



**The University of Hong Kong**  
**Department of Computer Science**  
**2023 – 2024**  
**COMP4801 Final Year Project**  
**Final Report**

**Instant Messaging Software for Academic Consultation**

Written by:

Chan Chin Hung (3035780879)

Team members:

Chan Chin Hung (3035780879)

Fung Ka Ho (3035902635)

Supervisor:

Dr. Chim, Tat Wing

## **Abstract**

Communication is essential for our daily life, as it allows message, information, and knowledge to be exchanged. Instant messaging software give us the opportunity to communicate with others and link everyone across time and distance. Nevertheless, asynchronous communication was adopted in higher education. This paper aims to implement an instant messaging software for academic consultation. It offers features such as real-time chatting, quizzes creating and responding, quiz question statistic information.

We will develop a progressive web app that address the limitations of current instant messaging software and adds new features to facilitate teaching and learning. React with Vite will be used for the frontend website development. Then, manifest and service worker files will be added to make it a valid Progressive Web App. Express framework was added on top of Node.js. Content Delivery network was applied to secure the access of the S3 bucket. The system features of the application are discussed in detail. Finally, the project limitations and difficulties are discussed.

## **Acknowledgement**

I would like to thank my supervisor Dr. Tim who provided a lot of suggestions on the system feature and the technology related to mobile app.

## Table of Contents

Abstract .....	II
Acknowledgement.....	III
List of Figures .....	VI
List of Tables.....	VIII
Abbreviations .....	IX
1 Introduction .....	1
1.1 Overview.....	1
1.2 Motivation.....	3
1.3 Objective .....	4
1.4 Report Outline .....	4
2 Methodology .....	5
2.1 Software Frameworks and Tools.....	5
2.1.1 Frontend.....	5
2.1.2 Backend.....	7
2.1.3 Database .....	8
2.1.4 Deployment .....	8
2.1.5 Content Delivery Network.....	9
2.1.6 Push Notification Service .....	9
2.2 Design and Implementation.....	10
2.2.1 System Architecture .....	10
2.2.2 Database Design .....	12
2.2.3 Authentication .....	14
2.2.4 Offline Caching .....	18
2.2.5 Manifest File Configuration .....	19
2.2.6 Real Time Communication.....	20
2.2.7 The Architecture of CloudFront and S3 Integration.....	22
2.2.8 The push notification.....	24

2.2.9 Deployment on EC2 .....	25
3 Result.....	26
3.1 Login .....	26
3.2 Registration .....	29
3.3 Home Page .....	32
3.4 Profile Page .....	35
3.5 Create/Join Chat Group .....	37
3.6 Real-time chatting .....	39
3.7 Chat Group Management .....	40
3.8 Quiz System Access .....	42
3.9 Quiz Creation .....	42
3.10 Quiz Answering and Updating .....	43
3.11 Quiz Information .....	46
3.12 Push Notification.....	49
4 Difficulties and Limitations.....	50
5 Conclusion.....	51
6 Reference.....	52
7 Appendix .....	55

## List of Figures

1	Leading mobile apps worldwide in 2022, by downloads .....	2
2	Number of mobile phone messaging app user worldwide from 2019 to 2025....	2
3	Relationship among service worker, user's device web browser and server.....	6
4	Hierarchical structure of document-based database.....	8
5	System Architecture Design .....	11
6a	Entity Relationship (ER) Diagram of the system.....	12
6b	JSON schema Diagram.....	14
7	JWT encoding and decoding structure.....	15
8	Email Verification and JWT Authentication Flow.....	16
9	Configuration of Transporter.....	17
10	Authentication Middleware in Express.js.....	17
11	Workbox Runtime Caching Configuration.....	18
12	manifest configuration of the PWA.....	19
13	Architecture of MongoDB change stream with Realm web.....	20
14	Configuration of data access rules on chatroom collection.....	21
15	code snippet of the JWT token generated using private key.....	22
16	The Flow of accessing S3 bucket via CloudFront.....	23
17	Sequence diagram for the push notification.....	24
18	Payload of Firebase Notification.....	25
19	Login page activated user identity select drop-down menu.....	26
20	Login page pre-populated email domain (for student).....	27
21	Login page pre-populated email domain (for teacher).....	27

22	Requesting one-time password.....	28
23	Retrieve an OTP generated by the system from the app user’s mailbox.....	29
24	Providing the received OTP to the designated field.....	29
25	Registration page OTP validation.....	30
26	Required personal information for registration (for student).....	31
27	Required personal information for registration (for teacher).....	31
28	File selector for uploading profile image.....	32
29	Layout of the Home page.....	33
30	Example of home page chat group list.....	34
31	Example of Home page quiz group list.....	34
32	Sorted chat group list creation.....	35
33	Sorted quiz group list creation.....	35
34	Profile page (for student) .....	36
35	Profile page (for teacher).....	36
36	Add new group page layout.....	38
37	Example of joining an existing chat group.....	39
38	Example of sending text messages and sharing documents to the chat group.....	40
39	Chat group information page (for group administrator).....	41
40	Chat group information page (for regular group member).....	41
41	Quiz page layout for teachers (left) and students (right).....	42
42	Example of quiz creation.....	43
43	Example of quiz submission.....	44
44	Notification after quiz submission.....	45
45	Quiz model answer reveal.....	45
46	Layout for the quiz information page.....	46

47 List of Submit and List of Unsubmit Example.....	47
48 Charts for displaying the correctness proportion and choices distribution.....	48
49 Algorithm for generating a list of unique colors.....	48
50 Notification message display on mobile phone.....	49
51 Notification message display on computer.....	50

**List of Tables**

1. Project Schedule.....	55
--------------------------	----



## Abbreviations

API	Application Programming Interface
APK	Android Application Package
AWS	Amazon Web Services
BSON	Binary JavaScript Object Notation
CRUD	Create, Read, Update, Delete
DOM	Document Object Model
EC2	Amazon Elastic Compute Cloud
ER	Entity Relationship
GraphQL	Graph Query Language
gRPC	Google Remote Procedure Call
HTTP	Hypertext Transfer Protocol
NoSQL	Not Only Structured Query Language
OTT	Over-the-Top
S3	Amazon Simple Storage Service
PWA	Progressive Web App
REST	Representational State Transfer
JSON	JavaScript Object Notation
TCP	Transmission Control Protocol
UI	User Interface
URL	Uniform Resource Locator

# **1 Introduction**

## **1.1 Overview**

In our fast-paced digital world, Instant Messaging (IM) has become an integral part of communication. It is a tool that not only ensure the ease of swift communication but also fosters a sense of closeness and immediacy among its users. IM has changed the way we maintain lasting relationships by means of breaking the physical barriers. It gave us the opportunity to immediately share memorable moments, celebrate events or festivals together and provide different support when apart by exchanging information in various format.

The trend of using mobile phone and OTT(Over-the-Top) service such as WhatsApp, Skype and WeChat, has grew steadily in recent years, which will be continued, and the IM software will dominate the mobile app development market. Drawing data from the 2022 Mobile App Download Report, TikTok, Instagram, and WhatsApp rank as the top three most downloaded apps, with the top nine mobile apps including IM services (Figure 1). A report, by L. Ceci in 2023 revealed that around 2.56 billion mobile phone users were using messaging apps for communication in 2019 with an anticipated increase to 3.51 billion by the year 2025 [1] (Figure 2). IM software has not only gained acceptance but is also increasingly becoming the preferred choice for users worldwide.

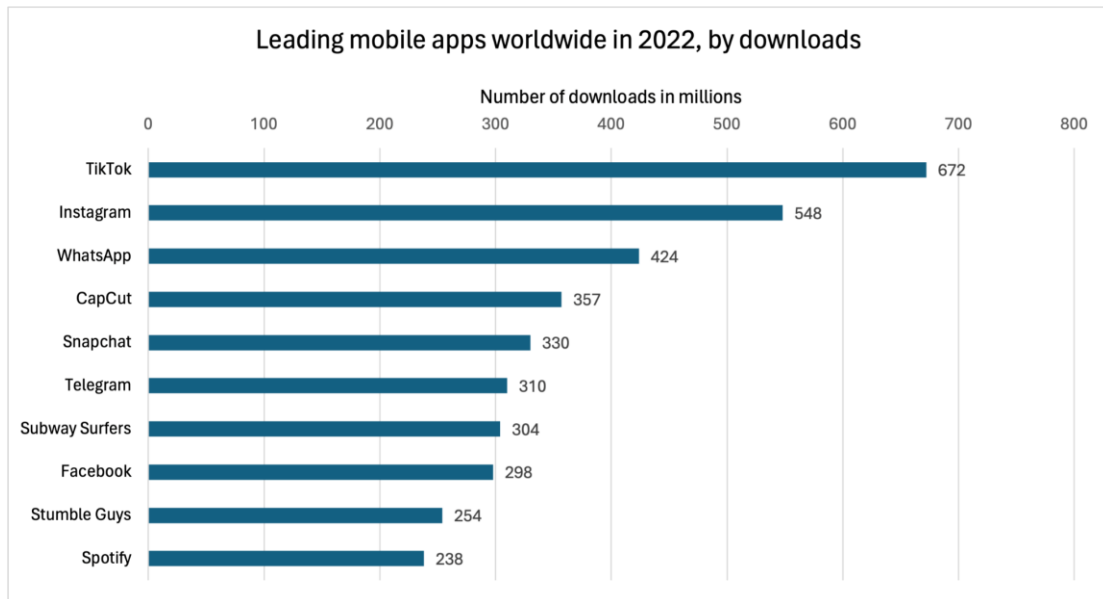


Figure 1: Leading mobile apps worldwide in 2022, by downloads [2]

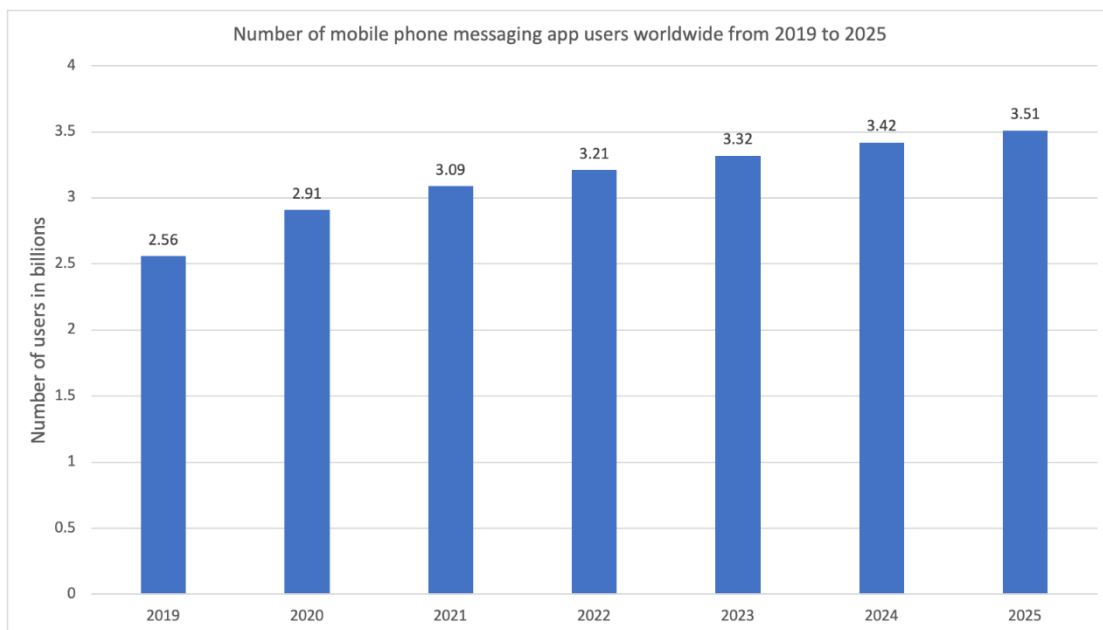


Figure 2: Number of mobile phone messaging app user worldwide from 2019 to 2025 [1]

Instant messaging can likewise be used for educational purposes, as it can support teaching and learning by sharing educational materials and improving the interaction between teacher and student as well as student-peer interaction. With the rise of education, during the COVID 19 outbreak virtual interaction through platforms such as Zoom Microsoft Teams and Kahoot has become a routine, for both teachers and

students. However, In the actual university education environment, students rarely have the opportunity to use instant messaging software to chat with professors or teaching assistants. They typically post messages on the forum provided by the university or email their professors in a formal tone. Quick responses and clarifications of doubts are not available to students immediately. They also struggle to stay a close connection with peers and teachers. These methods are more inconvenient and time-consuming than using instant messaging software, which allows for instant and casual chat without any format restriction.

## **1.2 Motivation**

The use of an instant messaging tool to support teaching and learning in higher education was investigated in the past. It demonstrated that the pedagogical intervention of WhatsApp outside school hours for the course could enhance students' understanding of the subject and improve their learning achievement [4]. There was a positive and significant relationship between the degree of immediacy exhibited by instructors and the level of cognitive and affective learning outcomes and motivation achieved by learners [5].

Nevertheless, WhatsApp or other instant messaging apps have several limitations. The material resource link has an expiration date, so it may not be accessible to students who joined late. They may have to request the teachers to resend it. Moreover, it is difficult for teachers to establish new chat dynamically since they are required to perform extensive configuration and subsequently deliver the invitation link to the students. As a result, this project aims to implement an instant messaging software for academic consultation that addresses the limitations of current instant messaging applications and posits a new feature to facilitate teaching and learning.

### **1.3 Objective**

This application will target at current students and teachers at HKU to enhance the learning experience and boost efficiency. This paper makes two contributions. First, it addresses the limitations of current instant messaging applications. It introduces a consultation platform for students to solve academic problems. The platform provides a private chat or group discussion function for students and teachers to communicate on academic issues. The user interface and the group management system will be enhanced to provide more intuitive functionality, enabling teachers to manage a large number of chat groups efficiently. The software also supports message storage, which allows users to review their past chat messages and files at all times. Second, it introduces new feature that combine functionality of Kahoot. This feature allows teachers to design multiple-choice and then send them to students via the software. Students can answer the questions and receive immediate feedback. The software also collects and analyzes the statistics of the quiz results, which can help teachers to adjust their teaching mode.

### **1.4 Report Outline**

The rest of the paper proceeds as follows. First, it presents the methodology of this project which includes the frontend, backend, database, deployment technologies and the reasons of choosing these frameworks (Section 2.1). It also describes the design concept of the application (Section 2.2). Moreover, it reports the results of the application mainly focus on the system feature (Section 3). Finally, it concludes by summarizing the main point of this paper and discussing difficulties and limitations as

well as the future plan of the project.

## **2 Methodology**

### **2.1 Software Frameworks and Tools**

This chapter will mainly introduce a set of tools and frameworks we have applied to develop the project. The software full stack development tools are introduced:

Frontend (Section 2.1.1), Backend (Section 2.1.2), Database (Section 2.1.3) and Deployment (Section 2.1.4) followed by the network and infrastructure services: Content Delivery Network(Section 2.1.5), Push notification service(Section 2.1.6).

#### **2.1.1 Frontend**

A progressive web app (PWA) will be built for the project. As it is relatively easy to build, maintain and release to users. A PWA has more capabilities than a normal web app, which makes it very user-friendly as it combines the best aspects of web and app experience and works offline or on low-quality network connections owing to service workers that cache the app data locally on the device [5]. Many existing PWA, such as Uber, Pinterest, and Starbucks, provide a responsive and intuitive UI that is similar to native applications.

In order to convert a normal website to a PWA, the service worker and manifest files are a must. The manifest file mainly configures the metadata about the web app, such as its name, icons, theme color, orientation, and start URL. The purpose of the manifest is to enable the web app to be added to the home screen of the user's device, and to customize its appearance and behavior when launched from there [6]. Whereas

a service worker is a background script that runs in another thread from the browser. Since it is able to act as a proxy to intercept and handle network requests, it allows the website to be visited normally even offline by retrieving network resources stored in the browser cache (Figure 3). One important feature is the push notification which allows the user to receive messages even if a browser is not open [5, 6].

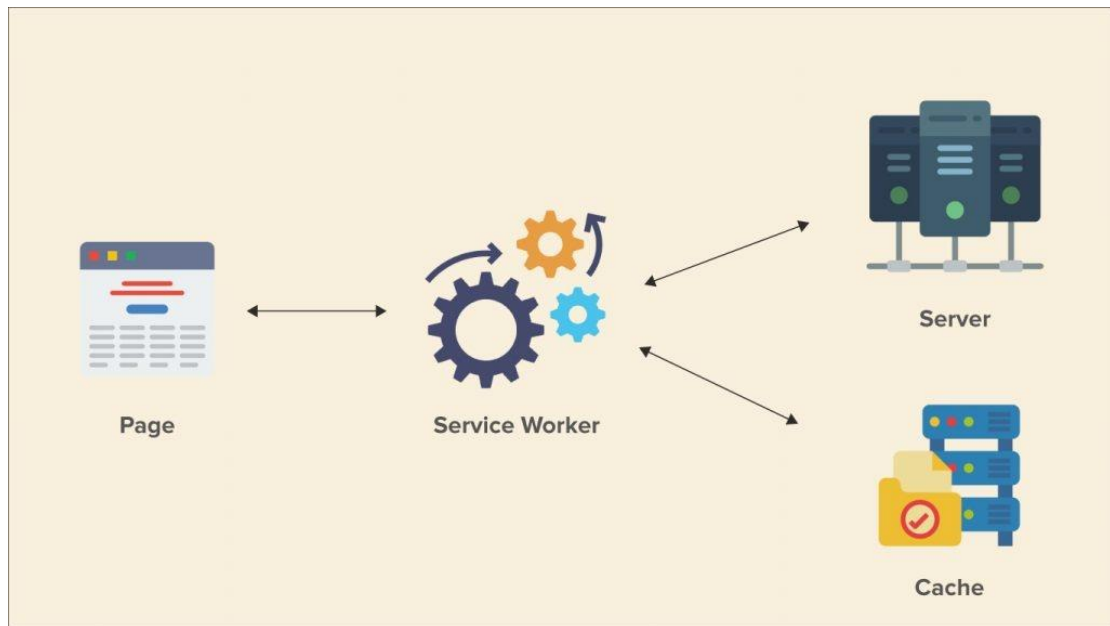


Figure 3: Relationship among service worker, user's device web browser and server [7].

React is a JavaScript library for creating web and native UI that was developed by Facebook in 2011. It uses components to make the code more reusable and maintainable as well as combines complex UI elements that improve the rendering speed with a virtual DOM method [8-9]. Compared with other frontend frameworks, react has a large eco-system that provides detailed documentation and adequate react UI libraries. Moreover, the Vite build tool is integrated together with react to speed up the development cycle as it makes the development server faster and real-time update of the source code can be achieved [8,10]. React with Vite was adopted for the frontend website development. Then, manifest and service worker files will be added

to make it a valid PWA.

### **2.1.2 Backend**

The project backend is based on Node.js, which is a popular and cross-platform JavaScript runtime environment for developing web servers. It uses an asynchronous event-driven model that can run smoothly on various operating systems and handle concurrent connections [11]. In contrast with other server frameworks, its comprehensive and reliable package manager npm provides over two million packages with detailed documentation [12]. It allows us to build complex server logic and reuse the features built by other developers. As a result, the backend server will be built by node.js.

Express framework was added on top of Node.js to offer a fast and flexible way to handle and create API requests and routing. Its middleware function allows us to break down the code into smaller modules such as error-handling, logging, authentication, etc [13]. These modules can be reused in different API endpoints and thereby speed up the backend development process.

The Server-Sent Event protocol also served for real-time communication. It can accommodate more complex needs that require the server to send messages directly to the client at any time. Unlike normal HTTP requests, it is a simple long-lived unidirectional HTTP connection, providing high efficiency and low latency connection [14], which is essential for our application, as we need to handle and transfer a large number of real-time chat data.



### 2.1.3 Database

MongoDb, which is a documented-based NoSQL database, will be used in this project. It stores data in a structure of BSON that is similar to JSON (JavaScript Object Notation), which is a data-interchange format in JavaScript and is simple and readable for humans. In MongoDB a collection is similar to a table in a database. Collections do not have a strict structural requirements like tables do. This allows documents, within a collection to vary in fields and structures (figure 4). Compared with traditional relational databases, it is efficient and flexible when performing data manipulation on huge datasets [15]. Additionally, there are multiple user-friendly packages that support the MongoDB driver connection and facilitate data manipulation, and data modeling such as mongoose, monk, etc. As a result, it is a suitable database for a Node.js server.

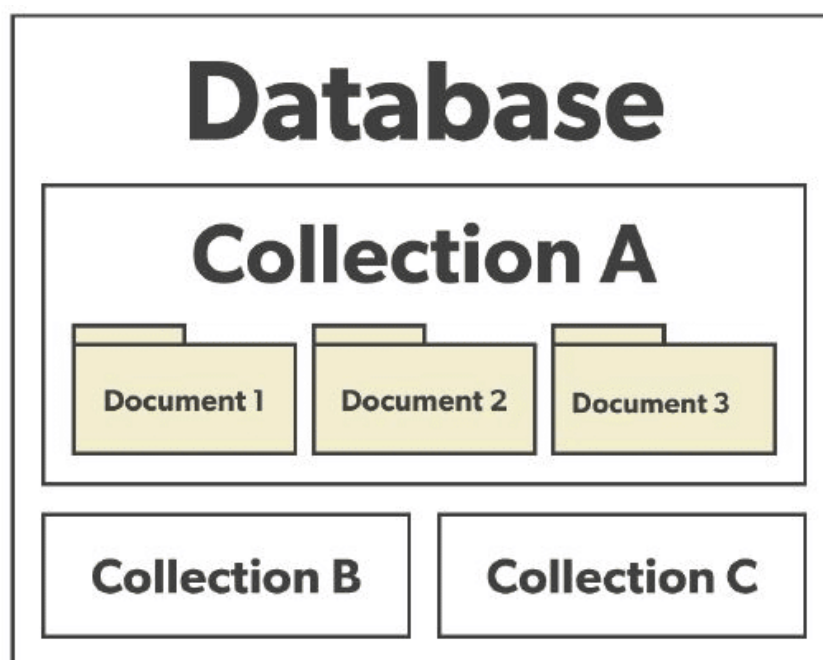


Figure 4: Hierarchical structure of document-based database [16]

### 2.1.4 Deployment

The server of the project was hosted on Amazon Elastic Compute Cloud (Amazon

EC2). Amazon EC2 provides a lot of types of virtual machine which allows users to rent according to their needs. Since it offers flexible and scalable virtual machine instances with plenty of built-in functionality and features, the maintenance cost is reduced as physical machines or other hardware are not required. The node.js server will be deployed on a 24/7 Amazon EC2 virtual machine instance.

The database was hosted on a cloud service called MongoDB Atlas which provides a multi-cloud database platform. Azure, AWS, Google Cloud are the famous cloud service providers that allow us to deploy the database on them. Moreover, MongoDB Atlas offers real-time analysis of the data and graphic interface as well as data visualization features which accelerate the development and debugging process [17]. However, we have to decide which cloud platform suits our needs best. Since the server is deployed on AWS, our database will deploy on the same cloud platform.

### **2.1.5 Content Delivery Network**

Typically, items stored in an S3 bucket can be accessed by the public allowing users to easily retrieve the data through a URL link. Nevertheless, this may lead to security problems that unrestricted access may cause enormous malicious downloads which will lead to high cost due to the increased network bandwidth consumption [18].

Moreover, how quickly users can access resources depends on their proximity to the S3 buckets region. Users locate near the S3 bucket region enjoy faster loading times for resources while those away may experience access speeds. Therefore, Amazon CloudFront CDN was introduced to improve resource access speed across various regions.

### **2.1.6 Push Notification Service**

Firestore Cloud Message (FCM) is crucial for delivering push notification in our

application. This feature keeps the user received chat messages in time without them having to keep the application the online. By linking Firebase with MongoDB, the registration token stored in MongoDB allows Firebase to target and send notifications to users smoothly.

## **2.2 Design and Implementation**

This section illustrates the design and implementation involved in building the software. First, design of the system is described in section 2.2.1. Second, the database schema design is illustrated in section 2.2.2. In section 2.2.3 The Authentication discuss the method for verifying user identity. In section 2.2.4 Offline Caching addresses how the system handles data when not connected to the internet. Thereafter, the manifest file configuration is explained in section 2.2.5 while the real time communication mechanism is detailed in section 2.2.6. 2.2.7 The Architecture of CloudFront and S3 Integration describes how the content delivery network and storage service work together. 2.2.8 The Push Notification section covers the method of sending information to users proactively. Lastly, 2.2.9 Deployment on EC2 outlines the steps for launching the system on the cloud computing platform.

### **2.2.1 System Architecture**

The whole system architecture describes the interactions between user devices and backend server including detail data communication methods, notification strategies, offline caching mechanisms and database connections (Figure 5).

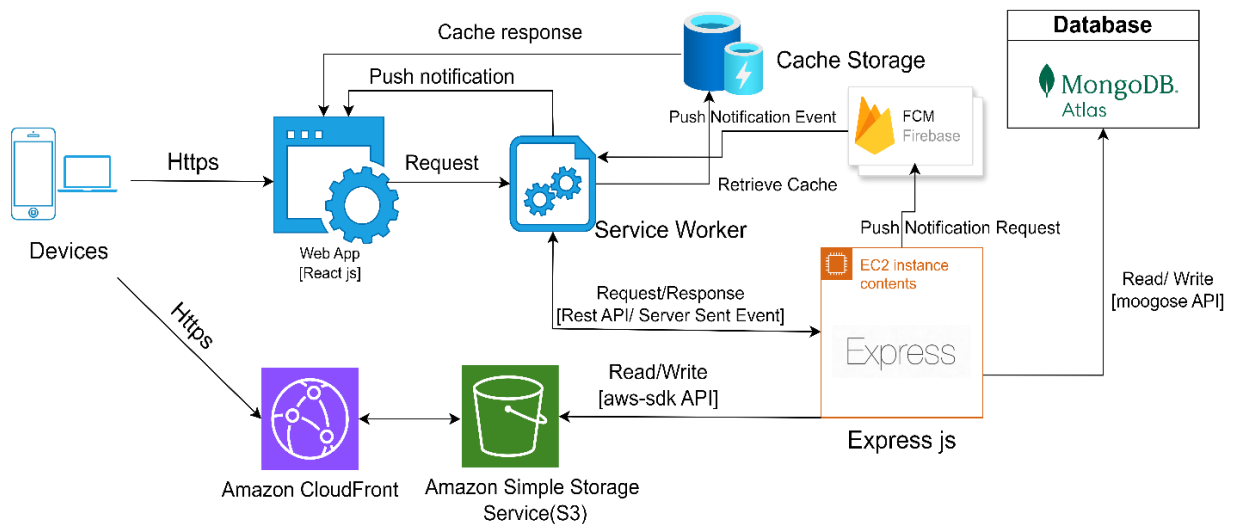


Figure 5: System Architecture Design

As for the frontend, clients can visit the official website and add the app to home page or download and install the application directly on their devices through the APK file. The users can open the app and it will automatically retrieve resources which will then be stored inside the browser cache. The browser obtains the resources from the cache if there are caches matches in the browser cache. Moreover, the app is designed to communicate with the Express JS server via Rest API endpoint to perform various operation such as data retrieval, data updates and deletions. The MongoDB change stream is open on the chatroom collection for the purpose of real-time communication. For offline users, the message will be displayed on the devices with the aid of the notification API and the Firebase Cloud Messaging. A notification will appear with the content of the message.

With respect to the backend, the server will handle the operational request from the client and perform corresponding CRUD operation to the database through mongoose API. The uploading and transferring of files will be temporarily stored in the AWS S3 bucket via aws-sdk API. The whole system architecture helps us facilitate the integration of new technologies and feature in the software development cycle.

## 2.2.2 Database Design

Furthermore, the overview of the whole system schema has been presented in Figure 6. It includes essential entities in our project such as user, chatroom and quiz, along with the relationship between each entity (figure 6.a). As a NoSQL database was used, the ER diagram was converted into a JSON schema diagram for practical implementation (figure 6.b).

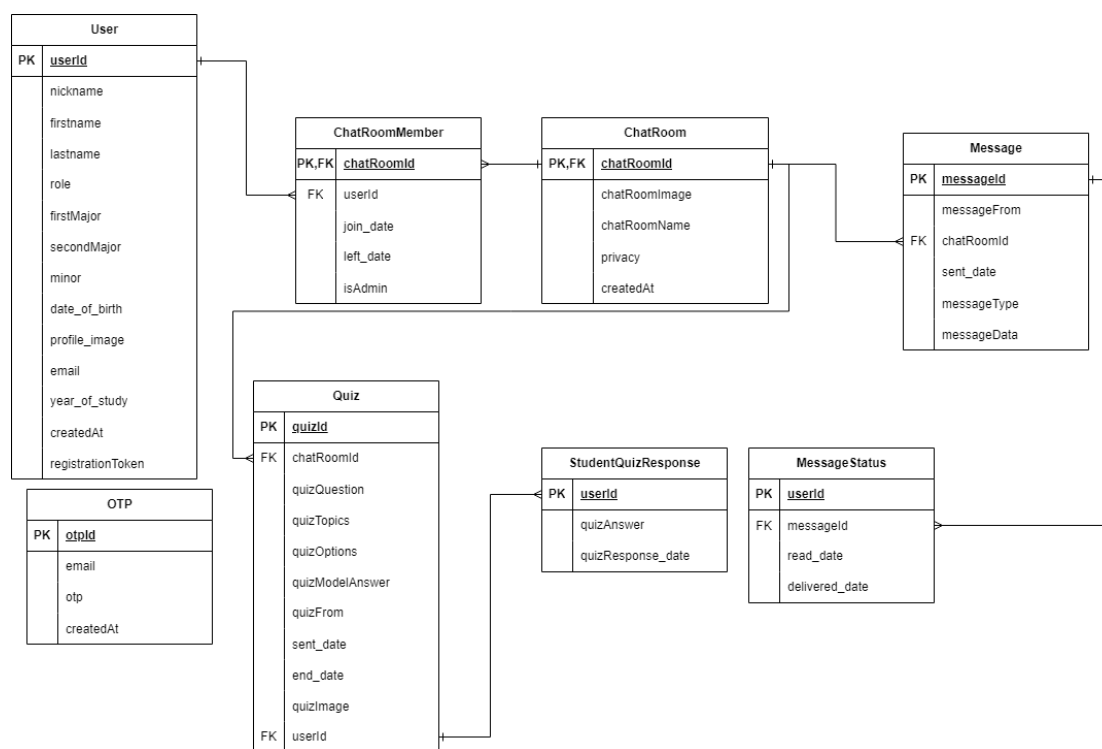


Figure 6a. Entity Relationship (ER) Diagram of the system

**User:** This entity models either a student or a teacher. It stores the user’s personal information such as, nickname, firstname, lastname, firstMajor, secondMajor, minor, date of birth, year of study, profile image, role, email , registrationToken and createdAt.

**ChatRoomMember:** This entity stores the information of the user in the chat room.

Basic information such as `join_date`, `left_date`, and `isAdmin` are stored.

**ChatRoom:** This entity models the chat room. It store the name and image data of the chatroom.

**Message:** This entity stores the message sent by the users. It store details of `sent_date`, `messageType`, `messageData`, `messageFrom`. Message are stored so that users can retrieve history of their conversation even if they join the group late. `MessageType` is the type of the message such as, text, image, txt, zip, pdf. `MessageData` is the content of the data which can be String and URL of the resources. As the chatroom may have multiple messages and the message only belongs to a chatroom, the relationship between the chatroom and the message is one-to-many.

**Quiz:** This entity store the quiz created by the teachers. It has attributes such as, `quizQuestion`, `quizOptions`, `quizTopics`, `quizModelAnswer` , `sent_date`, `end_date` and `quizImage`. `quizQuestion` is the question that is asked in the quiz and `quizTopics` is the related topics of the quiz. The relationship between the chatroom and the quiz is also one-to-many.

**studentQuizResponse:** In order to record students' responses, the entity model store `quizAnswer` and the `quizResponse_date` when the students submit their answer. The `userId` is used to identify the student who submitted their answer.

**MessageStatus:** In order to track whether the user has received or seen the message. This entity stored the `deliver_date` and `read_date` of the message by particular user.

**OTP:** This entity is used for verification of the OTP submitted by user. It contains attributes of `email`, `otp` and `createdAt`.

User	ChatRoom	OTP
<pre>{   "userId": ObjectID,   "nickname": String,   "firstname": String,   "lastname": String,   "role": String,   "email": String,   "firstMajor": String,   "secondMajor": String,   "minor": String,   "date_of_birth": Date,   "profile_image": String,   "year_of_study": String,   "createdAt": Date,   "registrationToken": String,   "chatRoomMember": [     {       "chatRoomId": ObjectID,       "join_date": Date,       "left_date": Date,       "isAdmin": Boolean     }   ] }</pre>	<pre>{   "chatRoomId": ObjectID,   "chatRoomImage": String,   "chatRoomName": String,   "privacy": String,   "createdAt": Date,   "message": [     {       "messageId": ObjectID,       "messageFrom": ObjectID,       "sent_date": Date,       "messageType": String,       "messageData": String,       "messageStatus": [         {           "userId": ObjectID,           "read_date": Date,           "deliver_date": Date         }       ]     }   ]   "quiz": [     {       "quizId": ObjectID,       "quizQuestion": String,       "quizTopics": [String],       "quizOptions": [String],       "quizModelAnswer": String,       "quizFrom": ObjectID,       "sent_date": Date,       "end_date": Date,       "quizImage": String,       "end_date": Date,       "studentQuizResponse": [         {           "userId": ObjectID,           "quizAnswer": String,           "quizResponse_date": Date         }       ]     }   ] }</pre>	<pre>{   "email": String,   "otp": String,   "createdAt": Date }</pre>

Figure 6b. JSON schema Diagram

There are a total three collection: User, ChatRoom and OTP. The chatRoomMember, the message, the quiz ,the studentQuizResponse as well as the MessageStatus entities are embedded in other entity models as there are various one-to-many relationships that could be transformed into a single document.

### 2.2.3 Authentication

As the HKU login API has some limitations and it may only support programming platforms such as JAVA, PHP and C#. As a result, authentication is performed by using the user's email address. The current mainstream authentication methods consist of two types: session-based authentication and token-based authentication. Session-based authentication relies on the session information. The server side will initiate the create of a session upon the user login. The session ID will be stored as a cookie inside the browser. Validation can be accomplished by checking the session ID from

the cookie against the corresponding session information stored in in-memory cache. Nevertheless, the user's state information is stored on the server. This poses challenges in distributed or stateless environments and subsequently hinder the distribution of user session across multiple servers. Therefore, token-based authentication will be adopted.

The image shows a web interface for encoding and decoding JWTs. On the left, under the heading "Encoded" with a sub-label "PASTE A TOKEN HERE", there is a text area containing a long alphanumeric string: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MzY1MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c`. On the right, under the heading "Decoded" with a sub-label "EDIT THE PAYLOAD AND SECRET", there are three sections: "HEADER: ALGORITHM & TOKEN TYPE" showing `{ "alg": "HS256", "typ": "JWT" }`; "PAYLOAD: DATA" showing `{ "sub": "1234567890", "name": "John Doe", "iat": 1516239022 }`; and "VERIFY SIGNATURE" showing the HMACSHA256 function signature: `HMACSHA256( base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret )`. There is a checkbox labeled "secret base64 encoded" which is currently unchecked.

Figure 7: JWT encoding and decoding structure

The JSON Web Tokens (JWT) authentication method is commonly used for authentication in the modern web technology. The JWT contains three parts which define the algorithm for encryption, the payload data, and the signature (Figure 7) [19]. The payload data can store frequently used data to minimize the number of data retrievals while the signature ensures that the integrity of the message is maintained.



The authentication flow is illustrated (Figure 8). The user first types in the email address. The server generates a one-time password and then sends an email to the user. Meanwhile, the server creates a document in the OTP collection for verification purposes. The user logs in or registers with the email and the one-time password. The server verifies the email and password. It then creates a JWT token based on the private key stored inside the server's environment variable and sends the token back to the user. For other requests that require identity authentication, the user can send request with a JWT header.

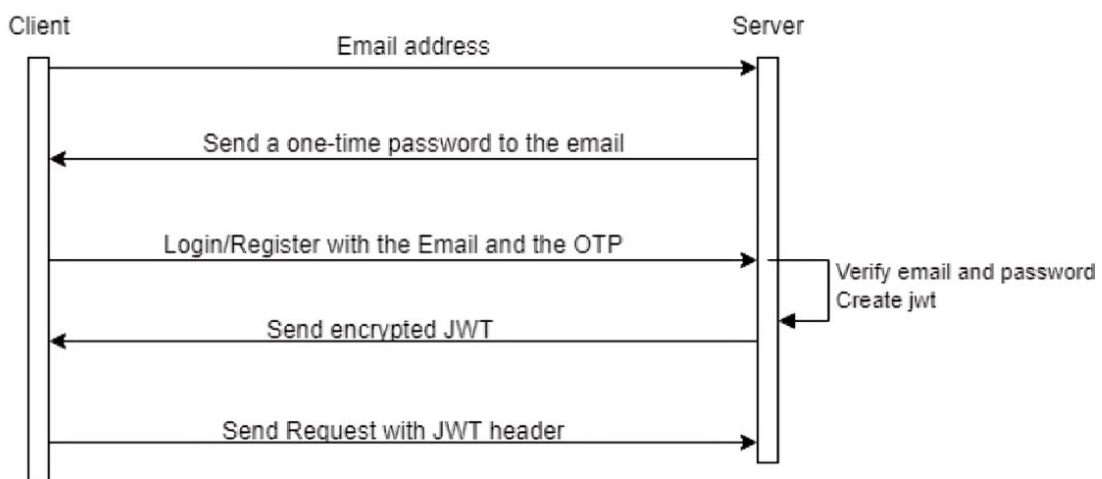


Figure 8: Email Verification and JWT Authentication Flow

In the context of the mail service, we opted not to use any existing mail services to send the OTP to the user. Instead, we decided to rely on the Node.js library, Nodemailer, to manage our Gmail account, for sending emails (Figure 9). As this service is free of charge with a daily cap of 500 recipients which suits our audience size well for sending OTP emails. Moreover, Google has shifted from email and password authentication to OAuth2 requiring us to create an application specific password for the Nodemailer. After defining the transporter, transport object is defined to send out emails.

```

const transporter = nodemailer.createTransport({
  service: "gmail",
  auth: {
    user: "fypimsac@gmail.com",
    pass: process.env.APP_PASSWORD,
  },
  port: 465,
  host: "smtp.gmail.com",
});

```

*Figure 9: Configuration of Transporter*

This is an example of authentication middleware in Express.js (Figure 10). The middleware allows us to verify a user's identity before proceeding with any steps. Most incoming requests go through this middleware. In the function we extract the token from the authorization headers. Then use the function to decode the token using the secret stored on the server as an environment variable. The token is attached to the req.user property. The next() function is called to allow the request to continue to the router handler.

```

function auth(req, res, next) {
  const token = req.headers.authorization;
  if (!token) {
    return res.status(401).json({ message: "Unauthorized" });
  }
  jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {
    if (err) {
      return res.status(401).json({ message: "Unauthorized" });
    }
    req.user = decoded;
    next();
  });
}

```

*Figure 10: Authentication Middleware in Express.js*

## 2.2.4 Offline Caching

The code snippet, for setting up the workbox configuration clearly outlines how assets and API responses are cached in the app (Figure 11). In order to make sure that resources can be accessed offline and load swiftly a global pattern was utilized to cache file formats like JavaScript, CSS, HTML and images. The runtime caching segment specifies using CacheFirst to enhance the overall performance and lessen server load, which is especially beneficial for endpoints that do not change frequently or demand data freshness.

```
workbox:{
  globalPatterns: ["**/*.{js,css,html,png,jpg,jpeg,json,svg}"],
  runtimeCaching:[
    {
      urlPattern: /\V(users|chatRoom|quiz)/,
      handler: "CacheFirst",
      options: {
        cacheName: "api-cache",
        expiration: {
          maxEntries: 100,
          maxAgeSeconds: 60 * 60 * 24,
        },
      },
    },
  ],
},
```

*Figure 11: Workbox Runtime Caching Configuration*

### 2.2.5 Manifest File Configuration

For the application to be installed on user's desktop or mobile device, it is necessary to configure the key properties in the manifest file: the name and icons. The detailed manifest configuration for the Progressive Web App (PWA) is shown in the diagram. It includes the apps name, "Instant Messaging Software For Academic Consultation " along with a shortened version "IM". Regarding the icons, a 192x192 pixel icon and a 512x512 pixel must be provided to allow browser to scale the icons to fit across various devices and platform. In addition, 'standalone' display mode is adopted to enable the app to operate like a mobile app rather, than a webpage.

```
manifest: {
  name: "Instant Messaging Software",
  short_name: "IM",
  description: "Instant Messaging Software",
  icons: [
    {
      src: "./src/assets/android-chrome-192x192.png",
      sizes: "192x192",
      type: "image/png",
    },
    {
      src: "./src/assets/android-chrome-512x512.png",
      sizes: "512x512",
      type: "image/png",
    },
  ],
  theme_color: "#000000",
  background_color: "#ffffff",
  display: "standalone",
}
```

Figure 12: manifest configuration of the PWA

## 2.2.6 Real Time Communication

In order to achieve a real time delivery of event without page refreshing, the real time communication was established using MongoDB change streams which are events coming right from the database operations log and they allow applications to access the real-time data changes without prior complexity and the need for a backend [20].

The database backend will open a change stream and initiate a Server-Sent Events when a collection is monitored.

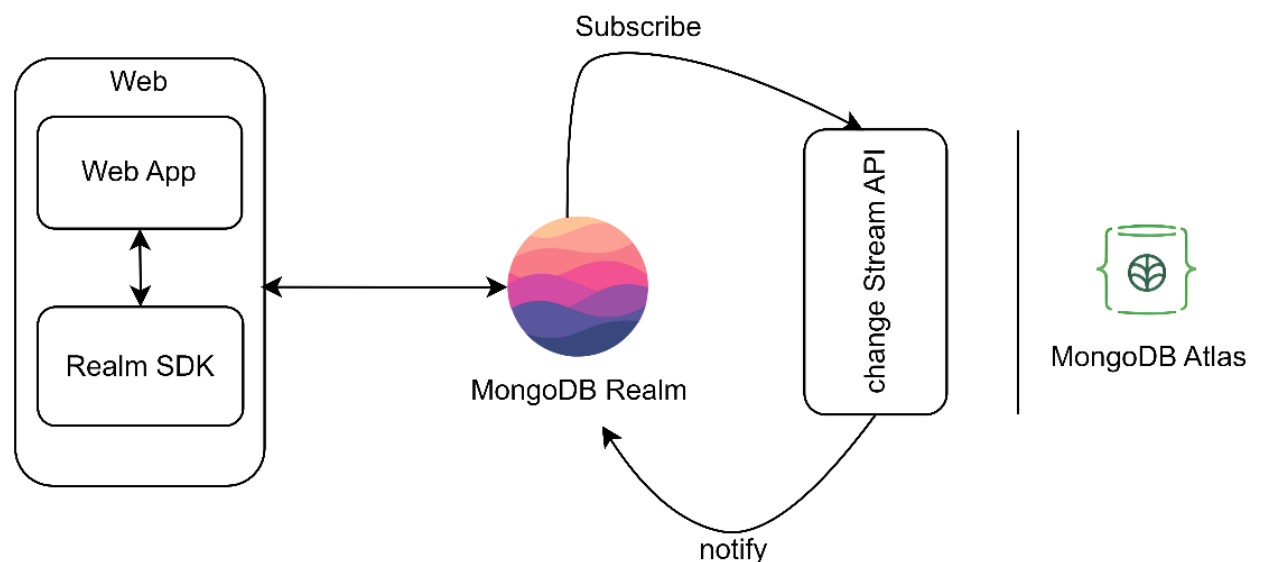


Figure 13: Architecture of MongoDB change stream with Realm web

For the purpose of ensuring the correct change stream is open on the chatroom collection, it is necessary to verify the user identity before monitoring begins. The MongoDB Realm Web SDK was integrated to allow users to access data stored in MongoDB Atlas with authentication (Figure 13). Custom JWT token authentication provider was used for user authentication. The consistent use of the same private key ensure that user can be authenticated with the same JWT token present in the authentication headers.

Data access rules must be assigned to each user to constrain what data they access. The Atlas app service allows us to configure dynamic permissions including both document-level and field-level permissions by setting up the roles to restricts access to sensitive information and prevent unauthorized writes [21].

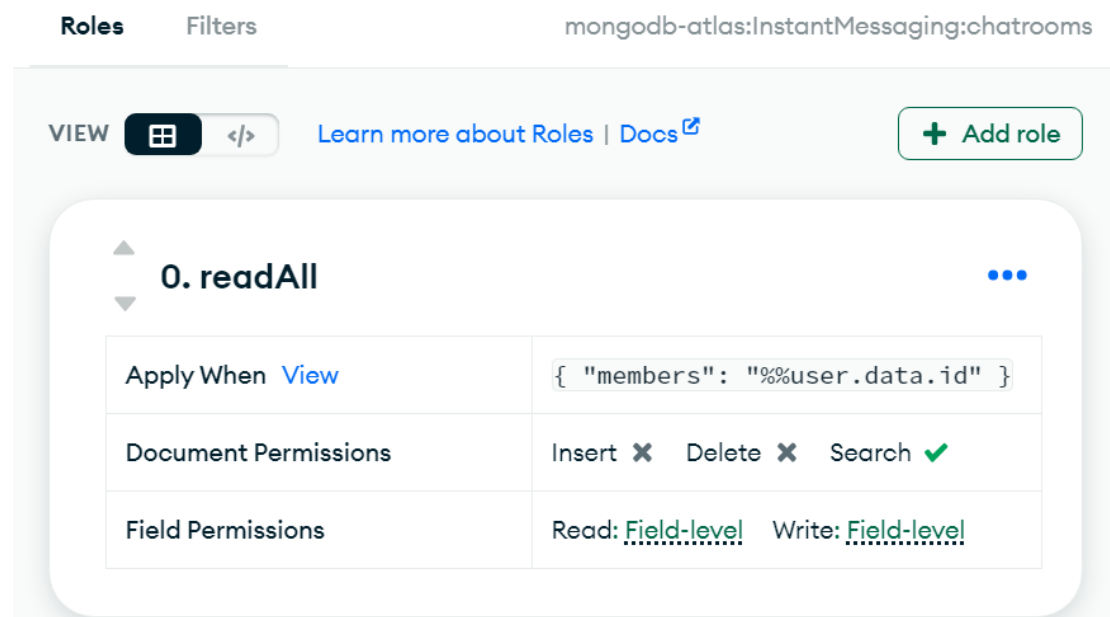


Figure 14: Configuration of data access rules on chatroom collection

Role predefines a set of permissions that regulate a user’s access at both the document and field levels, and they are evaluated using the “apply when” expressions (Figure 14). In our case, users are permitted to read the message field only if they are the members of the chatroom and they are not allowed to insert or delete any documents in the database. As a result, users’ application can keep track of the new message of chatroom that they belong to.

```
const token = jwt.sign(  
  { id: user._id,  
    sub: user._id,  
    aud: "application-0-eetka",  
    userId: user._id,  
  },  
  process.env.JWT_SECRET,  
  { expiresIn: "7d" }  
);
```

*Figure 15: code snippet of the JWT token generated using private key*

The 'aud' parameter is set up with the Realm Web App ID for validation purposes and the JWTs metadata is linked to the users ID. The Realm Web lets us to specify the location of the metadata by setting the mapping between the metadata field in the JWT and its corresponding attribute in the user profile. Additionally, the JWT's metadata cannot be changed by user as it invalidates the signature of the token. This ensures that the user cannot monitor other documents in the chatroom collection.

### **2.2.7 The Architecture of CloudFront and S3 Integration**

The resources of the S3 bucket are protected by the CloudFront with signed URL. The content delivery network (CDN) is a group of server and data centre distributed globally that cache the resources [18]. This allow users to connect to a data centre that is geographically closer them. Meanwhile, the signed URL was applied to allow users to grant temporary access to the private resource. The access control and expiration date can be configured to protect the S3 bucket.

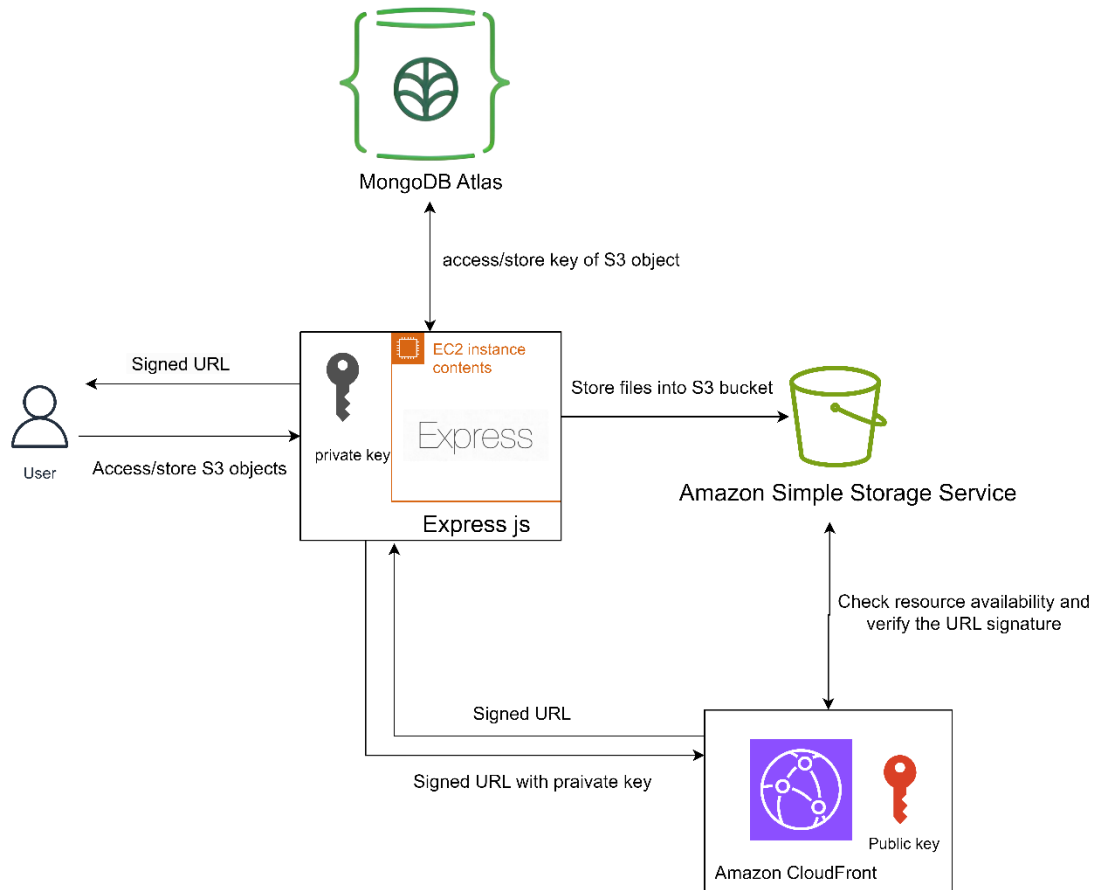


Figure 16: The Flow of accessing S3 bucket via CloudFront

When users access the files stored in S3, they send a post request to the server. The server retrieves the key of the S3 object from the MongoDB Atlas. The server uses the private key and the key to generate a signed URL and sends back to the client. Upon accessing the signed URLs, the CloudFront verifies the URL signature with the trusted public key and checks the availability of the resource and the permissions to ensure that the users have the right to access the object. When the users want to store files in the database, they first create an S3 key in the MongoDB Atlas and then upload the files to the S3 bucket.



## 2.2.8 The push notification

This sequence diagram describes the operation of push notification (Figure 17). When a user visits the website for the first time, the browser generates a pop-up window seeking permission for notification. Once the permission is granted, the service worker is registered. This service worker contains code that runs the Firebase configuration setting up the cloud messaging service. After the Firebase configuration is set up, the client generates a registration token, which is then be stored in MongoDB.

When the server needs to send notification to the clients, it retrieves the tokens from the database. Then, it makes use of the Firebase Cloud Messaging to send these notifications to the service worker. When the notification is received, the service worker calls the notification API to display the message.

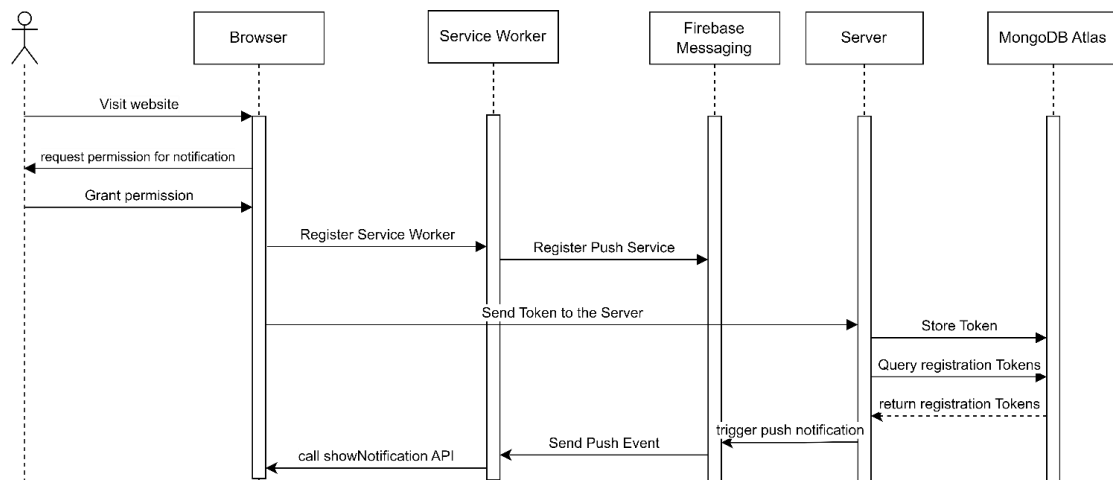


Figure 17: Sequence diagram for the push notification

The web push section was configured for delivering notifications in supported web browser (Figure 18). The object contains title, body, icon and image. The icon represents the user profile image. If the user does not have a profile image, the default image will be displayed. Mobile devices still make use of the web push to receive

message as the PWA can send web push notifications to users via background threads.

```
const payload = {
  webpush: {
    notification: {
      title: message.title,
      body: message.body,
      icon: message.icon ?? profile_image,
      image: message?.image,
    },
  },
  sound: "default",
  click_action: "OPEN_ACTIVITY",
  priority: "high",
  tokens: registrationTokens,
};
```

*Figure 18: Payload of Firebase Notification*

### 2.2.9 Deployment on EC2

The backend server was hosted on the free tier T2.micro instance, which offers 1 GB memory and processing capacity with 1vGPU. The region was set to ap-southeast-1 (Singapore) as it is the closet region to Hong Kong. Also, the deployed instance was configured with Amazon Linux 2 platform which is a Linux operating system from AWS. In order to allow user to visit the website, the port 80 and port 443 were opened to the public by setting up the inbound and outbound rules in the security group. For the purpose of the working of the mail service, the outbound rule for port 456 was also opened. As the project code was stored in the private repository in GitHub, we created a personal access token to grant a temporary access to the repository.

In addition, the Firebase Cloud Messaging requires the HTTPS protocol to work

properly, it is a must to get an SSL certificate and a domain for the EC2 instance. As a result, the domain name “fypimsac.xyz” was purchased from a domain name registrar.

### 3 Result

In this chapter, the result and system features of our final product are discussed.

#### 3.1 Login

For the user login system, a drop-down menu with default options was provided to streamline input and improve control over the user input (Figure 19). The user can change their user identity by selecting either “student” or “teacher” and the corresponding email domain will be adjusted to “@connect.hku.hk” (Figure 20) and “@hku.hk” (Figure 21).

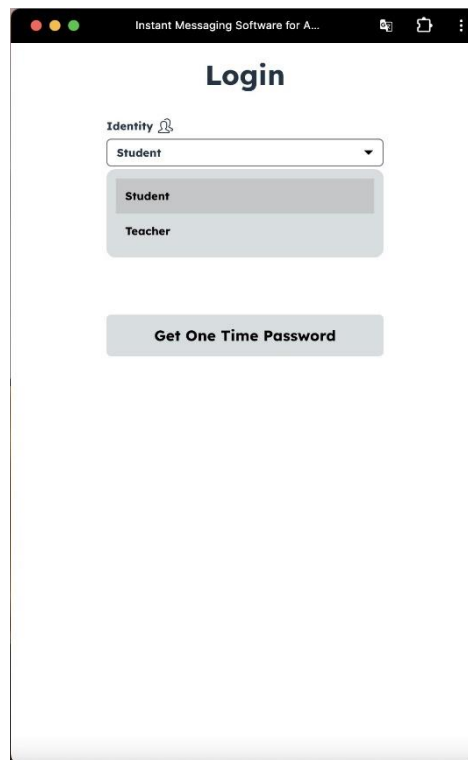
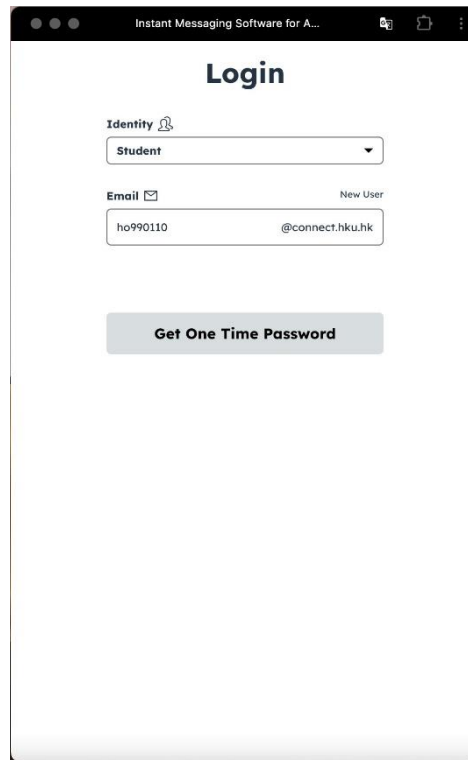


Figure 19: Login page activated user identity select drop-down menu

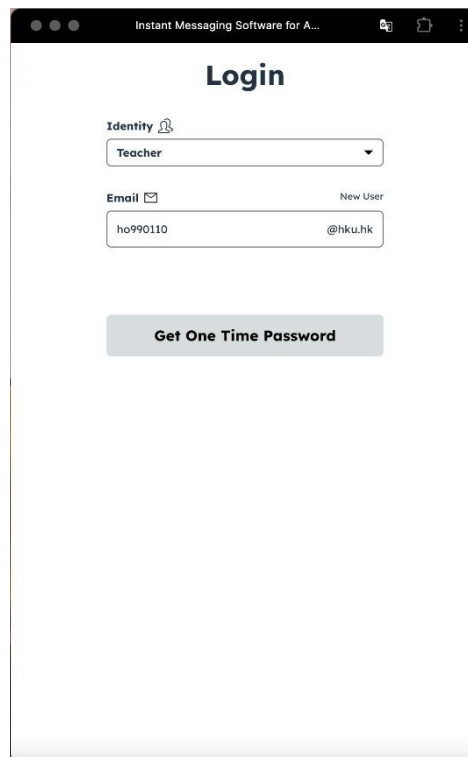
After selecting the user identity, the user can enter the email address of their HKU

PORTAL account by entering the UID and the email domain is determined by the user identity.



The screenshot shows a web browser window titled "Instant Messaging Software for A...". The page has a "Login" heading. Below it, there is an "Identity" section with a dropdown menu currently set to "Student". Underneath is an "Email" section with a "New User" link. The email input field contains "ho990110" and "@connect.hku.hk". At the bottom of the form is a button labeled "Get One Time Password".

Figure 20: Login page pre-populated email domain (for student)



The screenshot shows a web browser window titled "Instant Messaging Software for A...". The page has a "Login" heading. Below it, there is an "Identity" section with a dropdown menu currently set to "Teacher". Underneath is an "Email" section with a "New User" link. The email input field contains "ho990110" and "@hku.hk". At the bottom of the form is a button labeled "Get One Time Password".

Figure 21: Login page pre-populated email domain (for teacher)

After entering the email address, the user can click on the “Get one-time password” button to request to server to generate a unique 6 digit alphanumeric one-time-password (OTP). The OTP email will then be sent to the provided email address (Figure 23). After the successful of the sending of the OTP email, the one-time-password input field will be appeared to allow user to type in the OTP. Moreover, the notification indicating “OTP has been sent to your email” will be displayed at the top of the user’s screen. If there is an error in sending the OTP email, the notification indicating “Error sending OTP” will be displayed.

User retrieves the one-time-password in their HKU outlook mailbox and enter it into the OTP input field (Figure 22 and Figure 24). It is noted that the one-time-password can only be used within 5 minutes. After this time period, users need to click the resend button in order to obtain a new OTP which will replace the original one.

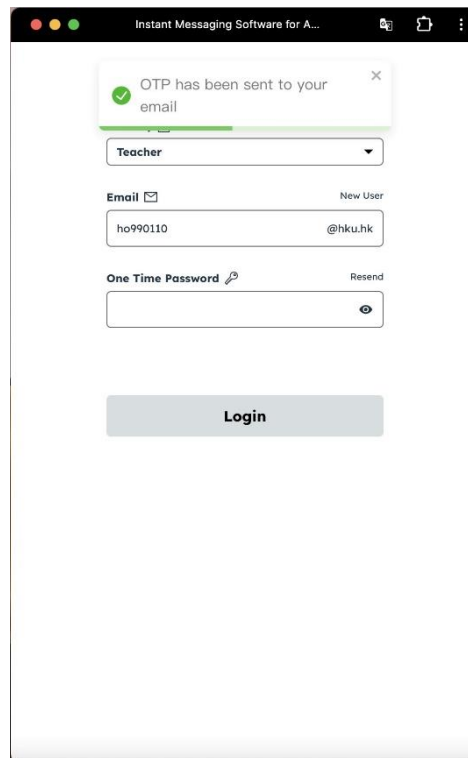


Figure 22: Requesting one-time password

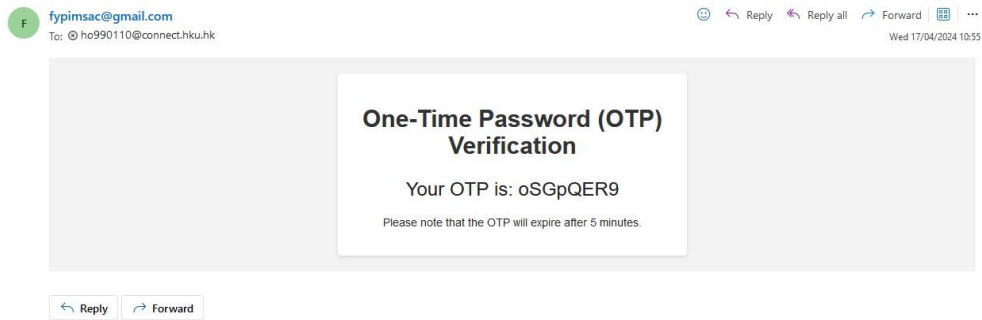


Figure 23: Retrieve an OTP generated by the system from the app user's mailbox

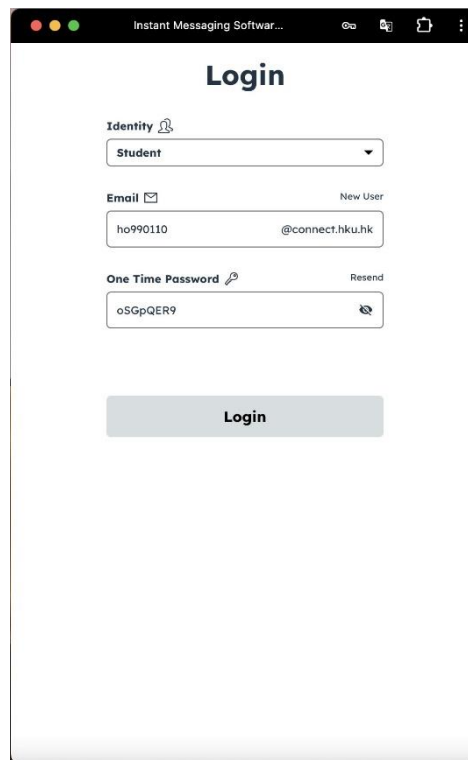


Figure 24: Providing the received OTP to the designated field

### 3.2 Registration

For new users, they can click the “New User” button to navigate to the registration page. The registration page for new users is similar to the login page, where they can input their email address to obtain one-time password. The key difference is that users have the option to switch back to the login page.

The screenshot shows a mobile application interface for registration. At the top, the window title is "Instant Messaging Softwar...". The main heading is "Registration". Below the heading, there are three input sections: "Identity" with a dropdown menu showing "Student", "Email" with the text "ho990110" and "@connect.hku.hk", and "One Time Password" with a "Resend" link. At the bottom, there are two buttons: "Verify One Time Password" and "Cancel".

Figure 25: Registration page OTP validation

After they receive the OTP email, users are allowed to click the “Verify One Time Password” button to validate the OTP (Figure 25). Following the successful verification, users are required to provide more detail personal information. The specific details required will be different based on the user’s role. For students, this includes providing their name (nickname, first name, last name), program (first major, second major, minor), year of study, profile image and date of birth (Figure 26). For teacher, they are only required to input the name, profile image and date of birth (Figure 27). Regarding the program input section, the students must at least type in the First major while the second major and minor field are optional. All other input fields are mandatory and they must be completed with information expect the image upload input field.

Figure 26: Required personal information for registration (for student)

Figure 27: Required personal information for registration (for teacher)



With the reference to the section for uploading a profile image, the format for uploading images has certain restrictions. Users can only upload images in “.jpg”, “.png” or “.svg” formats. Other file types will not appear in the file selection (Figure 28). After selecting an image, the name of the image will be displayed. Users can cancel their selection by clicking the “Remove” button. If the users do not choose an image, the default profile image will be applied. The profile image can be changed after the registration.

```
<input
  type="file"
  id="registrationProfileImage"
  className="registrationProfileImageSelector"
  accept="image/*"
  onChange={(e) => handleUploadedProfileImage(e)}
/>
```

*Figure 28: File selector for uploading profile image*

### **3.3 Home Page**

After users log in, the browser stores the JWT token on the browser cache. When the app is opened again, it will automatically redirect to the home page. As a result, users don't need to log in via OTP again. At the top of the home page, the user's nickname, role, and profile image are displayed. In the middle, there are two buttons, chat and quiz, which correspond to the two main features of our application (Figure 29).

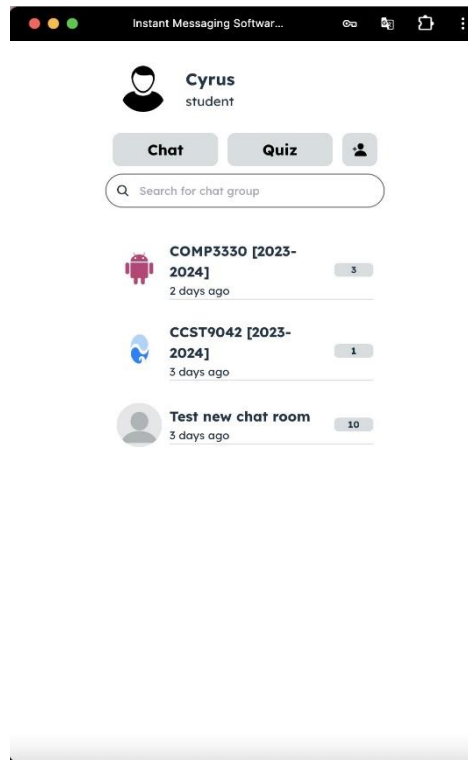


Figure 29: Layout of the Home page

Users are allowed to switch between the chat and quiz by clicking the respective buttons. In the chatroom section, a list of chatrooms the user has joined will be displayed. The chatroom's profile icon, name, number of unread messages and the timestamp of the last message will be shown (Figure 30). In the quiz section, a list of quiz rooms will be displayed only if a teacher is present in the corresponding chatroom (Figure 31). Similarly, the profile icon, name and the last quiz time will be displayed. If there is no scheduled quiz, "Undefined Quiz Date" will be shown.

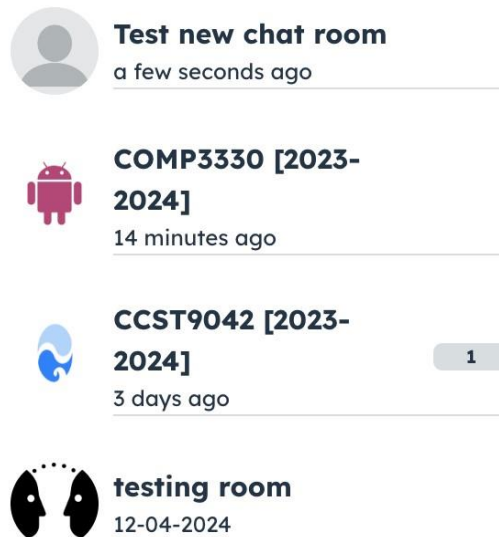


Figure 30: Example of home page chat group list

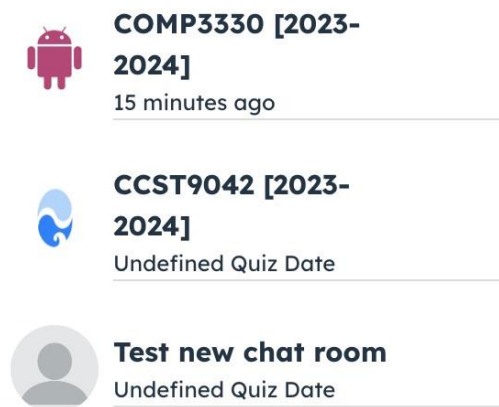


Figure 31: Example of Home page quiz group list

For the purpose of easy searching the chat groups and quiz groups, there is a search bar located below the “chat” and “quiz” button. Whenever the user types in characters, it automatically triggers the search algorithm. The search function looks for the exact sequence of characters that match the input and the searching algorithm is not case sensitive (Figure 32 and Figure 33).

```

//retrieve joined chat room list for chat system
const chatRoom = useGetChatRooms();
const allChatRooms = useMemo(() => {
  return (Array.isArray(chatRoom?.data) ? chatRoom?.data : [])
    .map((data_element) => ({
      chatRoomName: data_element.chatRoomName,
      chatRoomId: data_element.chatRoomId,
      chatRoomImage: data_element.chatRoomImage,
      chatRoomSentDate: data_element.sentDate,
      chatRoomCreatedAt: data_element.createdAt,
      chatRoomDateCombine:
        Number(
          moment(data_element.sentDate).format("YYYYMMDD") +
          moment(data_element.sentDate).format("HH:mm:ss").split(":").join("")
        ) ||
        Number(
          moment(data_element.createdAt).format("YYYYMMDD") +
          moment(data_element.createdAt).format("HH:mm:ss").split(":").join("")
        ),
      unseenMessageNumber: data_element.unseenMessageNumber,
    })))
    .sort(
      (compareObj1, compareObj2) =>
        compareObj2.chatRoomDateCombine - compareObj1.chatRoomDateCombine
    );
}, [chatRoom.data]);

```

*Figure 32: Sorted chat group list creation*

```

//retrieve joined quiz room list for quiz system
const quizRoom = useGetQuizRoom();
const allQuizRooms = useMemo(() => {
  return (Array.isArray(quizRoom?.data) ? quizRoom?.data : [])
    .map((data_element) => ({
      quizRoomName: data_element.chatRoomName,
      quizRoomId: data_element._id,
      quizRoomImage: data_element.chatRoomImage,
      quizRoomSentDate: data_element.lastQuizSentDate,
      quizRoomDateCombine:
        Number(
          moment(data_element.lastQuizSentDate).format("YYYYMMDD") +
          moment(data_element.lastQuizSentDate).format("HH:mm:ss").split(":").join("")
        ) || 0
    })))
    .sort(
      (compareObj1, compareObj2) =>
        compareObj2.quizRoomDateCombine - compareObj1.quizRoomDateCombine
    );
}, [quizRoom.data]);

```

*Figure 33: Sorted quiz group list creation*

### 3.4 Profile Page

User will be redirected to the profile page by clicking the header of the page. The profile page displays the details personal information of the user has filled in during the registration process. There are several input fields that allow users to edit the data.

These editable input fields have icons near their name to specify the editable input fields. For students, they are allowed to modify the nickname, profile image, second major and minor (Figure 34). Teachers, on the other hand, are allowed to modify the image and nickname (Figure 35). As the email address, the name of the user and ID are crucial for identifying users, they are not permitted to change this data.

The screenshot shows a web browser window with two tabs titled "Instant Messaging Software for A...". The active tab displays a "Profile" page. At the top, there are "Cancel" and "Save" buttons. The page is divided into two main sections. The left section, titled "Profile", features a profile picture placeholder with a "Change profile picture" link. Below this are input fields for "First name" (Ka Ho), "Last name" (Fung), "Nickname" (Cyrus), "User ID" (661fc03a281bb51b2d3099b8), "Email Address" (ho990110@connect.hku.hk), "First Major Program" (BEng Computer Science), "Second Major Program", and "Minor Program". The right section, titled "Change profile picture", contains a "Change profile picture" button and input fields for "First name" (Ka Ho), "Last name" (Fung), "Nickname" (Cyrus), "User ID" (661fc03a281bb51b2d3099b8), "Email Address" (ho990110@connect.hku.hk), "First Major Program" (BEng Computer Science), "Second Major Program", and "Minor Program". At the bottom right, there is a "Logout" button.

Figure 34: Profile page (for student)

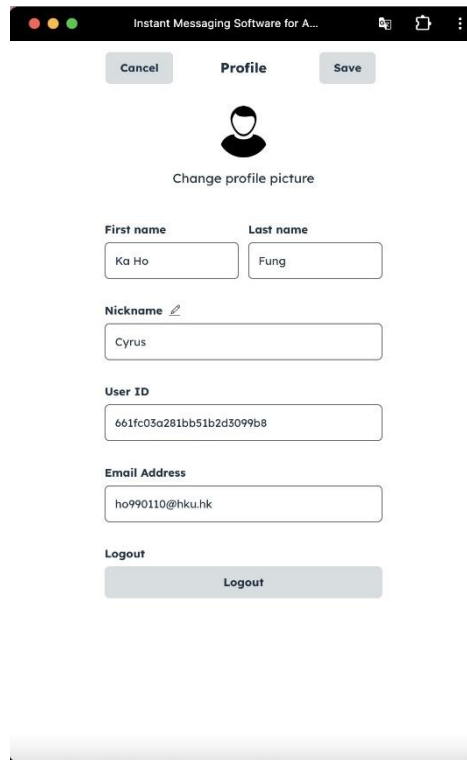
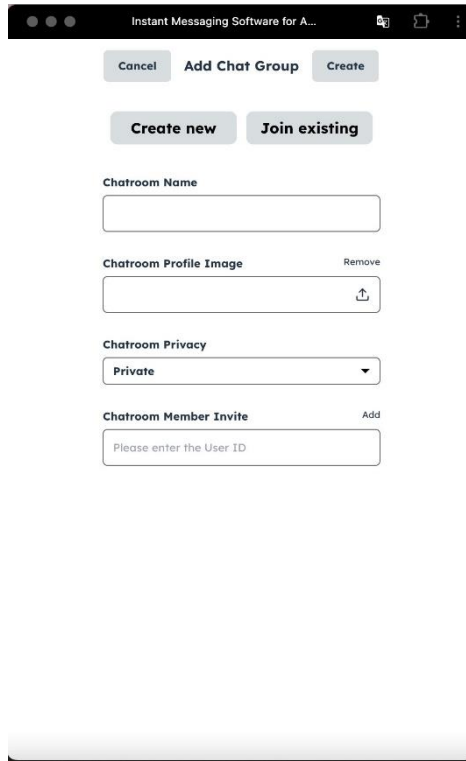


Figure 35: Profile page (for teacher)

### 3.5 Create/Join Chat Group

In order to join or create a new chat group, users can click on the “Add person” icon near the quiz button. Once clicked, they are redirected to new group page. They can switch between creating new group or joining existing group by clicking the “create new” or “join existing” buttons. For the creating a new group, users can fill in information such as the chatroom name, chatroom profile image, the chatroom privacy and the ID of the user being invited to the group (Figure 36). It is optional for the user to upload a profile image and invite more than one member by clicking the add button. However, it is necessary to invite at least one member.



*Figure 36: Add new group page layout*

Regarding the join existing group, a list of public groups is displayed that includes profile image, group name as well as the group ID (Figure 37). In case of duplicate group names, the ID can be used to distinguish the group. After selecting the desired group, users can click the join button at the top left corner to join the group.

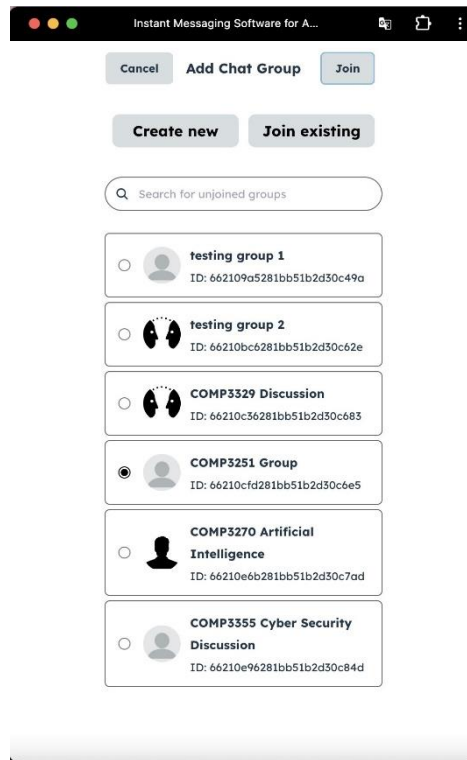


Figure 37: Example of joining an existing chat group

### 3.6 Real-time chatting

When users click on the one of the chatrooms, they will be directed to the chatroom. Chatroom message will then be fetched from the database where the past conversations are saved and the app displays the message starting from the point where the user last left off. There are three sections: the header displays the chatroom name and the icon, the message section shows the chatroom messages and the input box section enables users to type their message (Figure 38). Users can click on the header to navigate to the chatroom information page to view details information about the chatroom. In addition, the message section is scrollable so that users can scroll back and review past message history. To send new message, users are allowed to type characters in the text area or click the emoji icon to select the desired emoji. Apart from that, it is possible for the users to attach documents through a click on the



“+” icon.

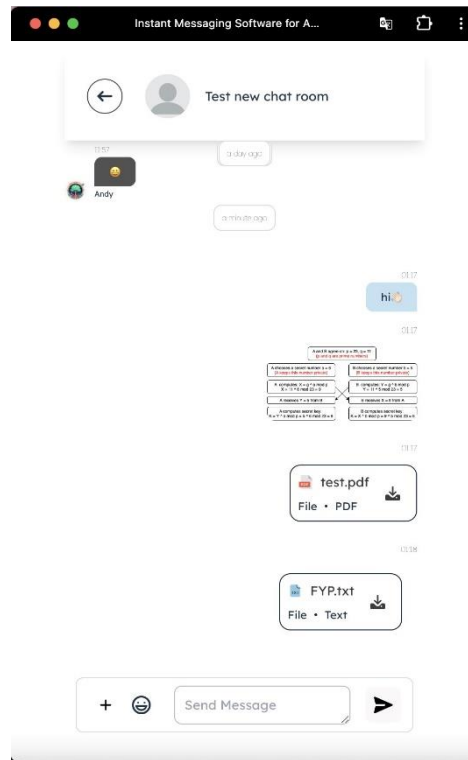


Figure 38: Example of sending text messages and sharing documents to the chat group

### 3.7 Chat Group Management

Regarding the chatroom information page, users are allowed to performance different actions based on their role. For the group administrators, they have certain privileges to manage the group. These privileges allow them to update the group profile image, the group name, privacy and invite new members (Figure 39). Moreover, they can manage the group member by assigning or withdrawing group administrators and by removing them from the group. In respect of non-administrative users, their access is limited to viewing the information of the chatroom (Figure 40). They can see details such as the chatroom name, profile image, and the details profile of other members.

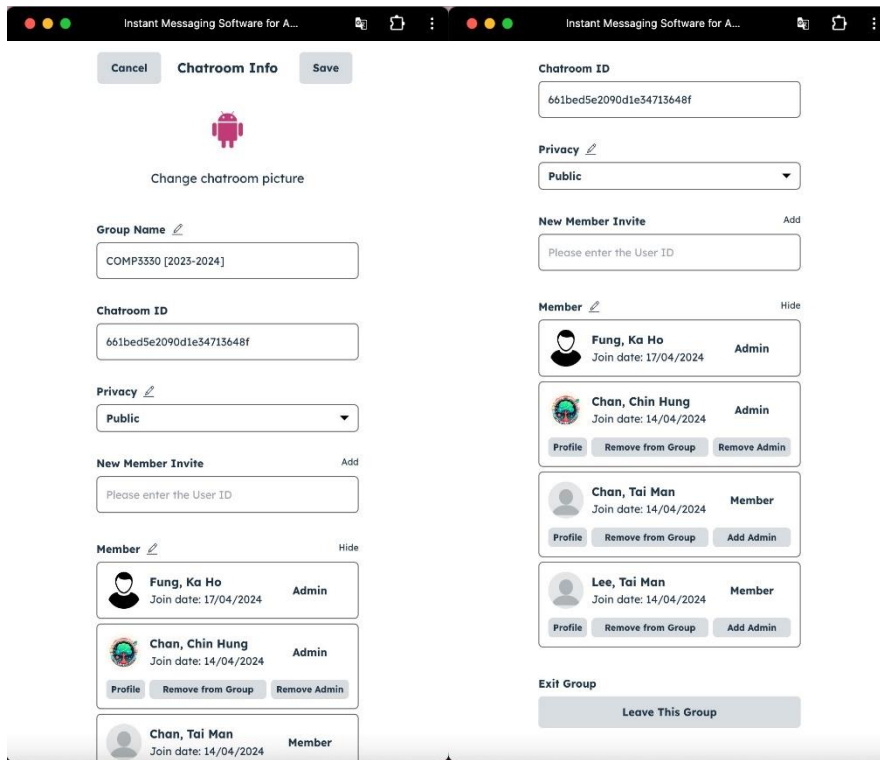


Figure 39: Chat group information page (for group administrator)

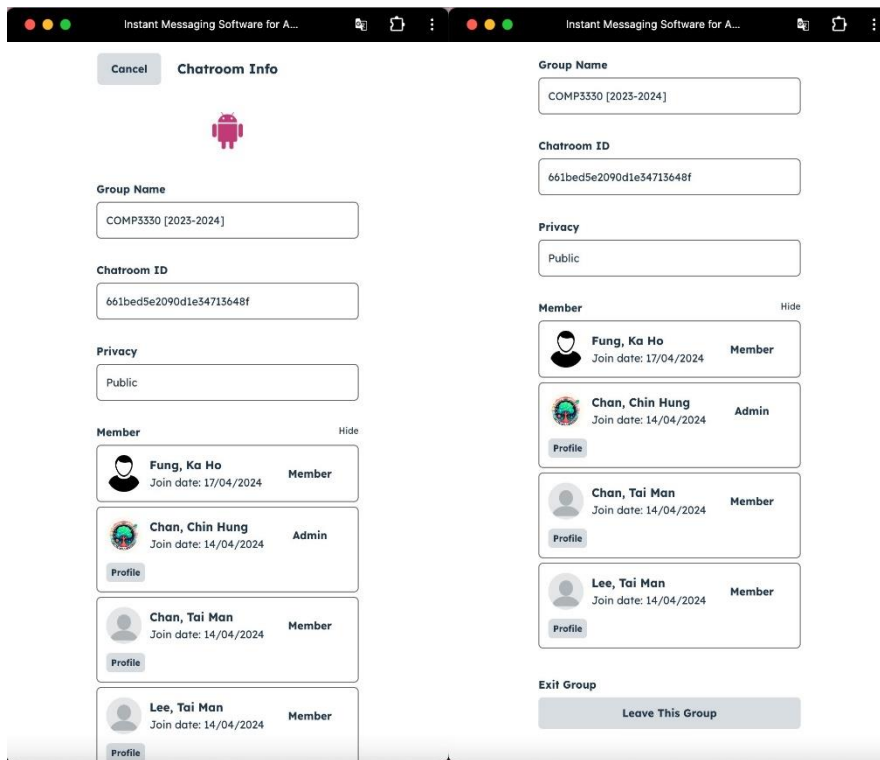


Figure 40: Chat group information page (for regular group member)

### 3.8 Quiz System Access

Similar to the chatroom sections, when users click on one of the quiz groups from the group list, they will be redirected to the quiz group associated with that group. A list of quizzes associated with the group will be displayed, arranged according with the start date of the quiz (Figure 41). Furthermore, the quiz topics are displayed to allow users to understand the content of the questions. They can have a quick overview of what subjects or themes the questions in each quiz will cover.

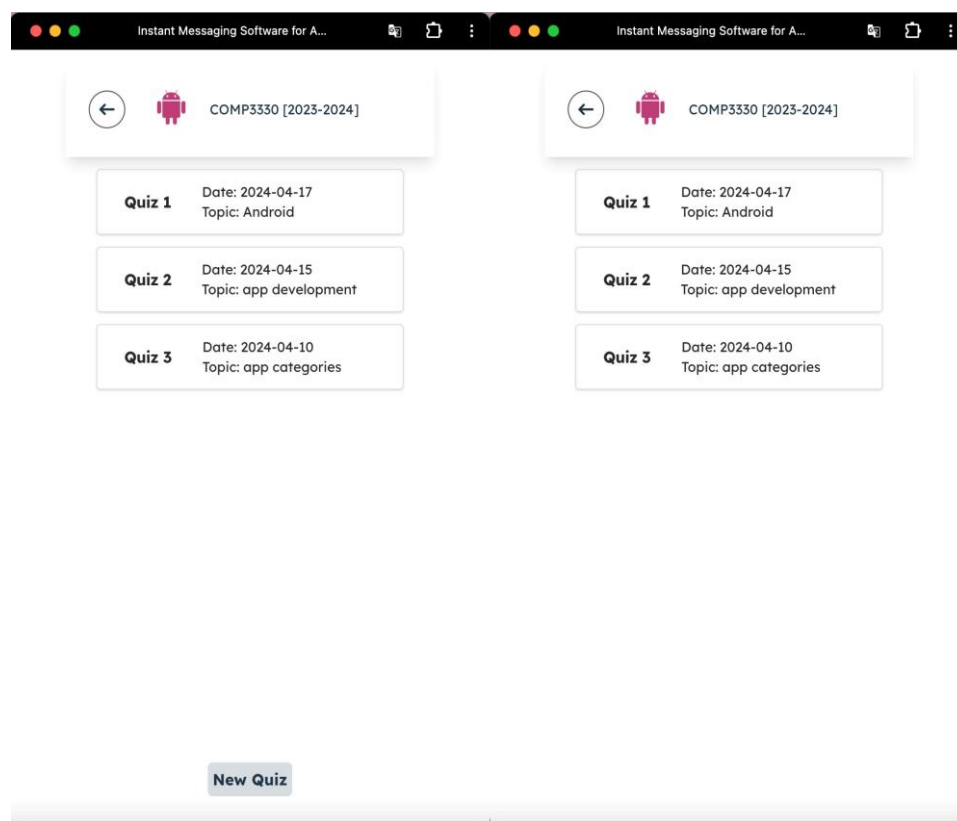


Figure 41: Quiz page layout for teachers (left) and students (right)

### 3.9 Quiz Creation

There is a “New Quiz” button at the bottom of the quiz page for teacher to create a new quiz. Teachers are allowed to design the quiz by setting the quiz topics, quiz question, quiz options and answer (Figure 42). Moreover, they can specify the end

dates for when the quiz will be available to the students and attach figure that may help student in answering the question. The format of the quiz is restricted to multiple choice questions as our focus is on encouraging student-teacher academic discussion. This type of quizzes can be delivered quickly to allow a greater number of questions during the lesson time. Additionally, it is a must for teachers to input at least two options, select one correct answer, and provide a question related to at least one topic.

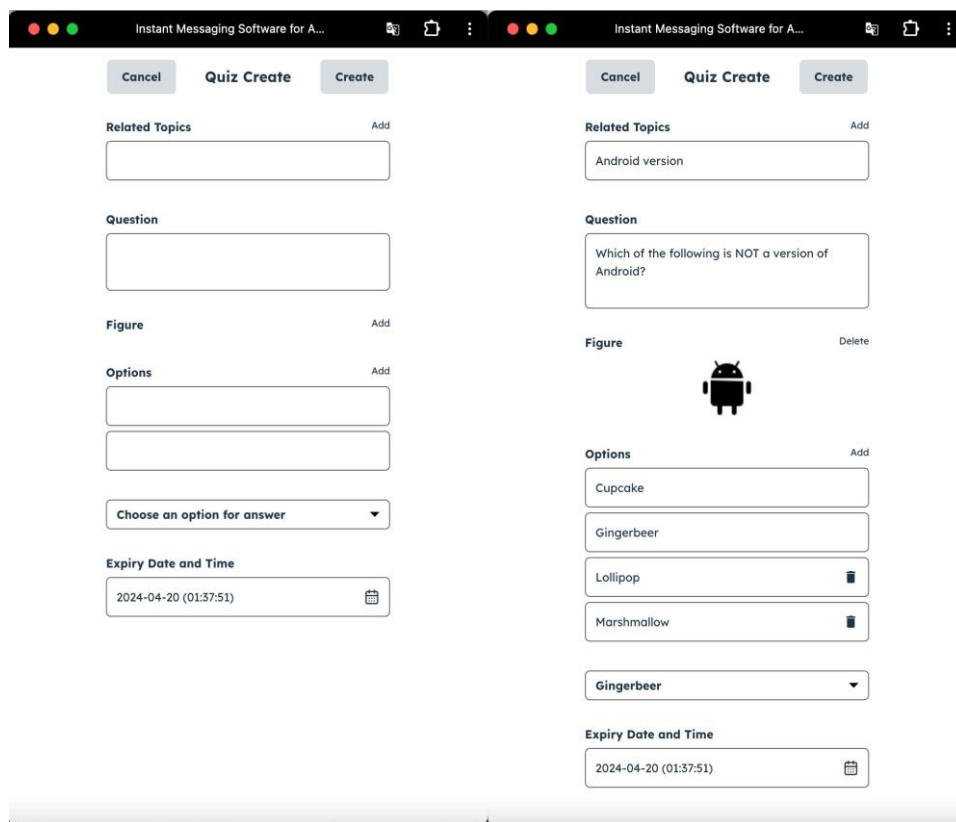
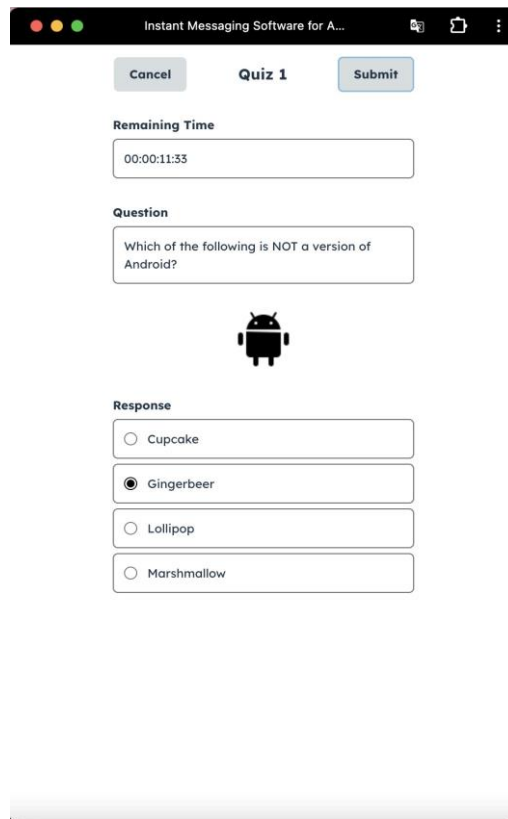


Figure 42: Example of quiz creation

### 3.10 Quiz Answering and Updating

Accessing the quiz question page is made possible with a simple click on the desired quiz. For the students, the quiz question, the figure as well as the option will be displayed. A countdown time will display the remaining time for students to complete the quiz at top. They are required to select one of the options and then click the submit

button to answer the question (Figure 43). If the time has just expired, they will be redirected to the quiz page. Moreover, students can review their choice before the deadline of the submission (Figure 44). Once the quiz deadline has passed, the model answer will be revealed (Figure 45).



*Figure 43: Example of quiz submission*

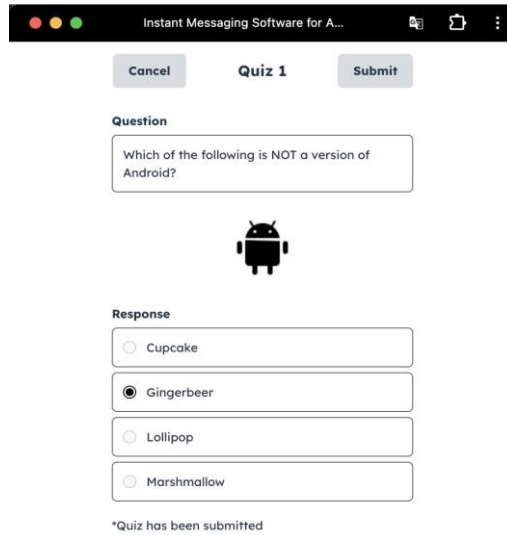


Figure 44: Notification after quiz submission

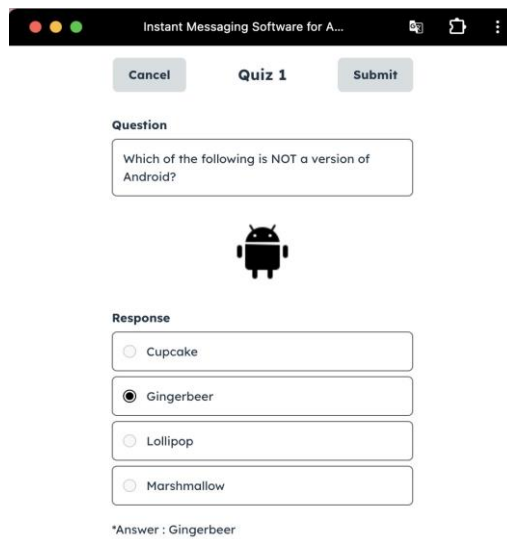


Figure 45: Quiz model answer reveal

Regarding teachers, they are allowed to modify the quiz information similar to the quiz creation process. However, students need to resubmit the quiz after the update of the quiz. The record of the previous submission will be cleared.

### 3.11 Quiz Information

Teacher can access the quiz information page by clicking the question information button at the bottom of the quiz question page. The question information section displays the details of statistics of the submissions. They can access this information to review student performance and gain insight into how well is doing overall (Figure 6).

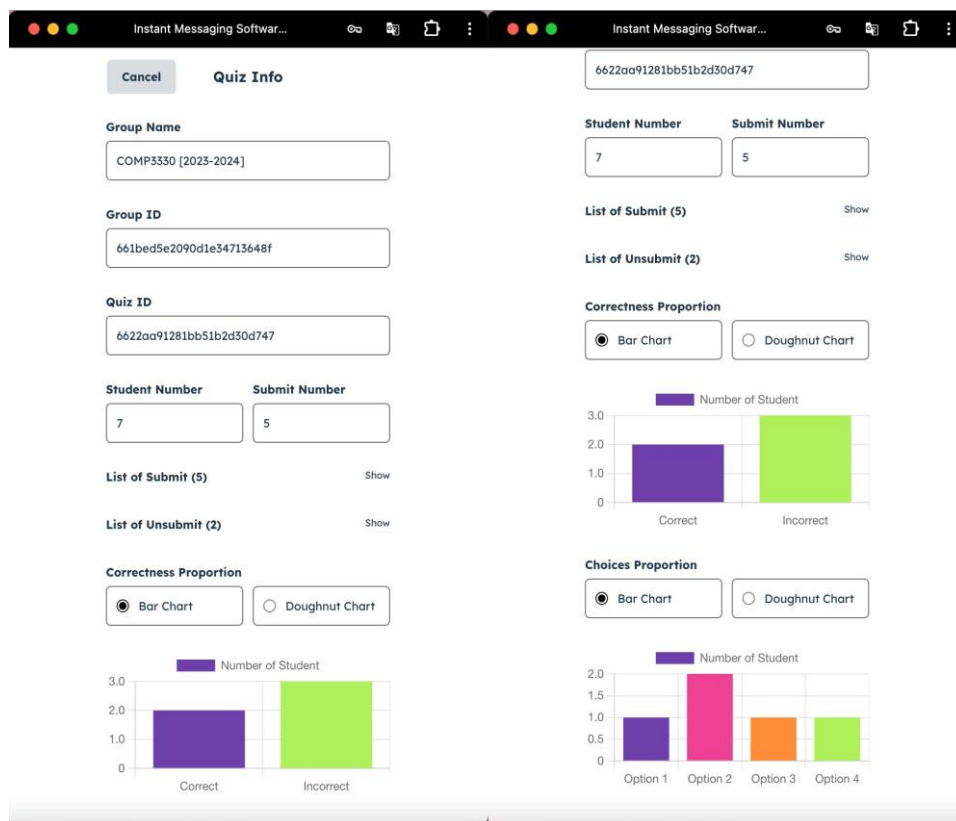


Figure 46: Layout for the quiz information page

The details of statistics information display the number of submitted student out of the total number of students. Teacher can review the details of each student submission by

clicking the “Show” button. A list of submitted student along with their submission date and their correctness will be shown (Figure 47).

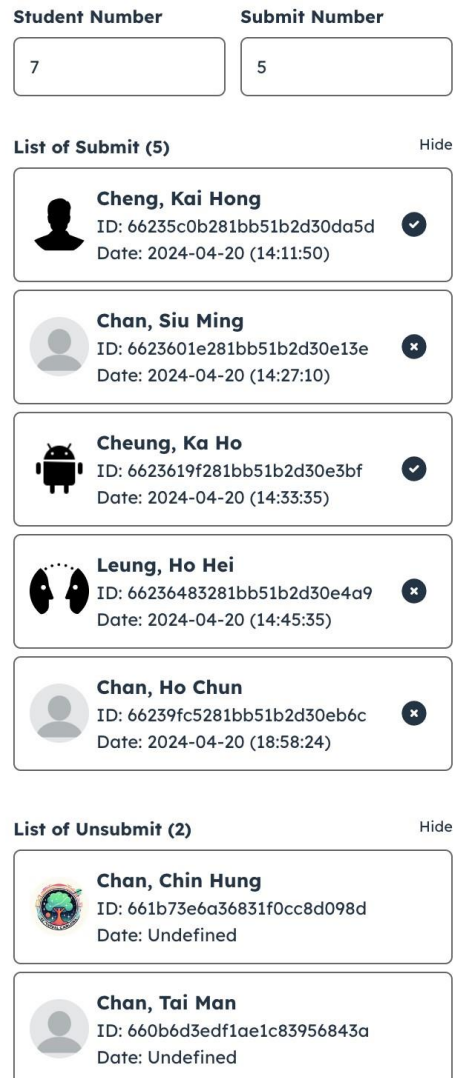


Figure 47: List of Submit and List of Unsubmit Example

Moreover, based on the student submission, a correctness proportion chat and the distribution of the choices will be displayed. The graph shows the percentage of correct answers submitted by students and a graph showing how the students’ answers are spread across the different options available in the quiz. Teachers are allowed to switch between the bar chat and the doughnut chart for better understanding of the



data (Figure 48). The colour of the options will be generated depending on the number of choices in order to make it easier to differentiate between them visually (Figure 49).

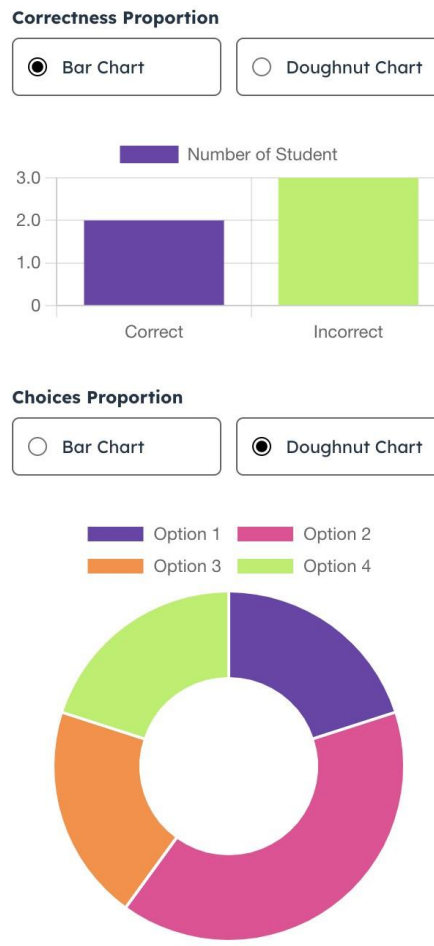


Figure 48: Charts for displaying the correctness proportion and choices distribution

```
const colorListGenerator = (numberOfColor, colorRangeLowerBound, colorRangeUpperBound) => {
  return Array(numberOfColor)
    .fill()
    .map(
      (colorElement, idx) =>
        d3.interpolateWarm(colorRangeLowerBound + idx * (colorRangeUpperBound - colorRangeLowerBound) / (numberOfColor - 1))
    );
}
```

Figure 49: Algorithm for generating a list of unique colors

### 3.12 Push Notification

The push notifications can inform users about new chat messages and encourage them to open the application. This allows important information to be promptly conveyed and keeps user connected and informed even the app is not currently open. The following diagrams display the notification message (Figure 50 and Figure 51).

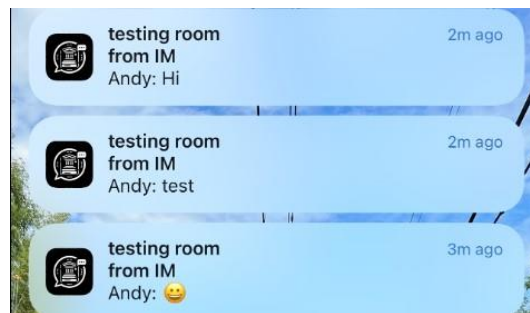


Figure 50: Notification message display on mobile phone



Figure 51: Notification message display on computer

#### **4 Difficulties and Limitations**

In our software development process, one of the most difficult challenges revolves around managing user roles. As our application is required to dynamically renders different content based on user identity, conditional rendering based on that is needed. The permission level control and the conditional rendering is tedious to implement and debug.

Moreover, the logic of the quiz section is somewhat complex, as duplicated submissions is not allowed and the correct answer should be revealed on time. A count down time need to keep track of the remaining time of the quiz. Various cases need to be handled correctly to ensure the quiz system does not contain any bugs. The current application does not provide a very user-friendly interface for desktop users as it operates as a standalone display. Further work can be done to enhance the UI and experience by implementing a new responsive web UI.

Furthermore, the app may suffer from security issue as end-to-end encryption was not adopted to safeguard data in both database and cloud storage. Users can directly view the message in the database or in S3 bucket.

## **5 Conclusion**

This paper aims to implement an Instant Messaging Software for Academic Consultation. It offers features such as real-time chatting, quizzes creating and responding, quiz question statistic information.

A PWA approach was adopted to build a user-friendly and cross platform application. React with Vite will be used for the frontend website development and manifest and service worker files will be added to make it a valid PWA. Express will be utilized to build the backend which will be hosted on an AWS EC2 server while MongoDB will be utilized to store data, which will be host on Mongo Atlas. The CloudFront and S3 bucket is responsible for retrieving and storing file data.

The project is finished according to the schedule. In the future, the software can explore different options and improve the web UI. Security issue regarding the data encryption could be explored like end-to-end encryption.

## 6 Reference

- [1] L. Ceci and A. 29, "Mobile messaging users worldwide 2025," Statista, <https://www.statista.com/statistics/483255/number-of-mobile-messaging-users-worldwide/#:~:text=Global%20number%20of%20mobile%20messaging%20users%202019%2D2025&text=In%202021%2C%20an%20estimate%20of,popular%20mobile%20messenger%20app%20worldwide> (accessed Sep. 23, 2023).
- [2] L. Ceci, "Leading mobile apps worldwide in 2022, by downloads" Statista, <https://www.statista.com/statistics/1285960/top-downloaded-mobile-apps-worldwide/> (accessed Mar. 10, 2024).
- [3]-S. So, 'Mobile instant messaging support for teaching and learning in higher education', *The Internet and Higher Education*, vol. 31, pp. 32–42, 2016.  
[doi:10.1016/j.iheduc.2016.06.001](https://doi.org/10.1016/j.iheduc.2016.06.001) (accessed Sep. 30, 2023)-
- [4]-D. Gachago, S. Strydom, P. Hanekom, S. Simons, and S. Walters, 'Crossing boundaries : lectures' perspectives on the use of WhatsApp to support teaching and learning in higher education', *Progressio*, vol. 37, no. 1, pp. 172–187, 2015.  
[doi:10.10520/EJC180393](https://doi.org/10.10520/EJC180393). (accessed Sep. 30, 2023)
- [5] D. Fortunato and J. Bernardino, "Progressive web apps: An alternative to the native mobile Apps," 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), Caceres, Spain, 2018, pp. 1- 6,  
[doi:10.23919/CISTI.2018.8399228](https://doi.org/10.23919/CISTI.2018.8399228). (accessed Sep. 30, 2023)
- [6] T. A. Majchrzak, A. Biørn-Hansen, and T.-M. Grønli. (2018) "Progressive Web Apps: the Definite Approach to Cross-Platform Development?" *Proc. 51st Hawaii Int.*

Conf. Syst. Sci. Available:

<https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1716&context=hicss-51> (accessed Sep. 30, 2023)

[7] L. Vu, “PWA Service Worker for Dummies” SimiCart Blog,

<https://www.simicart.com/blog/pwa-service-worker/> (accessed Mar. 20, 2024).

[8] React. Available: <https://react.dev/> (accessed Sep. 17, 2023).

[9] “Virtual dom and Internals,” React, <https://legacy.reactjs.org/docs/faq-internals.html> (accessed Sep. 17, 2023)

[10] Vite, <https://vitejs.dev/guide/> (accessed Sep. 24, 2023).

[11] Node.js. Available: <https://nodejs.org/en> (accessed Sep. 11, 2023).

[12] “NPM,” npm, <https://www.npmjs.com/> (accessed Sep. 11, 2023).

[13] Express. Available: <https://expressjs.com/> (accessed Sep. 11, 2023).

[14] MozDevNet, “Server-sent events - web apis: MDN,” MDN Web Docs,

[https://developer.mozilla.org/en-US/docs/Web/API/Server-sent\\_events](https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events) (accessed Apr. 25, 2024).

[15] “JSON and Bson,” MongoDB. Available: <https://www.mongodb.com/json-and-bson> (accessed Sep. 11, 2023).

[16] Narendra, “Cheatsheet To Document Oriented Database: Detailed breakdown,”

RedSwitches, <https://www.redswitches.com/blog/document-oriented-database/> (accessed Mar. 20, 2024).

[17] “MongoDB atlas database: Multi-cloud database service,” MongoDB,

<https://www.mongodb.com/atlas/database> (accessed Sep. 24, 2023).

[18] What is a CDN? - content delivery network explained - AWS,

<https://aws.amazon.com/what-is/cdn/> (accessed Mar. 15, 2024).

[19] “JSON web tokens introduction,” JSON Web Token Introduction,

<https://jwt.io/introduction> (accessed Dec. 25, 2024).

[20] MongoDB, Inc. “Change Streams.” MongoDB Documentation. [Online].

Available: <https://www.mongodb.com/docs/manual/changeStreams/> (accessed Mar. 15, 2024)

[21] MongoDB, Inc. “Role-based Permission” MongoDB Documentation. [Online].

Available: <https://www.mongodb.com/docs/atlas/app-services/rules/roles/> (accessed Mar. 16, 2024)

## 7 Appendix

Item	Date	Task
1	September 2023	<ul style="list-style-type: none"> <li>● Detail Project plan</li> <li>● Project webpage making</li> </ul>
2	October 2023	<ul style="list-style-type: none"> <li>● Graphical UI design and related research</li> <li>● Database design and related research</li> </ul>
3	November 2023	<ul style="list-style-type: none"> <li>● Graphical UI implementation</li> <li>● Database setup</li> <li>● System design and related search</li> </ul>
4	December 2023	<ul style="list-style-type: none"> <li>● Real-time chatting feature implementation (for both server side and client side)</li> <li>● Interactive quizzes creating and answering system implementation</li> <li>● Detailed interim report writing</li> <li>● First presentation preparation</li> </ul>
5	January 2024	
6	February 2024	
7	March 2024	<ul style="list-style-type: none"> <li>● System testing and debugging</li> <li>● System Optimization</li> </ul>
8	April 2024	<ul style="list-style-type: none"> <li>● Final report writing</li> <li>● Final presentation preparation</li> </ul>

Table 1. Project Schedule