



CS3219 Software Engineering Principles and Patterns

AY2023/2024 Semester 1

PeerPrep Project Report

Group 51

Team Member	Student Number
Haiqel Bin Hanaffi	A0235011B
Yap Pin Wei Zenith	A0234669R
Lee Kang Quan	A0229824X
Siew Hui Zhuan	A0219793N
Edward Mualim	A0221199A

Project Mentor: Akash Sihag

Table of Contents

1. Project Overview.....	3
2. Team Organisation and Contributions	4
3. Requirement Specification	8
3.1 Functional Requirements (FRs).....	8
3.2 Non-Functional Requirements (NFRs)	10
4. Application Architecture.....	11
4.1 Deprecated Past Architecture	11
4.2 Current Deployment-Level Architecture	13
4.3 Deprecated/Past Tech Stack.....	16
4.4 Current Tech Stack.....	16
5. Design and Implementation Process	19
5.1 Wireframing	19
5.2 Architecting Back End Services	20
5.3 Unit Testing.....	21
5.4 CI/CD	22
6. Nice-to-have Features Implemented.....	23
7. Challenges Faced	24
8. Areas of Improvement	25

1. Project Overview

This project is done as part of the module CS3219 Software Engineering Principles and Patterns by a group of five students from the School of Computing. Our work can be tracked on the team [Trello board](#). The goal of this project is to develop a full stack web application, PeerPrep, with the following minimum features:

1. A **user service** for profile management
2. A **matching service** for linking two users together from a given set of criteria
3. A **question service** for managing a library of coding questions
4. A **collaboration service** for two users to work on their coding skills together
5. Basic **UI** for users to interact with the application
6. **Deployment** of the application locally or via containerization

This adaptation of PeerPrep is a web application that allows its users to browse a library of coding questions and allows users to match with a peer and work on their coding skills together in a collaborative space consisting of a code editor and a text chat. Users may receive extra privileges within the platform by making a request to become an admin, which when approved by another admin, will allow them to add, remove and edit questions in the library. The authors of this project have containerized PeerPrep and are hosting it on Google Cloud using Google Kubernetes Engine (GKE) via the hostname app2.peerprepgroup51sem1y2023.xyz.

2. Team Organisation and Contributions

The team is organized into front-end and back-end developers as follows:

Front-end developers	Back-end developers
Haiqel Zenith Edward	Kang Quan Hui Zhuan

Individual Member contributions:

Member	Contributions
Haiqel	<ul style="list-style-type: none">• Front-end for question service• Front-end for SSO (Login with email and password, Sign Up and Sign Out)• Front-end for “404 page” when user visits an unknown page• Some of CI/CD in GitHub Actions• Some wireframes in Figma for other developers to use for front-end (Questions Page, SSO, Matching Page). Medium to High Fidelity• Maintainer for GCE & GKE server/cluster• Maintainer for DNS server from GoDaddy• Maintainer for NGINX Reverse Proxy for previous cloud deployment for Assignments• Local bare-metal Kubernetes deployment before GKE was used
Zenith	<ul style="list-style-type: none">• Front-end for question service• Front-end for profile page• Front-end for practice page, before and after user matched• Front-end for matching service pop up• Backend for collaboration service• SSO for Google, GitHub, and Twitter

	<ul style="list-style-type: none"> • Maintainer for NGINX Reverse Proxy for previous cloud deployment for Assignments • Setting up Google Kubernetes Engine • Wireframe for user page () and question page (high fidelity) • CI/CD in GitHub Actions for collaboration service and front end for GKE
Kang Quan	<ul style="list-style-type: none"> • Profile service backend API and database design, implementation and deployment • Front-end for validation of email address (login with email and password) • Front-end and backend for admin page (role segregation) • Front-end and backend for code editor (live collaboration and syntax highlighting) • Maintainer for NGINX Reverse Proxy for previous cloud deployment for Assignments • Bare metal Kubernetes deployment • Some of CI/CD using GitHub Actions
Hui Zhuan	<ul style="list-style-type: none"> • Question service backend and database design, implementation, testing, and deployment • Matching service backend and database design, implementation, and deployment • Automate continuous integration testing pipeline on GitHub Actions • Tooling, library, and technology research
Edward	<ul style="list-style-type: none"> • Front-end for question service • Error handling for adding questions • Wireframing for question page • Some of CI/CD using GitHub Actions

Sub-group contributions

Sub-group 1		Sub-group 2	
Member	Nice-to-have features	Member	Nice-to-have features
Haiqel	<ul style="list-style-type: none"> • N9 Deployment on GCP/GKE • N10 Scalability with Kubernetes Auto-Scaler • N11 API Gateway/ K8 Ingress • N12 Service Registry/Discovery with Kubernetes 	Kang Quan	<ul style="list-style-type: none"> • N1 Communication in Collaboration Service (text-chat) • N4 Enhancement of Question Service (tagging) • N5 Enhance collaboration service by providing an improved code editor with code formatting, syntax highlighting for one language, syntax highlighting for multiple languages • N8 Unit Testing, CI/CD
Zenith	<ul style="list-style-type: none"> • N9 Deployment on GCP/GKE • N10 Scalability with Kubernetes Auto-Scaler • N11 API Gateway/ K8 Ingress • N12 Service Registry/Discovery with Kubernetes 	Hui Zhuan	<ul style="list-style-type: none"> • N1 Communication in Collaboration Service (text-chat) • N4 Enhancement of Question Service (tagging) • N5 Enhance collaboration service by providing an improved code editor with code formatting, syntax highlighting for one language, syntax highlighting for multiple languages • N8 Unit Testing, CI/CD

		Edward	<ul style="list-style-type: none"> • N1 Communication in Collaboration Service (text-chat) • N4 Enhancement of Question Service (tagging) • N5 Enhance collaboration service by providing an improved code editor with code formatting, syntax highlighting for one language, syntax highlighting for multiple languages • N8 Unit Testing, CI/CD
--	--	--------	---

3. Requirement Specification

FR/NFR	Priority
F1 PeerPrep provides a user service.	High
F2 PeerPrep provides a matching service.	High
F3 PeerPrep provides a question service.	High
F4 PeerPrep provides a collaboration service.	High
F5 PeerPrep provides a user interface.	High
F6 PeerPrep goes through unit testing and CI/CD.	High
F7 PeerPrep is accessible via the internet.	High
F8 PeerPrep utilizes an API gateway.	High
F9 PeerPrep can handle a moderate number of users.	High
F10 PeerPrep uses service discovery.	High
NFR1 PeerPrep is responsive.	Medium
NFR2 PeerPrep has a healthy uptime.	Medium
NFR3 PeerPrep works across multiple browsers.	Medium
NFR4 PeerPrep is scalable.	Medium

3.1 Functional Requirements (FRs)

FR/Must-Have	Priority
F1.1 Allow users to update profile details. <ul style="list-style-type: none">1.1.1 Allow users to edit their username.1.1.2 Allow users to edit their first name.1.1.3 Allow users to edit their last name.1.1.4 Allow users to edit their age.	High
F1.2. Allow users to login with Single-Sign-On (SSO). <ul style="list-style-type: none">1.2.1 Allow users to sign in via Google.1.2.2 Allow users to sign in via Facebook.1.2.3 Allow users to sign in via GitHub.	High

<ul style="list-style-type: none"> 1.2.4 Allow users to sign in via Twitter / X. 	
F2.1 Allow users to match another user searching for the same type of question. <ul style="list-style-type: none"> 2.1.1 Allow users to match by programming language. 2.1.2 Allow users to match by question difficulty. 	High
F3.1 Allow users to view the questions. <ul style="list-style-type: none"> 3.1.1 Allow users to see question title. 3.1.2 Allow users to see question complexity. 3.1.3 Allow users to see question categories. 3.1.4 Allow users to see question description. 3.1.5 Allow users to see a list of all questions. 	High
F3.2 Allow users to delete questions. <ul style="list-style-type: none"> 3.2.1 Allow users to delete a question via a delete button. 	High
F3.3 Allow users to add questions <ul style="list-style-type: none"> 3.3.1 Allow users to enter question details in UI. 3.3.1 Allow users to add a new question via a save button. 	High
F3.4 Allow users to select questions. <ul style="list-style-type: none"> 3.4.1 Allow users to view question details after clicking on a question. 	High
F3.5 Allow users to edit questions. <ul style="list-style-type: none"> 3.5.1 Allow users to change title/complexity/description of a question. 3.5.2 Allow users to add and remove categories of a question. 	High
F4.1 Allow users to chat with another matched user.	High
F4.2 Allow user to code real time with another user.	High
F5 Provide the user with a simple interface.	High
F6.1 Support automated unit testing for individual components and functions.	High
F6.2 Automated deployment of clean build passing all checks.	High
F7.1 Users can access the web application through the public internet.	High
F7.2 Users can access the web application through a domain name.	High
F8.1. Create reverse proxy for service routing.	High

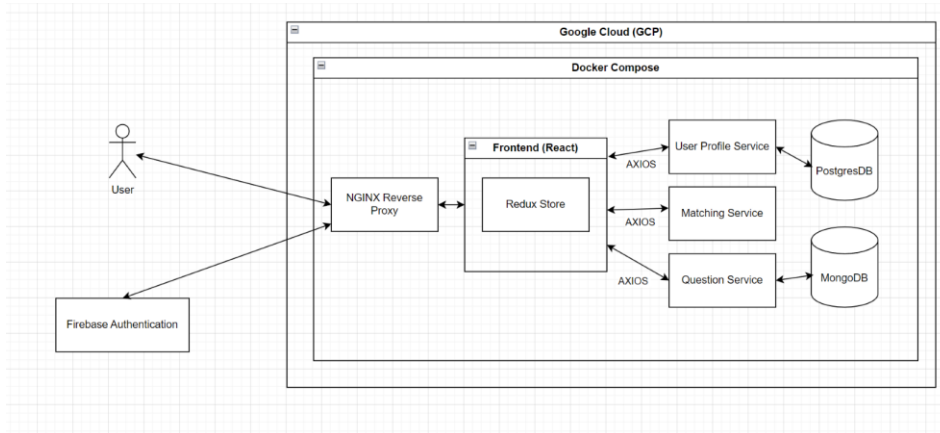
F8.2 Create authenticated/authorized protected routes.	High
F9.1 100 users can use the web application at the same time.	High
F10.1 Service Registry/Service Discovery with Kubernetes.	High

3.2 Non-Functional Requirements (NFRs)

NFR (Non-Functional Requirements)	Priority
NFR1.1 Updates to profile details should be reflected in the UI < 1 second	Medium
NFR1.2 User login process should take < 1 minute including UI navigation from home page.	Medium
NFR1.3 Matching service should return a match within 30 seconds if a valid match exists	Medium
NFR1.4 The system should display new messages (chat service) within 2 seconds	Medium
NFR1.5 Latency of code input to reflection in UI < 2 seconds	Medium
NFR1.6 CI/CD Jobs should be completed within 10 minutes per run.	Medium
NFR2.1 99.999% uptime	Medium
NFR2.2 The updated version of the app should be live within 5 minutes	Medium
NFR3.2 The app should be accessible over almost all modern browser engines (Chrome, Firefox, Safari etc.)	Medium
NFR4.1 The system should be able to handle up to 100 users without degradation to performance	Medium

4. Application Architecture

4.1 Deprecatd Past Architecture



Before our team ventured into learning and using Kubernetes, our team focused on deploying all our APIs, front-end and databases as containers in our original live server deployed on Google Cloud at 35.240.242.81. We exposed all our APIs through our reverse proxy which was "NGINX Reverse Proxy" and the reverse proxy acted as our API Gateway as well. Our reverse proxy was also containerized and all traffic from the internet had to go through our reverse proxy first before reaching other containers.

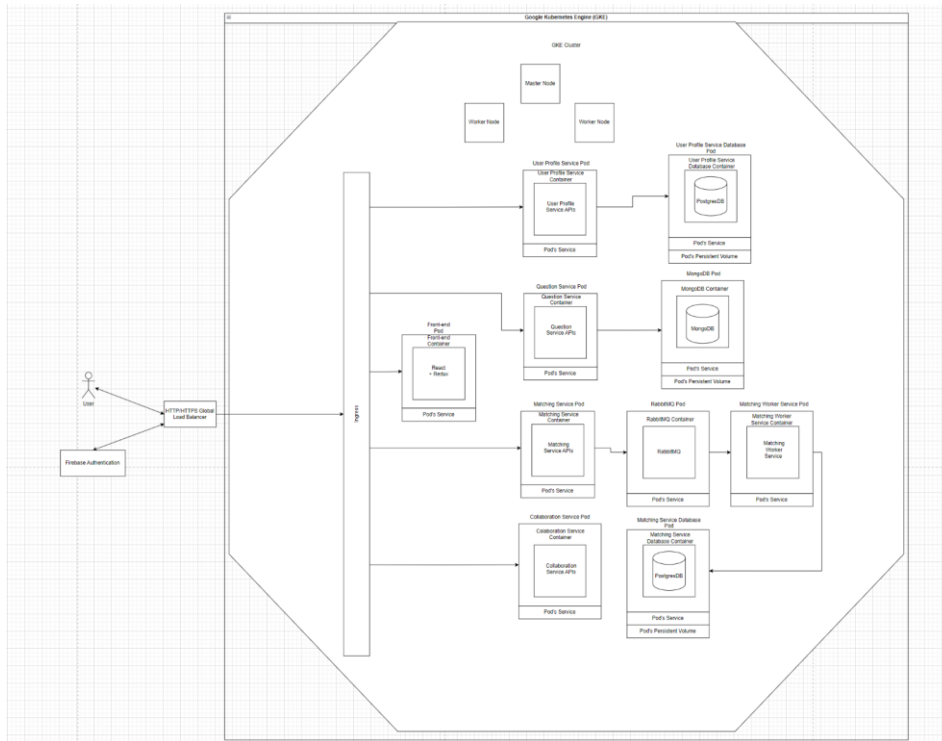
SOURCE	DESTINATION	SSL	ACCESS	STATUS
35.140.142.61:8000 Created: 1st November 2023	http://10.148.0.3:8000	HTTP only	Public	Disabled
api.peerprepgroup51sem1y2023.xyz Created: 1st October 2023	http://35.240.242.81:3100	Let's Encrypt	Public	Online
app.peerprepgroup51sem1y2023.xyz Created: 17th September 2023	http://35.240.242.81:3000	Let's Encrypt	Public	Online
collab.peerprepgroup51sem1y2023.xyz Created: 2nd November 2023	http://35.240.242.81:8576	Let's Encrypt	Public	Online
id.peerprepgroup51sem1y2023.xyz Created: 9th November 2023	http://35.240.242.81:3000	HTTP only	Public	Online
peerprepgroup51sem1y2023.xyz Created: 17th September 2023	http://35.240.242.81:3000	Let's Encrypt	Public	Online
proxy.peerprepgroup51sem1y2023.xyz Created: 17th September 2023	http://35.240.242.81:81	Let's Encrypt	Public	Online
www.peerprepgroup51sem1y2023.xyz	http://35.240.242.81:8000	Let's Encrypt	Public	Online

From the image above, we can see that all our APIs are being used through the reverse proxy. There was a need to expose most of our APIs through the reverse proxy as our front-end was Client-Side-Rendering (CSR) which meant that the front-end code would be expected to run by the client on their machine and the only way for the front-end to access our APIs is through the reverse proxy. Interestingly, our APIs communicate with the databases that are also containerized using the server's private IP since there was no need to expose our databases to the internet.

Furthermore, our team relied on another external service provider for authentication: Google's Firebase Authentication service. We settled on Google Firebase Authentication as we felt that it made integrating Single-Sign-On (SSO) features extremely easy, and we could prioritize other features better. ([NFR1.2](#))

However, our team noticed the limitations of using only docker compose for deploying our micro-services such as being unable to horizontally scale our APIs if there were thousands of users using our front-end and we noticed we did not have full control of the resources consumed by our containers.

4.2 Current Deployment-Level Architecture



Currently, our project is running inside a Google Kubernetes Engine (GKE) cluster which we migrated from our original server which was deployed using Google Compute Engine (GCE). In the image above, we can see that we have a main ingress acting as our gateway to all the pods. Currently, for each pod, there is only 1 container. For example, we decided that each pod will correspond to 1 container that focuses on 1 micro-service. For instance, a pod called "collaboration-service-deployment" only has 1 container which is "collaboration-service" in GKE. For each pod, there is a "service" attached to each pod which allows the ingress to communicate with all the pods through the "service" which is somewhat like a network card/interface for the pod to communicate with the ingress.

Free trial status: \$298.35 credit and 29 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform. DISMISS ACTIVATE

Google Cloud CS219 Group 51 Peer Prep Search (/) for resources, docs, products, and more Search

Kubernetes Engine Ingress Details REFRESH EDIT DELETE KUBECTL LEARN

Routes

Route	Service	Pod selector	Clusters	Pods
app2-peerprpgroup51sem1y2023.xyz/	front-end-deployment-service	app = front-end-app	peer-prnp-cluster	1/1
collabk8-peerprpgroup51sem1y2023.xyz/	collaboration-service-deployment-service	app = collaboration-service-app	peer-prnp-cluster	1/1
collabk8-peerprpgroup51sem1y2023.xyz/healthz/	collaboration-service-deployment-service	app = collaboration-service-app	peer-prnp-cluster	1/1
questionsk8-peerprpgroup51sem1y2023.xyz/api/	question-service-deployment-service	app = question-service-app	peer-prnp-cluster	1/1
questionsk8-peerprpgroup51sem1y2023.xyz/healthz/	question-service-deployment-service	app = question-service-app	peer-prnp-cluster	1/1
usersk8-peerprpgroup51sem1y2023.xyz/users/	profile-service-deployment-service	app = profile-service-app	peer-prnp-cluster	1/1
usersk8-peerprpgroup51sem1y2023.xyz/healthz/	profile-service-deployment-service	app = profile-service-app	peer-prnp-cluster	1/1
matchk8-peerprpgroup51sem1y2023.xyz/	matching-service-deployment-service	app = matching-service-app	peer-prnp-cluster	1/1
matchk8-peerprpgroup51sem1y2023.xyz/healthz/	matching-service-deployment-service	app = matching-service-app	peer-prnp-cluster	1/1

Serving pods

Service	Name	Status	Restarts	Created on ↑
collaboration-service-deployment-service	collaboration-service-deployment-77948f9645-m5q8	Running	0	Nov 12, 2023, 11:39:26 PM
matching-service-deployment-service	matching-service-deployment-6c475864c8-4wds5	Running	0	Nov 13, 2023, 1:09:02 AM
profile-service-deployment-service	profile-service-deployment-d5f9ddf-62wzh	Running	0	Nov 13, 2023, 1:36:51 AM
question-service-deployment-service	question-service-deployment-7cdfd5659-d6x76	Running	0	Nov 13, 2023, 1:42:44 AM
front-end-deployment-service	front-end-deployment-55754b9799-ijqz	Running	0	Nov 13, 2023, 2:21:18 PM

Load balancing resources

Our ingress is considered external by GKE as we decided for the ingress to be a NGINX version so that we can support web socket traffic. GKE automatically provisioned a “HTTP/HTTPS Global Load Balancer” for our ingress so that traffic will flow from the load balancer to the ingress and then to our pods.

Kubernetes Engine

Teams

Applications

Secrets & ConfigMaps

Storage

Object Browser

Backup for GKE

Networking

Services & Ingress

Network Function Optimiz...

Features

Feature Manager

Service Mesh

Security Posture

Marketplace

Release Notes

Editing Ingress Details

REFRESH EDIT DELETE KUBECTL

my-nginx-ingress

DETAILS TELEMETRY LOGS EVENTS **YAML**

SAVE CANCEL

Press Alt+F1 for Accessibility Options.

```

56 namespace: default
57 resourceVersion: "2082475"
58 uid: 1e2d590b-a68b-452c-9c2e-0eb216e9e169
59 spec:
60   rules:
61     - host: app2.peerprepgroup51sem1y2023.xyz
62       http:
63         paths:
64           - backend:
65               service:
66                 name: front-end-deployment-service
67                 port:
68                   number: 3000
69             path: /
70             pathType: Prefix
71     - host: collabk8.peerprepgroup51sem1y2023.xyz
72       http:
73         paths:
74           - backend:
75               service:
76                 name: collaboration-service-deployment-service
77                 port:
78                   number: 8576
79             path: /
80             pathType: Prefix
81           - backend:
82               service:
83                 name: collaboration-service-deployment-service
84                 port:
85                   number: 8576

```

We also used service discovery in our ingress so that our ingress can direct traffic from the load balancer to the correct pods based on the “service name” which corresponds to an internal private IP address that matches the expected pod which can be seen from the image above.

4.3 Depreciated/Past Tech Stack

Client: React, Redux Toolkit, React Router v6, Axios, MaterialUI

Animations: Lottiefiles

Server: Node, Express

Containerization: Docker, Docker Compose, Kubernetes

Cloud: Google Compute Engine (GCE)

Authentication: Google Firebase SSO

Databases: MongoDB, PostgreSQL

Message Broker: RabbitMQ

API Gateway: NGINX Proxy Manager

DNS Server: GoDaddy

CI/CD Tools: Watchtower, GitHub Actions

4.4 Current Tech Stack

Client: React, Redux Toolkit, React Router v6, Axios, MaterialUI, Monaco Editor

Animations: Lottiefiles

Server: Node.js, Express.js, Socket IO

Containerization: Docker, Kubernetes

Container Registry: GitLab

Cloud: Google Kubernetes Engine (GKE)

Authentication: Google Firebase SSO

Databases: MongoDB, PostgreSQL

ORMs: TypeORM, Prisma, Mongoose

Message Broker: RabbitMQ

API Gateway: External NGINX Kubernetes Ingress

DNS Server: GoDaddy

CI/CD Tools: GitHub Actions

In our current project, we removed NGINX Reverse proxy and replaced it with an external NGINX ingress since we migrated to Kubernetes from Docker Compose.

In the front-end, we used redux very heavily as our state manager as passing state to different components inside the front-end code was very difficult and we needed a single “source of truth” to share states between the components. React was also chosen because it was compatible with all browsers and most of our team members have experienced developing React ([NFR3.2](#)).

For the container registry, we chose Gitlab because we wanted to allow everyone to push their services as images easily to a common container repository that is free for everyone to use as Docker Hub requires payment.

For the databases, we chose MongoDB for non-relational database as we felt that it was the easiest database to work with and we chose PostgreSQL as most of our team members are familiar with PostgreSQL for a relational database.

peerpregroup51teamty2023.xyz
Domain Settings [Select a different domain](#)

DNS Records
Forwarding
Namerversers
Premium DNS
Hostnames
DNSSEC

DNS records define how your domain behaves, like showing your website content and delivering your email.

[Add New Record](#)

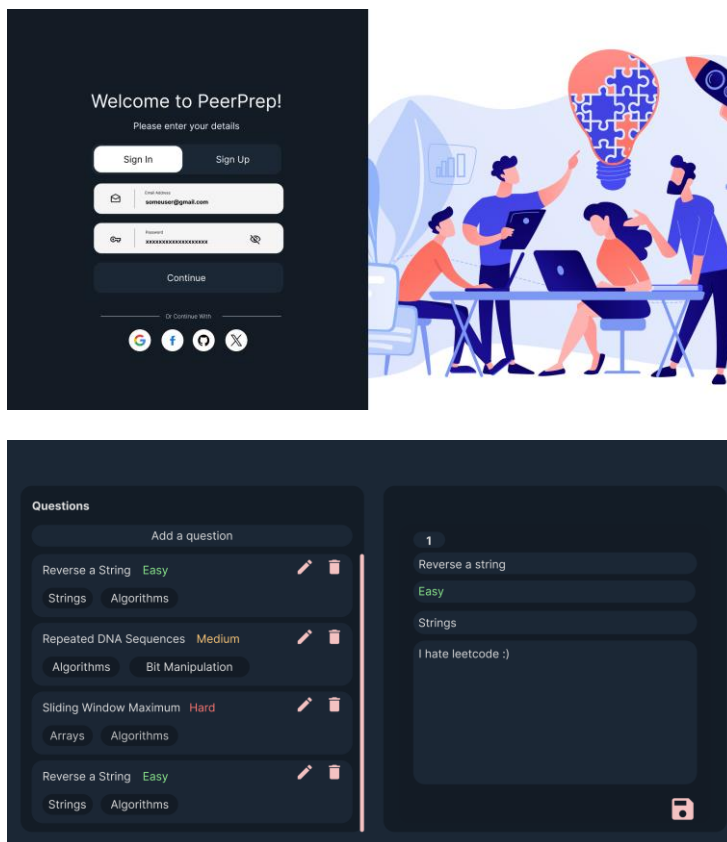
Type	Name	Data	TTL	Delete	Edit
A	match88	34.36.130.164	600 seconds		
A	parked	35.240.242.81	600 seconds		
A	postgress88	34.36.130.164	600 seconds		
A	proxy	35.240.242.81	600 seconds		
A	questions88	34.36.130.164	600 seconds		
A	users88	34.36.130.164	600 seconds		
NS	@	ns07.domaincontrol.com,	1 Hour	Can't delete	Can't edit
NS	@	ns08.domaincontrol.com,	1 Hour	Can't delete	Can't edit
CNAME	www	peerpregroup51teamty2023.xyz,	1 Hour		
CNAME	_domainconnect	_domainconnect.gsl.domaincontrol.com,	1 Hour		

For our DNS provider, we decided to go with GoDaddy, as they provided us with a simple dashboard to create multiple records for us to use and replace our public IP addresses with our chosen domain names as shown above.

5. Design and Implementation Process

5.1 Wireframing

Most of our front-end decisions were based on designs that we have designed beforehand on [Figma](#). We decided that our designs should be medium fidelity so that the front-end engineers would have a sufficient template to code out our features. Another consideration was the lack of time, and opting for high fidelity designs would not be practical. Using medium fidelity also increases the flexibility for our front-end engineers, leaving room for interpretation and iterative development. Some examples of our wireframes are shown below:



5.2 Architecting Back End Services

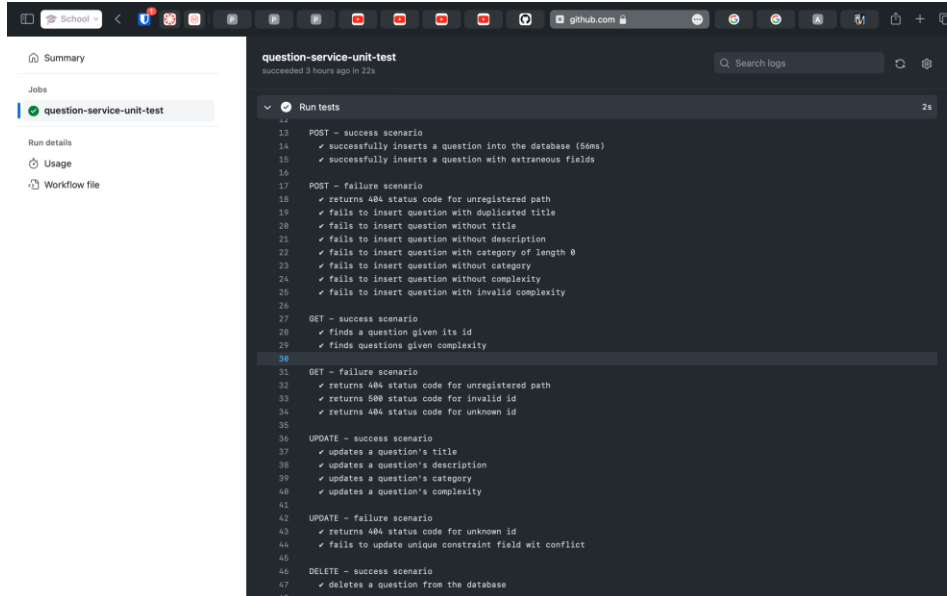
After determining the backend operations required by the frontend, we proceeded to design the backend services. The backend services that implement HTTP servers mostly follow a conventional MVC design pattern, except for matching service and collaboration service because of their real-time requirements using Socket.io. All the services were immediately containerized during the development process. You can find the details of the backend architecture for the following services in their respective README documents:

- [Profile service and admin role allocation](#)
- [Question service](#)
- [Matching service](#)
- [Collaboration service](#)

The services were also designed to meet the Non-Functional Requirements specified above. Major technical decisions such as using Socket.io were made for the purpose of ensuring real-time services work with very low latency (see [NFR1.1](#), [NFR1.3](#), [NFR1.4](#), [NFR1.5](#)). We also decided to go with Node.js and Express.js to build our HTTP servers because it can reliably handle large numbers of simultaneous client connections and requests ([NFR4.1](#)).

5.3 Unit Testing

The service that has implemented unit testing is Question Service. The author of the service has envisioned the development process of the service to be test-driven, hence he created the tests as soon as possible in the development cycle. Question Service utilizes Chai testing and assertion library, Supertest HTTP client, and Mongo-Unit for in-memory mock database. The tests cover all HTTP endpoints of the server and test the robustness of the server for handling various scenarios. A fixture of valid and invalid mock client message bodies was created to simulate various client errors. The tests check for the HTTP response codes and the body of the response to be returned to the client for mismatch against expected values. The testing strategy is to create test suites for each HTTP endpoint, each with positive (successful) and negative (unsuccessful) test cases. Mongo-unit is used to both provide a clean database for the tests and isolate the test database from development and production database. It is ephemeral and is meant to be destroyed and recreated between each test run. The tests are part of the CI/CD pipeline for the project.



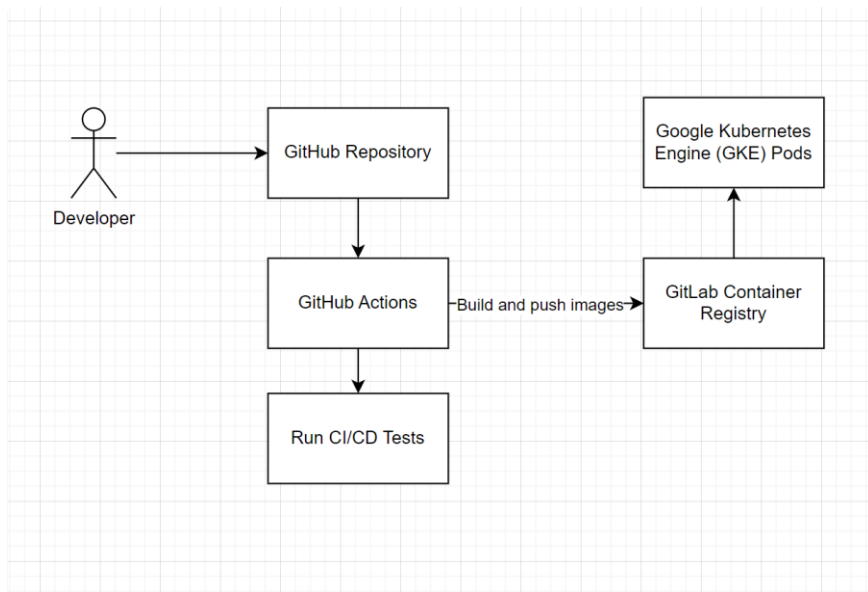
The screenshot displays a web interface for a CI/CD pipeline. On the left, a sidebar shows a 'Summary' section with a 'Jobs' list containing 'question-service-unit-test'. The main area shows the details of this job, which is titled 'question-service-unit-test' and 'succeeded 3 hours ago in 22s'. A 'Run tests' section is expanded, showing a list of 28 test cases. The tests are organized into groups: POST - success scenario (3 tests), POST - failure scenario (8 tests), GET - success scenario (2 tests), GET - failure scenario (3 tests), UPDATE - success scenario (4 tests), UPDATE - failure scenario (2 tests), and DELETE - success scenario (1 test). Each test case is marked with a green checkmark, indicating it passed. The tests cover various scenarios, including successful insertions, failures due to unregistered paths or invalid data, and successful updates and deletions.

```
question-service-unit-test
succeeded 3 hours ago in 22s

Run tests
11
12 POST - success scenario
13 ✓ successfully inserts a question into the database (50ms)
14 ✓ successfully inserts a question with extraneous fields
15
16 POST - failure scenario
17 ✓ returns 404 status code for unregistered path
18 ✓ fails to insert question with duplicated title
19 ✓ fails to insert question without title
20 ✓ fails to insert question without description
21 ✓ fails to insert question with category of length 0
22 ✓ fails to insert question without category
23 ✓ fails to insert question without complexity
24 ✓ fails to insert question with invalid complexity
25
26
27 GET - success scenario
28 ✓ finds a question given its id
29 ✓ finds questions given complexity
30
31 GET - failure scenario
32 ✓ returns 404 status code for unregistered path
33 ✓ returns 500 status code for invalid id
34 ✓ returns 404 status code for unknown id
35
36 UPDATE - success scenario
37 ✓ updates a question's title
38 ✓ updates a question's description
39 ✓ updates a question's category
40 ✓ updates a question's complexity
41
42 UPDATE - failure scenario
43 ✓ returns 404 status code for unknown id
44 ✓ fails to update unique constraint field wit conflict
45
46 DELETE - success scenario
47 ✓ deletes a question from the database
48
```

5.4 CI/CD

We use GitHub actions to automate our CI/CD tasks. For continuous integration, we ensure that each pull request must pass all checks before it can be merged to the master branch. The checks include unit tests for services that have been modified to ensure the developer does not introduce regression into the codebase. The jobs are also configured to not run if a service is not modified, which saves some time and keeps our CI/CD jobs running under 2 – 3 minutes ([NFR1.6](#)). Once a pull request has been merged to the master branch, GitHub actions perform the continuous deployment steps by building the application images and uploading the builds to our container registry on GitLab. It notifies all developers if any of the builds have failed and prevents them from being deployed, keeping the builds on the production server green. The new images are then automatically pulled by GKE from GitLab using the rolling update strategy. This ensures that the application remains available to users by gradually replacing instances of the old version with instances of the new version, reducing downtime ([NFR2.1](#), [NFR2.2](#)). This process is illustrated in the diagram below:



Commented [EM1]: moving this to CI CD section below

6. Nice-to-have Features Implemented

- N1 Communication in Collaboration Service (text-chat)
- N4 Enhancement of Question Service (tagging of questions based on category)
- N5 Enhance collaboration service by providing an improved code editor with code formatting, syntax highlighting for one language, syntax highlighting for multiple languages. Also, we added light and dark mode toggle switch.
- N8 Unit Testing, CI/CD
- N9 Deployment on GCP/GKE
- N10 Scalability with Kubernetes Auto-Scaler
- N11 API Gateway/ K8 Ingress
- N12 Service Registry/Discovery with Kubernetes

7. Challenges Faced

These were some of the challenges faced by us both individually and collectively.

- High initial learning curve
 - Most members of the team were not familiar with many of the tools and technologies used for the project and had to pick them up on the fly.
- Communication
 - Our varying backgrounds and experiences in software development created a communication gap and mismatch of expectations, which had to be aligned.
- Project management
 - The flat structure of our project team means it takes a longer time to decide the direction of development, which delayed the completion of our deliverables as we all agreed to be an egoless team.
 - Team syncs were held on major milestones instead of consistent intervals, causing significant overhead when keeping track of each other's progress.
 - Prioritizing completion of features instead of code quality caused us to rack up significant technical debt and lowered our test coverage.
- Technical difficulties
 - Some technologies that we hoped to use were not compatible with our tech stack, or not mature enough to be used in production such as Bun.js.
 - A significant amount of time was used to troubleshoot problems with Node.js Alpine distribution's C libraries which caused compatibility issues with some important project dependencies.
 - For Subgroup 1, we faced countless nights configuring the external NGINX ingress to work with Google's provisioned HTTP/HTTPS Global Load Balancer as we did not understand beforehand how "health checks" works in GKE.
 - Documentation quality and organisation for important technologies such as Docker and RabbitMQ leave a lot to be desired. We found more useful

information on unofficial forums and articles than on the official documentation website.

- GitLab went down once and broke our CD pipeline.

8. Areas of Improvement

The various areas that the team could have improved on include:

- Improve the standards for code reviews
- Prioritize writing tests
- Enforce consistent version control conventions
- Enforce consistent code quality standards
- Regular stand-ups for progress check
- Writing better documentations

Apart from the abovementioned areas the authors could have improved on, this adaptation of PeerPrep could also improve by having a way to filter and search questions within the question service page. It could also benefit from having a simple rating system of upvotes and downvotes for each question to keep track of their popularity.

Due to time constraints, the authors were also unable to implement a dashboard page as initially intended, which could have been helpful to users by making use of the abovementioned rating system. This could help identify questions by categories that are the most attempted or the highest rated, for example.