



CS3219
Software Engineering Principles and Patterns
Group 22
PeerPrep
Final Report

Ong Siying Natasha (A0207932E)
Robin Ho Liu Bin (A0150034Y)
Thuta Htun Wai (A0199811U)
Koh Vinleon (A0202155W)

Presented to: Prof. Bimlesh Wadhwa
Mentor: Nicholas Wee

Table of Contents

Background Introduction.....	3
Contributions	4
Project Requirements.....	5
Functional Requirements.....	5
Non-Functional Requirements	9
1. Reliability.....	10
2. Performance	13
3. Scalability.....	17
4. Security	19
5. Usability.....	22
6. Portability.....	23
Developer Documentation	24
Architecture Diagram.....	24
Tech Stack	25
Architectural Decision.....	25
Software Architecture Design.....	28
MVC Pattern.....	28
Clean Architecture.....	29
Messaging Pattern.....	30
Synchronous Message Call	30
Asynchronous Message Passing	30
Design Pattern.....	31
Chain of Responsibility Pattern	31
Singleton Pattern	32
Backend API	33
API Specifications	33
Error Resilience.....	33
API (User) Endpoints	34
Frontend	63
Application Demonstration.....	64
CI/CD	74
Development Process	77
Areas of Improvements.....	78
Reflections.....	79

Background Introduction

As Computer Science students, it is inevitable that we will encounter technical interviews during our future job search processes, and as a matter of fact, most of us already have while searching for internships. While preparing for technical interviews, it is likely that we will encounter challenging questions. Challenging technical questions typically require interviewees to come up with algorithms, and many lack that experience.

Having a platform to allow pair programming may progressively improve one's algorithmic design capability and eventually, one's confidence. Through verbal discussions with our peers, we can also practice articulating our thought processes to the other party, which is also an important skill for technical interviews. Furthermore, discussing algorithm problems with peers is also highly encouraged by the professors who teach algorithm-related modules. With these considerations in our mind, we have decided to come up with PeerPrep to empower students to improve their algorithmic thinking.

In addition, the new PeerPrep offers a new practice mode. Similar to platforms such as Leetcode and HackerRank, users can now view the available questions. However, instead of doing alone, they can choose a particular question to practice with another peer!

Our application supports pair programming where 2 users will be able to concurrently edit a piece of code in an editor, similar to a whiteboard-style programming interview. On top of that, users will be able to communicate with each other via chat and video on the same interface, encouraging communication between users to enhance their learning benefits.

Our application supports user authentication and user management via user and admin frontend, respectively.

Contributions

Name	Technical Contributions	Non-Technical Contributions
Vinleon	<p>Backend</p> <ul style="list-style-type: none"> • Match Microservice <ul style="list-style-type: none"> ◦ Matching Algorithms using Socket.IO ◦ Elo Match Pool ◦ Ratings • Editor Microservices • Chat Microservices • Video Chat Microservices <p>Frontend</p> <ul style="list-style-type: none"> • User Frontend <ul style="list-style-type: none"> ◦ User Authentication ◦ Profile ◦ Questions ◦ Matching ◦ Session (Editor + Chat + Video Chat) • Admin Frontend <ul style="list-style-type: none"> ◦ User Authentication <p>DevOps</p> <ul style="list-style-type: none"> • CI/CD + GitOps • Kubernetes Cluster Management • Ingress Controller & Service Discovery • Monitoring & Logging • Autoscaling • RabbitMQ • Frontend and Backend Deployment <p>Testing</p> <ul style="list-style-type: none"> • Editor Microservices • Chat Microservices 	<ul style="list-style-type: none"> • UI/UX for both frontends • Requirement and Design Report • Final Report
Natasha	<p>Backend</p> <ul style="list-style-type: none"> • User microservice <p>Frontend</p> <ul style="list-style-type: none"> • Admin User Dashboard <p>Testing</p> <ul style="list-style-type: none"> • User microservice 	<ul style="list-style-type: none"> • UI/UX for admin frontend's user management • Requirement and design report • Final report
Robin	<p>Backend</p> <ul style="list-style-type: none"> • Question Microservice <p>Frontend</p> <ul style="list-style-type: none"> • Admin Question Dashboard <p>Unit Testing</p> <ul style="list-style-type: none"> • Question Microservice 	<ul style="list-style-type: none"> • UI/UX for admin frontend's question management • Requirement and design report • Final report
Thuta	<p>Backend</p> <ul style="list-style-type: none"> • Match Microservices <p>Frontend</p> <ul style="list-style-type: none"> • Admin Match Dashboard <p>Unit Testing</p> <ul style="list-style-type: none"> • Match Microservices 	<ul style="list-style-type: none"> • UI/UX for admin frontend's match management • Requirement and Design Report • Final Report

Project Requirements

Functional Requirements

Revised from the functional requirements derived from the previous Requirement & Design report, these are the functional requirements we have worked on.

Indicator	Description
*	Low Priority
**	Medium Priority
***	High Priority

ID	Priority	Requirement Statement	Sprint	Completed By
F1	User Authentication Frontend			
F1.2	**	The web application must allow users to register for an account.	2	Vinleon
F1.3	**	The web application must allow users to sign in with a valid account.	2	
F1.4	**	The web application must allow users to logout of their account.	2	
F1.5	**	The web application must allow users to change their password.	2	
F2	User Profile Frontend			
F2.1	*	The web application must allow the users to view their profile.	2	Vinleon
F2.2	*	The user's profile consists of average ratings, elo and match history.	2	
F2.3	*	The user's profile includes the user's personal information, such as their display name and registered email.	2	
F3	Question Frontend			
F3.1	***	The web application should allow users to view a list of questions available.	2	Vinleon
F3.2	***	The web application should allow users to view the question details (title, descriptions, constraint, examples).	2	
F3.3	***	The web application should allow users to choose a question to find a partner to enter practice mode.	2	
F4	Match Frontend			

F4.1	Matchmaking Options			
F4.1.1	***	The web application should allow users to start matchmaking to start a coding session.	2	Vinleon
F4.1.2	***	The web application must allow the user to select the difficulty level before matchmaking.	2	
F4.1.3	**	The web application can allow the user to select the problem topic before matchmaking.	2	
F4.1.4	**	The user must be able to cancel the matchmaking at any time.	2	
F4.2	Matchmaking Process			
F4.2.1	***	The web application must attempt to find a match within 30 seconds and start a session.	2	Vinleon
F4.2.2	***	The web application must inform the user if unable to find a match within 30 seconds.	2	
F4.2.3	***	The web application must attempt to match the user with another user based on the difficulty level and problem topic chosen.	2	
F5	Session Frontend			
F5.1	Code Editor			
F5.1.1	***	The web application must allow users to type and edit in the code editor concurrently with his/her partner.	2	Vinleon
F5.1.2	**	The web application must show both users' typing pointers in real time.	2	
F5.2	Communication			
F5.2.1	**	The web application must allow users to communicate with each other through the text chatting interface.	1	Vinleon
F5.2.2	*	The web application allows users to join into a video chat.	2	
F5.3	Question and Match Details			
F5.3.1	**	The web application should allow users to view the question title and description.	2	Vinleon
F5.3.1	*	The web application should allow users to choose to view the hints for the question.	3	

F5.3.2	*	The web application should allow users to choose to view the examples (input & output) for the question.	3	
F5.3.3	*	The web application should allow users to choose to view the constraints for the question.	3	
F5.3.4	*	The web application should allow users to view the partner's information and matching criteria.	2	
F5.3.5	*	The web application should allow users to view the time left for the match.	2	
F5.4	Ending Session Frontend			
F5.4.1	***	The web application must allow users to end the session early or leave when the time's up.	2	Vinleon
F5.4.2	***	The web application must allow users to have the option to view the answer to the problem after the session has ended.	2	
F5.4.3	*	The web application must allow users to rate his/her partner's performance (on a scale of 1 to 5) after the session has ended.	2	
F5.4.4	*	The web application must update the user's elo upon rating given.	3	
F6	Admin Management (CRUD)			
F6.1	**	The web application must allow admins to create new administrators.	2	Natasha
F6.2	**	The web application must allow admins to view all the administrators.	2	
F6.3	*	The web application must allow admins to update other administrators.	2	
F6.4	**	The web application must allow admins to delete existing administrators.	2	
F7	User Management (CRUD)			Natasha
F7.1	**	The web application must allow admins to create new users.	2	
F7.2	**	The web application must allow admins to view all the user's information.	2	
F7.3	*	The web application must allow admins to update user's information.	2	
F7.4	**	The web application must allow admins to delete a user account.	2	

F8	Question Management (CRUD)			
F8.1	***	The web application must allow admins to create new questions.	2	Robin
F8.2	***	The web application must allow admins to view all the questions.	2	
F8.3	*	The web application must allow admins to sort and view questions based on difficulty level.	2	
F8.4	**	The web application must allow admins to update existing questions.	2	
F8.5	***	The web application must allow admins to delete existing questions.	2	
F9	Match Management			
F9.1	*	The web application must allow admins to update an existing match.	2	Thuta
F9.2	*	The web application must allow admins to view all the matches.	2	
F9.3	*	The web application must allow admins to delete a match.	2	

Non-Functional Requirements

Based on our requirement and design documentation, we have identified 7 Requirements Quality Attributes as part of our NFRs. Each will be elaborated in detail.

Priority	Attribute	Reason
1	Reliability	The web application should not encounter any huge bugs or crash during user consumption.
2	Performance	As the web application is highly user-interactive, the application's performance should be high enough such that it will not cause disruption to the user experience.
3	Scalability	The application must have a system with the ability to adapt to increasing user demands.
4	Security	As a web-based application, security must be prioritized to ensure that all user information is secure.
5	Usability	The web application must be straightforward, easy to use and intuitive, even for new users.
6	Portability	The web application should function regardless of a user's hardware device, web browser or operating system.

Constraints

1. The code editor is not capable of helping users to debug their code.
2. The web application only works with an active internet connection.
3. The web application is not optimized on mobile devices.

1. Reliability

Single Point of Failure (SPOF)

To always ensure the high reliability of our application status uptime, our application needs to be fault-tolerant and have plans to mitigate the SPOF concerns. Mentioned below are some of the measures we have put in place for our application.

EC2 Instance

At all times, two EC2 instances across two availability zones would be running to ensure higher availability. This also ensures that the clusters will always have additional pods available for auto-scaling as well as to allow isolation between critical and non-critical services.

i-0a4174cf8d281f42e	Running	t3a.medium	2/2 checks passed	No alarms	+	ap-southeast-1a
i-0194a6adc199dfffbb	Running	t3a.medium	2/2 checks passed	No alarms	+	ap-southeast-1b

ALB available in 2 availability zones

Ingress Controller (API Gateway)

The AWS Application Load Balancer (ALB) is used as a single point of entry to fulfil the API gateway requirement as part of the project requirements. To address the SPOF, the load balancers are available in at least 2 availability zones. In any scenario, when one is down, another one will take over. As such, this ensures the high availability of our application.

Availability Zones [subnet-060170bfe95c72bfc - ap-southeast-1b](#)
IPv4 address: Assigned by AWS

[subnet-0865b706b7d49f14e - ap-southeast-1a](#)
IPv4 address: Assigned by AWS

ALB available in 2 availability zones

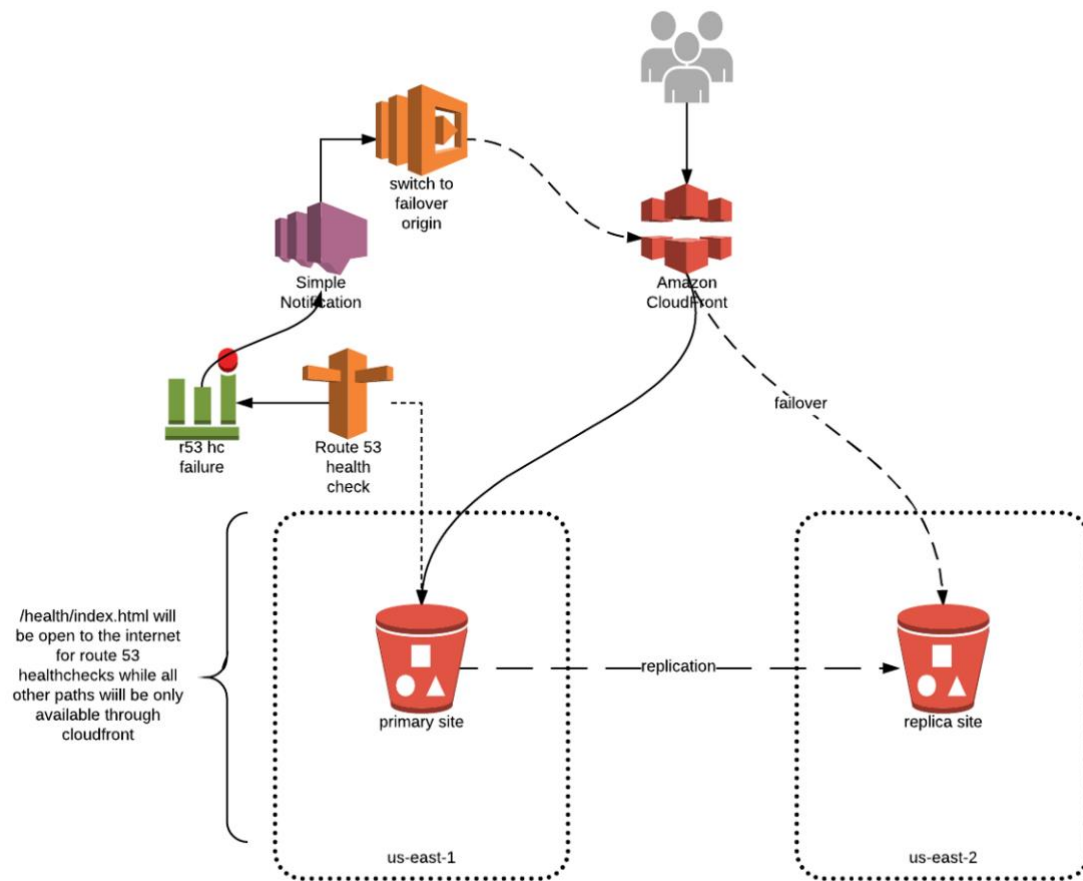
Pods (Microservices Instance)

At all times, there are two pods running per Microservice. In the scenario where one pod crashes, the ALB will route incoming requests to the alternative pod to fulfil the user's requests.

NAME	READY	STATUS	RESTARTS	AGE
chat-85f8bc945d-rn6xw	1/1	Running	0	21h
chat-85f8bc945d-xgq9q	1/1	Running	0	12h
editor-75c69c4889-rzj4m	1/1	Running	0	21h
editor-75c69c4889-wzxxz	1/1	Running	0	12h
match-6c4c9ffff8-4hwnt	1/1	Running	0	21h
match-6c4c9ffff8-clj68	1/1	Running	0	12h
question-6b4dbc7f99-lfhd5	1/1	Running	0	12h
question-6b4dbc7f99-w4mdz	1/1	Running	0	12h
user-6c95584798-6zqr9	1/1	Running	0	21h
user-6c95584798-p47mg	1/1	Running	0	12h
video-chat-88cd4667c-kk7jh	1/1	Running	0	12h
video-chat-88cd4667c-m6ggg	1/1	Running	0	17h

2 instances running per Microservices

CloudFront



CloudFront failover plan

The user's frontend CloudFront distribution has a failover plan where it will route the traffic to a replica site when the primary site fails the health check.

Data Reliability

Complex Database Queries

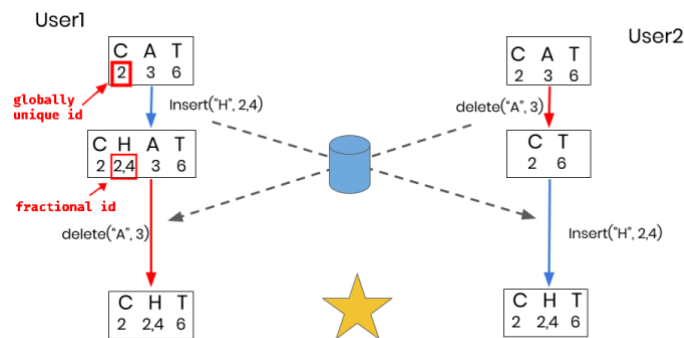
For complex database queries that may involve querying multiple documents or microservices, a MongoDB client session is used for benefits like causal consistency, retryable writes, and transactions. In any scenario of failure from any database operation, the client session will be aborted, and the database will be able to roll back safely to avoid inaccurate or partially updated information at all times.

```
/**
 * @description If there is a match that meets the requirement -> update. Else create new record.
 * @function findMatch
 */
export default async function findMatch(payload: Partial<IMatch> & { question_id: string } & { user_id: string }) {
  const session = await mongoose.startSession();
  session.startTransaction();
```

Code snippet from findMatchController in Match MS

Concurrency Issue

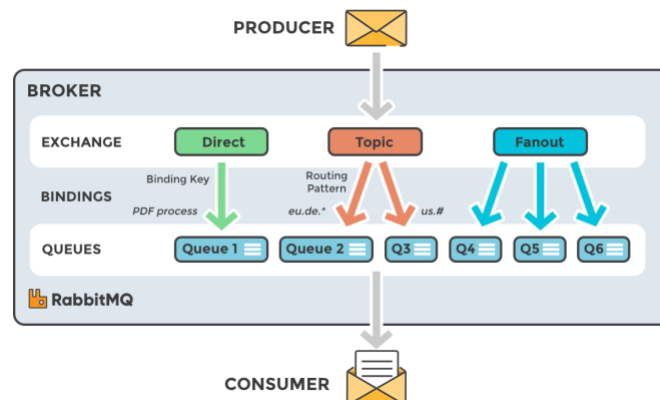
As part of the real-time collaborative editor for PeerPrep, there might be conflicting and concurrency issues when two users edit the same text data at the same time. To ensure some form of consistency, **Yjs**, a conflict-free replicated data types (CRDT) library, was utilized to mitigate this issue.



Source: <https://conclave-team.github.io/conclave-site/>

Message Queue

Inter-service communication is done through CloudAMQP, a RabbitMQ as Service Provider. This allows asynchronous requests and more fault tolerance. If any of the services are down temporarily, the request will be persistent in the queue and processed upon recovery.



Overview of RabbitMQ

2. Performance

Abstracted from Design & Requirement Documentation:

1. The code editor should reflect code changes within 1 to 2 seconds
2. Support up to 100 concurrent users performing meaningful actions
3. Pages should load within 5 seconds

Code Editor Performance

As mentioned in the concurrency issue under Reliability, the code editor in PeerPrep uses Yjs for CRDT. At the same time, the Monaco Editor binding library (y-monaco) developed by Yjs is also used as part of our implementation which is empowered by WebSockets. As it is a live, collaborative code editor, the response time is expected to be quick.

Check WebSocket Availability

Test results : - wss://server.peerprep.tech/editor

New Test

Result URL: wss
Load Time: 2 sec(s)
Tested on: 5 Nov 2021 09:51:46 PM

[Check Website Availability](#) | [Ping Website](#) | [DNS Analysis](#) | [Find IP](#) | [Find Location](#)

Websocket Availability Results: 5 Nov 2021 09:51:44 PM

				<div><div>DNS</div><div>Time taken for the server to resolve DNS name to IP Address</div></div> <div><div>Connect</div><div>Time taken to setup the between connection Site24x7 and website</div></div> <div><div>Handshake Time</div><div>The time taken to complete the handshake</div></div> <div><div>PingPong Time</div><div>The time taken for the WebSocket to provide a pong response for the ping.</div></div>			
Location	Status	IP	(ms)	(ms)	(ms)	(ms)	Response Time (ms)
Fremont-CA - US	OK	13.228.104.5	206	184	544	182	1,116
Toronto - CA	OK	18.136.127.196	40	542	1,945	234	2,761
Amsterdam - NL	OK	18.136.127.196	15	181	539	180	915
Melbourne - AUS	OK	13.228.104.5	32	112	303	98	545
Singapore - SG	OK	13.228.104.5	370	5	10	4	389

Response Time of Editor's WebSocket across 5 countries

Changes should be reflected in about 389 milliseconds for a user in Singapore, whenever there are code changes in the collaborative code editor.

API Endpoint Performance

Load Testing

Load testing is done on our backend endpoint using k6 using a user flow (scenario) approach. Using a user flow approach simulates the high traffic scenario where many concurrent users perform meaningful actions instead of hammering the API endpoints.

```
running (3m13.1s), 000/100 VUs, 603 complete and 0 interrupted iterations
default ✓ [=====] 100 VUs 3m0s

  ■ simple user journey

    ✓ is status 200
    ✓ is api key present

checks.....: 100.00% ✓ 1206 × 0
data_received.....: 9.2 MB 48 kB/s
data_sent.....: 604 kB 3.1 kB/s
group_duration.....: avg=31.67s min=13.59s med=32.43s max=39.51s p(90)=35.88s p(95)=37.99s
http_req_blocked.....: avg=8.32ms min=0s med=1µs max=196.38ms p(90)=2µs p(95)=15.41ms
http_req_connecting.....: avg=541.29µs min=0s med=0s max=11.48ms p(90)=0s p(95)=4.4ms
http_req_duration.....: avg=7.81s min=5.76ms med=1.11s max=31.51s p(90)=27.6s p(95)=30.19s
  { expected_response:true }...: avg=12.64s min=13.37ms med=2.41s max=31.51s p(90)=30.19s p(95)=30.99s
http_req_failed.....: 50.00% ✓ 1206 × 1206
http_req_receiving.....: avg=382.54µs min=19µs med=127µs max=7.48ms p(90)=819.9µs p(95)=967.34µs
http_req_sending.....: avg=120.96µs min=9µs med=87µs max=5.17ms p(90)=202µs p(95)=248µs
http_req_tls_handshaking.....: avg=7.51ms min=0s med=0s max=179.91ms p(90)=0s p(95)=10.26ms
http_req_waiting.....: avg=7.8s min=5.58ms med=1.1s max=31.51s p(90)=27.6s p(95)=30.19s
http_reqs.....: 2412 12.493689/s
iteration_duration.....: avg=31.67s min=13.59s med=32.43s max=39.51s p(90)=35.88s p(95)=37.99s
iterations.....: 603 3.123422/s
vus.....: 28 min=28 max=100
vus_max.....: 100 min=100 max=100
```

The average response duration is 7.81s for a user journey from:

- Logging In
- Viewing Match History
- Viewing Question
- Select a Question
- Changing display name
- Logout

Due to the match, the editor and chat system requires two concurrent users to join in the same room, it was difficult to model it as a test. The sockets are tested separately.

Caching

Several endpoints retrieve a large amount of data from the database (e.g., list of questions and match history). Repeatedly retrieval from the persistent database may slow down the performance. Hence, an auxiliary Redis Database is used in several microservices as a cache. Caching such information serves to reduce the load times.

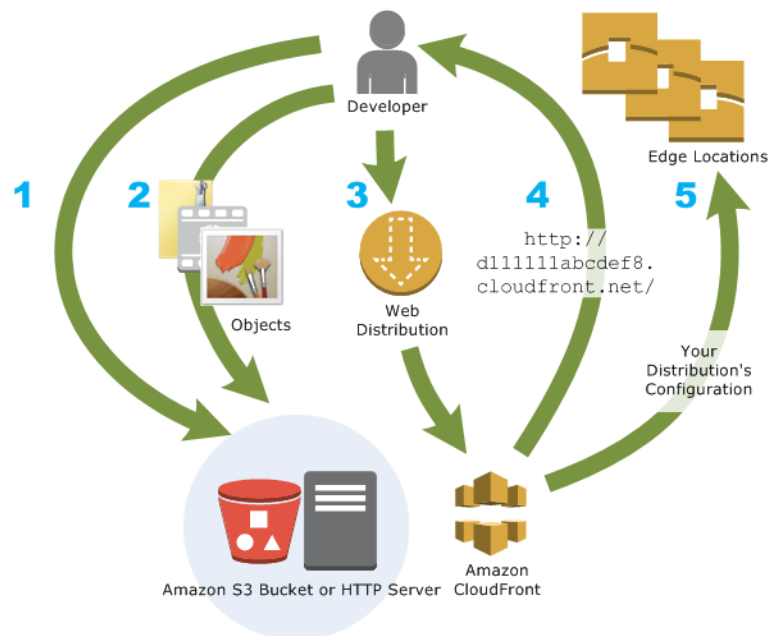
Server-Side Pagination

On top of caching, server-side pagination is also used to improve the performance even further for API endpoints which retrieves more than one data.

Website Performance

Content Delivery Network (CDN)

Amazon CloudFront is a fast content delivery network (CDN) service. It caches the content to its edges allowing fast loading of the static site. Once it is cached, it allows users from all over the world to access the cache files from the nearest edge location instead of retrieving from the actual S3 web hosting bucket.



Overview of CloudFront Failover Plan

Loading Time

The average loading time of the user frontend (peerprep.tech) is around 1 ~ 3.15 seconds on different desktop devices and operating systems, which is within the threshold we have defined. The performance metrics exclude mobile devices as stated in the constraints.

GTmetrix Grade ?

A	Performance ?	Structure ?
	93%	97%

Web Vitals ?

Largest Contentful Paint ?	Total Blocking Time ?	Cumulative Layout Shift ?
1.6s	48ms	0.04

Performance Metrics

The following metrics are generated using Lighthouse Performance data.

Metric details ☐ OFF

First Contentful Paint ?	Good - Nothing to do here 917ms	Time to Interactive ?	Good - Nothing to do here 1.0s
Speed Index ?	Good - Nothing to do here 1.1s	Total Blocking Time ?	Good - Nothing to do here 48ms
Largest Contentful Paint ?	OK, but consider improvement 1.6s	Cumulative Layout Shift ?	Good - Nothing to do here 0.04

Performance Metrics from GTmetrix

















Desktop Performance Insights

[NAVIGATION TIMING API \(in sec.\)](#)

[PAGE RESOURCE SUMMARY](#)

[CPU PROCESSING BREAKDOWN \(in sec.\)](#)

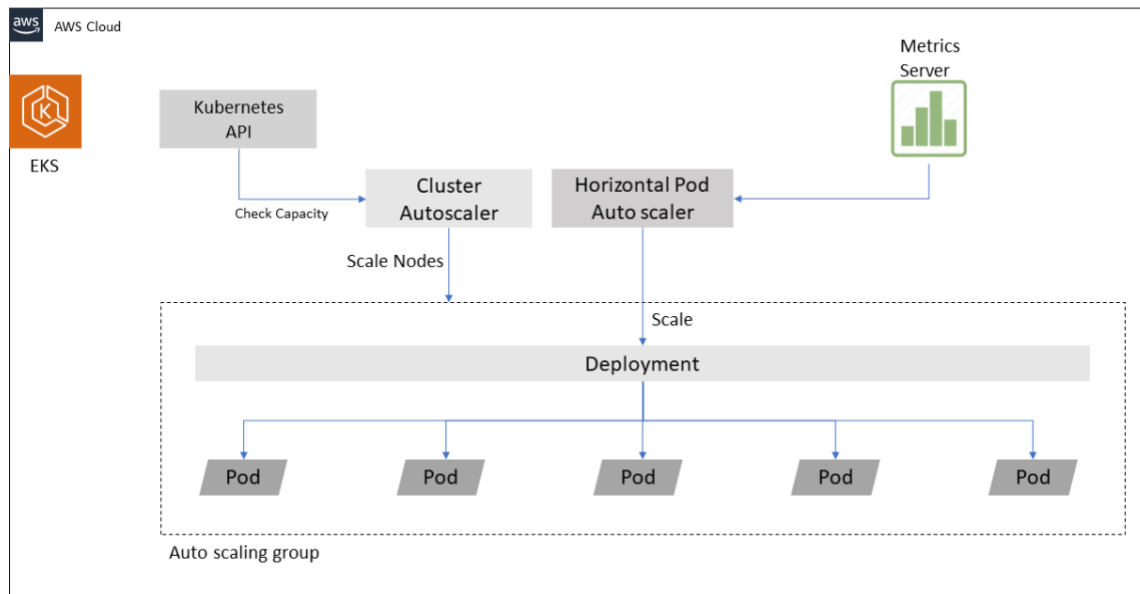
View timing information for critical events during page load. Visualize or download HAR logs for all browser network requests.

Device & Browser		DNS Lookup ↕	TTFB↕	Response↕	DOM Processing ↕	Onload↕	Page Load↕	HAR Log	Launch a Real browser
OS X Big Sur	 14	0.01	0.21	0.13	1.00	0	1.00	 	 Start now
OS X Catalina	 93	0	0.87	0.44	1.82	0	1.82	 	 Start now
OS X Catalina	 95	0	0.97	0.82	3.13	0	3.13	 	 Start now
OS X Catalina	 95	0	0.98	0.86	3.15	0	3.15	 	 Start now

Performance of peerprep.tech from SpeedLab

3. Scalability

Architecture Scaling



Overview of Scaling

PeerPrep handles the scalability by using:

- Cluster Autoscaler
- Horizontal Pod Autoscaler

Cluster Autoscaler allows automatic adjustment of the number of nodes in the cluster when pods fail or are rescheduled onto other nodes. This works together with Horizontal Pod Autoscaler, where when a surge in traffic is witnessed in a particular microservice, it will scale up to meet the demand.

The Kubernetes cluster is set to have a minimum of 2 nodes and maximum of 5 nodes. It scales based on the CPU and capacity utilization of the pods. Each pod is set to have a minimum of 2 pods and maximum of 4 pods. The scaling will occur when the pod's CPU utilization hits 80%.

```
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
kohvinleon@Kohs-Mac-mini ~/D/C/o/c/s/manifest (master)> k get nodes
NAME                                STATUS    ROLES    AGE    VERSION
ip-10-0-0-107.ap-southeast-1.compute.internal Ready    <none>   2d10h v1.21.4-eks-033c
e7e
ip-10-0-1-125.ap-southeast-1.compute.internal Ready    <none>   6m19s v1.21.4-eks-033c
e7e
ip-10-0-1-29.ap-southeast-1.compute.internal Ready    <none>   85m    v1.21.4-eks-033c
e7e
kohvinleon@Kohs-Mac-mini ~/D/C/o/c/s/manifest (master)>

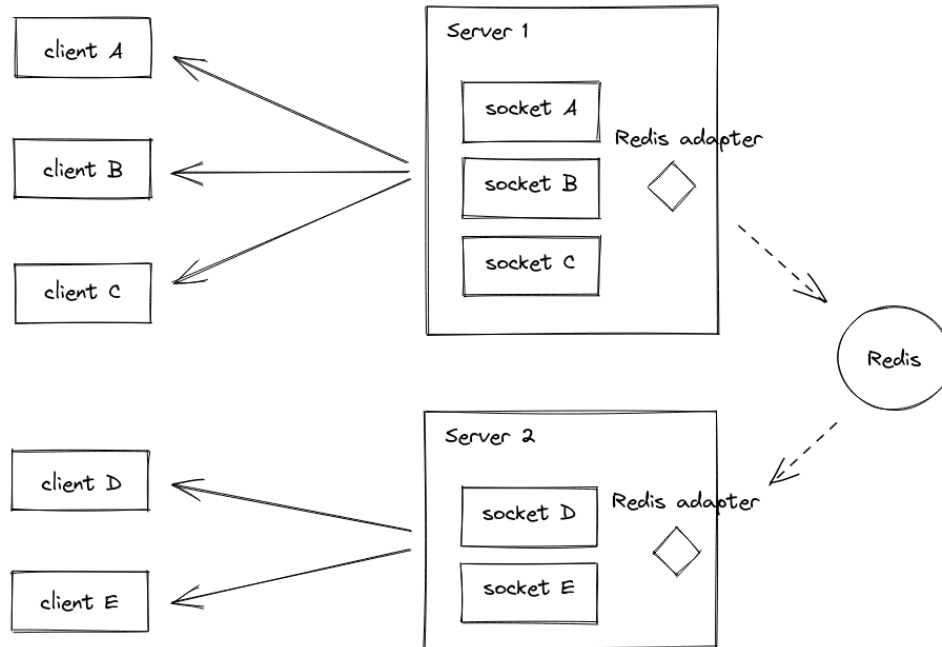
kohvinleon@Kohs-Mac-mini ~/D/C/o/c/s/manifest (master)> k get hpa -n peerprep
NAME      REFERENCE          TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
user      Deployment/user    239%/80%  2         4         4          64s
kohvinleon@Kohs-Mac-mini ~/D/C/o/c/s/manifest (master)>
```

Example of Cluster Autoscaler and HPA

WebSocket Scalability

For the Editor microservice, it utilizes WebSocket. For Match and Chat microservices, it utilizes Socket.IO. Both technologies open a two-way interactive communication session between the user's browser and a server.

One of the known limitations of WebSockets is that by itself, it is not scalable. To mitigate the scalability issue, Redis adapter is used to replace the default in-memory adapter to provide Pub/Sub functionality to ensure the events are properly routed to all clients.



Overview of Socket.IO on Multiple Nodes communicating using Redis

Limitation in WebRTC Scalability

For the Video Chat microservice, WebRTC is used. WebRTC is a popular technology that provides real-time communication, which is commonly used in video-conferencing applications. However, WebRTC was designed with scaling not in mind. Hence there are limitations to this aspect of scalability.

As it is an MVP feature and not an important functionality of PeerPrep as stated in the functional requirement, we decided not to explore the possibility of scaling WebRTC.

4. Security



Website Security

As introduced in November 2021, CloudFront Distribution now allows defining of custom header response policies, which used to be only achievable by using a Lambda Edge function previously.

A custom policy is created to be attached as the Viewer Response of the CloudFront Distribution that follows the Mozilla Observatory Security requirements.

Security headers info		
Strict-Transport-Security max-age: 63072000 (seconds) preload IncludeSubDomains Origin override	X-Content-Type-Options Origin override	X-Frame-Options Origin: DENY Origin override
X-XSS-Protection enabled Block Origin override	Referrer-Policy same-origin Origin override	Content-Security-Policy frame-ancestors 'none'; default-src 'none'; img-src 'self'; script-src 'self'; style-src 'self'; object-src 'none' Origin override

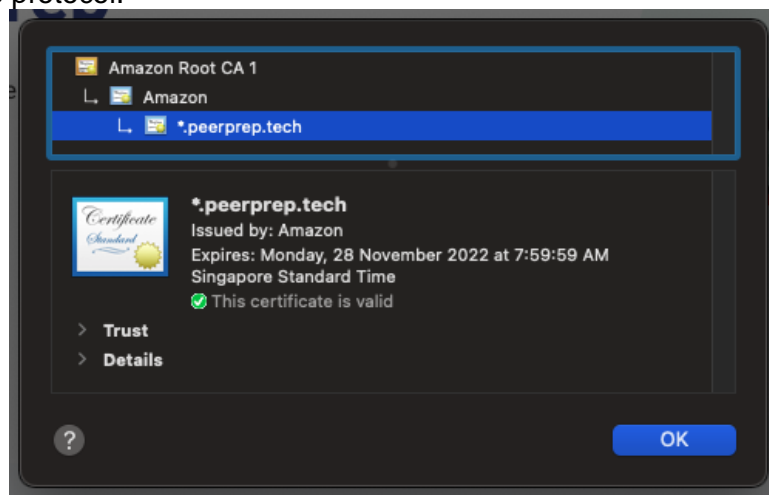
Security Headers defined for peerprep.tech and admin.peerprep.tech

Scan Summary		Scan Summary	
	Host: peerprep.tech		Host: admin.peerprep.tech
	Scan ID #: 22561312 (unlisted)		Scan ID #: 22552865 (unlisted)
	Start Time: November 8, 2021 10:25 AM		Start Time: November 8, 2021 1:01 AM
	Duration: 8 seconds		Duration: 5 seconds
	Score: 120/100		Score: 120/100
	Tests Passed: 11/11		Tests Passed: 11/11


Test Scores				
Test	Pass	Score	Reason	Info
Content Security Policy	✓	+10	Content Security Policy (CSP) implemented with <code>default-src 'none'</code> and no <code>'unsafe'</code>	i
Cookies	—	0	No cookies detected	i
Cross-origin Resource Sharing	✓	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	i
HTTP Public Key Pinning	—	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	i
HTTP Strict Transport Security	✓	0	HTTP Strict Transport Security (HSTS) header set to a minimum of six months (15768000)	i
Redirection	✓	0	Initial redirection is to HTTPS on same host, final destination is HTTPS	i
Referrer Policy	✓	+5	Referrer-Policy header set to <code>"no-referrer"</code> , <code>"same-origin"</code> , <code>"strict-origin"</code> or <code>"strict-origin-when-cross-origin"</code>	i
Subresource Integrity	—	0	Subresource Integrity (SRI) is only needed for html resources	i
X-Content-Type-Options	✓	0	X-Content-Type-Options header set to <code>"nosniff"</code>	i
X-Frame-Options	✓	+5	X-Frame-Options (XFO) implemented via the CSP <code>frame-ancestors</code> directive	i
X-XSS-Protection	✓	0	X-XSS-Protection header set to <code>"1; mode=block"</code>	i

Mozilla Observatory Security Test Score for peerprep.tech & admin.peerprep.tech

All PeerPrep endpoints have their own SSL certificate, which restricts to only communicating over the HTTPS protocol.

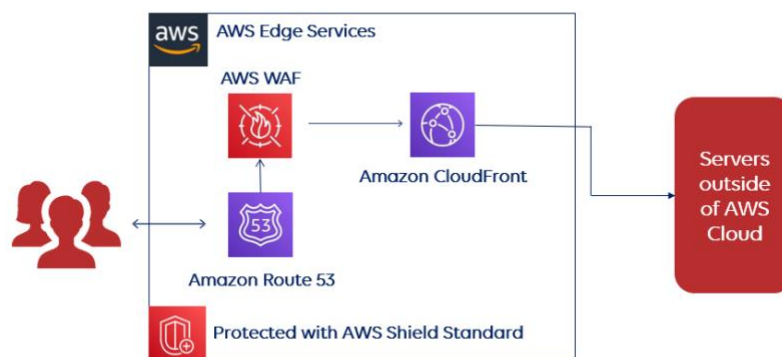


PeerPrep's SSL Certificate

Scan Summary	
	Host: peerprep.tech (99.86.37.29)
	Scan ID #: 46561990
	End Time: November 8, 2021 10:25 AM
	Compatibility Level: Intermediate
	Certificate Explainer: 188695716

Mozilla Observatory Security TLS Observatory Test

In addition, Amazon CloudFront comes with AWS Shield Standard which automatically protects against common DDoS attacks. Click-jacking is denied by X-Frame-Options as set in the policy headers.

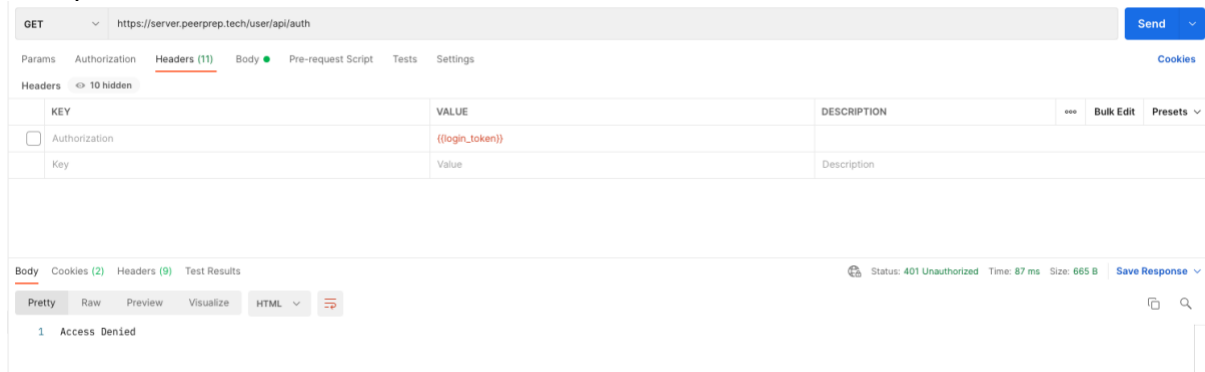


Source: <https://www.stormit.cloud/post/cloud-ddos-protection-how-to-mitigate-all-risks>

API Endpoint Security

JWT Authentication

A JWT Token is generated when a user is logged in, and a token validator middleware is implemented in the backend services to ensure that the user is logged in before performing certain actions such as finding a match. This provides a level of protection to ensure no data corruption.



Denied Access to the Endpoint if JWT is not present/valid

Role-Based Authorization

The API endpoints are split between the user (/api) and admin to define a role-based authorization control. The endpoints defined for the user (/api) are within their level of access rights, while endpoints defined for admin (/admin) usually consist of privileges such as CRUD.

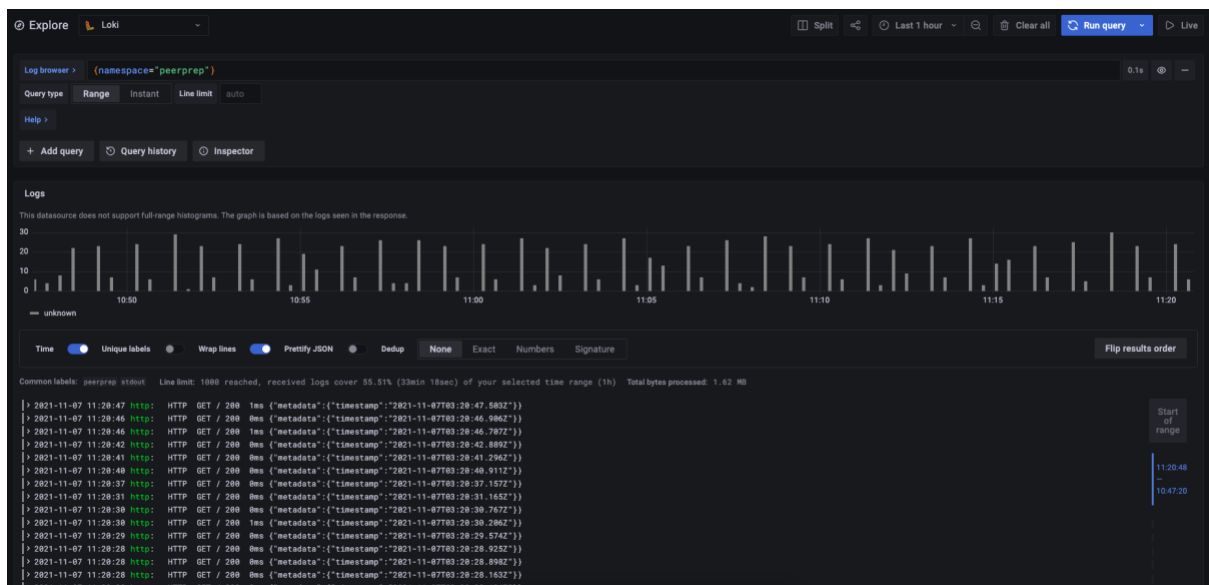
As mentioned in the JWT Authentication, the admin endpoints require a valid token as well. This prevents unauthorized users from accessing the endpoint. On top of that, the JWT token generated upon logging in uses different secrets for administrators. Hence, a normal user won't be able to access it as well.

Database Security

In our application, user passwords are not stored directly into the database. Instead, only password hashes are stored. Bcrypt is used in our application's implementation to hash user passwords with the help of salt and a secret key. In the case of a data leak and all user information is exposed, passwords will not be recoverable from their salted and hashed values.

Logging

All the microservices in the PeerPrep cluster logs down all the HTTP calls and requests that were made. By using Loki and Promtail, it allows a centralized gathering of logs which is displayed in the Grafana Dashboard. This allows monitoring of suspicious activities such as unsolicited access to the database and tracing of query that caused the server to crash if any.



Screenshot of Grafana Dashboard

5. Usability

An accessibility test was conducted, indicating problems for older users, people with disabilities or accessibility needs. PeerPrep passed all the accessibility tests, which means any common users can use it without problem.

Level	WCAG 2.1	Section 508 - 2017	Key
A	●	●	● Pages with level A issues are unusable for some people
AA	●	●	● Pages with level AA issues are very difficult to use
AAA	●	●	● Pages with level AAA issues can be difficult to use

Accessibility Test from PowerMapper

However, the test has flagged out a level A issue on *CSS animations that run for more than 5 seconds without giving the user a way to turn them off*. This animation is generated by Nuxt.js during the compilation of the static site, hence it's non-controllable from developer perspective.

6. Portability

Browser Compatibility

We chose to use Nuxt.js, since it is a framework of Vue.js. It is cross-browser compatible since most of its basic components are built from JavaScript. It supports almost all the most commonly used browsers and their various versions.

Summary

Issues

Pages

<https://peerprep.tech/>

Errors

Accessibility

Compatibility

Search

Standards

Usability

This tab shows pages that exhibit browser-specific behavior, or trigger browser bugs.

Browser	IE	Edge	Firefox	Safari	Opera	Chrome	iOS	Android	Key
Version	11	95	93	15	80	95	≤ 14	15	
Critical Issues	✓	✓	✓	✓	✓	✓	✓	✓	<div> <div></div> <div>Missing content or functionality</div> </div>
Major Issues	✓	✓	✓	✓	✓	✓	✓	✓	<div> <div></div> <div>Major layout or performance problems</div> </div>
Minor Issues	✓	✓	✓	✓	✓	✓	✓	✓	<div> <div></div> <div>Minor layout or performance problems</div> </div>

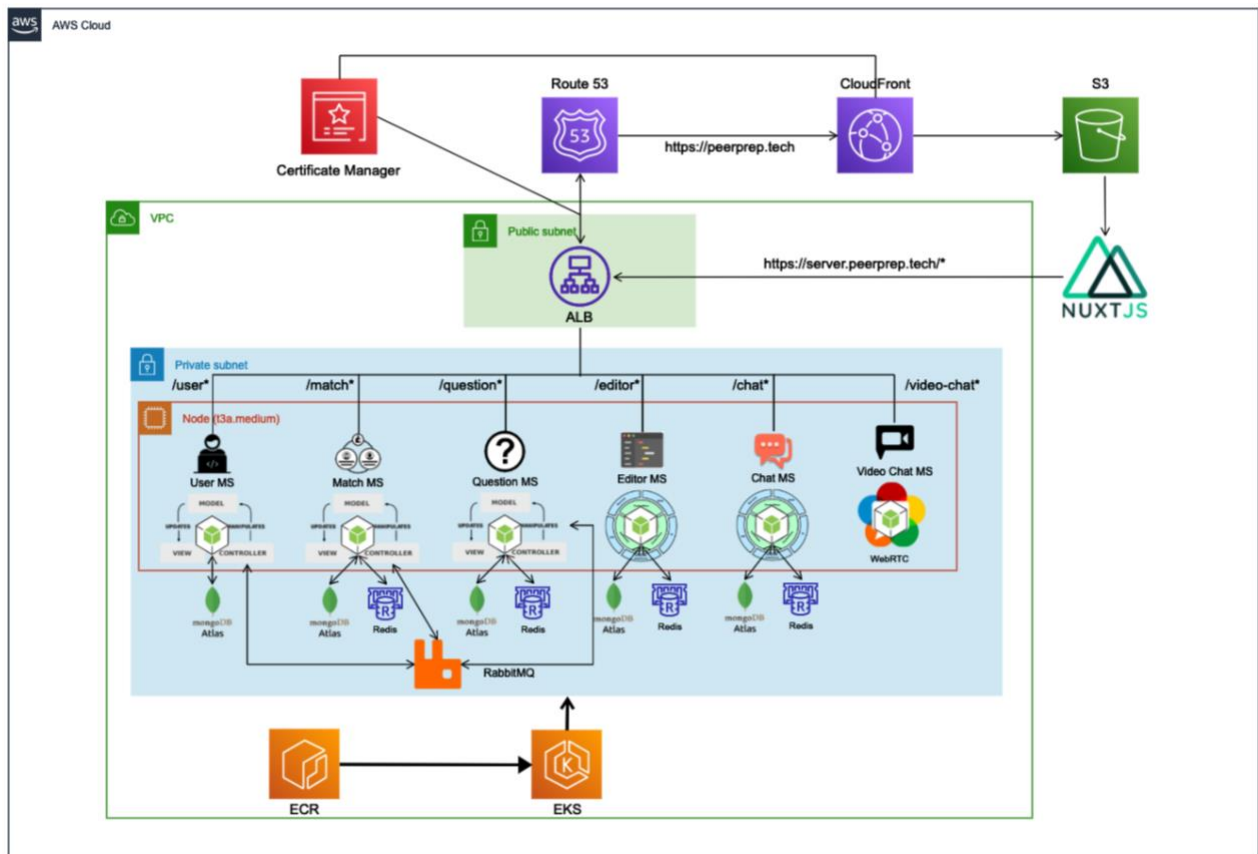
Compatibility Test from PowerMapper

Glossary

Terms	Definitions
Session	An event where a match discusses and works on a technical question in a collaborative whiteboard coding environment.
Matchmaking	The process of finding a suitable partner for a user to start a session.
Match	A pair of users who fit each other's technical requirements.
Rating	Score (from 1 to 5) given to a user by other users after a coding session based on compatibility.
GUI	The Graphical User Interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicators such as primary notation, instead of text-based user interfaces, typed command labels or text navigation.

Developer Documentation

Architecture Diagram



Architecture Diagram of PeerPrep

1. **Route 53** receives the inbound traffic from end-users. The route input will direct the user to either user frontend (<https://peerprep.tech>) or admin frontend (<https://admin.peerprep.tech>) to the CloudFront CDN, which access the static files generated by Nuxt.js.
2. **SSL certificate** is used to authorize HTTP Requests for both CloudFront and ALB.
3. When requests are made to the backend server via the ALB Alias (<https://server.peerprep.tech>), the requests are direct to the **ALB** in the **Public Subnet** of the **Virtual Private Cloud (VPC)**
4. The microservice layer resides in the **Private Subnet**, which is only accessible through ALB, providing a single point of entry.
5. The ALB, which also functions as an **ingress and service discovery** will direct the traffic to any of the instances of the requested service (e.g. <https://server.peerprep.tech/user> will route to the User Microservice)
6. Each service minimally follows an **MVC** (User, Match, Question) or **Clean Architecture** (Editor, Match) Design Pattern.
7. Each service has its own persistent database (**MongoDB**) and/or an ElastiCache (**Redis**) database in the Data Layer.
8. Interservice communication is done through **RabbitMQ**, where calls can be either RPC (if a response is required) or Pub/Sub (no response required)

Tech Stack

Some of the decisions are explained in the architectural decision section.

Component	Tech Stack
Frontend	NuxtJS (Framework of Vue.js)
Backend	NodeJS (with ExpressJS)
Database	MongoDB (Atlas) & Redis
Container Registry	Amazon Elastic Container Registry (ECR)
Deployment	Amazon Elastic Compute Cloud (EC2)
Clusters	Amazon EKS (Production) & GCP GKE (Staging)
Web Hosting	Amazon S3 & Cloudfront (as CDN)
Pub-Sub Messaging	CloudAMQP (RabbitMQ) & Redis
CI/CD	Github Actions (CI) & Flux Version 2 (CD, GitOps)
Monitoring & Logging	Prometheus, Grafana, Loki & Promtail
Infrastructure Tool	Terraform
Project Management Tool	Github Issues

Architectural Decision

1. System Architecture	
Problems Observed	A more conventional Monolithic Architecture has certain limitation: <ol style="list-style-type: none">1. Tightly coupled2. Limitation in size and complexity3. Size may slow down startup time4. Difficult to scale5. Less team autonomy and more potentially more merge conflicts
Architectural Decision	Adoption of Microservices Architecture <ol style="list-style-type: none">1. Allows breaking up a large application into loosely coupled modules2. Isolation of fault3. Able to scale a deployment instead of the entire backend4. Allows more team autonomy
Assumption	NIL
Alternatives	Monolithic Architecture
Justification	Microservices architecture allows ease in scaling and more team autonomy - each member can work on their assigned tasks with little to no dependency on other members.

2. Kubernetes Cluster	
Presumption	The Cloud Provider chosen for the PeerPrep project is AWS, as it provides a myriad of services that are beneficial to the project development. To have more centralized management of resources, EKS is chosen to be the production cluster.
Problems Observed	<ol style="list-style-type: none"> 1. Expensive to maintain EC2 instances on AWS 2. Not ideal to do thorough testing as additional costs are incurred if the system or application bugs out and may over-consume the resources 3. Limited by the credit limit, which may end up with some constraints in testing or forgoing some good practices
Architectural Decision	Splitting between Staging and Production Clusters <ul style="list-style-type: none"> - Staging Cluster hosted in GCP's GKE - Production Cluster hosted in AWS EKS
Assumption	Deployment between EKS and GKE to share same/similar resources files (manifests, ingress, etc.) with minimal modification.
Alternatives	Digital Ocean, Azure Kubernetes Service (AKS)
Justification	<p>EKS provides an upper hand in terms of usability and feasibility in deploying resources, but it comes with the high cost of maintaining EC2 instances - which may exceed the reimbursement amount.</p> <p>For this project, the staging backend is deployed to GKE to utilize the \$300 trial credits. Even though it is not ideal to have our clusters in a separate cloud provider and environment, it happens to work well in our development process.</p> <p>Furthermore, it gives insights into its respective platform's benefits and limitations. In addition, having a less restricted budget constraint on GKE also estimated how much resources we need on the actual production cluster EKS platform to achieve cost savings.</p>

3. API Gateway & Service Discovery	
Problems Observed	<ol style="list-style-type: none"> 1. No single point of entry to microservices 2. Manual routing 3. Security Issue (multiple endpoints) 4. Manual configuration of HTTPS per service
Architectural Decision	Using Amazon's Application Load Balancer (ALB) as an ingress controller as a single point of entry to the microservices.
Assumption	NIL
Alternatives	AWS API Gateway, Nginx Ingress Controller

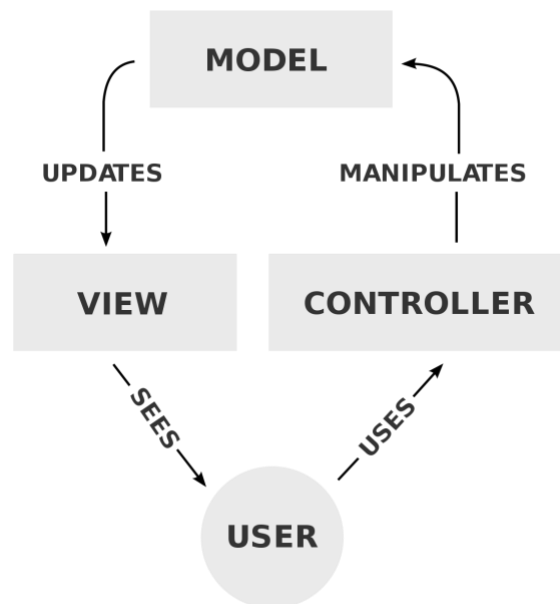
Justification	<p>After reviewing the functional and non-functional requirements, ALB fits the project requirements the best since the production cluster is already in Amazon EKS.</p> <p>In addition, it also provides additional benefits such as</p> <ul style="list-style-type: none"> • Content Based routing • Native HTTP/2 support • Native Websocket support • CloudWatch Metrics integration for service monitoring and alerting • EC2 Container Service (ECS) integration for managed container orchestration • AWS Certificate Manager (ACM) integration for free SSL certificates <p>At the same time, using ALB also fulfils the service discovery requirement as it performs content-based routing and target groups routing.</p> <p>For example, requests sent to <ALB_ENDPOINT>/user will be redirected to any instances with API containers listening on port 3001.</p>
----------------------	---

4. Interservice Communication	
Presumption	Interservice communication is done through REST API
Problems Observed	<ol style="list-style-type: none"> 1. Tighter Coupling which may breaks if the port number changed 2. Blocking Process 3. Inadequate Error Handling - request is gone after it fails
Architectural Decision	Using CloudAMQP, a RabbitMQ as Service platform.
Assumption	NIL
Alternatives	Traversing through the internal application load balancer and Kafka
Justification	<p>RabbitMQ is a mature message broker which provides several useful capabilities such as RPC, Pub/Sub, Topic, Routing, and others.</p> <p>Messages can be either synchronous or asynchronous. In the event that the CloudAMQP crashes, the message is still persistent in the queue and will be processed upon recovery.</p> <p>Asynchronous interservice communication can eliminate the blocking issue. Pub/Sub is especially useful as it can broadcast to multiple consumers without blocking.</p> <p>In addition, RabbitMQ allows extensibility as additional RabbitMQ capabilities such as Routing can be easily added to the existing codebase.</p>

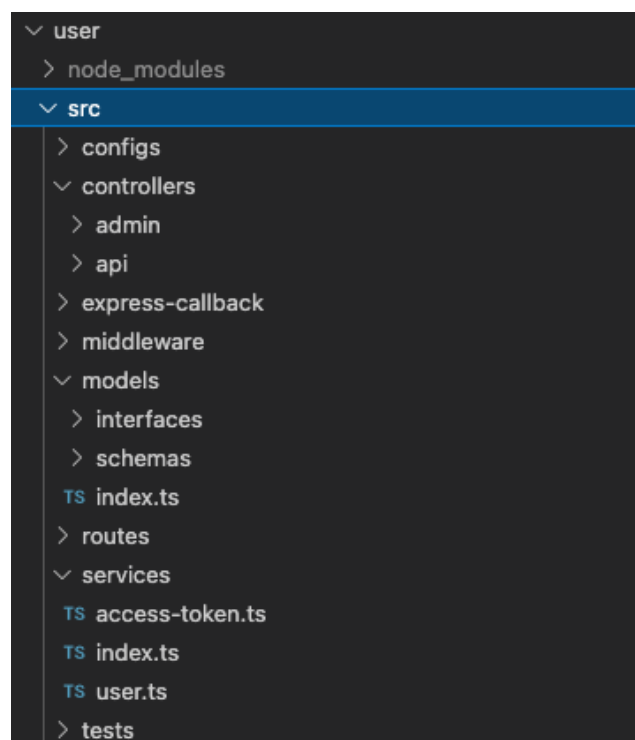
Software Architecture Design

MVC Pattern

The user, question, and match microservices adopt a **MVC (model-view-controller)** pattern as it separates concerns between model, view, and controller, reduces coupling and allows better software maintainability.



Overview of MVC



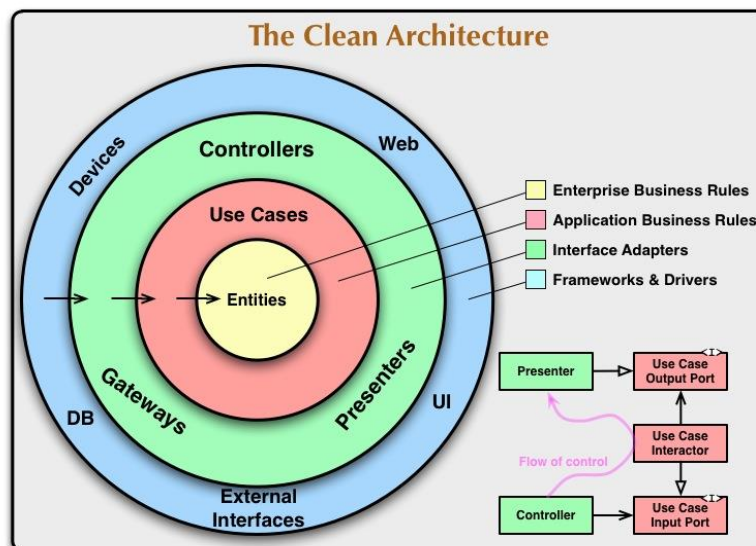
Folder Structure of User Microservice

Clean Architecture

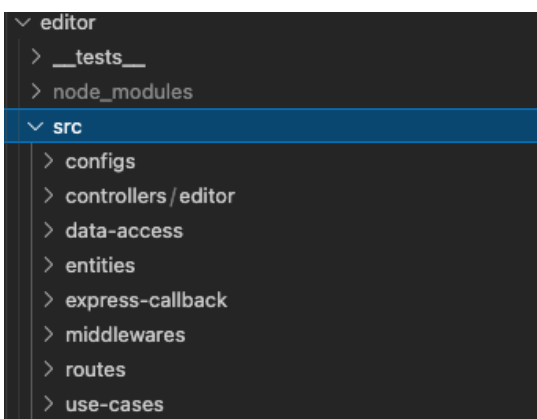
The Chat and Editor Microservices adopts the clean architecture design. This allows the designing of applications with very low connection and independent of technical implementation details, such as databases and frameworks. This also improves the testability of the application.

By adopting Clean Architecture, it fulfils the SOLID principles, which are

1. Single responsibility principle
2. Open-closed principle
3. Liskov substitution principle
4. Interface segregation principle
5. Dependency inversion principle



Clean Architecture Diagram



Folder Structure of Editor MS

```
/**
 * @description Use case to create a new editor
 * @function createEditor
 */
const createEditor = makeCreateEditor({ editorDb: EditorDb });

/**
 * @description Use case to get editor by ID
 * @function getEditorById
 */
const getEditorById = makeGetEditorById({ editorDb: EditorDb });
```

Port & Adapter

Messaging Pattern

Synchronous Message Call

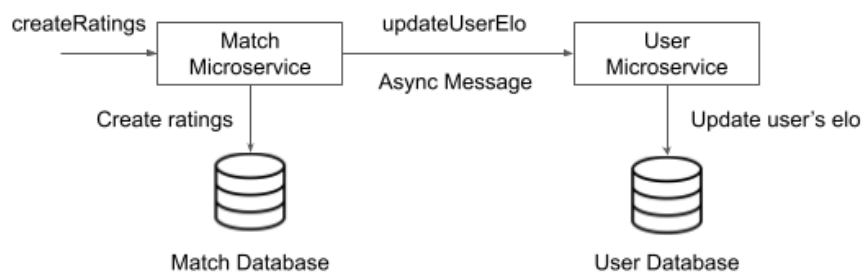
Remote Procedure Call (RPC) is used as one of the interservice communication methods. The match microservice needs to communicate with the user and question microservices to search for a partner and a suitable question, respectively. All the information is required before the user can start their match session; hence RPC fits the criteria. An detailed explanation and diagram is included in the Match Microservice [API](#) section.

Asynchronous Message Passing

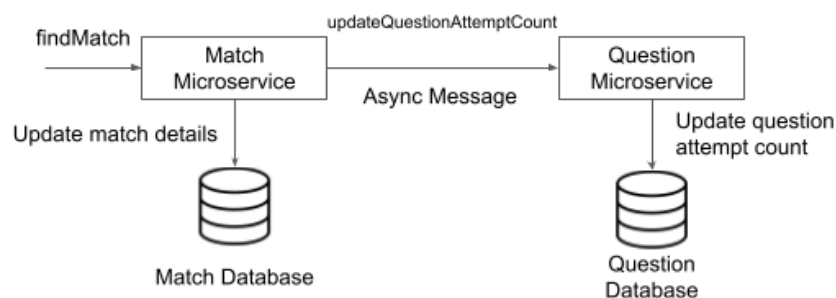
The P2P and Pub/Sub mechanism allows sending messages/requests to a single consumer or interested parties depending on requirements. This allows persistent asynchronous communication, and it is guaranteed that the messages will be eventually inserted in the recipient's queue.

This mechanism is used in 2 areas:

- Updating of user's elo after his/her partner has given the rating
- Updating of number of question attempts



Async Message Passing between Match MS and User MS

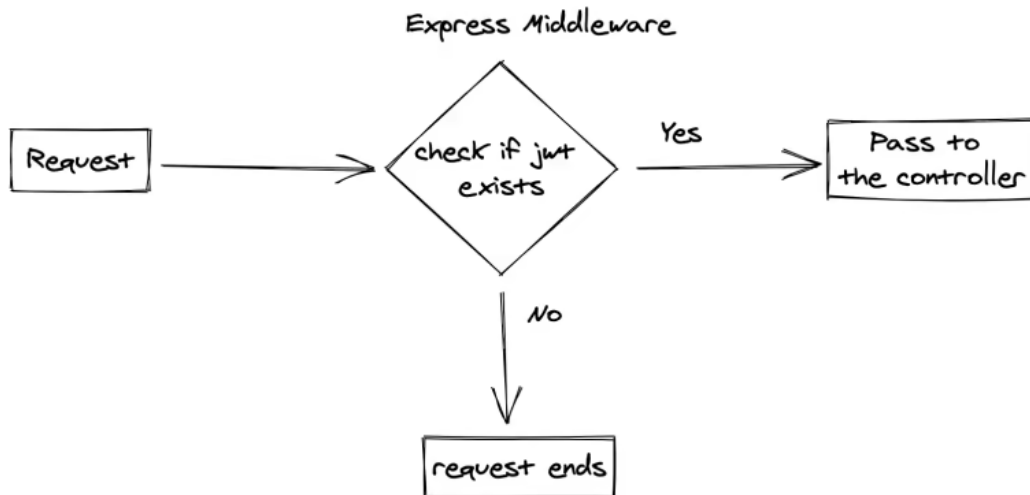


Async Message Passing between Match MS and Question MS

Design Pattern

Chain of Responsibility Pattern

The chain of responsibility allows an object to go through a chain of conditions or functionalities. Instead of managing all the functionalities and conditions in one place, it splits into chains of conditions that an object must pass through. This pattern is adopted in various areas namely input validation, JWT token and access controls.



Chain of Responsibility Example

```
import _ from "lodash";
import { Request, Response, NextFunction } from "express";

import Validator, { Rules } from "validatorjs";

export default function makeValidator(rules: Rules) {
  return async function validatorMiddleware(
    req: Request,
    res: Response,
    next: NextFunction,
  ): Promise<void | Response<any, Record<string, any>>> {
    const user = _.get(req, "user");
    const body = _.get(req, "body");
    const params = _.get(req, "params");
    const query = _.get(req, "query");

    const request_body = Object.assign({}, user, body, params, query);
    const validation = new Validator(request_body, rules);
    const passed = validation.passes();
    if (passed) {
      return next();
    }

    res.set({
      "Content-Type": "application/json",
    });
    res.type("json");
    return res.status(422).send(validation.errors);
  };
}
```

Express validator middleware that checks the validity of input

Singleton Pattern

The singleton pattern implies that there should be only one instance for a class. This is used in instantiating Redis to ensure only a single instance is instantiated and be used across the application.

```
import { createClient, RedisClient } from "redis";
import { promisify } from "util";

export default class Redis {
  private static redis_instance: Redis;
  redis_client: undefined | RedisClient;
  getAsync!: (key: string) => Promise<string | null>;
  setAsync!: (key: string, records: any, mode: string, duration: number) => Promise<any>;

  constructor() {
    if (Redis.redis_instance) {
      return;
    }
    const REDIS_ENDPOINT = process.env.REDIS_ENDPOINT;
    if (!REDIS_ENDPOINT) {
      console.warn("Redis Endpoint not found. Redis is not established");
      return;
    }
    const redis_client = createClient(REDIS_ENDPOINT, { auth_pass: process.env.REDIS_PASSWORD });
    redis_client.on("connect", () => {
      console.log("Succesfully connected to Redis");
    });

    this.redis_client = redis_client;
    this.getAsync = promisify(redis_client.get).bind(redis_client);
    this.setAsync = promisify(redis_client.set).bind(redis_client);
    Redis.redis_instance = this;
  }

  static getInstance(): Redis {
    if (Redis.redis_instance) {
      return Redis.redis_instance;
    }

    new Redis();
    return Redis.redis_instance;
  }
}

const redisClient = Redis.getInstance();

export { redisClient };
```

Singleton Pattern of instantiating Redis Client

Backend API

Backend Endpoint: <https://server.peerprep.tech>

Staging Backend Endpoint: <https://server-staging.peerprep.tech>

API Specifications

- The results returned by the API must be data or errors (following the [Google JSON guide](#)).
- Typically for **GET** requests, there are two variants: one will get a specific record by ID while the other will get all the records from the database.
- For **DELETE**, there are two variants: one will perform a soft delete while the another will perform a hard delete.

Error Resilience

API endpoints which requires route parameter (e.g. GET /api/user/:note_id) or message body (e.g. POST /api/user/) uses a validator middleware (Validatorjs) to ensure the required data is passed in along with the request. We can also specify the type of data or regex to be checked against with.

Example of a validator (create-user.ts)

```
const createUserRules = {  
  display_name: "required|string",  
  email: "required|email",  
  password: "required|string|min:8|confirmed",  
};  
  
export default createUserRules;
```

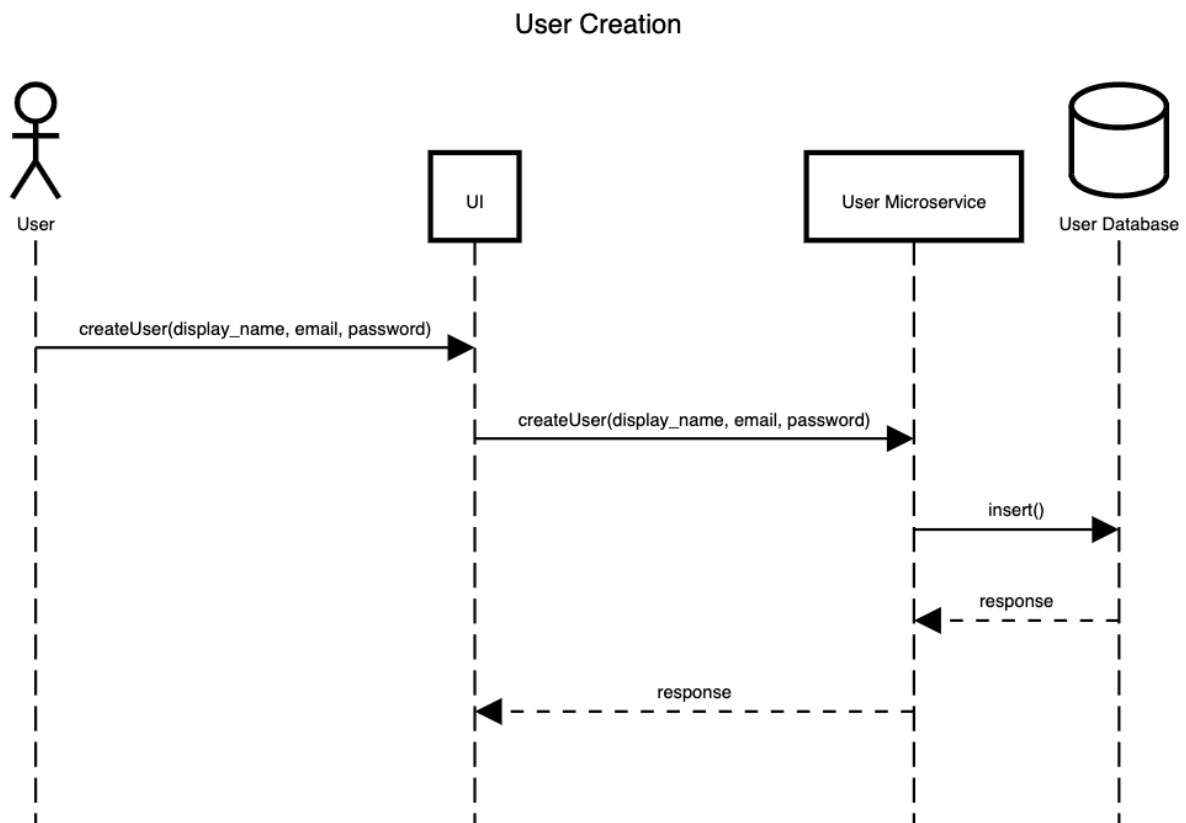
- **display_name** is required, and it should be a format of string
- **email** is required and it should be a format of email
- **password** is required and it should have minimum of 8 characters

If the data passed in fails the validation (e.g., missing route parameter, missing data in the message body, invalid data type, fails the regex), the status code is **422**.

If an error is encountered during the execution of a query, such as a record not found or an internal error, the status code will be **404**.

API (User) Endpoints

User Microservices



Method	Route	Description
POST	/user/api	Creates a new user
POST	/user/api/login	Login user
POST	/user/api/logout	Logout user
GET	/user/api/auth	Authenticate user's token validity
PUT	/user/api	Update user's profile

Create User

Method: POST

Route: /user/api

Description: Creates a new user

Data: **display_name**: string, **email**: string, **password**: string

Example Input	Output
<pre>{ "display_name": "prouser", "email": "prouser@gmail.com", "password": "p@ssw0rd" }</pre>	<pre>{ "data": { "_id": "618a1d6e3a7ca5989c5e55e8", "display_name": "prouser", "email": "prouser@gmail.com", "role": "user", "elo": 1000, "created_at": "2021-11-09T07:04:14.020Z", "updated_at": "2021-11-09T07:04:14.020Z", "__v": 0 } }</pre>

Login User

Method: POST

Route: /user/api/login

Description: Login user

Data: **email**: string, **password**: string

Example Input	Output
<pre>{ "email": "prouser@gmail.com", "password": "p@ssw0rd" }</pre>	<pre>{ "data": { "_id": "618a1d6e3a7ca5989c5e55e8", "display_name": "prouser", "email": "prouser@gmail.com", "role": "user", "elo": 1000, "created_at": "2021-11-09T07:04:14.020Z", "updated_at": "2021-11-09T07:04:14.020Z", "__v": 0 }, "login_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoiaWoidXNlciIsImhhdCI6MTYzNjQ0MjQxNiwiZXhwIjojY4MDAwMDE2fQ.4REdfXL7NhcF-EpCQ3yPwIE0FH2IjmG1ADb4BrAOVGM" }</pre>

Auth User**Method:** GET**Route:** /user/api/auth**Description:** Authenticate user**Header:** Authorization (JWT Token)**Output**

```
{
  "data": {
    "_id": "618a1d6e3a7ca5989c5e55e8",
    "display_name": "prouser",
    "email": "prouser@gmail.com",
    "role": "user",
    "elo": 1000,
    "created_at": "2021-11-09T07:04:14.020Z",
    "updated_at": "2021-11-09T07:04:14.020Z",
    "__v": 0
  }
}
```

Update User**Method:** PUT**Route:** /user/api/auth**Description:** Update existing user**Header:** Authorization (JWT Token)**Data:** _id: string, display_name: string (optional), password: string (optional)**Example Input**

```
{
  "_id": "618a1d6e3a7ca5989c5e55e8",
  "display_name": "prouser1"
}
```

Output

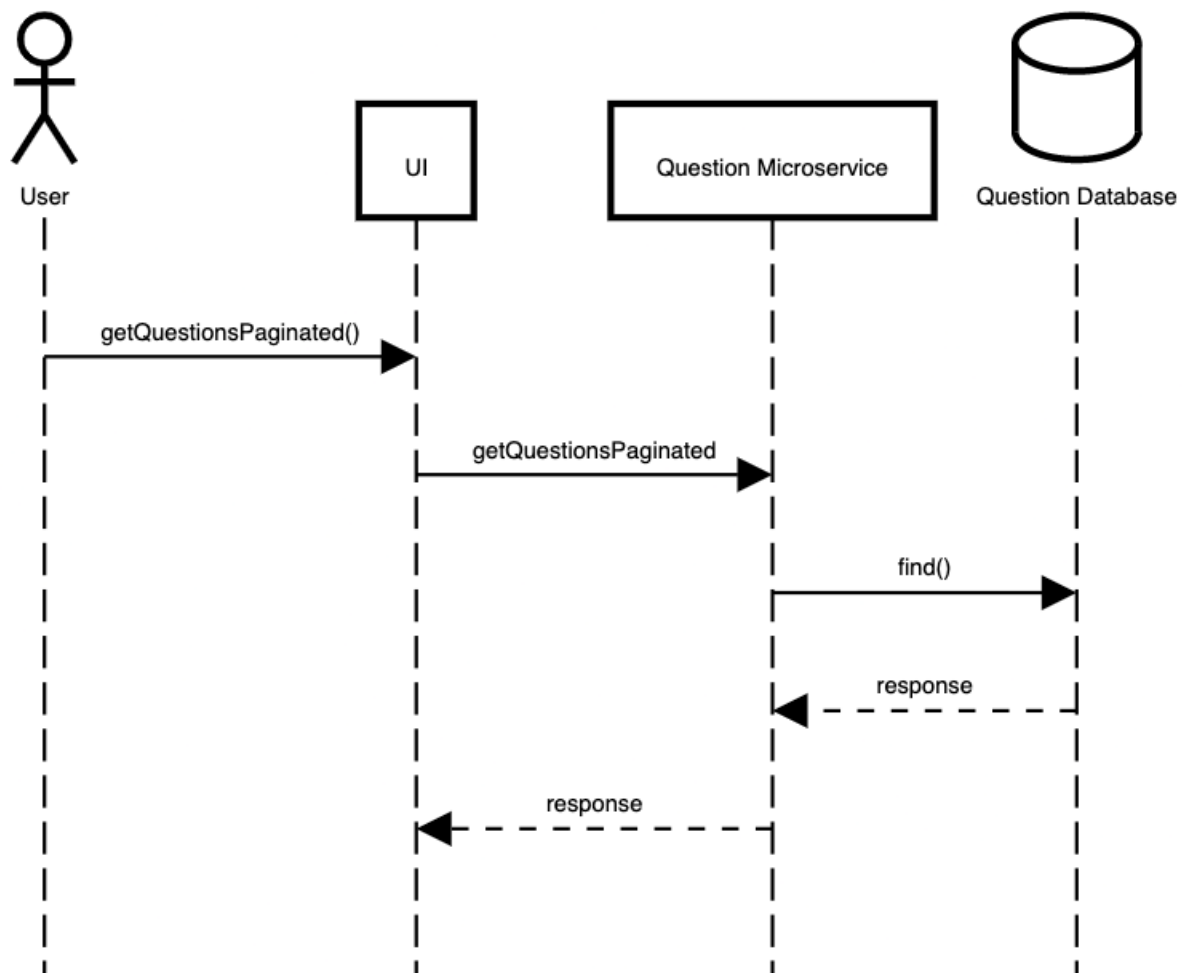
```
{
  "data": {
    "_id": "618a1d6e3a7ca5989c5e55e8",
    "display_name": "prouser1",
    "email": "prouser@gmail.com",
    "role": "user",
    "elo": 1000,
    "created_at": "2021-11-09T07:04:14.020Z",
    "updated_at": "2021-11-09T07:36:07.891Z",
    "__v": 0
  }
}
```

Logout User**Method:** POST**Route:** /user/api/logout**Description:** Logout user**Header:** Authorization (JWT Token)**Data:** email: string

Example Input	Output
<pre>{ "email": "prouser@gmail.com" }</pre>	<pre>{ "data": true }</pre>

Question Microservices

Viewing Question



Method	Route	Description
GET	/question/api	Get questions paginated
GET	/question/api/topic	Get the available topics
GET	/question/api/featured	Get the featured topics
GET	/question/api/:question_id	Get question details

Get Questions Paginated

Method: GET

Route: /question/api

Description: Get questions paginated

Parameter (Optional): **difficulty_levels**: array, **topics**: array, **query**: string, **page**: number

Example Output

```
{
  "data": [
    {
      "_id": "6188aa54741c47b88e17896a",
      "title": "Two Sum",
      "description": "Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.\nYou may assume that each input would have exactly one solution, and you may not use the same element twice.\nYou can return the answer in any order.",
      "difficulty": "easy",
      "recommended_duration": 30,
      "topic": "Algorithms",
      "hints": [
        "A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations."
      ],
      "examples": [
        {
          "input": "nums = [2,7,11,15], target = 9",
          "output": "[0,1]",
          "_id": "6188aa54741c47b88e17896b"
        }
      ],
      "constraints": [
        "2 <= nums.length <= 10",
        "-109 <= nums[i] <= 109",
        "-109 <= target <= 109",
        "Only one valid answer exists"
      ],
      "solution": "class Solution {\npublic:\n    vector<int> twoSum(vector<int>& nums, int target) {\n\n    }\n};",
      "number_of_attempts": 0,
      "created_at": "2021-11-08T04:40:52.478Z",
      "updated_at": "2021-11-08T04:40:52.478Z"
    }
  ],
  "pagination": {
    "current_page": 1,
```

```
"from": null,  
"to": null,  
"per_page": 15,  
"total": 1,  
"total_pages": 1  
}  
}
```

Get Questions Topics

Method: GET

Route: /question/api/topic

Description: Get question topics

Example Output

```
{  
  "data": [  
    "Algorithms",  
    "Data Structures"  
  ]  
}
```

Get Featured Topics

Method: GET

Route: /question/api/featured

Description: Get featured question topics

Example Output

```
{  
  "data": [  
    {  
      "_id": "Algorithms",  
      "count": 1,  
      "number_of_attempts": 0  
    }  
  ]  
}
```


Get Question By ID

Method: GET

Route: `/question/api/question_id`

Description: Get a specific question by its id

Parameter: `question_id`: string (required)

Example Output

```
{
  "data": {
    "_id": "6188aa54741c47b88e17896a",
    "title": "Two Sum",
    "description": "Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.\nYou may assume that each input would have exactly one solution, and you may not use the same element twice.\nYou can return the answer in any order.",
    "difficulty": "easy",
    "recommended_duration": 30,
    "topic": "Algorithms",
    "hints": [
      "A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations."
    ],
    "examples": [
      {
        "input": "nums = [2,7,11,15], target = 9",
        "output": "[0,1]",
        "_id": "6188aa54741c47b88e17896b"
      }
    ],
    "constraints": [
      "2 <= nums.length <= 10",
      "-109 <= nums[i] <= 109",
      "-109 <= target <= 109",
      "Only one valid answer exists"
    ],
    "solution": "class Solution {\n  public int[] twoSum(int[] nums, int target) {\n    for (int i = 0; i < nums.length; i++) {\n      for (int j = i + 1; j < nums.length; j++) {\n        if (nums[j] == target - nums[i]) {\n          return new int[] { i, j };\n        }\n      }\n    }\n    // In case there is no solution, we'll just return null\n    return null;\n  }\n}",
    "number_of_attempts": 0,
    "created_at": "2021-11-08T04:40:52.478Z",
    "updated_at": "2021-11-08T08:22:21.825Z",
    "__v": 0
  }
}
```

Match Microservices

Method	Route	Description
GET	/match/api/user/:user_id	Get match history paginated
GET	/match/api/:match_id	Get the match details
PUT	/match/api/end	Ends a match
POST	/match/api/rating	Creates a rating
GET	/match/api/rating/:receiver_id	Get average ratings by receiver_id (user)

Get Match History Paginated

Method: GET

Route: /match/api/user/:user_id

Description: Get match history paginated of the user

Header: Authorization (JWT Token)

Parameter: user_id: string (required)

Output

```
{
  "data": [
    {
      "_id": "61894b4cfd707d769fc0125e",
      "partner1_id": "6176a4bfe9bc6fe2726410d7",
      "partner2_id": "617b81a6e48a7dc8f13c8c3b",
      "question_id": "6188aa54741c47b88e17896a",
      "status": "in-progress",
      "mode": "elo",
      "match_requirements": {
        "programming_language": "python",
        "question_mode": "timed",
        "elo_match_pool": "61894b4795da03efceb7c1b0"
      },
      "matched_at": "2021-11-08T16:07:40.637Z",
      "updated_at": "2021-11-08T16:07:41.133Z",
      "created_at": "2021-11-08T16:07:40.643Z",
      "__v": 0,
      "meta": {
        "partner1_display_name": "Jon",
        "partner2_display_name": "vinleon",
        "question_title": "Two Sum"
      }
    }
  ]
}
```

Get Match

Method: GET

Route: `/match/api/match_id`

Description: Get match details

Header: `Authorization` (JWT Token)

Parameter: `match_id`: string (required)

Output

```
{
  "data": {
    "_id": "61894b4cfd707d769fc0125e",
    "partner1_id": "6176a4bfe9bc6fe2726410d7",
    "partner2_id": "617b81a6e48a7dc8f13c8c3b",
    "question_id": "6188aa54741c47b88e17896a",
    "status": "in-progress",
    "mode": "elo",
    "match_requirements": {
      "programming_language": "python",
      "question_mode": "timed",
      "elo_match_pool": "61894b4795da03efceb7c1b0"
    },
    "matched_at": "2021-11-08T16:07:40.637Z",
    "updated_at": "2021-11-08T16:07:41.133Z",
    "created_at": "2021-11-08T16:07:40.643Z",
    "__v": 0,
    "meta": {
      "partner1_display_name": "Jon",
      "partner2_display_name": "vinleon",
      "question_title": "Two Sum"
    },
    "question": {
      "_id": "6188aa54741c47b88e17896a",
      "title": "Two Sum",
      "description": "Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.\nYou may assume that each input would have exactly one solution, and you may not use the same element twice.\nYou can return the answer in any order.",
      "difficulty": "easy",
      "recommended_duration": 30,
      "topic": "Algorithms",
      "hints": [
        "A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations."
      ],
      "examples": [
```

```

    {
      "input": "nums = [2,7,11,15], target = 9",
      "output": "[0,1]",
      "_id": "6188aa54741c47b88e17896b"
    }
  ],
  "constraints": [
    "2 <= nums.length <= 10",
    "-109 <= nums[i] <= 109",
    "-109 <= target <= 109",
    "Only one valid answer exists"
  ],
  "solution": "class Solution {\n  public int[] twoSum(int[] nums, int target) {\n    for (int i = 0; i < nums.length;\n    i++) {\n      for (int j = i + 1; j < nums.length; j++) {\n        if (nums[j] == target - nums[i]) {\n          return\n          new int[] { i, j };\n        }\n      }\n    }\n    // In case there is no solution, we'll just return null\n    return\n    null;\n  }\n}",
  "number_of_attempts": 14,
  "created_at": "2021-11-08T04:40:52.478Z",
  "updated_at": "2021-11-08T16:07:40.758Z",
  "__v": 0
},
"partner1": {
  "_id": "617b81a6e48a7dc8f13c8c3b",
  "display_name": "Jon",
  "email": "fatedtime@hotmail.com",
  "role": "user",
  "elo": 1600,
  "created_at": "2021-10-29T05:07:50.615Z",
  "updated_at": "2021-11-03T17:41:08.826Z",
  "__v": 0
},
"partner2": {
  "_id": "6176a4bfe9bc6fe2726410d7",
  "display_name": "vinleon",
  "email": "kohvinleon@gmail.com",
  "role": "user",
  "created_at": "2021-10-25T12:36:15.923Z",
  "updated_at": "2021-11-02T15:02:51.131Z",
  "__v": 0,
  "elo": 1500
}
}
}

```

End Match

Method: PUT

Route: `/match/api/end`

Description: End a match

Header: `Authorization` (JWT Token)

Data: `match_id`: string

Example Input	Output
<pre>{ "match_id": "61894b4cfd707d769fc0125e" }</pre>	<pre>{ "data": true }</pre>

Create Ratings

Method: POST

Route: `/match/api/rating`

Description: Create ratings

Header: `Authorization` (JWT Token)

Data: `match_id`: string, `giver_id`: string, `rating`: number

Example Input	Output
<pre>{ "match_id": "61894b4cfd707d769fc0125e", "giver_id": "6176a4bfe9bc6fe2726410d7", "rating": 5 }</pre>	<pre>{ "data": true }</pre>

Get Average Ratings

Method: PUT

Route: `/match/api/rating/:receiver_id`

Description: End a match

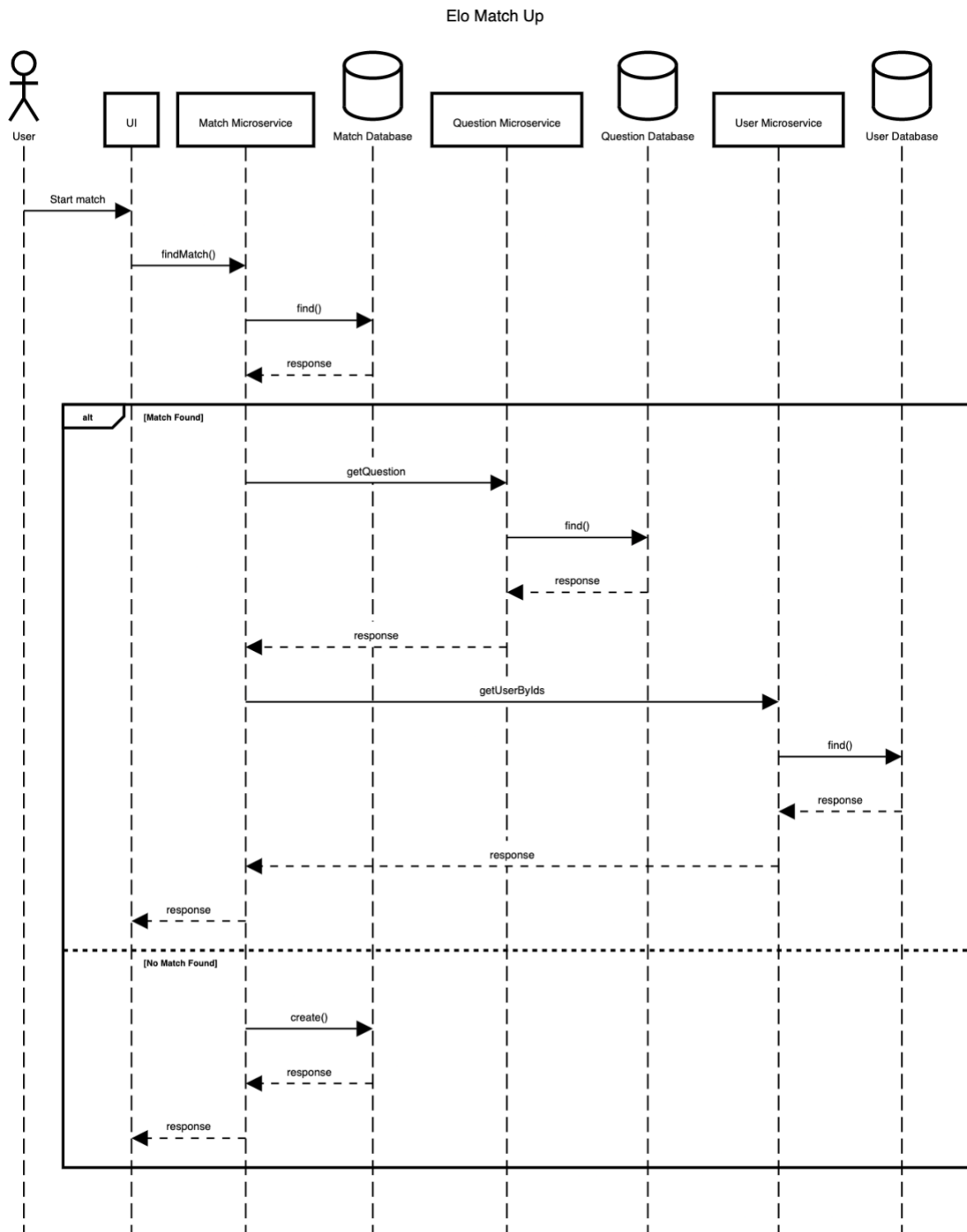
Header: `Authorization` (JWT Token)

Parameter: `receiver_id`: string (required)

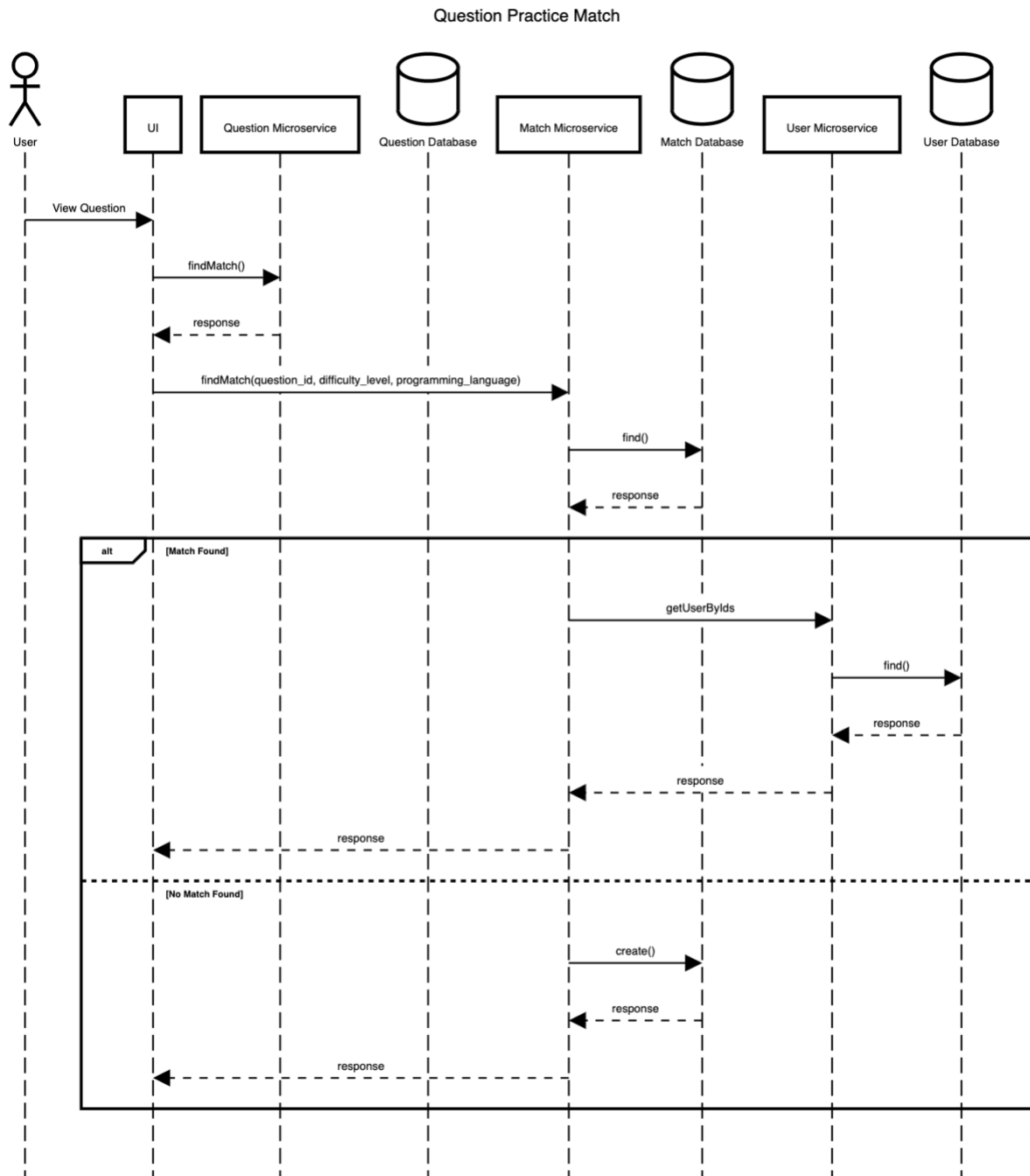
Output
<pre>{ "data": 4.333333333333333 }</pre>

Real-time Match System

For matching, it uses Socket.IO to enable real-time communication between the user and the server to simulate a queue in searching for matches.



Sequence Diagram for Elo Match Up



Sequence Diagram for Question Practice Match

As shown in the sequence diagrams, for matching, it is split into 2 types:

1. Elo Match Up
2. Question Practice Mode

Elo Match Up

When a user searches for a match, it will first search in the database whether there is a user that matches their requirement (e.g., within the ± 250 from the user's current elo, same difficulty level and chosen same programming language).

Question Practice Mode

Users can search for practice matches via the Question's Page dialog.

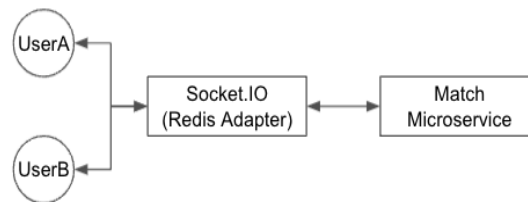
When a user searches for a match, it will first search in the database whether there is a user that matches their requirement (e.g., selected question and chosen same programming language).

Matching Mechanism

If there is no match, the Match Microservice will create a match in the database with the status “waiting”. The user will be put in the room in the Socket.IO of the match_id for 30 seconds until a match is found. The match_id is generated uniquely; hence no third party can listen in to this room.

If there is a match, the Match Microservice will do an RPC call to the Question Database to pick a question of the difficulty and topic chosen randomly. At the same time, the Match Microservice will also do an RPC call to the User Database to retrieve both users’ information for the ease of supporting the editor and chat at a later stage. Finally, both users will be redirected to the session page.

If there is no match, a notification will be sent to inform the user. Then, the user can choose to queue again or abort.



Overview of Real-time Match System

Chat Socket.IO

Socket	Route	Payload	Description
On	elo_matching	match_details: { user_id: string, programming_language: string, difficulty: string, user_elo: string, topic: string }	Search for a elo match
On	question_matching	match_details: { user_id: string, programming_language: string, match_requirements: { question_id: string question_mode: string }}}	Search for a practice question match
Emit	matched	match_details: Match	Match found, redirect user to session page
Emit	waiting	match_id or elo_match_pool_id	Put the user to wait
On	elo_cancel		Cancels a Elo match
On	question_cancel		Cancels a practice question match

Editor & Chat

Method	Route	Description
GET	/editor/api/match/:match_id	Get editor history by match_id
GET	/chat/api/match/:match_id	Get chat history by match_id
POST	/editor/api/execute	Execute code

Get Editor by Match ID

Method: GET

Route: /editor/api/match/:match_id

Description: Get editor history by match_id

Header: Authorization (JWT Token)

Parameter: match_id: string (required)

Output

```
{
  "data": {
    "_id": "618a9717b22002b1c25c467f",
    "match_id": "618a953b87749e5fb984a10d",
    "content": "def greet(name):\n    \"\"\"\n    This function greets to\n    the person passed in as\n    a parameter\n    \"\"\"\n    print('Hello, ' + name + '. Good morning!')\ngreet('Jon')",
    "created_at": "2021-11-09T15:43:19.142Z",
    "updated_at": "2021-11-09T15:43:23.980Z"
  }
}
```

Get Chat by Match ID

Method: GET

Route: /chat/api/match/:match_id

Description: Get chat history by match_id

Header: Authorization (JWT Token)

Parameter: match_id: string (required)

Output

```
{
  "data": {
    "_id": "618a971cb9a2b66819359d66",
    "match_id": "618a953b87749e5fb984a10d",
    "content": [
      {
        "user_id": "6176a4bfe9bc6fe2726410d7",
        "display_name": "vinleon",
        "message": "Hello there",
        "time_sent": "2021-11-09T15:40:10.362Z",
        "_id": "618a971cb9a2b66819359d67"
      },
      {

```

```

        "user_id": "617b81a6e48a7dc8f13c8c3b",
        "display_name": "Jon",
        "message": "Hey\n",
        "time_sent": "2021-11-09T15:40:14.962Z",
        "_id": "618a971cb9a2b66819359d68"
    }
],
"created_at": "2021-11-09T15:43:24.173Z",
"updated_at": "2021-11-09T15:43:24.173Z"
}
}

```

Execute Code

Method: POST

Route: `/editor/api/execute`

Description: Execute code

Header: `Authorization` (JWT Token)

Data: `code`: string, `language`: string, `input`: string (optional)

Example Input	Output
<pre> { "code": "def greet(name):\n \"\\\"\\\"\\\" This function\n greets to\\n the person passed in as\\n a\n parameter\\n \"\\\"\\\"\\\" print(\"Hello, \" + name + \".\n Good morning!\")\ngreet(\"Jon\")", "language": "python" } </pre>	<pre> { "data": { "sourceCode": "def greet(name):\n \"\\\"\\\"\\\" This\n function greets to\\n the person passed in as\\n a\n parameter\\n \"\\\"\\\"\\\" print(\"Hello, \" + name + \". Good\n morning!\")\ngreet(\"Jon\")", "status": 0, "errorCode": 0, "error": null, "outputType": 0, "output": "Hello, Jon. Good morning!\n", "outputStyle": null, "date": "0001-01-01T00:00:00", "language": "py", "input": null, "id": 0 } } </pre>

Real-time Editor, Chat and Video Chat

For the code editor, it is using y-monaco, which is powered by WebSockets. The WebSockets is managed by the editor microservices. Similar to Match, the chat microservices use Socket.IO (Redis Adapter). Both WebSockets and Socket.IO are used to allow real-time communication between the client browser and the server.

Similar to Match, when users are in a match session, the user will be put into the same room in the socket identified by the uniquely generated match_id using MongoDB ObjectId.

The client will listen to the socket of the room identified by match_id. This allows the client to update when incoming changes are detected.

By using Socket.IO with the Redis Adapter, Socket.IO uses the Redis Pub/Sub capability under the hood to propagate messages between servers, and servers will propagate messages to clients. Redis is also used to cache data. If the user happens to be disconnected, the contents sent previously will still be available.

Editor WebSocket

Socket	Route	Payload	Description
On	message	WSData	Updates the chat editor information such as code and cursor.
on	close	NIL	Close the socket session and save into persistent database

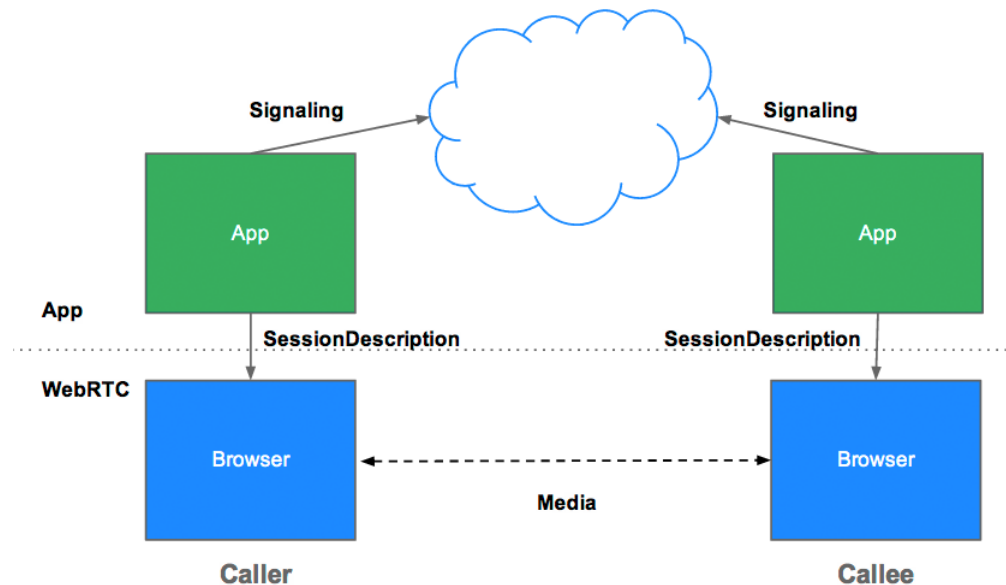
Chat Socket.IO

Socket	Route	Payload	Description
On	room	room (match_id): string	Joins the room (channel)
On / Emit	message	match_id: string, payload: { user_id: string, display_name: string, message: string, time_sent: Date }	Sends/Receive chat messages
Emit	end_session	match_id: string payload: [{ user_id: string, display_name: string, message: string, time_sent: Date }]	Ends the session and save information to persistent database

Video Chat

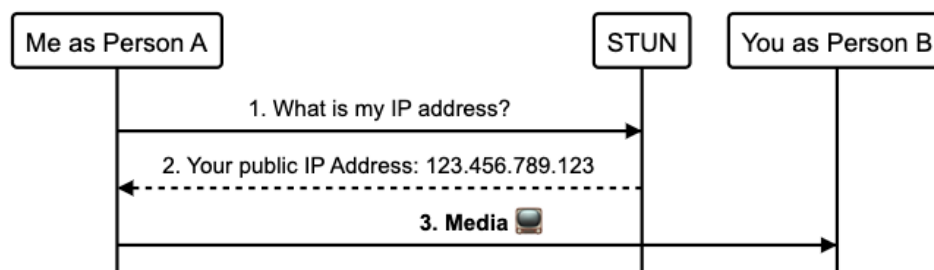
For Video Chat microservice, it uses **WebRTC** to allow real-time communication capabilities and it is used in popular video conferencing application as well. It uses signalling. It is the process of finding each other and then coordinating communication through an exchange of media information.

Signalling makes use of the **Session Description Protocol** (SDP) for gathering network information such as Internet Protocol address (IP address) and port numbers used for media exchange



Overview of Video Chat via WebRTC

A STUN (Session Traversal Utilities for Nat) server is also used to retrieve the remote peer's public IP addresses. This information will then be part of the SDP information you need to send over to your peers when you have just arrived.



How STUN works

Admin Endpoints

Question Microservices

Method	Route	Description
GET	/question/admin	Get all questions
GET	/question/api/:question_id	Get a question
POST	/question/admin	Creates a new question
PUT	/question/admin	Updates a new question
DELETE	/question/admin/:question_id	Deletes a specific question

Get All Questions

Method: GET

Route: /question/admin

Description: Get all questions in the database

Header: Authorization (JWT Token)

Parameter (Optional): difficulty_levels: array, topics: array, query: string, page: number

Example Output

```
{
  "data": [
    {
      "_id": "6188aa54741c47b88e17896a",
      "title": "Two Sum",
      "description": "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.",
      "difficulty": "easy",
      "recommended_duration": 30,
      "topic": "Algorithms",
      "hints": [
        "A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations."
      ],
      "examples": [
        {
          "input": "nums = [2,7,11,15], target = 9",
          "output": "[0,1]",
          "_id": "6188aa54741c47b88e17896b"
        }
      ]
    }
  ],
}
```

```

    "constraints": [
      "2 <= nums.length <= 10",
      "-109 <= nums[i] <= 109",
      "-109 <= target <= 109",
      "Only one valid answer exists"
    ],
    "solution": "class Solution {\npublic:\n    vector<int> twoSum(vector<int>& nums, int target) {\n\n    }\n};",
    "number_of_attempts": 0,
    "created_at": "2021-11-08T04:40:52.478Z",
    "updated_at": "2021-11-08T04:40:52.478Z"
  }
],
}

```

Get Question By ID

Method: GET

Route: /question/admin/:question_id

Description: Get a specific question by its id

Header: Authorization (JWT Token)

Parameter: question_id: string (required)

Example Output

```

{
  "data": {
    "_id": "6188aa54741c47b88e17896a",
    "title": "Two Sum",
    "description": "Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.\nYou may assume that each input would have exactly one solution, and you may not use the same element twice.\nYou can return the answer in any order.",
    "difficulty": "easy",
    "recommended_duration": 30,
    "topic": "Algorithms",
    "hints": [
      "A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations."
    ],
    "examples": [
      {
        "input": "nums = [2,7,11,15], target = 9",
        "output": "[0,1]",
        "_id": "6188aa54741c47b88e17896b"
      }
    ]
  },
}

```

```

"constraints": [
  "2 <= nums.length <= 10",
  "-109 <= nums[i] <= 109",
  "-109 <= target <= 109",
  "Only one valid answer exists"
],
"solution": "class Solution {\n  public int[] twoSum(int[] nums, int target) {\n    for (int i = 0; i <
nums.length; i++) {\n      for (int j = i + 1; j < nums.length; j++) {\n        if (nums[j] == target - nums[i]) {\n
return new int[] { i, j };\n      }\n    }\n    // In case there is no solution, we'll just return null\n
return null;\n  }\n}",
"number_of_attempts": 0,
"created_at": "2021-11-08T04:40:52.478Z",
"updated_at": "2021-11-08T08:22:21.825Z",
"__v": 0
}}

```

Create a New Question

Method: POST

Route: /question/admin/

Description: Create a new question

Header: Authorization (JWT Token)

Data: **title:** string (required), **description:** string (required), **difficulty:** string (required), **hints:** array, **solution:** string (required), **recommended_duration:** number (required), **examples:** array, **constraints:** array

Example Input	Example Output
<pre> { "title": "Two Sum", "description": "Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.\nYou may assume that each input would have exactly one solution, and you may not use the same element twice.\nYou can return the answer in any order.", "difficulty": "easy", "recommended_duration": 30, "topic": "Algorithms", "hints": ["A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations."], </pre>	<pre> { "data": { "_id": "6188aa54741c47b88e17896a", "title": "Two Sum", "description": "Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.\nYou may assume that each input would have exactly one solution, and you may not use the same element twice.\nYou can return the answer in any order.", "difficulty": "easy", "recommended_duration": 30, "topic": "Algorithms", "hints": ["A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you </pre>

<pre> "examples": [{ "input": "nums = [2,7,11,15], target = 9", "output": "[0,1]", "_id": "6188aa54741c47b88e17896b" }], "constraints": ["2 <= nums.length <= 10", "-109 <= nums[i] <= 109", "-109 <= target <= 109", "Only one valid answer exists"], "solution": "class Solution {\n public int[]\n twoSum(int[] nums, int target) {\n for (int i = 0; i <\n nums.length; i++) {\n for (int j = i + 1; j <\n nums.length; j++) {\n if (nums[j] == target -\n nums[i]) {\n return new int[] { i,\n j };\n }\n }\n }\n // In case\n there is no solution, we'll just return null\n return\n null;\n }\n}" </pre>	<p>can come up with optimizations."</p> <pre> }, "examples": [{ "input": "nums = [2,7,11,15], target = 9", "output": "[0,1]", "_id": "6188aa54741c47b88e17896b" }], "constraints": ["2 <= nums.length <= 10", "-109 <= nums[i] <= 109", "-109 <= target <= 109", "Only one valid answer exists"], "solution": "class Solution {\n public int[] twoSum(int[]\n nums, int target) {\n for (int i = 0; i < nums.length; i++)\n {\n for (int j = i + 1; j < nums.length; j++) {\n if (nums[j] == target - nums[i]) {\n return new int[]\n { i, j };\n }\n }\n }\n // In case there is\n no solution, we'll just return null\n return null;\n }\n}", "created_at": "2021-11-08T04:40:52.478Z", "updated_at": "2021-11-08T08:22:21.825Z", "__v": 0 } </pre>
--	--

Update a Question

Method: PUT

Route: /question/admin/

Description: Updates an existing question

Header: Authorization (JWT Token)

Data: _id: string (required), title: string, description: string, difficulty: string, hints: array,

solution: string, recommended_duration: number, examples: array, constraints: array

Example Input	Example Output
<pre> { "_id": "6188aa54741c47b88e17896a", "topic": "Data Structures" } </pre>	<pre> { "data": { "_id": "6188aa54741c47b88e17896a", "title": "Two Sum", "description": "Given an array of integers nums and an\ninteger target, return indices of the two numbers such that\nthey add up to target.\nYou may assume that each input\nwould have exactly one solution, and you may not use the\nsame element twice.\nYou can return the answer in any\norder.", </pre>

	<pre> "difficulty": "easy", "recommended_duration": 30, "topic": "Data Structures", "hints": ["A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations."], "examples": [{ "input": "nums = [2,7,11,15], target = 9", "output": "[0,1]", "_id": "6188aa54741c47b88e17896b" }], "constraints": ["2 <= nums.length <= 10", "-109 <= nums[i] <= 109", "-109 <= target <= 109", "Only one valid answer exists"], "solution": "class Solution {\n public int[] twoSum(int[] nums, int target) {\n for (int i = 0; i < nums.length; i++) {\n for (int j = i + 1; j < nums.length; j++) {\n if (nums[j] == target - nums[i]) {\n return new int[] { i, j };\n }\n }\n // In case there is no solution, we'll just return null\n return null;\n }\n}", "created_at": "2021-11-08T04:40:52.478Z", "updated_at": "2021-11-09T08:22:21.825Z", "__v": 0 }} </pre>
--	---

Delete a Question

Method: DELETE

Route: `/question/admin/:question_id`

Description: Delete an existing question

Header: `Authorization` (JWT Token)

Parameter: `:question_id`: string (required)

Example Output

```
{
  "data": {
    "_id": "6188aa54741c47b88e17896a",
    "title": "Two Sum",
    "description": "Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.\nYou may assume that each input would have exactly one solution, and you may not use the same element twice.\nYou can return the answer in any order.",
    "difficulty": "easy",
    "recommended_duration": 30,
    "topic": "Algorithms",
    "hints": [
      "A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations."
    ],
    "examples": [
      {
        "input": "nums = [2,7,11,15], target = 9",
        "output": "[0,1]",
        "_id": "6188aa54741c47b88e17896b"
      }
    ],
    "constraints": [
      "2 <= nums.length <= 10",
      "-109 <= nums[i] <= 109",
      "-109 <= target <= 109",
      "Only one valid answer exists"
    ],
    "solution": "class Solution {\n  public int[] twoSum(int[] nums, int target) {\n    for (int i = 0; i < nums.length; i++) {\n      for (int j = i + 1; j < nums.length; j++) {\n        if (nums[j] == target - nums[i]) {\n          return new int[] { i, j };\n        }\n      }\n    }\n    // In case there is no solution, we'll just return null\n    return null;\n  }\n}",
    "created_at": "2021-11-08T04:40:52.478Z",
    "updated_at": "2021-11-08T08:22:21.825Z",
    "deleted_at": "2021-11-12T12:20:52.258Z",
    "__v": 0
  }
}
```

Match Microservices

Method	Route	Description
GET	/match/admin	Get all matches
GET	/match/admin/:match_id	Get the match details
PUT	/match/admin	Update a match
DELETE	/match/admin/:match_id	Soft-delete a match

Get Match

Method: GET

Route: /match/admin

Description: Get all match details

Header: Authorization (JWT Token)

Output

```
{
  "data": [
    {
      "match_requirements": {
        "programming_language": "python",
        "question_mode": "timed",
        "elo_match_pool": "61894b4795da03efceb7c1b0"
      },
      "_id": "61894b4cfd707d769fc0125e",
      "partner1_id": "6176a4bfe9bc6fe2726410d7",
      "partner2_id": "617b81a6e48a7dc8f13c8c3b",
      "question_id": "6188aa54741c47b88e17896a",
      "status": "completed",
      "mode": "elo",
      "matched_at": "2021-11-08T16:07:40.637Z",
      "updated_at": "2021-11-09T11:14:08.120Z",
      "created_at": "2021-11-08T16:07:40.643Z",
      "__v": 0,
      "completed_at": "2021-11-09T11:08:59.638Z"
    },
    {
      "match_requirements": {
        "programming_language": "python",
        "question_mode": "timed",
        "elo_match_pool": "61890df24b01c92a293d181a"
      },
      "meta": {
        "partner1_display_name": "Jon",

```

```

        "partner2_display_name": "vinleon",
        "question_title": "Two Sum"
    },
    "_id": "61890df84b01c92a293d1820",
    "partner1_id": "617b81a6e48a7dc8f13c8c3b",
    "partner2_id": "6176a4bfe9bc6fe2726410d7",
    "question_id": "6188aa54741c47b88e17896a",
    "status": "completed",
    "mode": "elo",
    "matched_at": "2021-11-08T11:46:00.048Z",
    "updated_at": "2021-11-08T12:19:07.690Z",
    "created_at": "2021-11-08T11:46:00.050Z",
    "__v": 0,
    "completed_at": "2021-11-08T12:19:07.689Z"
}
]
}

```

Get Match by ID

Method: GET

Route: /match/admin/:match_id

Description: Get match details

Header: Authorization (JWT Token)

Parameter: match_id: string (required)

Output

```

{
  "data": {
    "_id": "6188f87d99acfb641c63d70",
    "partner1_id": "6176a4bfe9bc6fe2726410d7",
    "partner2_id": "617b81a6e48a7dc8f13c8c3b",
    "question_id": "6188aa54741c47b88e17896a",
    "status": "in-progress",
    "mode": "elo",
    "match_requirements": {
      "programming_language": "python",
      "question_mode": "timed",
      "elo_match_pool": "6188f87899acfb641c63d6a"
    },
    "matched_at": "2021-11-08T10:14:21.593Z",
    "updated_at": "2021-11-08T10:14:21.594Z",
    "created_at": "2021-11-08T10:14:21.594Z",
    "__v": 0
  }
}

```

```
}
```

Update Match

Method: PUT

Route: `/match/admin`

Description: Update a match's details

Header: `Authorization` (JWT Token)

Example Input	Output
<pre>{ "_id": "6188f87d99acfa641c63d70", "status": "completed" }</pre>	<pre>{ "data": { "_id": "6188f87d99acfa641c63d70", "partner1_id": "6176a4bfe9bc6fe2726410d7", "partner2_id": "617b81a6e48a7dc8f13c8c3b", "question_id": "6188aa54741c47b88e17896a", "status": "completed", "mode": "elo", "match_requirements": { "programming_language": "python", "question_mode": "timed", "elo_match_pool": "6188f87899acfa641c63d6a" }, "matched_at": "2021-11-08T10:14:21.593Z", "updated_at": "2021-11-09T14:10:52.418Z", "created_at": "2021-11-08T10:14:21.594Z", "__v": 0 } }</pre>

Delete Match

Method: DELETE

Route: `/match/admin/:match_id`

Description: Soft-Delete a Match

Header: `Authorization` (JWT Token)

Parameter: `match_id`: string (required)

Output
<pre>{ "is_deleted": true, "data": { "match_requirements": { "programming_language": "python", "question_mode": "timed", "elo_match_pool": "6188f87899acfa641c63d6a" } } }</pre>

```
},  
  "_id": "6188f87d99acfa641c63d70",  
  "partner1_id": "6176a4bfe9bc6fe2726410d7",  
  "partner2_id": "617b81a6e48a7dc8f13c8c3b",  
  "question_id": "6188aa54741c47b88e17896a",  
  "status": "completed",  
  "mode": "elo",  
  "matched_at": "2021-11-08T10:14:21.593Z",  
  "updated_at": "2021-11-09T14:10:52.418Z",  
  "created_at": "2021-11-08T10:14:21.594Z",  
  "__v": 0  
}  
}
```

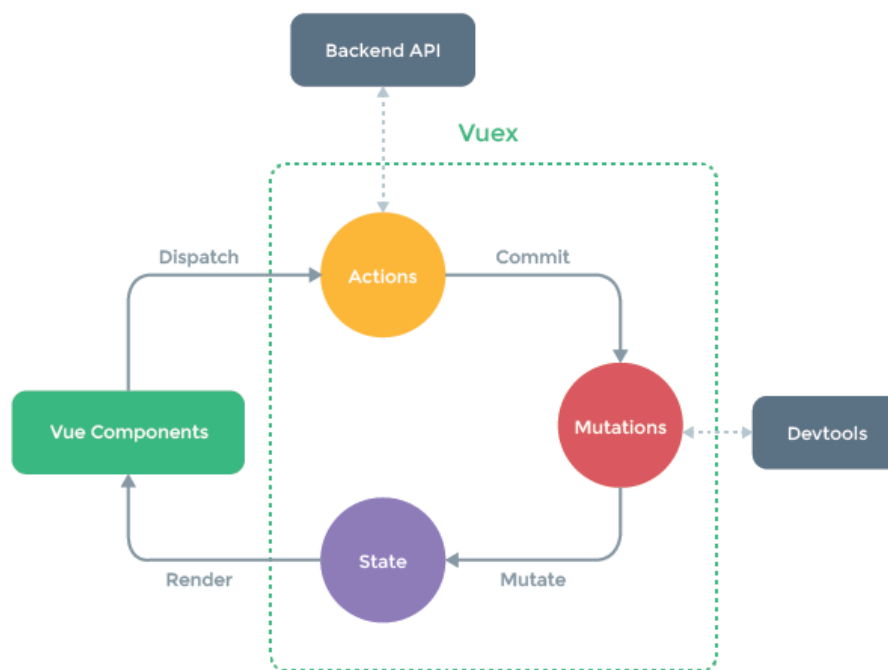
Frontend

Tech Stack & Libraries Used

1. Nuxt.js
2. Vuex
3. Vuetify
4. Yjs & y-monaco (For Editor)

Nuxt.js is a higher-level framework that builds on top of Vue. It simplifies the development of universal or single page Vue apps. Nuxt.js abstracts away the details of server and client code distribution so we can focus on application development.

Vuex is a state management pattern and library for Vue.js applications. It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion.



Overview of Vuex

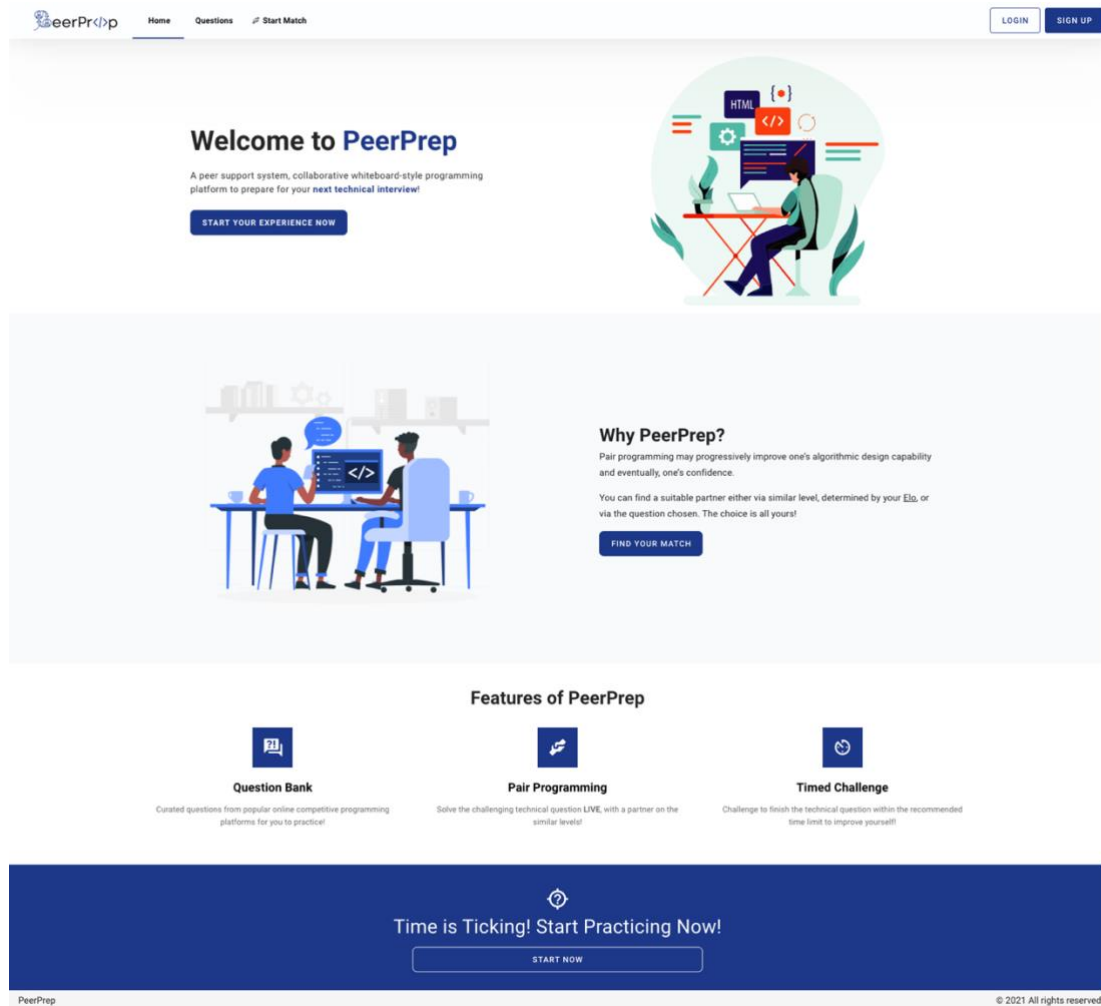
Source: <https://conclave-team.github.io/conclave-site/>

Application Demonstration

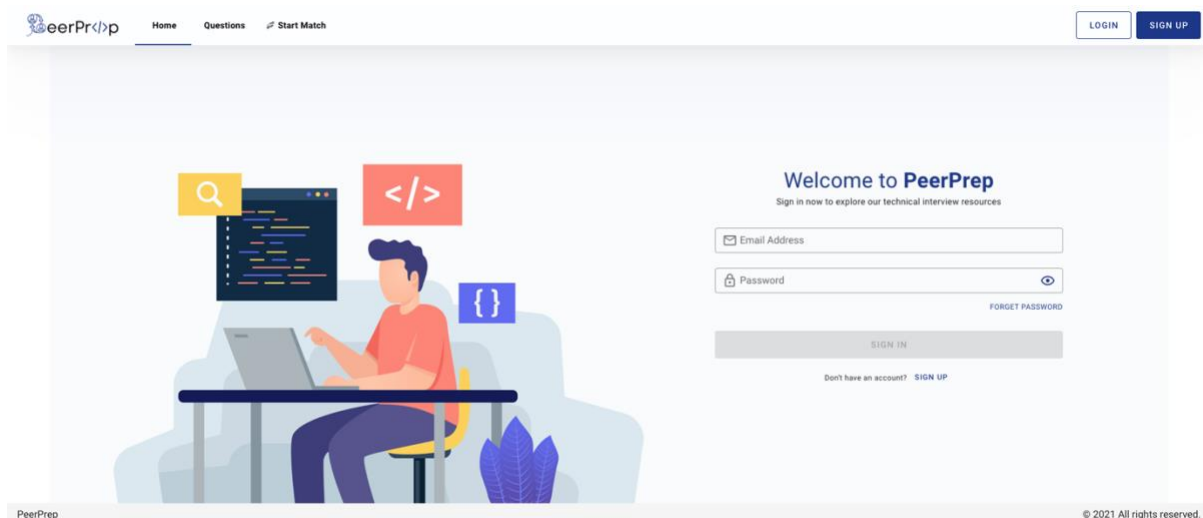
Website Endpoint: <https://peerprep.tech>

Staging Website Endpoint: <https://staging.peerprep.tech>

Upon landing on the website, the user is greeted with some basic information such as motivation and features.

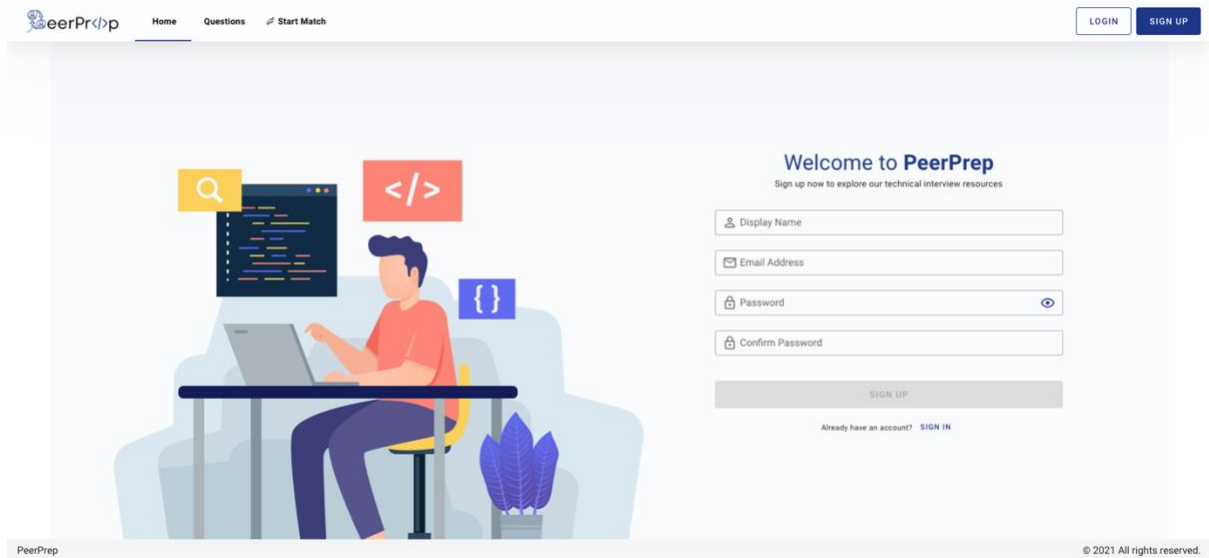


PeerPrep Landing Page



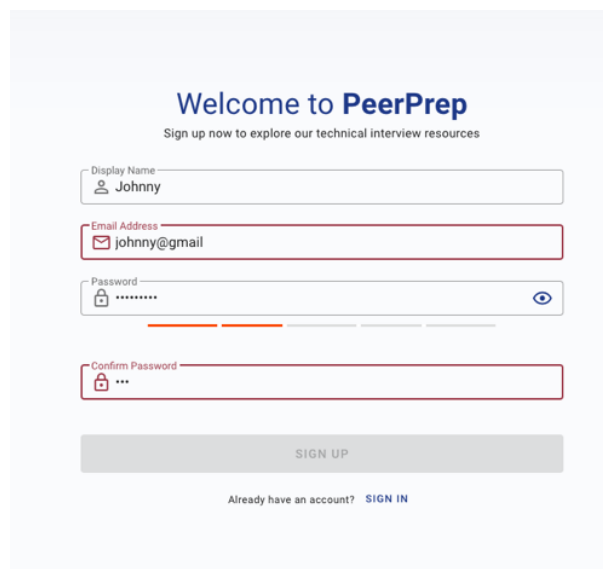
Login Page

User can login and register for an account to gain the full privilege on the website.



Registration Page

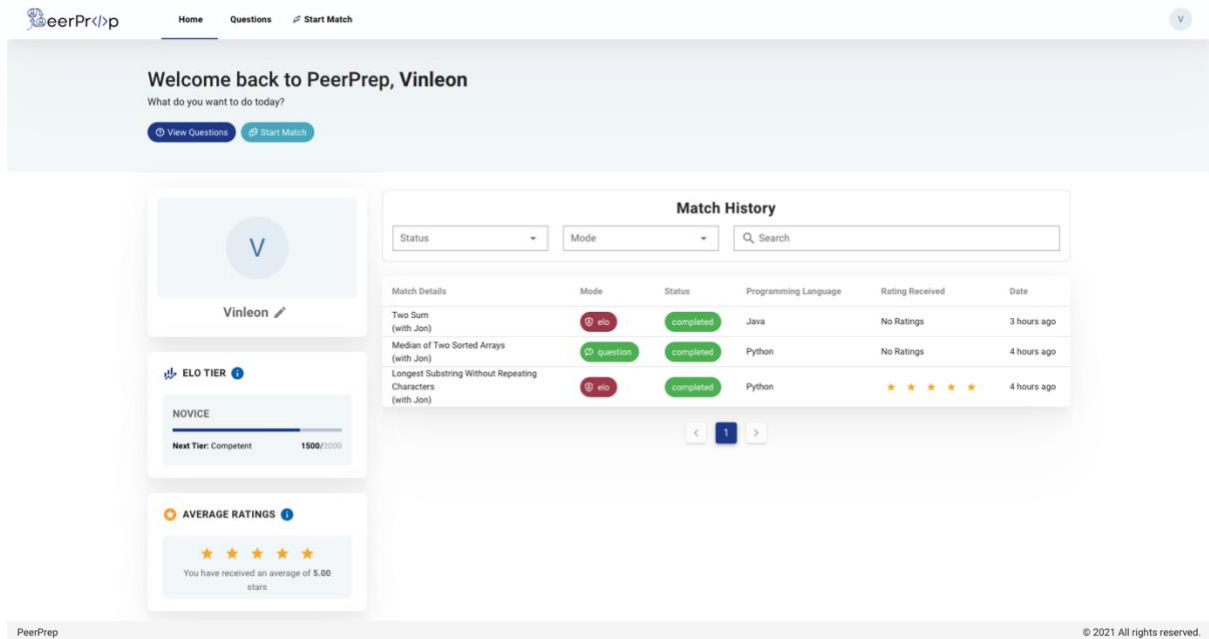
On registration page, some client-side validation measures are put in place to ensure accurate information are input.



Client-Side Validation & Password Strength

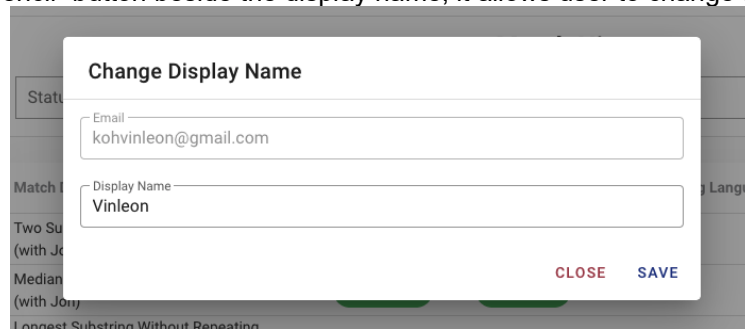
Profile

Upon logging in, the user is greeted with their profile page, dubbed as the user dashboard. It consists of information such as display name, elo tier, average ratings, and match history. There are also quick actions available to users at the top of the page.



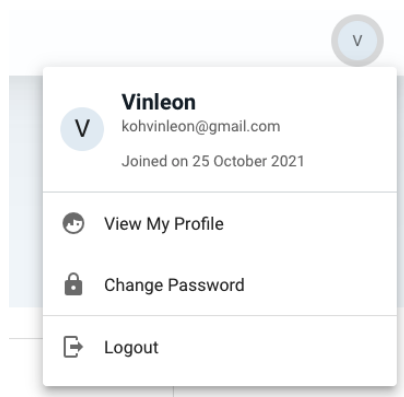
User Dashboard

By clicking the “pencil” button beside the display name, it allows user to change their display name.

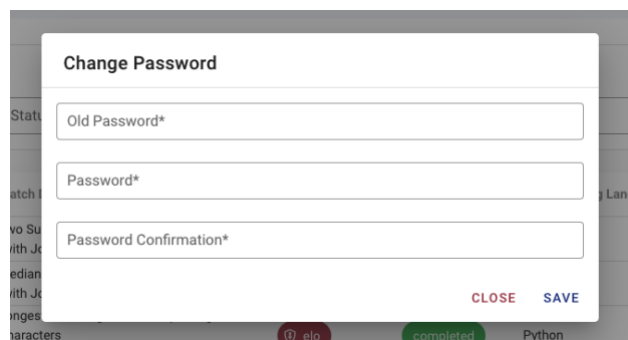


Change Display Name Dialog

By clicking the “avatar” button on the toolbar, it opens a mini menu where user can either view their profile, change password or logout.



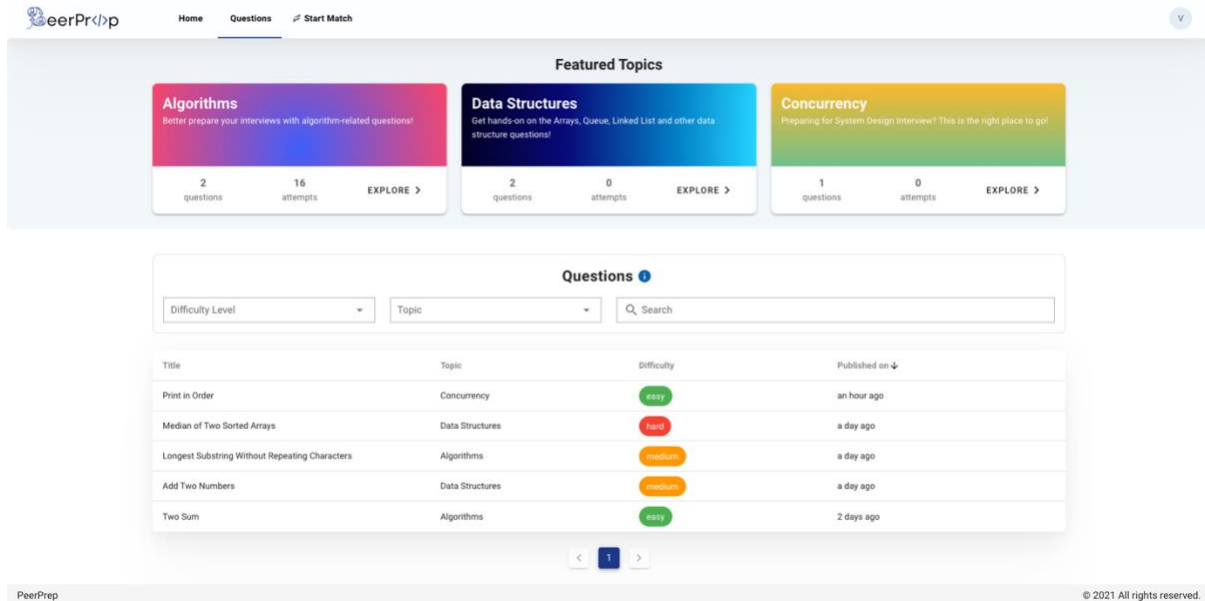
User's Action Menu



Change Password Dialog

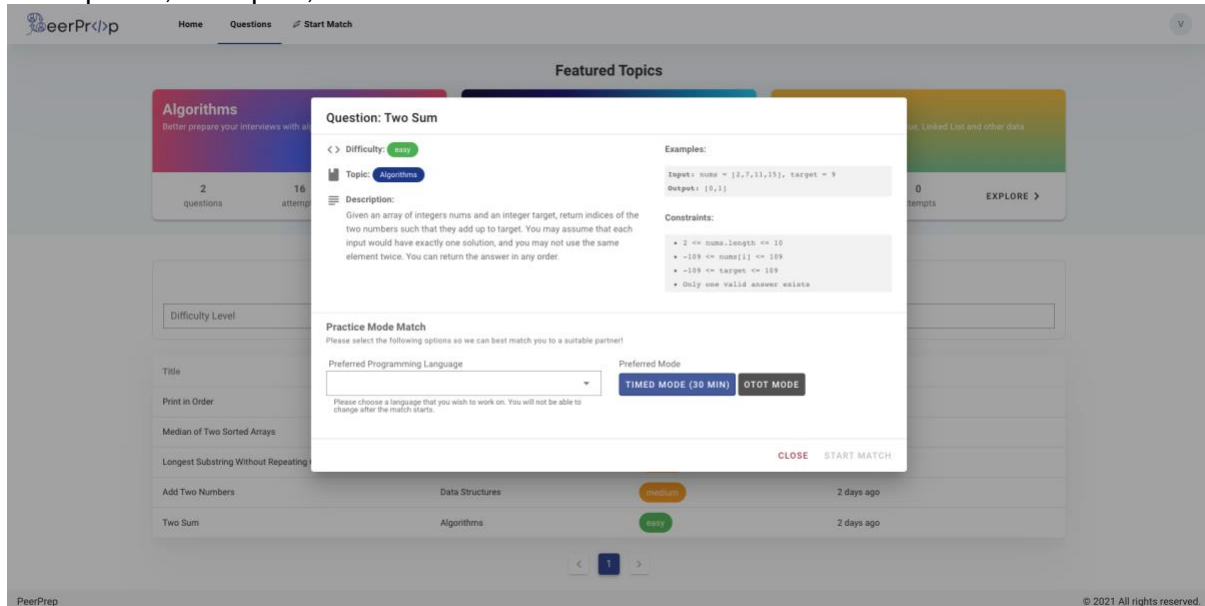
Questions

The user is redirected to the list of questions available on PeerPrep by clicking the question tab on the app bar. This allows users to view the available questions and find a partner to practice the question together.



List of Questions Available on PeerPrep

User can click on the individual questions to view the detailed question, consist of the descriptions, examples, and constraints.

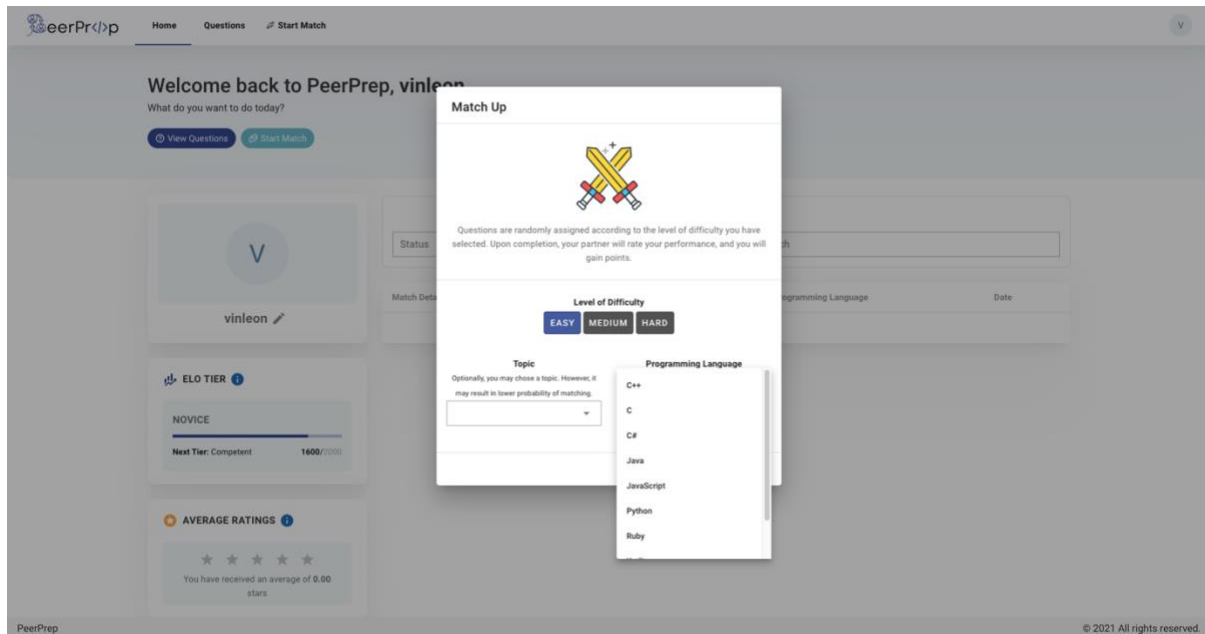


Clicking a particular question pops up a dialog that contains more details about the question and gives the user the option to find a partner to enter practice mode.

The practice mode match process is similar to the Elo Match process which will be explained in the following section. The only additional constraint is the question, while Elo Match's question is picked randomly to provide a sense of challenge.

Matching

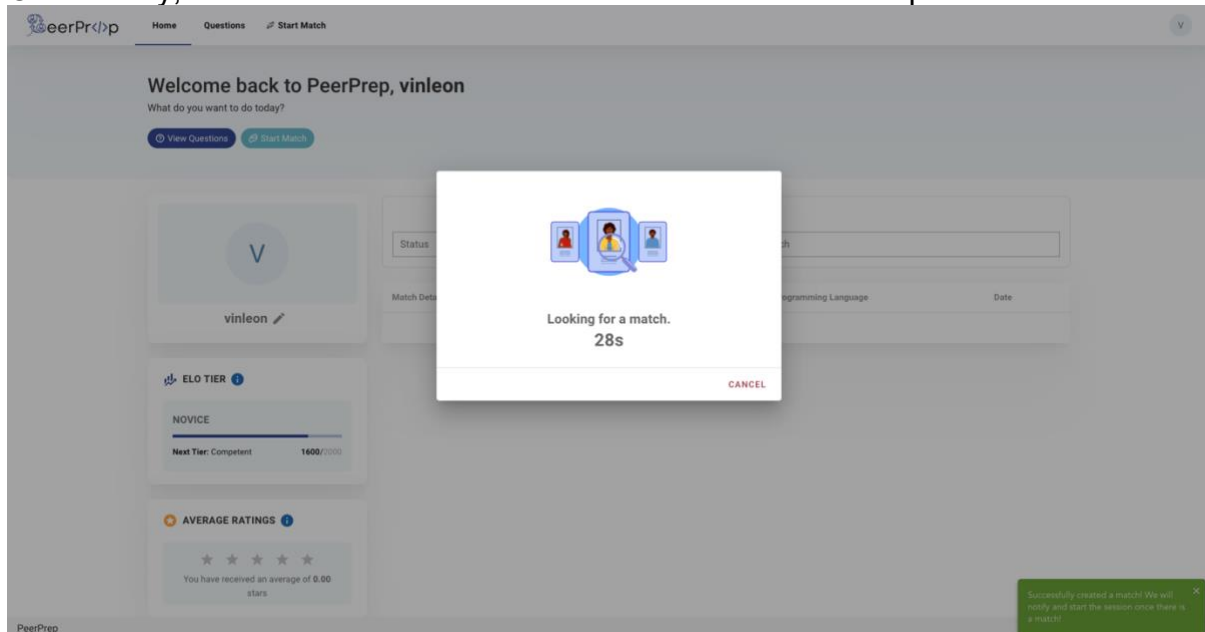
A match up dialog is open when the user clicks on the “Start Match” button on the app bar or from the quick actions from the user dashboard. Users have to choose their preferred level of difficulty, topic (optional) and programming language.



Match Options (Difficulty Level, Topics, Programming Languages)

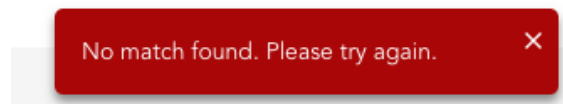
The reason for fixing the programming language is to ensure users matched can communicate and solve the question efficiently. Switching of language in the middle of the question may cause unhappiness to another partner.

Once ready, user can click on “Start Match” and user will be enqueued.



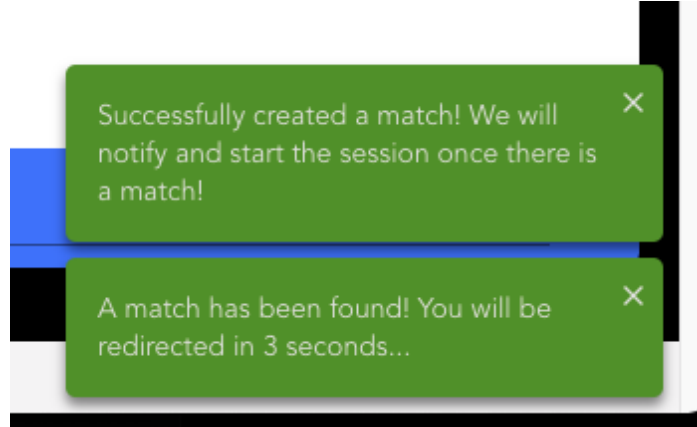
User enqueued to find a match

If no match is found within 30 seconds, the match will be cancelled, and the dialog will be closed. Users have to re-queue again and may want to lax their requirements.



Notification if no match found after 30 seconds

If there is a match found within 30 seconds, user will be notified and redirected to the session page.



Notification informing user that their match will be starting

Session (Match)

The session page is where an ongoing match is occurring.

- If the user chooses OTOT mode, the session is valid for 12 hours or ended by the user manually, whichever earlier.
- If the user chooses Elo Match or Question Timed Mode, the session is valid within the question recommended duration or ended by the user manually, whichever earlier.

The page is split into 3 sections:

1. Questions and Match Details

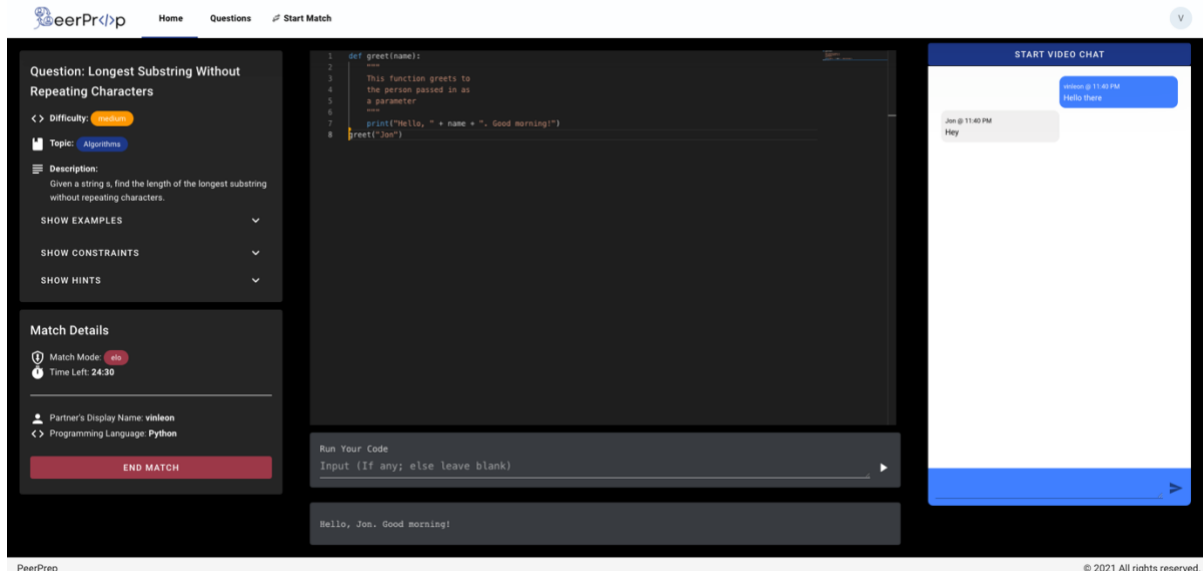
- a. Users can view the description of the question, examples, constraints and hints (if any).
- b. Users can view the match details with information such as how much time is left, the partner's name, and the programming language chosen. Users can also end their match session.

2. Code Editor

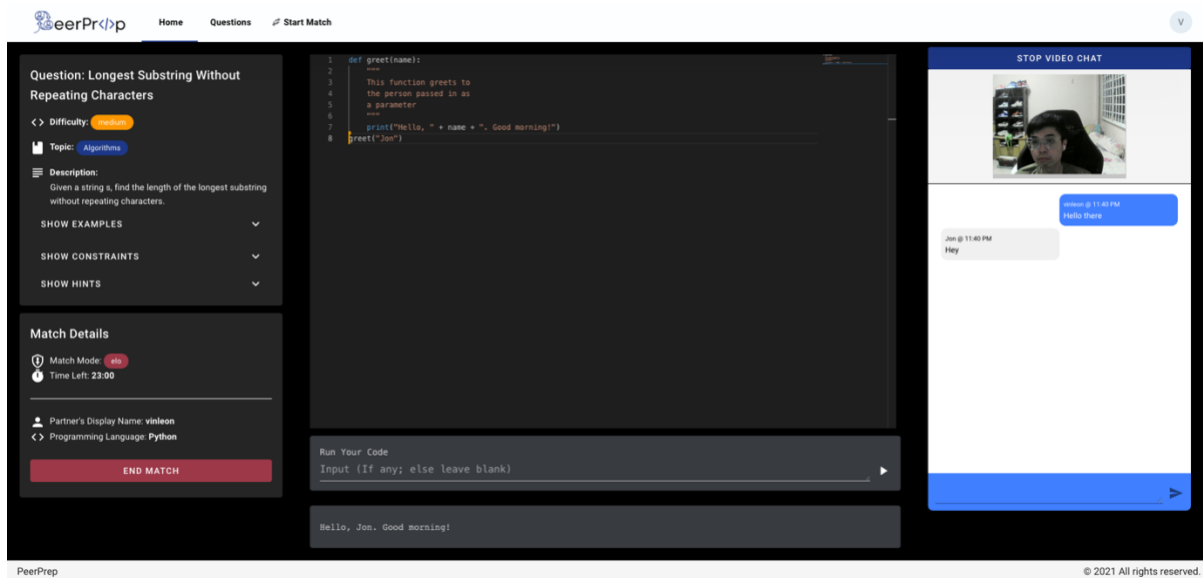
- a. A real-time live collaborative code editor is available to the users to work on their problems.
- b. Users can also execute their code via the code executor at the bottom of the code editor. Outputs (or errors) will be shown to user.

3. Chat and Video Chat

- a. The primary form of communication offered to users is via message. Users can type their ideas or communicate with their partners via the chat panel.
- b. Alternatively, users can also choose to initiate a video chat to discuss ideas over video and voice.



Match Session

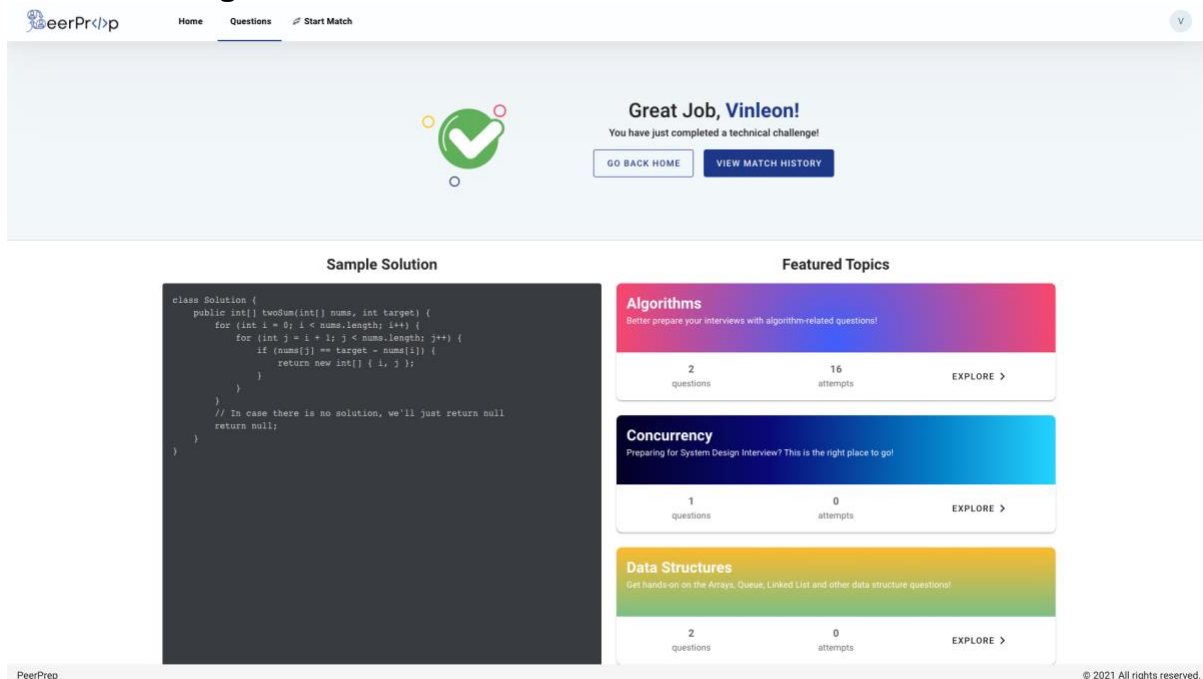


Match Session with Video Chat switched on

Thank You Page


After the match session has ended, user will be redirect to the thank you page, which it includes a sample solution.


Thank You Page for Practice Mode



Sample Solution and Featured Topics shown to user

Thank You Page for Elo Match

Home Questions Start MatchV



Great Job, Vinleon!
You have just completed a technical challenge!
[GO BACK HOME](#) [VIEW MATCH HISTORY](#)

Sample Solution

```
class Foo {  
    public Foo() {  
    }  
  
    public void first(Runnable printFirst) throws InterruptedException {  
        // printFirst.run() outputs "first". Do not change or remove this line.  
        printFirst.run();  
    }  
  
    public void second(Runnable printSecond) throws InterruptedException {  
        // printSecond.run() outputs "second". Do not change or remove this line.  
        printSecond.run();  
    }  
  
    public void third(Runnable printThird) throws InterruptedException {  
        // printThird.run() outputs "third". Do not change or remove this line.  
        printThird.run();  
    }  
}
```

Rate your Partner! 😊

Please help us by rating your partner! This allows us to improve our matching criteria.
It will help us to match users with higher compatibility in the future!

1 is the lowest satisfaction, 5 is highest satisfaction


★ ★ ★ ★ ★


[SUBMIT ➡](#)

PeerPrep© 2021 All rights reserved.

Sample Solution and Rating Dialog

For Elo Match, a rating dialog will be displayed to user. If user is given a rating, he/she will be given a basic 50 + number of stars * 10 amount of elos.

Home Questions Start MatchV




Great Job, Vinleon!
You have just completed a technical challenge!
[GO BACK HOME](#) [VIEW MATCH HISTORY](#)

Sample Solution

```
class Foo {  
    public Foo() {  
    }  
  
    public void first(Runnable printFirst) throws InterruptedException {  
        // printFirst.run() outputs "first". Do not change or remove this line.  
        printFirst.run();  
    }  
  
    public void second(Runnable printSecond) throws InterruptedException {  
        // printSecond.run() outputs "second". Do not change or remove this line.  
        printSecond.run();  
    }  
  
    public void third(Runnable printThird) throws InterruptedException {  
        // printThird.run() outputs "third". Do not change or remove this line.  
        printThird.run();  
    }  
}
```

Rate your Partner! 😊



Your ratings has been received!
We hope you have enjoy PeerPrepping!

PeerPrep

Thank you for your ratings! We hope you have enjoy PeerPrepping! ✕

Notification and message shown after user have given their rating.

Match History

Users can view their match history by clicking in from the Profile page. It will show the question they have attempted, code and chat history.



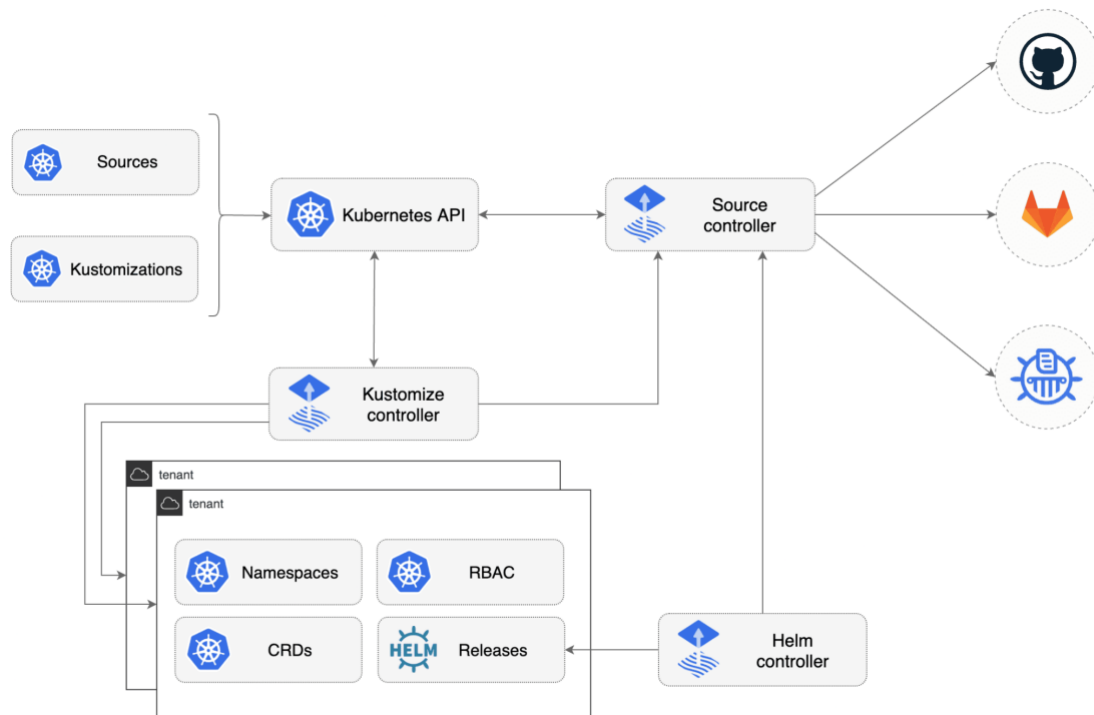
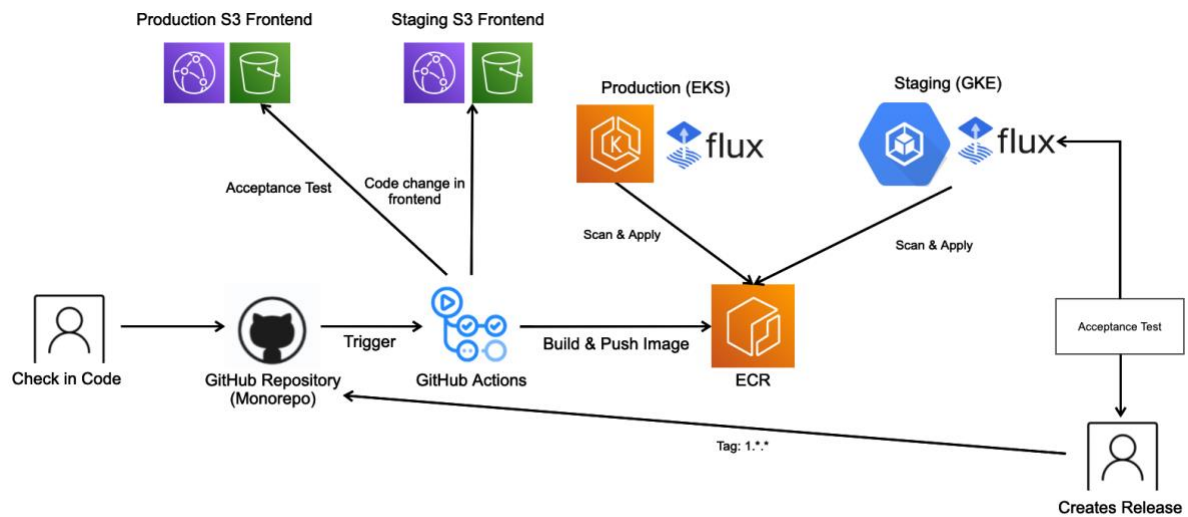
Match Session History

CI/CD

Continuous Delivery



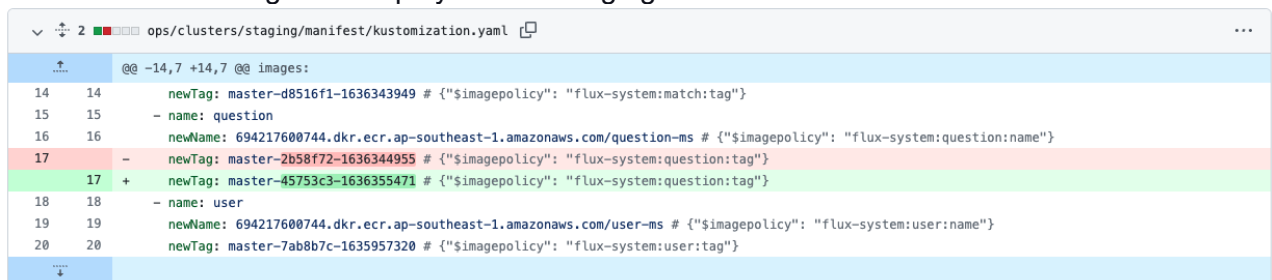
We have adopted the **Continuous Delivery** approach. A diagram to illustrate CI/CD process:



Flux v2 Overview

Backend CI/CD

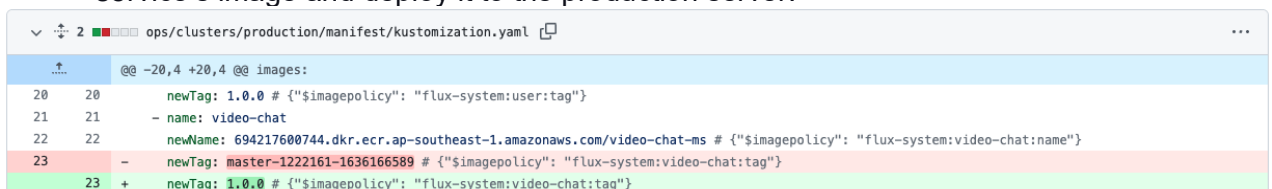
1. As Microservices architecture is adopted in this project, each service has its own folder.
2. Once code modification is detected in the individual folder (e.g., user), the user's GitHub Action workflow is triggered to perform CI (test & build-and-push).
3. The GitHub Action workflow will perform unit testing and/or integration testing (API request tests).
4. Once the test has passed, it will build the Docker image and pushed to its respective ECR repository with a tag with the following format: `${GITHUB_REF##*/}-${GITHUB_SHA:0:7}-${date +%s}`, e.g. `master-7ab8b7c-1635957304`.
5. Flux v2 (in the staging cluster) will monitor and scan the ECR repository every 1 minute. Once it detects a new image with the tag as mentioned in (4), Flux v2 will update the `kustomization.yaml` file in the staging cluster to update the new tag of the service's image and deploy it to the staging server.



		ops/clusters/staging/manifest/kustomization.yaml	
		@@ -14,7 +14,7 @@ images:	
14	14	newTag: master-d8516f1-1636343949 # {"\$imagepolicy": "flux-system:match:tag"}	
15	15	- name: question	
16	16	newName: 694217600744.dkr.ecr.ap-southeast-1.amazonaws.com/question-ms # {"\$imagepolicy": "flux-system:question:name"}	
17	-	newTag: master-2b58f72-1636344955 # {"\$imagepolicy": "flux-system:question:tag"}	
17	+	newTag: master-45753c3-1636355471 # {"\$imagepolicy": "flux-system:question:tag"}	
18	18	- name: user	
19	19	newName: 694217600744.dkr.ecr.ap-southeast-1.amazonaws.com/user-ms # {"\$imagepolicy": "flux-system:user:name"}	
20	20	newTag: master-7ab8b7c-1635957320 # {"\$imagepolicy": "flux-system:user:tag"}	

Example of Flux updating the clusters/staging/kustomization.yaml

6. Then, acceptance testing is conducted and is done via self-testing or user testing to ensure that the features are production ready.
7. Once it passes the acceptance tests, a manual deployment will be performed by creating a tag and release in the GitHub repository.
8. GitHub Action workflow (`production-backend.yaml` & `production-frontend.yaml`) will build and tag the image in the format `1.*.*`, e.g. `1.0.1`
9. Flux v2 (in the production cluster) will monitor and scan the ECR repository every 10 minutes. Once it detects a new image with the tag as mentioned in (8), Flux will update the `kustomization.yaml` in the production cluster to update the new tag of the service's image and deploy it to the production server.



		ops/clusters/production/manifest/kustomization.yaml	
		@@ -20,4 +20,4 @@ images:	
20	20	newTag: 1.0.0 # {"\$imagepolicy": "flux-system:user:tag"}	
21	21	- name: video-chat	
22	22	newName: 694217600744.dkr.ecr.ap-southeast-1.amazonaws.com/video-chat-ms # {"\$imagepolicy": "flux-system:video-chat:name"}	
23	-	newTag: master-1222161-1636166589 # {"\$imagepolicy": "flux-system:video-chat:tag"}	
23	+	newTag: 1.0.0 # {"\$imagepolicy": "flux-system:video-chat:tag"}	

Example of Flux updating the clusters/production/kustomization.yaml

Frontend CI/CD

1. PeerPrep has two frontends - user and admin. Each frontend is contained within its own folder. Once code modification is detected in a particular folder, a GitHub Action workflow will be triggered.
2. The respective GitHub Action workflow will first perform tests.
3. Once all the tests have passed, it will build the static files to be deployed to the staging S3 bucket and invalidate the staging's CloudFront Distribution.
4. Acceptance testing is done via self-testing or user testing to ensure that the features are production ready.
5. Similar to the backend CI/CD, once it passes the acceptance tests, a manual deployment will be performed by creating a tag and release in the GitHub repository.
6. Similar to the staging workflow, it will build the static files to be deployed to the production S3 bucket and perform invalidation in the production's CloudFront Distribution.

Backend Testing

Each backend is tested thoroughly using Mocha & Chai for User, Match and Question microservices and Jest for Editor and Chat microservices.

For User, Match and Question Microservices CI Test

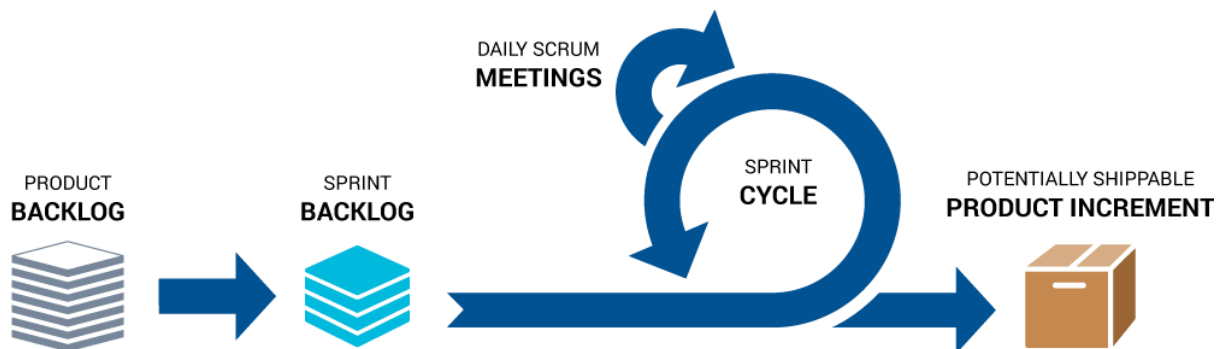
- API Integration Test is conducted to ensure all the endpoints are accessible and behave as expected.

Editor and Chat Microservices CI Tests

- Unit Testing is conducted on most of the components, namely the entities and the use cases. This allows testing that the entities work when stubbing with random values and with the business logic use-cases.
- API Integration Test is done using Supertest, which test that the endpoints are accessible and behave as expected.

Development Process

Our team has decided to adopt the **agile methodology** for this project.



Adopting an agile methodology is highly advantageous as our requirements can grow when we start our implementation, and new features can be added to the product backlog according to requirements.

Throughout the project, we have divided our development process into 4 sprints:

1. **Sprint 1** (03 October to 17 October) - Microservices and DevOps
2. **Sprint 2** (18 October to 25 October) - Frontend Development
3. **Sprint 3** (26 November to 1 November) - Testing, Bug Fixing and DevOps
4. **Sprint 4** (2 November to 10 November) - Testing, Documentation and DevOps

By every end of the Sprint, deployments are done to its respective platform to ensure whatever we have done are working on a staging/production environment.

On the days where the member is working on the project, the members are required to fill up the standup logs to keep track of their own progress.

Standup Template

1. What have you done yesterday (or the previous day you have worked on the project)?
2. What will you be doing today?
3. Are you encountering any blockers?

Optional

4. Do you need to get into a call for assistance or discussions?

The standup allows each group member to better support each other, especially as we do not have physical lessons (hence, no physical interaction). The group members can aid by providing existing knowledge to another member's future tasks or solutions to blockers.

Areas of Improvements

A more attractive, gamification-like concept for PeerPrep

Currently, there is the Elo and Ranking System in place. In the future iteration of PeerPrep, more gamification concepts can be integrated to entice users to continually use the platform, such as a leaderboard and some form of digital rewards in solving certain challenges.

Increase options for the matching process

Currently, users have to pick a topic and difficulty for the kind of questions that they want to attempt. We can implement a system that will slowly loosen the question criteria if they cannot find a suitable match for the current parameters. For example, after 30 seconds, users who have not found their match yet could be matched to other users choosing the same topic, but with different difficulties. This will help to increase the likelihood of a match being successful.

Automated Approach in Populating the Question Database

The current method of inserting questions into the database requires an admin to create each question and provide the various details of the question individually, which can be a long and tedious process. An area of improvement is to use an automated approach of scraping data from websites such as Leetcode or Hackerrank to obtain many questions in a quick and efficient method. PeerPrep could also have links for users to look up the scraped questions from the corresponding websites to get more information about the questions.

Reflections

This project allowed us to go through the whole process of turning an idea into something tangible and functional. From defining functional and non-functional requirements to set up the backend microservices and front end, we had to learn a lot of new technology to pull this off.

There is a steep learning curve in picking up DevOps, and it's rather challenging to debug compared to backend and frontend development. This promotes exploration, and all the concepts taught during the lectures and tutorials are applicable. It also brings in new insights and technologies that are used and practised in DevOps, which is relatively uncommon.

Having regular sprints was very useful, since within each sprint, we could focus on a different aspect of the project. For example, for the first sprint, we focused on the backend microservices, and for sprint 2 we focused on hooking up the back end to the front end. We also concentrated on testing for sprint 3, to utilise Continuous Integration into our project. This was very useful to delineate the entire time period into small chunks with attainable goals, so we can focus on the task at hand.