

4.3 Architecture Decisions

4.3.1 Microservices over Monolithic Design

Our team has decided to use Microservice architecture for the project, we have a frontend service connected to an API gateway acting as the single entry point into the system. The API gateway is connected to the other 7 services and each of the services are running on different domains (AWS) / IP addresses or port numbers (Local). Each service in the backend has their own set of databases, if persistent storage is needed.

Our team chose microservice over monolithic due to the strengths of it:

1. Self-contained services
2. Eliminates single-point of failure
3. Loosely coupling of services
4. Independently deployable
5. Independently scalable

4.3.2 Database per Service

Since there are multiple backend services that require some persistence of storage, our team decided to integrate the Database per Service pattern to keep each microservice persistent data private and accessible only through its own API routes.

Using such a pattern helps to keep our services loosely coupled, and any changes to one backend service database does not affect the other services' databases. This also allows services and their respective databases to be deployed and scaled independently. This comes in nicely when different databases have different data storage requirements.

4.3.3 Redis Caching

Our team has decided to use Redis for in-memory caching mechanism (AWS ElastiCache) as we want to ensure that our clients are able to get speedy responses. For instance, our matching service requires users to be matched within 30 seconds if possible, and many users trying to get matched up within a short amount of time. Redis being an in-memory caching mechanism helps to provide quicker response to our clients as it minimizes the number of queries made to our databases by decreasing data access latency and ease load on our databases. With AWS ElastiCache, we also get to enjoy the benefits of fast, scalable and easy to use managed service. (ElastiCache cluster with 3 nodes for scalability).

4.4 Design Principles & Patterns

4.4.1 Facade Pattern

The Gateway service used the Facade pattern to unify all the API from different microservices. A unified interface makes the different microservices easier to use and the frontend just needs to interact with the gateway with a single entry point. This reduces complexity and allows easier expansion of additional microservices in the future.

4.4.2 Publisher-Subscriber Pattern

Publisher-Subscriber Pattern is used in the communication and collaboration to allow users to communicate in match. Publisher-Subscriber Pattern allows us to synchronize various tools that we provide by publishing any update or changes to the various events and subscribing to get all updates. For example, in the collaboration service, any changes made are being published to the specific events like 'Coding' and 'Drawing' in order for different tools like the code editor and whiteboard to be synchronized between the two users. Publisher-Subscriber Pattern allows for potential expansion for multi-users matches where multiple users can easily subscribe to the same event.

5 Microservices Implementation

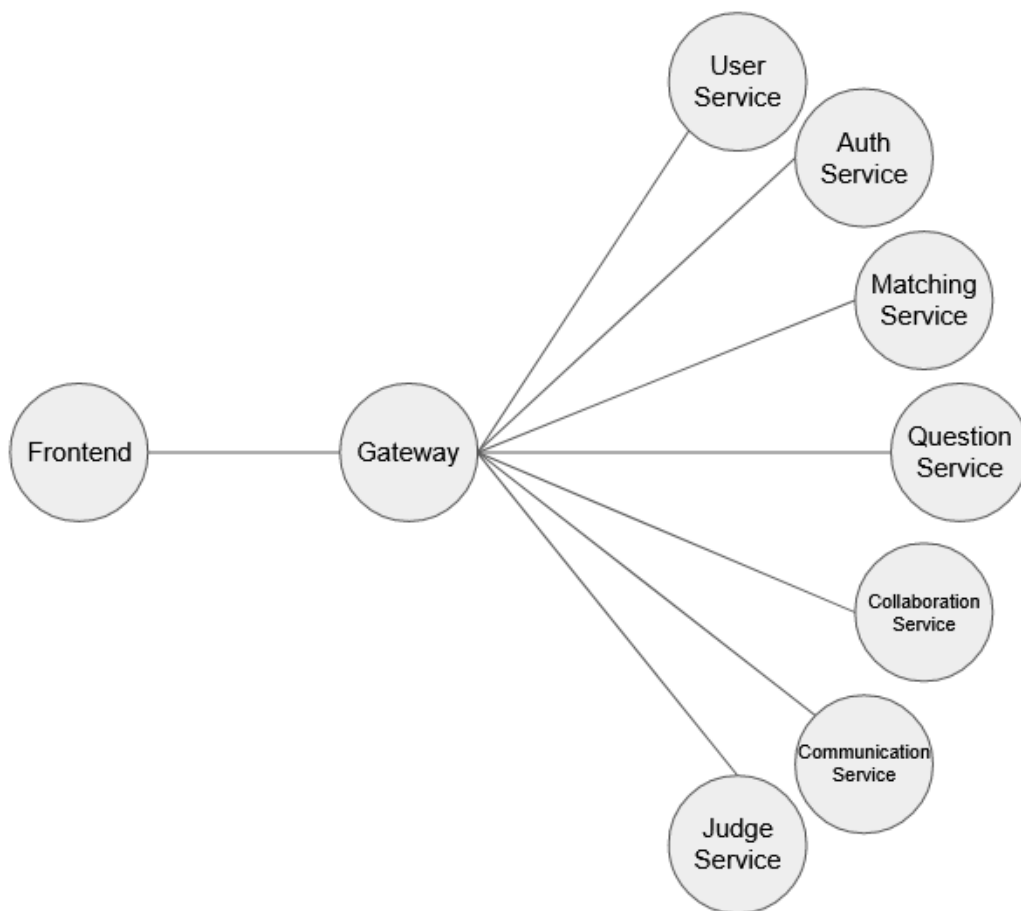
Our team implemented the following microservices to support features that would help the platform achieve its purpose and enhance a user's experience. These features include: creating and managing an account for a user, authenticating users, allowing users to communicate with each other through different mediums (Video, Voice, and Text), collaboration with each other in a live coding environment (shared IDE) and drawing of diagram (Virtual Whiteboard), the compilation of code with output for a variety of supported languages and viewing past attempted question and written code. The following table highlights the services we implemented ourselves and the external services that we relied on and adapted:

Services	External Services
Gateway Service	-
Authentication Service	-
User Service	MongoDB Atlas
Matching Service	MongoDB Atlas
Question Service	MongoDB Atlas
Collaboration Service	-
Communication Service	PeerJS
Judge Service (Code Compilation)	-
History Service	MongoDB Atlas

This section will highlight each microservice, the routes for each endpoint and include sequence diagrams of selected scenarios to showcase the flow of data through various interactions to achieve our features.

5.1 Gateway Service

The gateway service acts as the entry point to our PeerPrep backend system. It acts as a middleman and encapsulates the complex internal system architecture and plays the role of an orchestrator to redirect all requests to each different internal microservice. Additionally, the gateway protects the public endpoints of internal microservices from incoming requests by communicating with the auth microservice. This can be seen as the usage of the facade architectural pattern because the gateway implements a single interface in front of a complex system to improve its usability and provide loose coupling. This helps allow us to maintain and change or migrate the location of the backend microservices with no impact to the client.



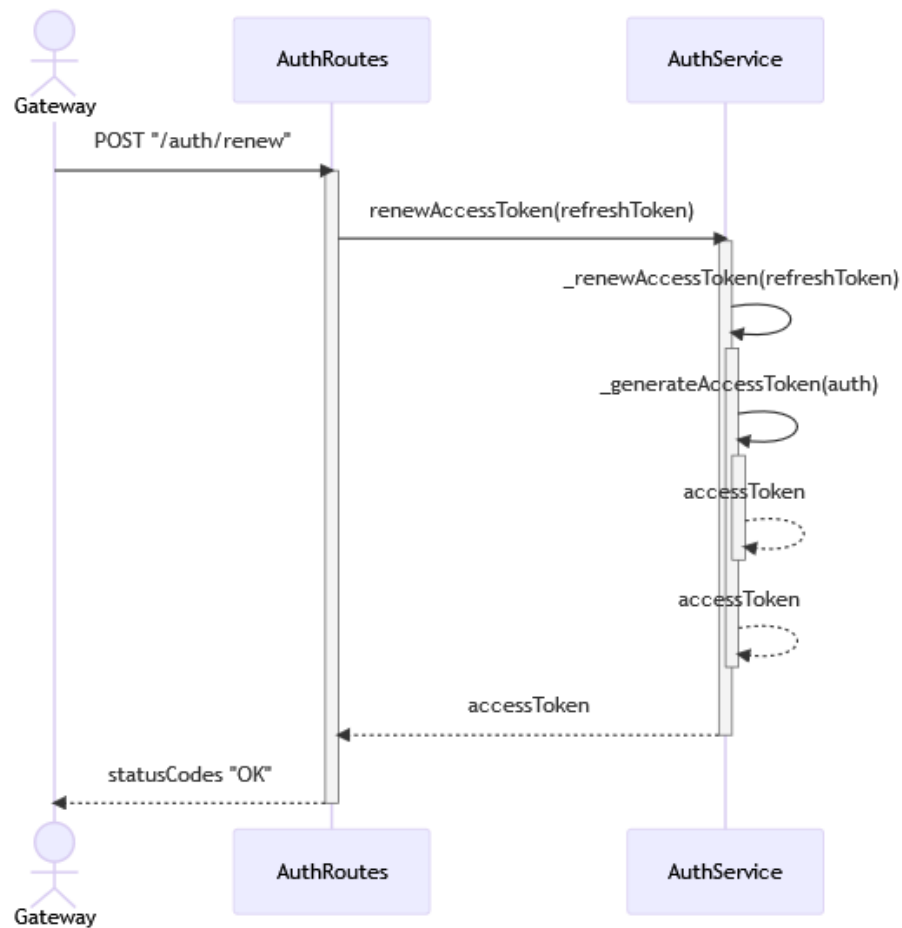
5.2 Auth Service

The auth service is implemented as a backend microservice process to maintain a list of users that are logged in and consists of routes for creating and verifying access and refresh tokens. It ensures that only the authorized users can access protected PeerPrep backend services.

Auth services	Route	Request Type	Request Parameters/ Body/Cookies
Generate access and refresh token	/generate	POST	userId, username
Verify if the user is authenticated and it's access token is not expired or compromised	/verify	POST	refreshToken
Renew the currently logged in user access token after 15 minutes	/renew	POST	refreshToken
Revoke the refresh token of the logged out user	/revoke	POST	refreshToken

The /generate returns a response body containing the generated access token. At the same time, a refresh token will also be generated and appended to the response as a httpOnly cookie. Routes that require the refresh token as a parameter will automatically specify the httpOnly cookie.

Renewal of user access token



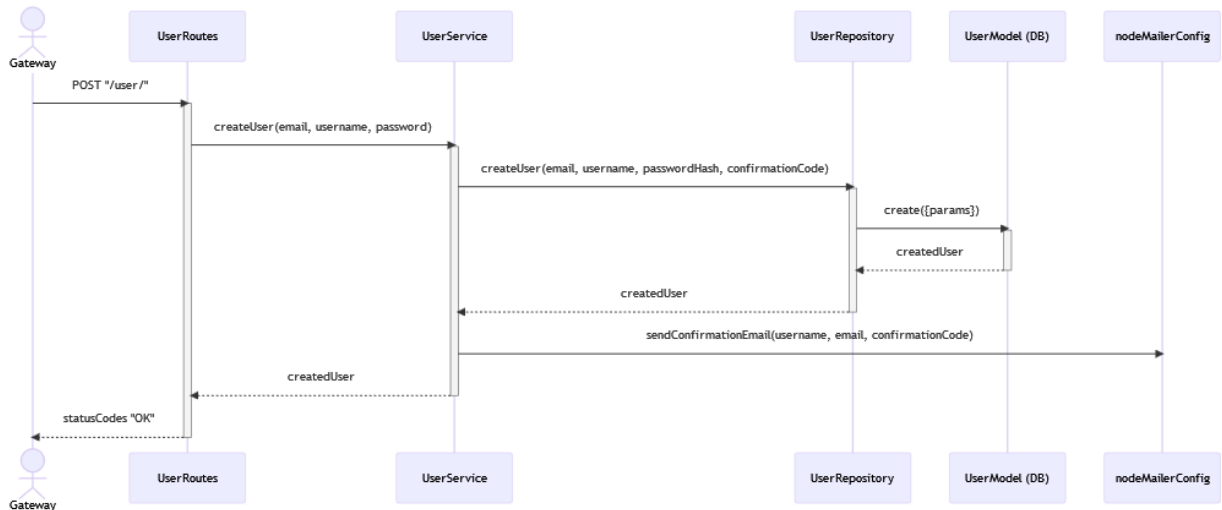
1. Gateway sends a POST request to authRoutes “/renew” with the refreshToken to get a new accessToken.
2. AuthRoutes will invoke AuthService’s renewAccessToken(), passing in the refreshToken as parameter.
3. AuthService will then invoke its’ _renewAccessToken() passing in the refreshToken.
4. Within _renewAccessToken(), AuthService will invoke _generateAccessToken(), passing in the auth decoded from verifying the refreshToken.
5. Within _generateAccessToken(), it will resign the auth, thus renewing it for another 15 minutes.
6. AuthService will then return the renewed accessToken back to AuthRoutes, and AuthRoutes will return statusCodes “OK” to gateway.

5.3 User Service

The user service is implemented as a backend microservice process to allow users to create a new account, sign in, update their password, reset their password, and delete their account. It supports the following user services:

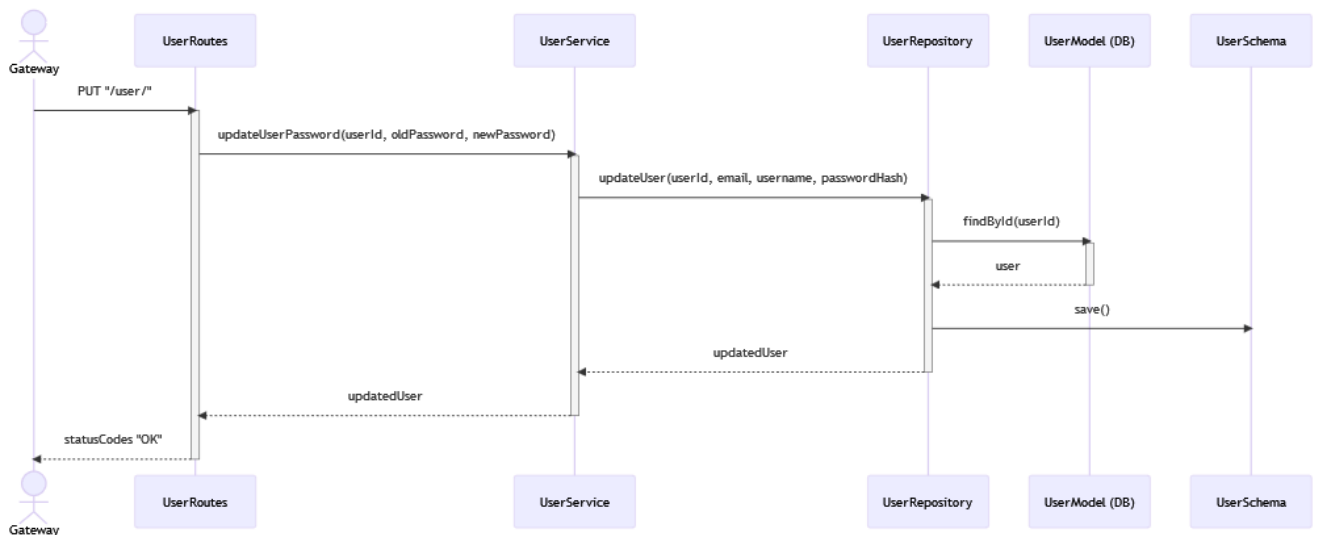
User services	Route	Request Type	Request Parameters/ Body
Get user by email	/	GET	email
Create new account	/	POST	email, username, password
Update user password	/	PUT	userId, oldPassword, newPassword
Delete user account	/:userId	DELETE	userId
Verify the user account	/verify	POST	email, password
Confirm the user account	/confirm/:confirmationCode	GET	confirmationCode
Send password reset link to user email	/passwordReset	POST	email
Reset the password of the user account	/passwordReset/:userId/:token	POST	userId, token, newPassword

Create new account



1. Gateway sends a POST request to the UserRoutes with the email, username , and password to create a new account.
2. UserRoutes then invokes createUser() belonging to UserService with the parameters.
3. UserService will generate the password hash and confirmation code (to be sent to the user account) and parse the email, username, hash, and confirmation code as parameters into createUser() and invoke it.
4. Invoking createUser() of UserRepository will invoke create() with the same parameters and create the new user object in the database. create() will return the createdUser object back to UserRepository.
5. UserRepository will return the user object "createdUser" back to UserService. UserService will then invoke the sendConfirmationEmail with username, email, and confirmation code as parameters.
6. UserService will return the createdUser back to UserRoutes and UserRoutes will return status ok to the gateway.

Update user password



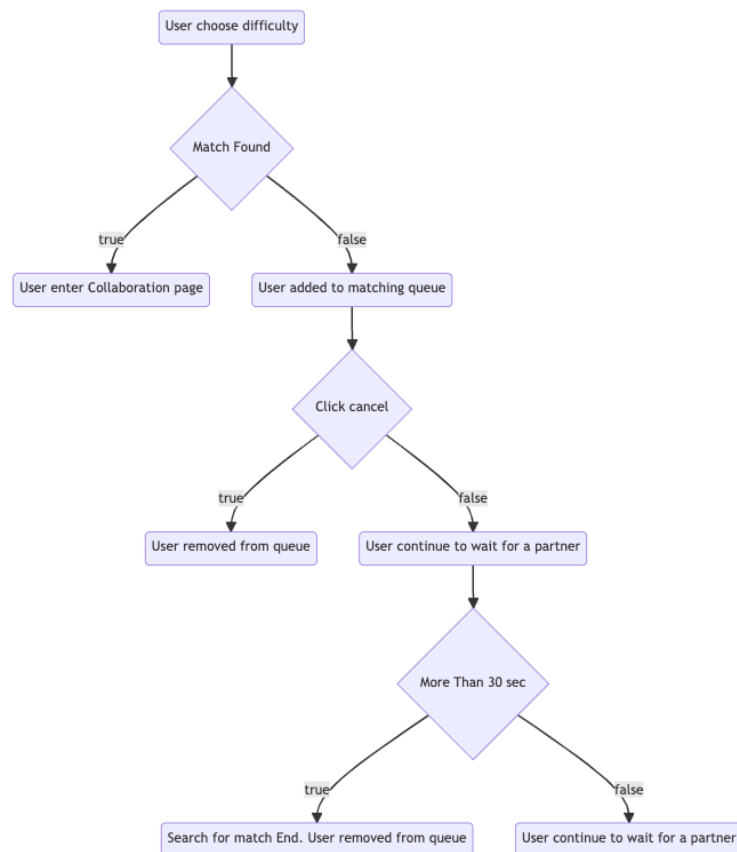
1. Gateway sends a PUT request with `userId`, `oldPassword` and `newPassword` to `UserRoutes` to request to update the user password.
2. `UserRoutes` will invoke `updateUserPassword()` in `UserService`, parsing in the parameters.
3. `UserService` will compute the password hash and invoke `UserRepository`'s `updateUser()`, parsing in `userId`, `email`, `username` and the `passwordHash`.
4. `UserRepository` will invoke `findById()`, and obtain the user object from `UserModel` to be updated.
5. Then `UserRepository` will update the `passwordHash` of the user object and invoke the object's `save()`.
6. `UserRepository` then returns the `updatedUser` back to `UserService`. `UserService` will then return `updatedUser` back to `UserRoutes`. `UserRoutes` will then return `statusCode "OK"`.

5.4 Matching Service

The matching-service is implemented as a backend microservice process to allow users to get a match based on the question difficulty they have selected.

When the matching-service receives a match request from a user, the matching-service will look for potential matches by iterating through its database and attempt to find match entries that are within a 30 seconds time range and of the same question difficulty. If the match cannot be found, the user's information will be inserted into the database as a match entry. The match entry also specifies the entry creation time, for other users to match upon within the 30 seconds window. At the same time, the service will place the user's socket inside a socket.io room of the match entry's id.

Once a match is found, the user's socket will be attached to the socket.io room of the selected match entry's id. As a result, both users' sockets will be attached to the same socket.io room. The socket can thereafter be used for emitting and subscribing various events that will be used for in other backend services.

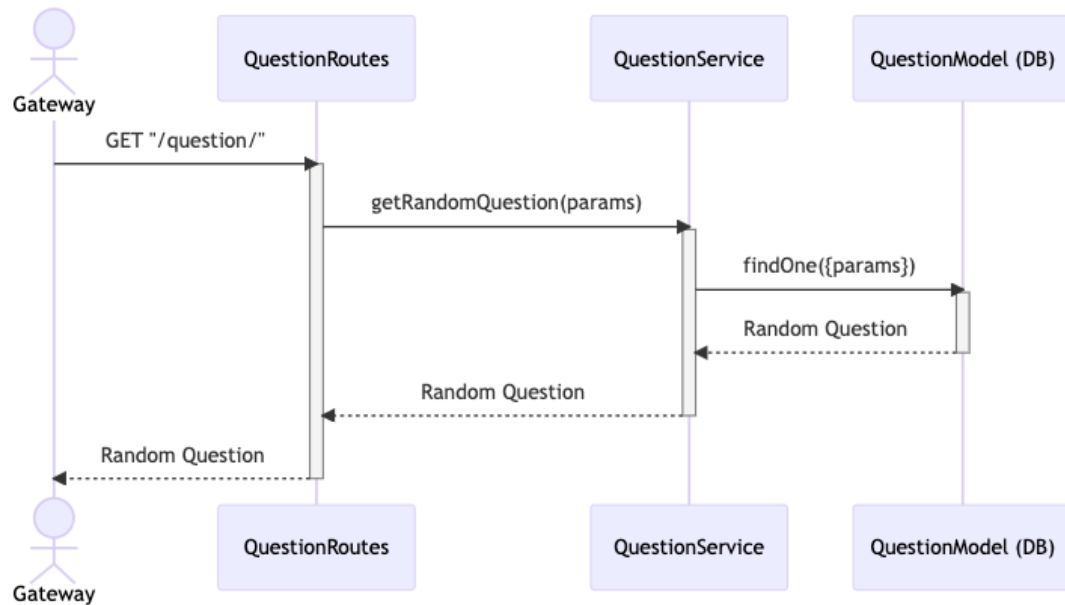


5.5 Question Service

The question service is implemented as a backend microservice process to allow users to get a random question based on the difficulty they have selected. It supports the following question services:

Question services	Route	Request Type	Request Parameters/ Body
Get user a random question	/	GET	
Get user a easy random question	/easy	GET	
Get user a medium random question	/medium	GET	
Get user a hard random question	/hard	GET	
Get user the question based on question id	/id	GET	id
Get user the relevant questions based on search	/search	GET	searchTerm

Get user a random question



1. Gateway sends a GET request to the QuestionRoutes to get a random question.
2. QuestionRoutes then invokes getRandomQuestion() belonging to QuestionService.
3. Invoking getRandomQuestion() of QuestionRoutes will invoke findOne() with the search condition parameters and get one random question from the database.
4. QuestionService will return the question object "Random Question" back to QuestionRoutes.
5. QuestionRoutes will return the question object back to the gateway.

5.6 Collaboration Service

The collaboration service provides real time coding collaboration between users. Collaboration service used Socket.io to sync up code and drawing. Publisher-Subscriber pattern is used to relay changes between users. Each pair of users is assigned to a room id which is generated from the Match Service. The user's socket is subscribed to various events to coordinate various activities like drawing and coding.

Event	Purpose
Language	To sync up the programming language selected by the "interviewee" between the two users
CurrentCode	Whenever there is a change to the code, the user's socket will emit the changes and the other user that is subscribed to this event, will be able to view update in real time
Drawing	Whenever there is a change to the drawing, the user's socket will emit the changes and the other user that is subscribed to the four events, will be able to view drawing and also draw at the same time.
UndoDrawing	
RedoDrawing	
ClearDrawing	
LeaveRoom	When one user leave the match, Socket will push the information to the observer(another user) so that match can be ended
SubmitCode	When one user (interviewee) submit the code, Socket will push the information to the observer(another user) so that the role of the interviewee and interviewer will be swap and a new question will be set

5.7 Communication Service

Chat (Message)

Socket.io is used to send messages between two users during a match. Socket.io used PubSub design to relay messages between users. Each pair of users is assigned to a room id which is generated from the Match Service. Communication service will group the two users' sockets together based on the room id. Each user will emit and receive messages from this room. New messages received from the socket will be appended to the list of messages and will be displayed in the UI.

Video Chat

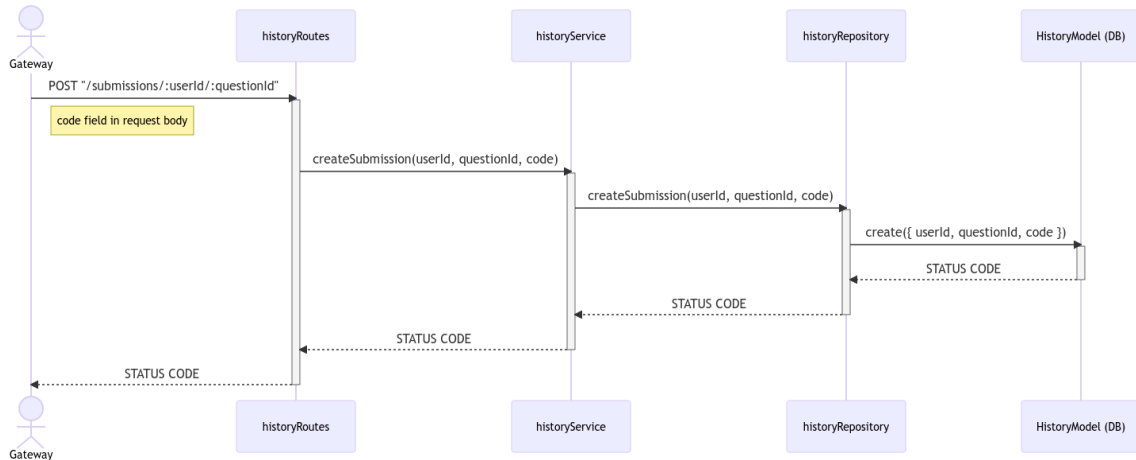
Video Chat is implemented using PeerJs which is based on WebRTC. WebRTC allows users to communicate using a peer-to-peer connection which allows for lower latency and better video chat communication experience. In order for the both users to establish peer-to-peer connection, one of the users (user2) will make a call (signaling) to the other user (user1) via PeerJs server. The other user (user1) will then answer the call and the peer-to-peer connection is then established. We are using the PeerJs online server to broker the connection and no peer-to-peer data is transferred to the PeerJs server.

5.8 History Service

The history service is a backend microservice process that manages code submission history for each user and the questions they have attempted and submitted. It allows a user to view their most recent past code submissions for various questions they have attempted

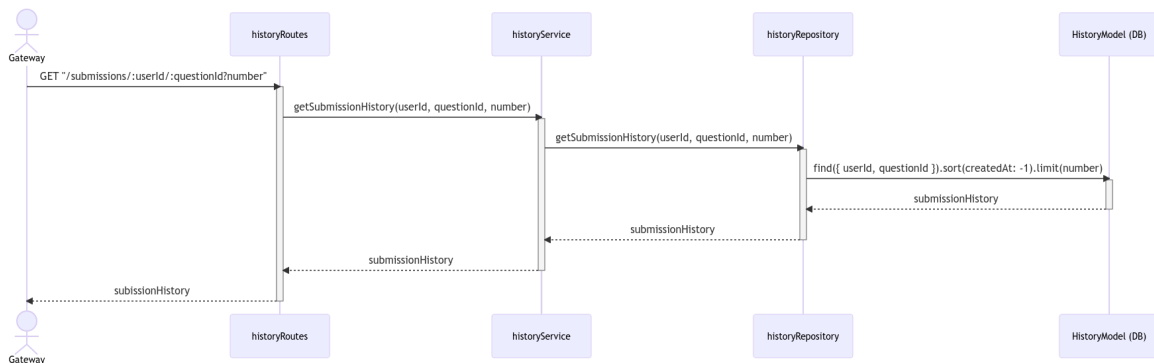
History services	Route	Request Type	Request Parameters/ Body
Create a submission	/submissions/:userId/:questionId	POST	Params: userId, questionId Body: Code field
Get submission history for a specified user and question	/submissions/:userId/:questionId	GET	Params: userId, questionId Query string: number
Get submission history for a specified user	/submissions/:userId	GET	Params: userId Query string: number

Create submission history



1. Gateway sends a POST request to the historyRoutes router to create a code submission for a userId and questionId specified in the request parameters
2. The router calls createSubmission of historyService with the userId and questionId from the request parameters and the code field from the request body
3. The historyService calls createSubmission of historyRepository
4. The historyRepository creates the document in the database
5. The appropriate status code is then passed back to the Gateway eventually

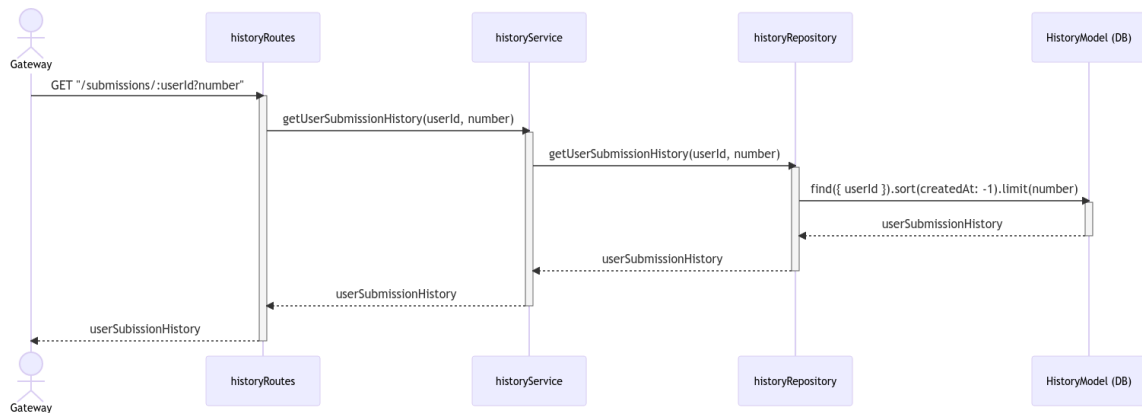
Get user question submission history



1. Gateway sends a GET request to the historyRoutes router to get the most recent number of code submissions for a specified user and question from the request parameters and query string
2. The router calls getSubmissionHistory of historyService with the userId, questionId and number of submissions from the request parameters and query string
3. The historyService calls getSubmissionHistory of historyRepository

4. The historyRepository finds matching documents in the database and sorts them by createdAt in descending order to get the most recent submissions before using number to limit how many documents will be returned
5. The submissionHistory object is then passed back to the Gateway eventually

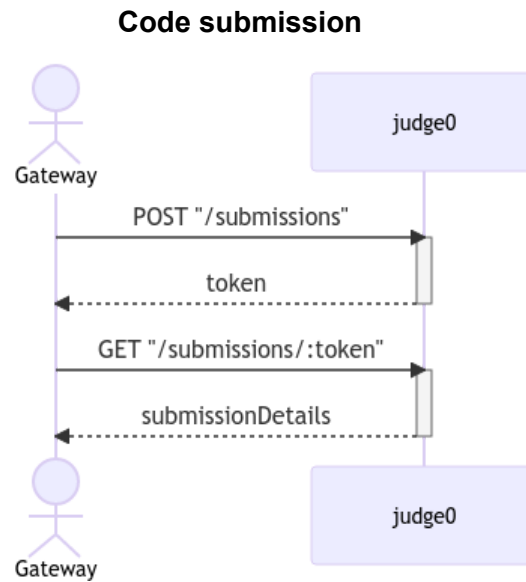
Get user submission history



1. Gateway sends a GET request to the historyRoutes router to get the most recent number of code submission for a specified user in the request parameters and the query string
2. The router calls getUserSubmissionHistory of historyService with the userId from the request parameters and number from the query string
3. The historyService calls getUserSubmissionHistory of historyRepository
4. The historyRepository finds matching documents in the database and sorts them by createdAt in descending order to get all the submissions for a user before using number to limit how many documents will be returned
5. The userSubmissionHistory object is then passed back to the Gateway eventually

5.9 Judge Service

The judge service provides users the ability to compile code with input test cases and observe the output. We used an instance of judge0 because it was easy, simple to use and had comprehensive documentation. A code submission creates a response with a token. Further requests can be sent to check on the status of the code submission using this token and view the compiled output once compilation is done processing.

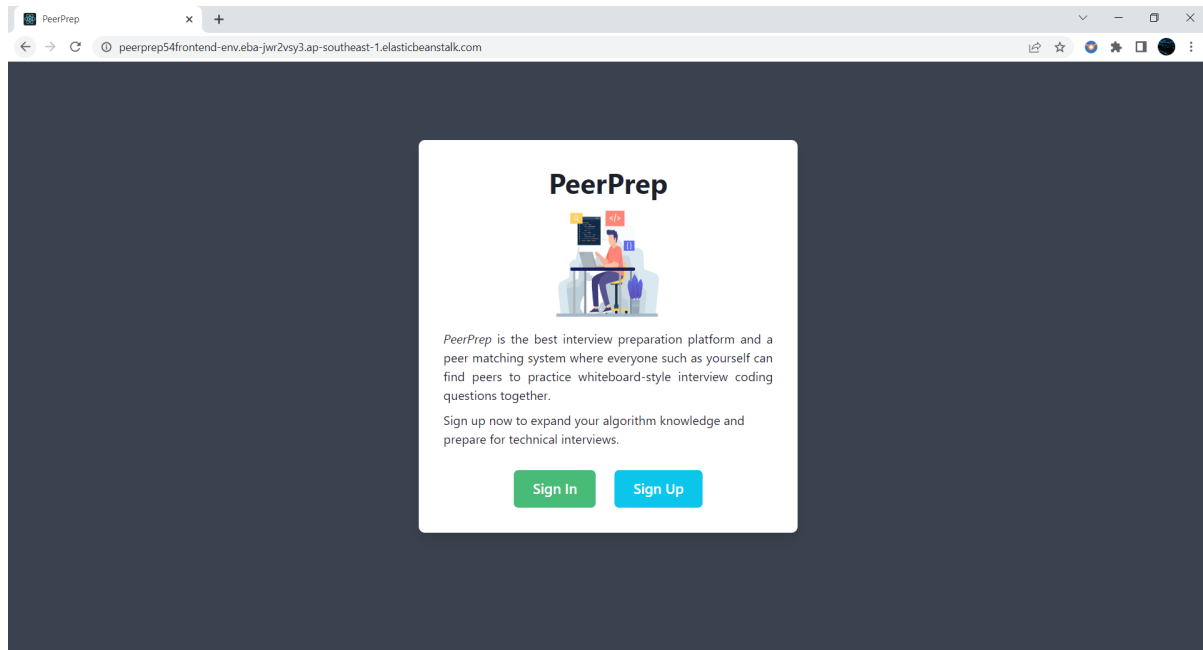


6 User Interface

This section showcases our user interface and guides usage of the PeerPrep platform.

6.1 Landing Page

When users first arrive on our platform they will be directed to our landing page. This prompts them to either sign in if they have already created an account or to sign up and register.



6.2 Signup Page

Upon clicking the sign up button on the landing page, users will navigate to our signup page where it prompts them with a form to fill in required details for the registration process

Registration

Email Address *

Username *

Password * Confirm Password *

Sign Up

Already a user? [Sign In](#)

When a user signs up with their NUS email, a confirmation email will be sent to their school inbox containing a verification link for their PeerPrep account which will expire after a duration of time.

Registration

Email Address *
yongliangang@u.nus.edu

Username *
yongliangang

Password * Confirm Password *

Sign Up

Already a user? [Sign In](#)

Success
Account successfully created! Please check your email

PeerPrep - Please confirm your account

peerprepteam54@gmail.com
To: Ang Yong Liang
- External Email -

Email Confirmation

Hello yongliangang

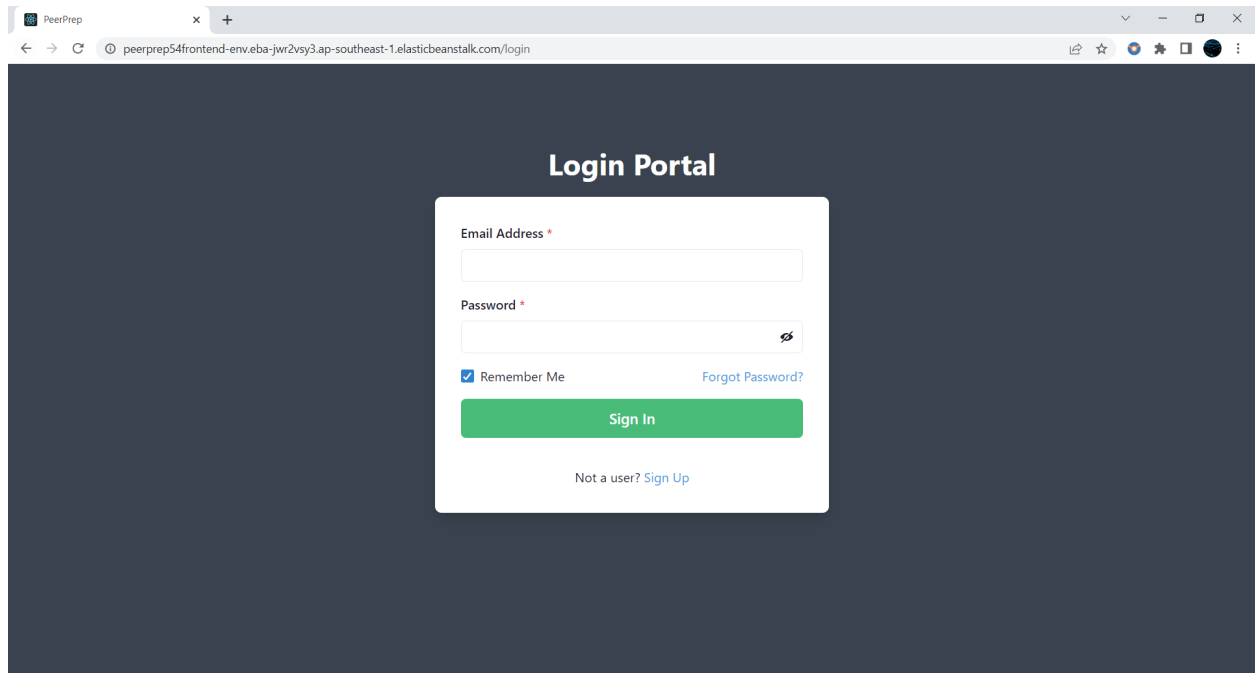
Thank you for signing up with PeerPrep. Please confirm your email by clicking on the following link
[Click here](#)

Happy coding,
PeerPrep Team 54

Reply Forward

6.3 Login Page

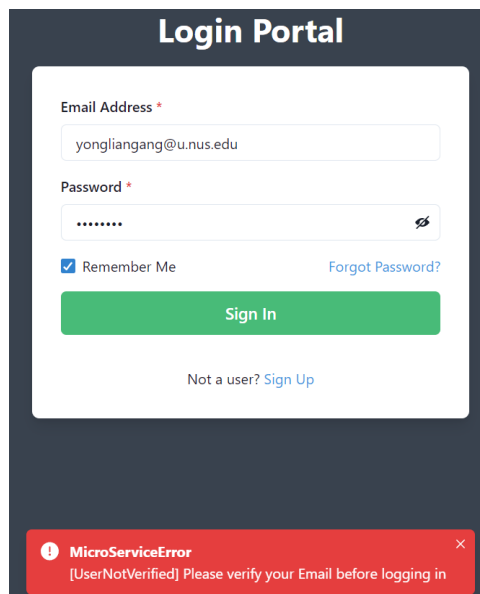
Upon clicking the sign in button on the landing page, users will navigate to our login page. Here users are required to use their NUS email and password they have signed up with.



The screenshot shows a web browser window with the URL `peerprep54frontend-env.eba-jwr2vvy3.ap-southeast-1.elasticbeanstalk.com/login`. The page features a dark blue background with a white login form in the center. The form is titled "Login Portal" and contains the following elements:

- Email Address ***: A text input field.
- Password ***: A password input field with a toggle icon for visibility.
- ☒ **Remember Me**: A checkbox with the label "Remember Me".
- [Forgot Password?](#): A link to the forgot password page.
- Sign In**: A green button to submit the login form.
- [Not a user? Sign Up](#): A link for new users to sign up.

If the user is not verified, they would be unable to login to the main page. This ensures that the email address is valid and belongs to a NUS student fulfilling our F-U-2 functional requirement.

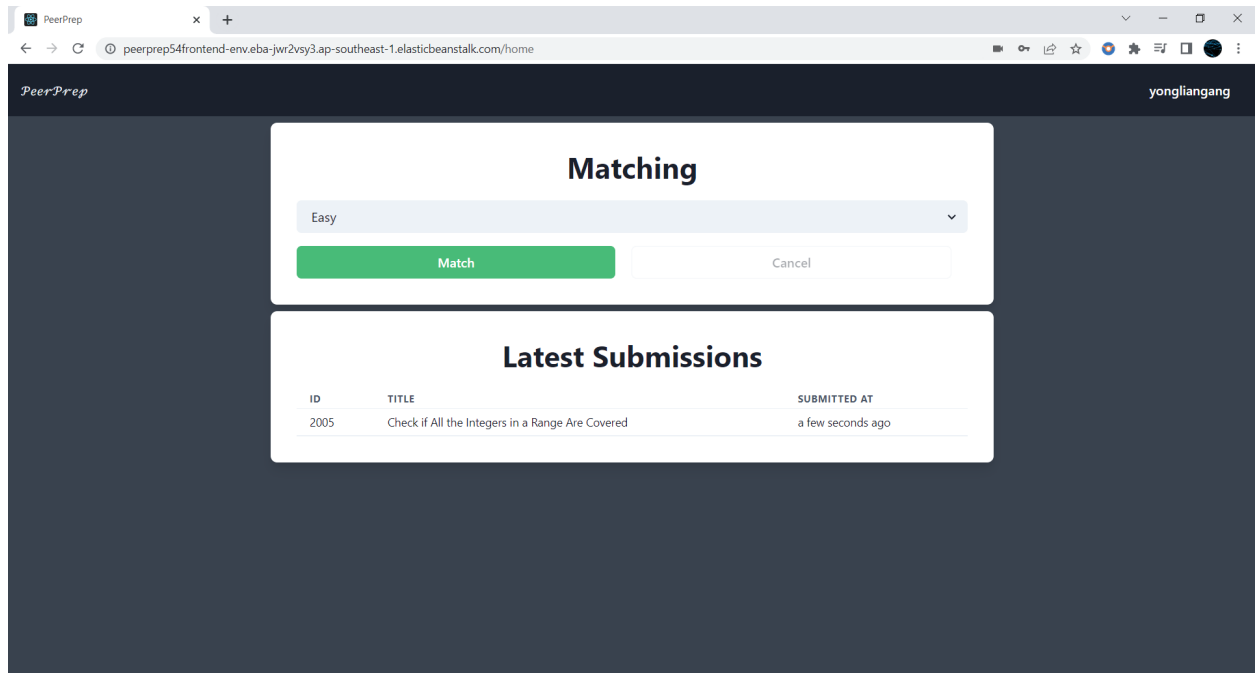


This screenshot shows the same login form as the previous one, but with the email address `yongliangang@u.nus.edu` entered. Below the form, a red error message is displayed:

MicroServiceError
[UserNotVerified] Please verify your Email before logging in

6.4 Main Page

Upon a successful login from the login page, users will navigate to the main page of the platform which has two parts. The top portion allows them to match with other students based on their selected level of difficulty of questions they wish to attempt. The bottom portion shows a history of their most recent submissions.



Users may navigate to view a specific submission history by clicking on the title.



6.5 Collaboration Page

Upon a successful match from the main page, users will navigate to the collaboration page which displays a random question of their selected level of difficulty on the left portion of the page. An interactive code editor is displayed on the right portion of the page where users can work collaboratively on the solution in real time. Users can select their preferred programming language under the IDE Options dropdown menu. On this page users take turns being the interviewer and interviewee, where one user submits their solution and then the other does providing mutual feedback through the communication mediums we have provided.

The screenshot displays the PeerPrep collaboration interface. On the left, a problem titled "Increasing Order Search Tree" is shown. It asks the user to rearrange a binary search tree in in-order so that the leftmost node is the root, and every node has no left child and only one right child. An example is provided: a tree with root 5 and nodes 1, 2, 3, 4, 6, 7, 8, 9 is transformed into a right-skewed tree where nodes are in increasing order. The input and output arrays are also shown.

On the right, the code editor is active, showing a C++ solution. The code defines a `TreeNode` struct and a `Solution` class with a method `increasingBST` that takes a `TreeNode*` root and returns a `TreeNode*`.

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
8  *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
9  *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left)
10 * };
11 */
12 class Solution {
13 public:
14     TreeNode* increasingBST(TreeNode* root) {
15     }
16 };
17
```

Under the white board tab, both users can draw sketches in the virtual white board in real time to explain their thought process to the other user playing the role of the interviewer.

PeerPrep

peerprep54frontend-env.eba-jwr2v3y3.ap-southeast-1.elasticbeanstalk.com/collaboration

SubmitLeave Room

Increasing Order Search Tree

Easy

StackTreeDepth-First SearchBinary Search TreeBinary Tree

Input: root = [5,3,6,2,4,null,8,1,null,null,null,7,9]

Output: [1,null,2,null,3,null,4,null,5,null,6,null,7,null,8,null,9]

Example 2:

Code EditorWhite BoardChat BoxVideo / Audio

4

1245

Under the Chat Box tab, users can send messages to each other.

PeerPrep

peerprep54frontend-env.eba-jwr2v3y3.ap-southeast-1.elasticbeanstalk.com/collaboration

SubmitLeave Room

Increasing Order Search Tree

Easy

StackTreeDepth-First SearchBinary Search TreeBinary Tree

Given the root of a binary search tree, rearrange the tree in **in-order** so that the leftmost node in the tree is now the root of the tree, and every node has no left child and only one right child.

Example 1:

Input: root = [5,3,6,2,4,null,8,1,null,null,null,7,9]

Output: [1,null,2,null,3,null,4,null,5,null,6,null,7,null,8,null,9]

Example 2:

Code EditorWhite BoardChat BoxVideo / Audio

yongliangang2:43:24 PM

Hello

wejiro65642:44:13 PM

Hello world

Enter your message

Send

Under the video/audio tab, users are able to video chat each other to ask and answer questions to simulate an actual interview process.

33

PeerPrep

peerprep54frontend-env.eba-jwr2v3y3.ap-southeast-1.elasticbeanstalk.com/collaboration

SubmitLeave Room

PeerPrep

Easy

StackTreeDepth-First SearchBinary Search TreeBinary Tree

Increasing Order Search Tree

Given the root of a binary search tree, rearrange the tree in **in-order** so that the leftmost node in the tree is now the root of the tree, and every node has no left child and only one right child.

Example 1:

Input: root = [5,3,6,2,4,null,8,1,null,null,7,9]

Output: [1,null,2,null,3,null,4,null,5,null,6,null,7,null,8,null,9]

Example 2:

Code EditorWhite BoardChat BoxVideo / Audio

wejiro6564

yongliangang

Users are able to compile code to test whether their solution is working.

PeerPrep

peerprep54frontend-env.eba-jwr2v3y3.ap-southeast-1.elasticbeanstalk.com/collaboration

SubmitLeave Room

PeerPrep

Easy

ArrayHash TablePrefix Sum

Check if All the Integers in a Range Are Covered

You are given a 2D integer array `ranges` and two integers `left` and `right`. Each `ranges[i] = [starti, endi]` represents an **inclusive** interval between `starti` and `endi`. Return `true` if each integer in the inclusive range `[left, right]` is covered by **at least one** interval in `ranges`. Return `false` otherwise.

An integer `x` is covered by an interval `ranges[i] = [starti, endi]` if `starti <= x <= endi`.

Example 1:

Input: `ranges = [[1,2],[3,4],[5,6]]`, `left = 2`, `right = 5`

Output: `true`

Explanation: Every integer between 2 and 5 is covered:

- 2 is covered by the first range.
- 3 and 4 are covered by the second range.
- 5 is covered by the third range.

Example 2:

Input: `ranges = [[1,10],[10,20]]`, `left = 21`, `right = 21`

Output: `false`

Explanation: 21 is not covered by any range.

Constraints:

- `1 <= ranges.length <= 50`
- `1 <= starti <= endi <= 50`
- `1 <= left <= right <= 50`

Code EditorWhite BoardChat BoxVideo / Audio

IDE Options

Code Compilation

200410

Compile Code

```

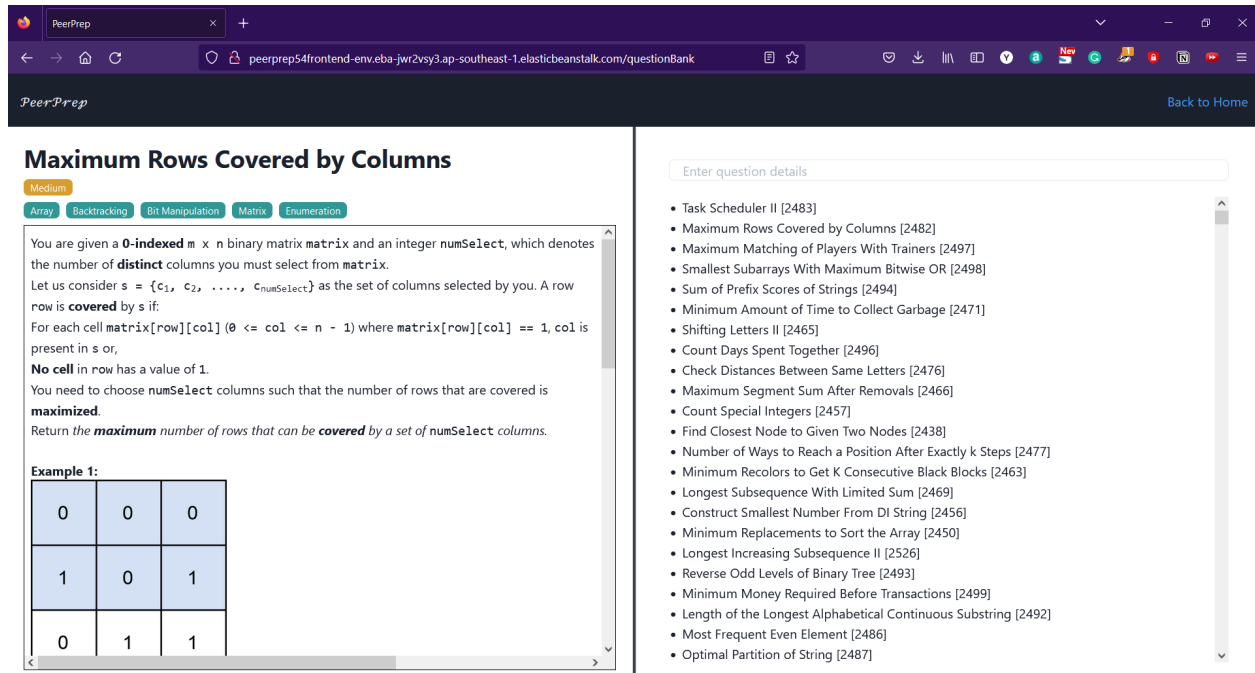
1 x = input()
2 print(int(x) + 210)

```

34

6.6 Question Search

Users can navigate to a question bank search by clicking on the username in the top right corner from the main page. By entering question details, users can filter and search for questions.



Maximum Rows Covered by Columns

Medium

Array Backtracking Bit Manipulation Matrix Enumeration

You are given a **0-indexed** $m \times n$ binary matrix **matrix** and an integer **numSelect**, which denotes the number of **distinct** columns you must select from **matrix**.
Let us consider $s = \{c_1, c_2, \dots, c_{\text{numSelect}}\}$ as the set of columns selected by you. A row is **covered** by s if:
For each cell $\text{matrix}[\text{row}][\text{col}]$ ($0 \leq \text{col} \leq n - 1$) where $\text{matrix}[\text{row}][\text{col}] == 1$, col is present in s or,
No cell in row has a value of 1.
You need to choose **numSelect** columns such that the number of rows that are covered is **maximized**.
Return the **maximum** number of rows that can be **covered** by a set of **numSelect** columns.

Example 1:

0	0	0
1	0	1
0	1	1

Enter question details

- Task Scheduler II [2483]
- Maximum Rows Covered by Columns [2482]
- Maximum Matching of Players With Trainers [2497]
- Smallest Subarrays With Maximum Bitwise OR [2498]
- Sum of Prefix Scores of Strings [2494]
- Minimum Amount of Time to Collect Garbage [2471]
- Shifting Letters II [2465]
- Count Days Spent Together [2496]
- Check Distances Between Same Letters [2476]
- Maximum Segment Sum After Removals [2466]
- Count Special Integers [2457]
- Find Closest Node to Given Two Nodes [2438]
- Number of Ways to Reach a Position After Exactly k Steps [2477]
- Minimum Recolors to Get K Consecutive Black Blocks [2463]
- Longest Subsequence With Limited Sum [2469]
- Construct Smallest Number From DI String [2456]
- Minimum Replacements to Sort the Array [2450]
- Longest Increasing Subsequence II [2526]
- Reverse Odd Levels of Binary Tree [2493]
- Minimum Money Required Before Transactions [2499]
- Length of the Longest Alphabetical Continuous Substring [2492]
- Most Frequent Even Element [2486]
- Optimal Partition of String [2487]

7 Milestones

7.1 Gantt Chart

FRs								
F-A Auth Service								
S/N	Functional Requirements	W1-6	W7	W8	W9	W10	W11	W12
F-A-1	The auth service will allow a user to log into their accounts by entering the email and password they signed up with							
F-A-2	The auth service will allow a user to log out of their account							
F-A-3	The auth service will verify a user's access token before accessing protected resources							
F-A-4	The auth service will allow a user to renew their access token using a refresh token							
F-A-5	The auth service will allow a user to choose whether to persist their login session							
F-U User Service								
S/N	Functional Requirements	W1-6	W7	W8	W9	W10	W11	W12
F-U-1	The user service will allow a user to create an account with an email address and a password							
F-U-2	The user service will send a confirmation email when a new user signs up and creates an account							
F-U-3	The user service should ensure that every account created has a unique email address							
F-U-4	The user service will allow a user to reset their password via their email address							
F-U-5	The user service will allow a user to delete their account							
F-U-6	The user service will allow a user to update their current password to a new password if they remember their current password							
F-U-7	The user service will allow a user to view their own settings							

F-M Matching Service								
S/N	Functional Requirements	W1-6	W7	W8	W9	W10	W11	W12
F-M-1	The matching service will allow a user to select a difficulty level (Easy, Medium, Hard, Random) of the question they wish to attempt							
F-M-2	The matching service will match two users waiting for a question of the same level of difficulty and allocate them to the same room							
F-M-3	The matching service will notify a user that no match is available if a match cannot be found							
F-M-4	The matching service should allow a user to cancel the match search at any time during the search							
F-M-5	The matching service should allow a user to leave a room once matched							
F-Q Question Service								
S/N	Functional Requirements	W1-6	W7	W8	W9	W10	W11	W12
F-Q-1	The question service will generate a random question from a question bank for the user based on their selected level of difficulty							
F-Q-2	The question service should provide an interface for users to view questions in the question pool							
F-H History Service								
S/N	Functional Requirements	W1-6	W7	W8	W9	W10	W11	W12
F-H-1	The history service will show a user their past submissions for a question							
F-H-2	The history service will provide a user interface for users to view the details of their submissions							
F-CL Collaboration Service								
S/N	Functional Requirements	W1-6	W7	W8	W9	W10	W11	W12
F-CL-1	The collaboration service will allow a user to type code in the provided editor							
F-CL-2	The collaboration service will allow a user to view shared updates in the provided code editor during the coding session							
F-CL-3	The collaboration service will allow a user to leave the coding session							

F-CL-4	The collaboration service will allow a user to sketch on a collaborative whiteboard							
F-CL-5	The system should show user what the other user is sketching							
F-CL-6	The system should allow users to compile and run their own solution							
F-CL-7	The system should allow user to specify input before their run their code							
F-CL-8	The system should allow the roles of interviewer/interviewee to be swapped the first interviewee submit their solution							
F-CM Communication Service								
S/N	Functional Requirements	W1-6	W7	W8	W9	W10	W11	W12
F-CM-1	The communication service will allow a user to view other user messages during the coding session							
F-CM-2	The communication service will allow a user to chat over video and voice call during the coding session							
F-CM-3	The communication service will allow a user to toggle turning on and off their video							
F-CM-4	The communication service will allow a user to toggle turning on and off their audio							
NFRs								
Availability Attribute								
S/N	Non Functional Requirements	W1-6	W7	W8	W9	W10	W11	W12
NF-A-1	The system should not take more than 10 seconds to load when user send a request							
NF-A-2	The system should bring the user back to main page after failing to find a match for the user within 30 seconds							
Reliability Attribute								
NF-R-1	The system should ensure the data is being saved properly when a network error occurred							
Security Attribute								

NF-SE-1	The system should revoke access tokens that were expired							
NF-SE-2	The system should only generate access tokens with a lifespan of 15 minutes							
NF-SE-3	The system should not persistently store any user's access token/refresh token							
NF-SE-4	The system should only provide the client with a refresh token using httpOnly secure cookie							
NF-SE-5	Users' passwords should be hashed and salted before storing in the DB							
NF-SE-6	User's email should be restricted to only contain "u.nus.edu" within the domain.							
NF-SE-7	Users' password length should be within 8 to 16 characters							
NF-SE-8	The user should not be able to login if they are not email verified							
NF-SE-9	Backend microservices are only accessible via the gateway by authenticated users through the frontend							
Usability Attribute								
NF-U-1	The system should be able to persist the login session after page refreshes.							
NF-U-2	The system should only record compilable submissions							
Scalability Attribute								
NF-SC-1	The system should be able to scale up or down to match demands							

7.2 CI/CD

Our team used Github Actions for CI/CD. Workflows for certain microservices were defined to be run upon certain triggers.

Example of auth service CI workflow file. It will invoke the test cases using JEST in our `./backend/auth-service` folder. The output workflow can be seen on the right.

```
1 name: Auth Service CI
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10  build-and-test:
11    runs-on: ubuntu-latest
12    env:
13      ENV: PROD
14      PORT: 8001
15    strategy:
16      matrix:
17        node-version: [16.x]
18    steps:
19      - uses: actions/checkout@v3
20      - name: Use Node.js ${{ matrix.node-version }}
21        uses: actions/setup-node@v3
22        with:
23          node-version: ${{ matrix.node-version }}
24      - name: Install Dependencies
25        working-directory: ./backend/auth-service
26        run: npm ci
27      - name: Run Tests
28        working-directory: ./backend/auth-service
29        run: npm test
```

Summary

Jobs

Run details

Usage

Workflow file

build-and-test (16.x)
succeeded 1 hour ago in 14s

- Set up job
- Run actions/checkout@v3
- Use Node.js 16.x
- Install Dependencies
- Run Tests
 - Run npm test
 - auth-service@1.0.0 test
 - jest --testTimeout=20000 --runInBand
 - PASS tests/routes.test.js
 - PASS tests/services/authService.test.js
 - Test Suites: 2 passed, 2 total
 - Tests: 11 passed, 11 total
 - Snapshots: 0 total
 - Time: 1.386 s
 - Ran all test suites.
- Post Use Node.js 16.x
- Post Run actions/checkout@v3
- Complete job

Example auth service CD workflow file. It will be triggered when the prod branch is updated.

```
1 name: Auth Service CD
2
3 on:
4   push:
5     branches: [prod]
6
7 jobs:
8   deploy:
9     runs-on: ubuntu-latest
10    env:
11      AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
12      AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
13      AWS_DEFAULT_REGION: ${ secrets.AWS_DEFAULT_REGION }
14    steps:
15      - uses: actions/checkout@v2
16      - name: Install Python 3.9
17        uses: actions/setup-python@v2
18        with:
19          python-version: 3.9
20      - name: Install EB CLI using pip
21        run: |
22          python -m pip install --upgrade pip
23          pip install awsebcli
24      - name: Deploy to Elastic Beanstalk
25        working-directory: ./backend/auth-service
26        run: |
27          eb deploy
```

Auth Service CD

Merge pull request #160 from CS3219-AY2223S1/update-readme #8

Summary

Jobs

Run details

Usage

Workflow file

deploy
succeeded 4 minutes ago in 59s

- Set up job
- Run actions/checkout@v2
- Install Python 3.9
- Install EB CLI using pip
- Deploy to Elastic Beanstalk
 - Run eb deploy
 - Creating application version archive "app-221109_083957399037".
 - uploading Peerpreauthservice-env/app-221109_083957399037.zip to S3. This may take a while.
 - upload Complete.
 - 2022-11-09 08:40:00 INFO Environment update is starting.
 - 2022-11-09 08:40:06 INFO Deploying new version to instance(s).
 - 2022-11-09 08:40:09 INFO Instance deployment: You didn't specify a Node.js version in the configuration file.
 - 2022-11-09 08:40:26 INFO Instance deployment completed successfully.
 - 2022-11-09 08:40:32 INFO New application version was deployed to running EC2 instances.
 - 2022-11-09 08:40:32 INFO Environment update completed successfully.
- Post Install Python 3.9
- Post Run actions/checkout@v2
- Complete job

8 Future Enhancements

There are a couple of features that we were not able to implement in the final product due to limited time and budget constraints.

8.1 Enhance matching criteria

Besides matching users based on difficulty, additional options like question categories could allow users to filter the type of question that they would like to work on. This would provide a better experience to the user as users can pick the weaker topics that they need extra practice on or topics that are commonly tested by companies.

8.2 Allowing match to be more than 2 users

Instead of keeping each match to only two users, allowing more users in a match could better simulate an actual interview where there are multiple interviewers interviewing an interviewee. However, this could result in a longer session due to multiple roles switching and also scalability issues. Hence, more testing is required to find the maximum suitable number of users in each match room.

8.3 Providing solution code

Having solution code allows the user to review a question and understand how to solve a problem. It also allow us to create an additional feature whereby user can run test case and find out what is the correct output and what is their output

8.4 Adding review to each session

Allowing users to provide comments and a score on how one has performed in the session would allow users to know what are the areas they can improve. This could enhance the user experience as it allows the user to perform better in the actual interview. The score could then be tabulated and averaged for a user's past attempts to show how other users perceive how prepared a user may be for their future interview.

8.5 End-to-end and Load testing

More load tests can be done to test whether our application can handle multiple requests without failing and to see if our system could accommodate more users concurrently. End-to-end testing could allow us to replicate real user experience and to see whether our application works for devices of different specifications.

8.6 Persistency of collaboration data

Adding persistence of collaboration data using Kafka could allow our system to ensure that users remain in the collaboration page even after refreshing the page. This would improve the user experience of PeerPrep.

9 Reflections

Although it would seem that microservices would be easy to develop due to looser coupling as compared to a monolithic architecture, we experienced challenges with trying to handle environment variable bloat that crept up on us towards the end. Our gateway forwarded requests from the frontend to each internal backend microservice, but adapting this to work with publish and subscribe events from the frontend to each internal microservice was not trivial. We introduced redis caching on the gateway for microservices that used websockets to get pub/sub to work. Deploying each microservice and managing their separate environment variables also proved to be difficult. This gave us insight on what our future internships or jobs may entail when we have to work on production systems that use microservices.

Our team faced a lot of difficulties with using AWS for deployment. It was a virgin experience for us and we encountered numerous issues that were difficult to debug given that the error logs on AWS do not provide much details of what went wrong. It took a lot of searching and trial and error to fix some of the issues we experienced. Our team also lacked budget for deployment and we did not even realize how much costs we incurred until it was too late having ended up burning a hole in our pockets. This provided our team with insight on how much actual software development costs in the real world and why development decisions can have a significant impact on the finances of a business.