



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

**“Instant Messaging Software for Academic Consultation”**

Fung Ka Ho (3035902635)  
Chan Chin Hung (3035780879)  
Supervised by Dr. T.W. Chim

## **Abstract**

These days, in-person interactions or email exchanges are the primary ways for students to consult their teachers. Since students have to create an email with a serious tone and style or return to school, these are not a handy way for them to consult their teachers.

Moreover, the one-way teaching approach has been widely adopted by the teachers nowadays. Insufficient interaction between teachers and students during lessons has become a common phenomenon. This is one of the reasons that reduces student's learning efficiency.

In view of this, this project aims to develop an instant messaging (IM) software so that students can communicate with their teachers in a more convenient manner. Moreover, in order to facilitate discussions during lessons, the quiz creating and answering system will be incorporated within the IM software.

As of the time of writing this interim report, the database design and setup, as well as most of the graphical user interface (GUI) design and implementation has been completed. Currently, all the functionalities including in both the client side and server side are being conducted.

## **Acknowledgment**

We would like to take this opportunity to express our deep and sincere gratitude to Dr. Chim, Tat Wing for providing us with lots of valuable guidance throughout this final year project.

# Table of Contents

<b>Abstract</b> .....	<b>I</b>
<b>Acknowledgment</b> .....	<b>II</b>
<b>Table of Contents</b> .....	<b>III</b>
<b>List of Figures</b> .....	<b>IV</b>
<b>List of Tables</b> .....	<b>V</b>
<b>Abbreviations</b> .....	<b>VI</b>
<b>1. Introduction</b> .....	<b>1</b>
1.1 Background.....	1
1.2 Motivation.....	2
1.3 Existing Instant Messaging Software and Online Learning Platform .....	2
1.4 Overview and Objectives .....	2
1.5 Report Outline.....	3
<b>2. Methodology</b> .....	<b>4</b>
2.1 Implementation .....	4
2.1.1 Frontend .....	4
2.1.2 Backend.....	5
2.1.3 Database .....	5
2.1.4 Deployment.....	6
2.2 Testing.....	6
2.2.1 Unit Testing .....	6
2.2.2 API automated Testing .....	7
2.2.3 Integration Testing .....	7
<b>3. Result and Discussion</b> .....	<b>8</b>
3.1 Current Results.....	8
3.1.1 System Architecture.....	8
3.1.2 Database Design.....	9
3.1.3 Authentication.....	10
3.1.4 Graphical User Interface Design and Implementation.....	12
3.1.4.1 Login Page Graphical User Interface.....	12
3.1.4.2 Registration Page Graphical User Interface.....	14
3.1.4.3 Main Page Graphical User Interface.....	16
3.1.4.4 Profile Page Graphical User Interface .....	16
3.1.4.5 Quiz Create Page Graphical User Interface.....	17
3.1.4.6 Chat Page Graphical User Interface.....	18
3.2 Limitations and Difficulties .....	19
<b>4. Future Work</b> .....	<b>20</b>
<b>5. Conclusion</b> .....	<b>21</b>
<b>Reference</b> .....	<b>22</b>

## List of Figures

Figure 1: Number of mobile phone messaging app users worldwide from 2019 to 2025 [P.1]

Figure 2: System Architecture Design [P.8]

Figure 3: Database data model diagram for this project [P.10]

Figure 4: Documents in User, ChatRoom, OTP collections (based on data model diagram) [P.10]

Figure 5: JWT encoding and decoding structure [P.11]

Figure 6: Email Verification and JWT Authentication Flow [P.12]

Figure 7: Login Page (for Student) [P.13]

Figure 8: Login Page (for Teacher) [P.13]

Figure 9: Login Page after pressing the “Get One Time Password” button [P.14]

Figure 10: Registration Page after entering the OTP [P.14]

Figure 11: Registration Page for student (after verifying OTP) [P.15]

Figure 12: Main Page [P.16]

Figure 13: Profile Page [P.17]

Figure 14: Quiz Create Page [P.18]

Figure 15: Chat Page [P.18]

## List of Tables

Table 1: Schedule [P.20]

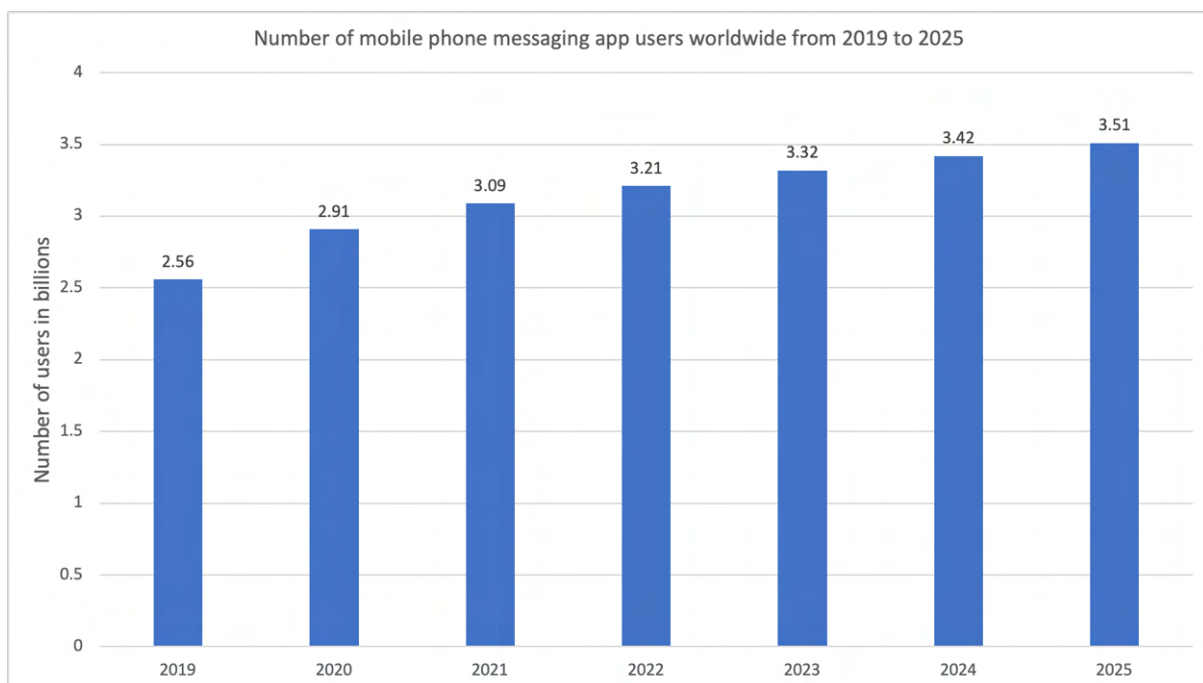
## Abbreviations

Instant Messaging	(IM)
Graphical user interface	(GUI)
Amazon Web Services	(AWS)
Binary Javascript Object Notation	(BSON)
Create, Read, Update, Delete	(CRUD)
Document Object Model	(DOM)
Amazon Elastic Compute Cloud	(EC2)
Entity Relationship	(ER)
Graph Query Language	(GraphQL)
Google Remote Procedure Call	(gRPC)
Hypertext Transfer Protocol	(HTTP)
Not Only Structured Query Language	(NoSQL)
Progressive Web App	(PWA)
Representational State Transfer	(REST)
JavaScript Object Notation	(JSON)
Transmission Control Protocol	(TCP)
Uniform Resource Locator	(URL)
One-Time Password	(OTP)

# 1. Introduction

## 1.1 Background

Instant messaging (IM) software refers to a category of online applications that enable users to communicate textually in real-time. The advent of IM software has revolutionized the way we communicate in our personal lives, providing a seamless and efficient way to connect with others. IM software has become deeply ingrained in our daily routine from casual conversations to sharing amusing anecdotes, this gives rise to the popularity of IM software. According to the statistics given by L. Ceci in 2023, the number of IM software users worldwide in 2019 was approximately 2.56 billion (see Figure 1) [1]. This number has increased over the years. By 2025, the number is expected to reach 3.51 billion (see Figure 1) [1].



*Figure 1: Number of mobile phone messaging app users worldwide from 2019 to 2025 [1]*  
*The horizontal axis of the figure represents the year while the vertical axis represents the number of mobile phone messaging app users worldwide in billions.*

The popularity of instant messaging (IM) software can be attributed to its convenience and user-friendly nature. With just IM software, the app users can conduct real-time chat less formally without any time and geographical constraints. Moreover, these apps offer users various ways to express their thoughts and emotions, such as through text, and multimedia



elements like emojis, voice messages, or photos. Such features greatly enhance the efficiency of communication.

## **1.2 Motivation**

Although using IM software brings convenience and is popular in daily conversation, it is not a common tool used by students to consult their teachers. At present, students mainly consult their teachers via in-person interactions or email. Compared to using IM software, both approaches are relatively inconvenient and time-consuming as they require students to spend time going back to school or draft an email with a serious tone and formal formatting to ask a question rather than asking the question more directly and immediately.

Additionally, the majority of teaching methods employed by educators today are one-way, with the teacher serving as the sole source of knowledge, while the students passively receive and absorb information [2]. This leads to a lack of interaction between teachers and students during lessons and hinders effective learning. Research conducted in 2021 has shown that increased interaction between teachers and students can enhance both learning motivation and efficiency [3]. Therefore, there is a need to establish effective teacher-student interaction during lessons.

## **1.3 Existing Instant Messaging Software and Online Learning Platform**

Although students and teachers currently have the option to conduct communication through existing IM software like WhatsApp, Telegram, or Skype and engage in quiz creation and response through game-based learning platforms like Kahoot, it is not a seamless and efficient approach. This is due to the need to install multiple applications on their devices to access all the necessary features, making it inconvenient for conducting these academic activities.

## **1.4 Overview and Objectives**

Taking the factors shown in section 1.2 and section 1.3 into consideration, we would like to develop an IM software specifically for students to consult their teachers. Furthermore, we would like to incorporate a quiz creation and answering system within the IM software to facilitate teacher-student interaction during lessons. Here are the project objectives:

- To integrate both the live chat and the quiz creation and response features into one app
- To give teachers a proper graphical interface so they may create, modify, and distribute interactive assessments in class to promote communication between students and teachers
- To serve as a consultation platform for students' academic inquiry
- To enable academic conversations between instructors and students in groups or in private
- To enable message archiving so that users of the app can review previous conversations

### **1.5 Report Outline**

This report consists of three remaining sections. Section 2 will delve into the methodology, explaining the software and tools utilized in this project. Section 3 will present the results and discussion, encompassing the progress made thus far and the challenges encountered. Lastly, section 4 will focus on the conclusion.

## **2. Methodology**

### **2.1 Implementation**

The implementation of the instant messaging software involves four main stages: Frontend (Section 2.1.1), Backend (Section 2.1.2), Database (Section 2.1.3), and Deployment (Section 2.1.4). These are described in more detail below.

#### **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 [4]. Many existing PWA, such as Uber, Pinterest, and Starbucks, provide a responsive and intuitive UI that is similar to native applications. Therefore, we will build a PWA.

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 [5]. 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. One important feature is the push notification which allows the user to receive messages even if a browser is not open [4, 5].

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 [6-7]. 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 will be used 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 [6,8]. 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 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 [9]. In contrast with other server frameworks, its comprehensive and reliable package manager npm provides over two million packages with detailed documentation [10]. 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 frameworks will be used 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 [11]. These modules can be reused in different API endpoints and thereby speed up the backend development process.

The WebSocket protocol will also be used for real-time communication. Unlike normal HTTP requests, it reduces the number of TCP handshakes and allows bi-directional communication, providing high efficiency and low latency connection [12], 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. BSON has extra features that allow effective indexing and querying of data. [13-14] Therefore, it is a suitable database for a Node.js server. 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.

## 2.1.4 Deployment

The server of the project will be hosted on Amazon Elastic Compute Cloud (Amazon EC2). Amazon EC2 provides a lot of types of virtual machines 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 is 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 [16]. 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.

## 2.2 Testing

The testing of the software involves three main methods: unit testing (Section 2.2.1), API automated testing (Section 2.2.2) and integration testing (Section 2.2.3). These are explained in more detail below.

### 2.2.1 Unit Testing

Vitest will be used as the testing framework because it is vite-native. Vitest is built on top of the Jest which is the most popular JavaScript testing framework. Jest provides many testing features such as snapshot testing, mocking and coverage reports and these API and ecosystem libraries can be accessed by Vitest [17]. In addition, it is relatively easy to configure and build through a single vite.config.js file in contrast with Jest which requires an extra jest.config.js file. With Vite, Jest has many more limitations. For example, it is not injected into all the modules and it may not work properly for coverage tests [18].

Vitest will be used by us to test the low-level logic and functions as well as the UI content. The react testing library, which is a testing utility, will be used together with Vitest. As it allows direct working with the actual DOM nodes. In this way, Code can be written to simulate the user interaction with the DOM and verify the real content that appears on the

website[19]. We will test each page by rendering its react DOM and then verify the content. For instance, the text content, the image source URL and the buttons will be checked.

### **2.2.2 API automated Testing**

Server security is crucial as it holds a lot of business logic and data. Even though the website can restrict the user's input and limit access to the endpoint, there are still other ways to directly communicate with the endpoint. This poses potential dangers to the API. Therefore, the API endpoint security will be tested.

Postman is a well-known tool for backend developers and software testers to test the API. It provides a comprehensive set of tools that cover various aspects such as testing, design, mock server, and API detection. It also supports a wide range of protocols such as gRPC, REST, GraphQL, etc [20]. We focus on testing features and writing automation scripts to test all the HTTP API and WebSocket API.

### **2.2.3 Integration Testing**

Although the Vitest has the browser mode feature which allows testing in a native browser, it is still in the early stages of development and there are potentially a lot of bugs and issues in the code [8]. Another tool called Puppeteer is a node js library that can be used to control the Chromium browser. It offers lots of API which allows us to perform actions such as clicking buttons, key press on the keyboard, navigating to other pages, and helping us to test the real scenario of the software. Online and offline behavior and the actual user interaction with the software will be tested.

### 3. Result and Discussion

This section will summarize the project's current results as well as the limitations and difficulties encountered. The project's current results will be discussed in part 3.1, while the limitations and difficulties encountered will be discussed in section 3.2.

#### 3.1 Current Results

##### 3.1.1 System Architecture

The whole system architecture describes the interaction between user devices and backend server including detailed data communication methods, notification strategies, offline caching mechanisms and database connections (Figure 2).

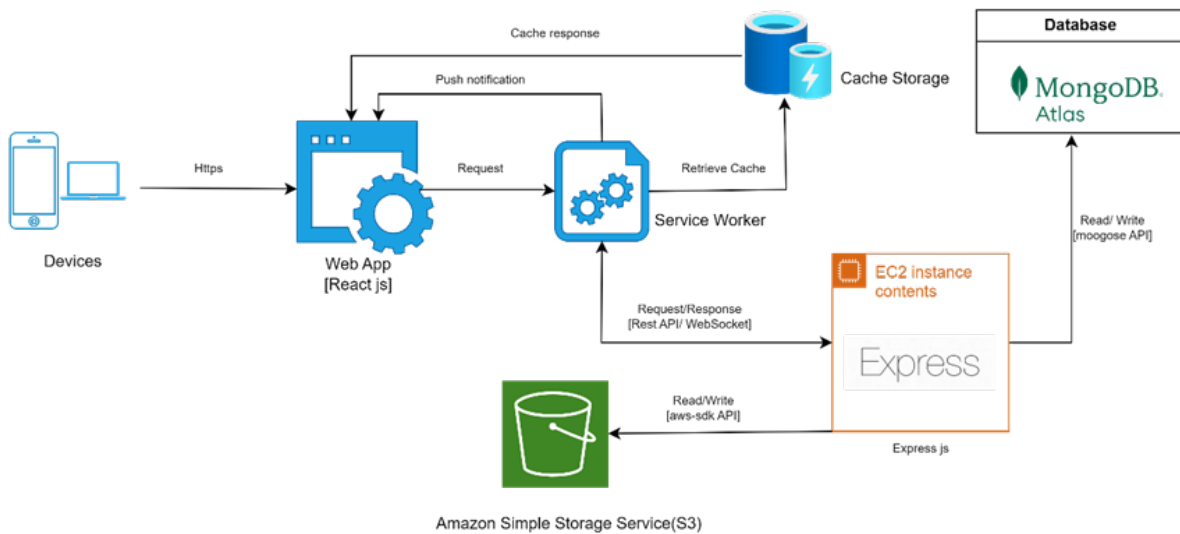


Figure 2: System Architecture Design

As for the frontend, clients can visit the official website and add the app to the 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 operations such as data retrieval, data updates and deletions. The devices also establish a WebSocket connection with other users for the purpose of real-time communication. For offline users, the message will be

displayed on the devices with the help of the notification API. 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 such as, image, audio, videos will be temporarily stored in the AWS S3 bucket via aws-sdk API.

### **3.1.2 Database Design**

There are three main collections in this database, which includes the User collection, the ChatRoom collection, as well as the OTP collection. PK and FK indicate the field is the primary key and foreign key respectively.

In the User collection, it has the fields nickname, firstname, lastname, role, program, age, profile\_image, email, year\_of\_study, create\_at, and chatRoomMember, where chatRoomMember field is an array of embedded documents and each embedded document has fields join\_date, left\_date, isAdmin, and chatRoomId. The chatRoomId field in each embedded document refers to a unique document in the ChatRoom collection.

In the ChatRoom collection, each document has the fields chatRoomId, chatRoomImage, chatRoomName, message, and quiz, where message and quiz are two different arrays of embedded documents. For each embedded document in the message array, it has fields messageId, messageFrom, send\_date, message\_type, messageData, userId, and messageStatus, where messageStatus is an array of embedded documents. Each embedded document in the messageStatus array has fields userId, read\_date, and deliver\_date. For each embedded document in the quiz array, it has fields quizId, quizQuestion, quizModelAnswer, quizType, quizFrom, sent\_date, quizImage, and studentQuizResponse, where studentQuizResponse is an array of embedded documents. Each embedded document in the studentQuizResponse array has fields userId, quizAnswer, and quizResponse\_date.

In the OTP collection, it has fields email, otp and createdAt. This collection will only be used in the login process or account registration process.



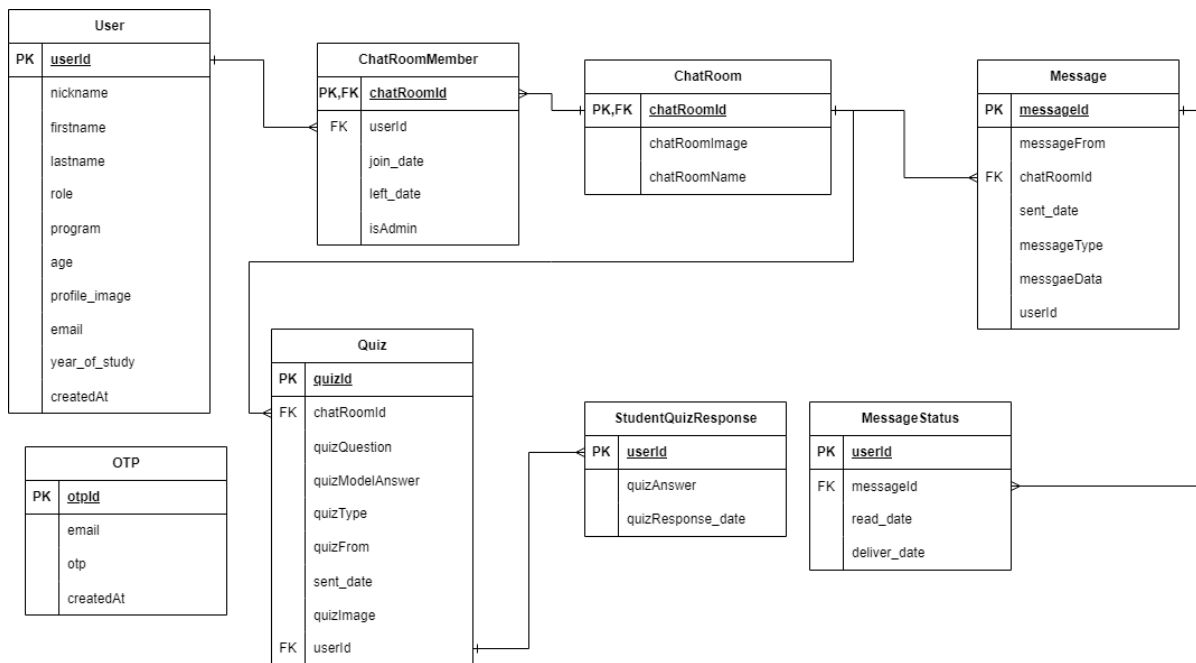


Figure 3: Database data model diagram for this project

User	ChatRoom	OTP
<pre>{   "userId": ObjectID,   "nickname": String,   "firstname": String,   "lastname": String   "role": String,   "email": String,   "program": String,   "age": Number,   "profile_image": String,   "year_of_study": Number,   "createdAt": Date,   "chatRoomMember": [     {       "chatRoomId": ObjectID,       "join_date": Date,       "left_date": Date,       "isAdmin": Boolean     }   ] }</pre>	<pre>{   "chatRoomId": ObjectID,   "chatRoomImage": String,   "chatRoomName": String,   "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,       "quizModelAnswer": String,       "quizType": String,       "quizFrom": ObjectID,       "sent_date": Date,       "sent_image": String,       "quizImage": String,       "studentQuizResponse": [         {           "userId": ObjectID,           "quizAnswer": String,           "quizResponse_date": Date         }       ]     }   ] }</pre>	<pre>{   "email": String,   "otp": String,   "createdAt": Date }</pre>

Figure 4: Documents in User, ChatRoom, OTP collections (based on data model diagram)

### 3.1.3 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 creation 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 sessions across multiple servers. Therefore, token-based authentication will be adopted.

The JSON Web Tokens (JWT) authentication method is commonly used for authentication in modern web technology. The JWT contains three parts which define the algorithm for encryption, the payload data, and the signature (Figure 5). The payload data can store frequently used data to minimize the number of data retrievals while the signature verifies that the integrity of the message is maintained.

The image shows a web interface for encoding and decoding JWTs. On the left, under the heading 'Encoded', there is a text area containing a long alphanumeric string: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MzkwMjQyLm51KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c`. On the right, under the heading 'Decoded', there is a structured view of the token's components. The 'HEADER: ALGORITHM & TOKEN TYPE' section shows a JSON object: `{ "alg": "HS256", "typ": "JWT" }`. The 'PAYLOAD: DATA' section shows a JSON object: `{ "sub": "1234567890", "name": "John Doe", "iat": 1516239822 }`. The 'VERIFY SIGNATURE' section shows the HMACSHA256 algorithm and a text input field for the secret key, with a checkbox for 'secret base64 encoded'.

Figure 5: JWT encoding and decoding structure

The authentication flow is illustrated below (Figure 6).

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 requests with a JWT header.

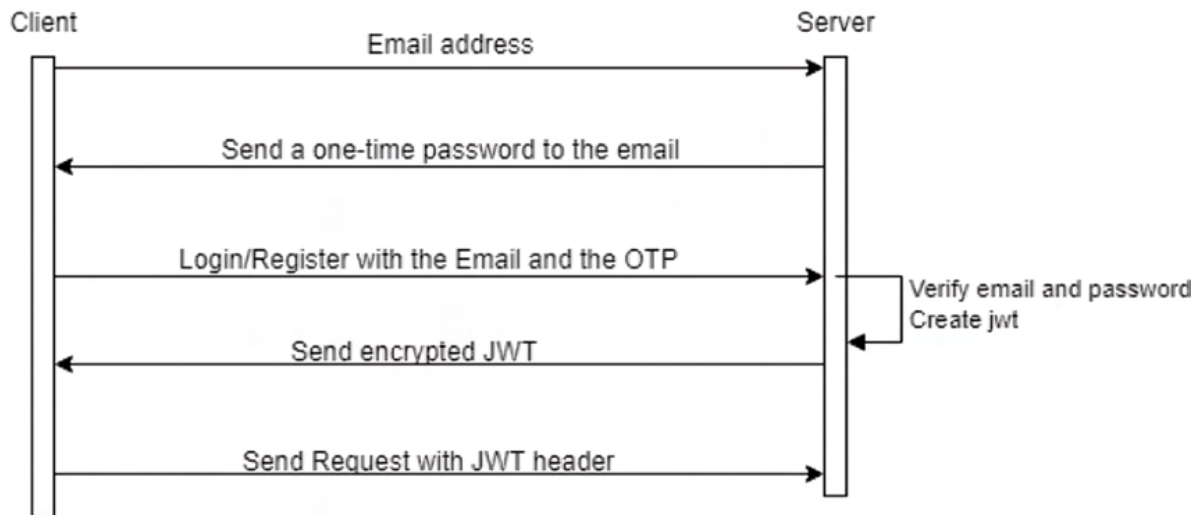


Figure 6: Email Verification and JWT Authentication Flow

### 3.1.4 Graphical User Interface Design and Implementation

#### 3.1.4.1 Login Page Graphical User Interface


On the login page, in order to simplify the information input process and ensure the information provided by the users is accurate, we have provided a drop-down menu for users to select their identity. Since our IM software is only intended for use by members of the HKU community, users can choose to identify as either students or teachers. Just below the identity selection drop-down menu, we have provided an email input field for users to enter their HKU email address. Depending on the identity selection given by the users, the label near the email input field will automatically adjust to the respective email domain. For students, the domain will be "@connect.hku.hk," while for teachers, it will be "@hku.hk."


*Figure 7: Login Page (for Student)*


*Figure 8: Login Page (for Teacher)*

After entering the HKU email address, the users must click the “Get One Time Password” button to get the OTP. Subsequently, an OTP will be generated by the system and sent to the user’s mailbox. This verification mechanism guarantees that only HKU students or teachers can access the system. It is important to note that the OTP must be used within five minutes to complete the login process, otherwise, the user will be required to obtain a new OTP again. In case someone using the app for the first time does not have an account, they can proceed to the registration page by clicking the "New User" button.

**Login**

Identity 

Email  New User

One Time Password  Resend


**Login**


Figure 9: Login Page after pressing the “Get One Time Password” button


### 3.1.4.2 Registration Page Graphical User Interface

Similar to the login process, users are required to first select their identity and provide their HKU email address in order to get an OTP. As long as the users receive the OTP via their mailbox, they have to enter it into the designated input field and proceed by clicking the “Verify One Time Password” button. This step is to make sure that only students or teachers in HKU can register for an account.

**Registration**

Identity 

Email 

One Time Password  Resend

**Verify One Time Password**

**Cancel**

Figure 10: Registration Page after entering the OTP

Once the OTP is successfully authenticated, additional input fields will dynamically appear for users to complete the registration process. For students, they have to provide their name, program of study, year of study, date of birth, as well as the profile image. In the program of study section, it is not mandatory for students to specify their second major and minor program. To simplify the process, a user-friendly drop-down menu is available for students to easily indicate their current year of study.

The registration form is organized into several sections:

- Name** (with a person icon): Includes three input fields for 'FirstName', 'LastName', and 'NickName'.
- Program** (with a list icon): Includes three input fields: 'FirstMajor', 'SecondMajor' (with 'Optional' text), and 'Minor' (with 'Optional' text).
- Year of Study** (with a calendar icon): A dropdown menu currently showing 'Year 1'.
- Date of Birth** (with a calendar icon): An empty input field with a calendar icon on the right.
- Profile Image** (with a person icon and a 'Remove' link): An empty input field with an upload icon on the right.

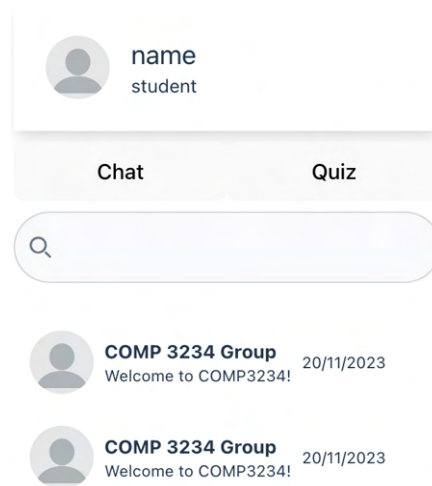
At the bottom of the form are two large buttons: 'Submit' and 'Cancel'.

Figure 11: Registration Page for student (after verifying OTP)

For teachers, the registration process follows a similar outline, with the exception that they are not required to input information regarding their program of study or year of study and these information input sections will not be included in the user interface.

### 3.1.4.3 Main Page Graphical User Interface

On the Main page, users will see their name and profile image displayed at the top. Just below, the "Chat" and "Quiz" buttons act as entry points to the app's main features, which include academic consultation as well as the quiz creating and answering system. By clicking the "Chat" button, users gain access to a comprehensive contact list, offering both one-to-one and group chat options. This feature fosters seamless communication for academic consultation purposes. On the other hand, selecting the "Quiz" button presents users with a list of groups, each housing multiple quizzes created by teachers. This section provides an opportunity for students to participate in quizzes and improve their educational experience.



*Figure 12: Main Page*

### 3.1.4.4 Profile Page Graphical User Interface

When users click their name which is shown at the top of the main page, they will be directed to the profile page. In the profile page, the users can review and modify their personal information. However, it's crucial to note that certain fields like name, email address, and user ID are fixed and cannot be modified as these parameters cater to essential functionalities. For example, the student name can be used by the teacher to identify students and the email address can be used for restricting the system access.

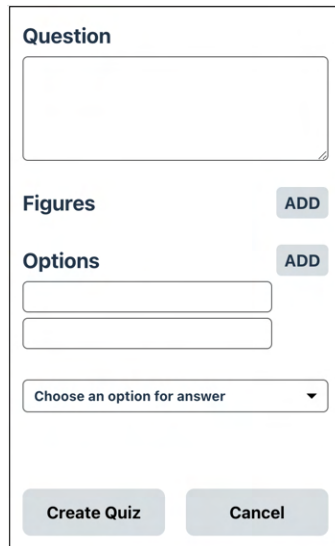
The image shows a mobile application profile page. At the top left is the title "Profile" and at the top right is a "cancel" button. Below the title is a circular profile picture placeholder with the text "Change profile picture" underneath it. The form contains several input fields: "First name" and "Last name" (two separate boxes), "Nickname" (one wide box), "User ID" (one wide box), and "Email address" (one wide box). At the bottom of the form is a "Save Changes" button.

*Figure 13: Profile Page*

### **3.1.4.5 Quiz Create Page Graphical User Interface**

The quiz create page can only be accessed by teachers to create quizzes. The primary goal of this project is to facilitate classroom discussions through simple questions and answers. Considering the limited screen size of mobile phones, we will initially focus on multiple-choice questions. On the Quiz Create page, teachers can type out their questions in the text area provided. Additionally, they can optionally attach some images to the questions by simply clicking the “ADD” button. When the “ADD” button in the image upload section is pressed, a pop-up window will appear and the teacher can choose the desired images to upload. Following this, teachers are required to provide choices for the multiple-choice question. Initially, two choices are mandatory, but teachers can add extra options by clicking the “ADD” button in the options define section.





The image shows a 'Quiz Create Page' form. It features a large text input field at the top labeled 'Question'. Below this are two sections: 'Figures' and 'Options', each with an 'ADD' button. The 'Options' section includes two smaller text input fields. At the bottom of the options section is a dropdown menu with the text 'Choose an option for answer'. At the very bottom of the form are two buttons: 'Create Quiz' and 'Cancel'.

Figure 14: Quiz Create Page

### 3.1.4.6 Chat Page Graphical User Interface

The name of the chat group (including both group and one-to-one chat) as well as its profile picture will appear at the top of the chat page. To facilitate easy navigation, a return button is placed next to the chat group profile image to allow users return back to the main page whenever necessary. Moreover, the chat messages will be presented just below the name and profile picture of the conversing group. Users can type and send chat messages to other users at the bottom of the page. By clicking the button next to the message input area, users can optionally attach documents or images to their messages.

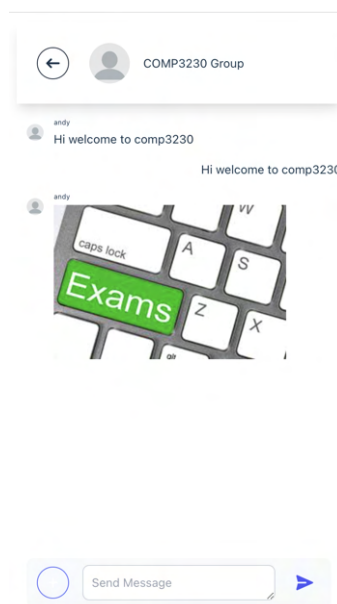


Figure 15: Chat Page

### **3.2 Limitations and Difficulties**

Given our lack of familiarity with MongoDB database usage and PWA development using React and Node.js, we have to invest an indeterminate amount of time acquiring knowledge from scratch. This includes comprehending the setup procedures for the MongoDB database and its working principle, mastering database interaction techniques, as well as understanding the programming language used in React and Node.js for mobile application development. As a result, allocating time to learn all the required knowledge and reserve time to implement all the features and functionalities of the IM software in this project within the time limit is the biggest difficulty. To shorten the time for learning the required knowledge, we will change our learning strategies and focus on mastering the essential knowledge required for the project. This entails prioritizing the fundamental concepts and knowledge of MongoDB, React, and Node.js that are directly applicable to our goals, rather than attempting to delve into every aspect in detail. Additionally, to effectively navigate this learning journey, we will not only utilize various online resources but also consult our supervisor more frequently. Seeking their guidance will ensure that we acquire knowledge in the right direction and promptly address any challenges or obstacles encountered along the way.

## 4. Future Work

Further details of future work on this project are shown in Table 1 below. After finishing the system design, database design and its setup, as well as the UI design and its implementation, the next crucial step will be the implementation of all functionalities on both the server side and the client side. This involves incorporating features such as real-time chatting and a system for creating and answering quizzes. After that, system testing and debugging and its optimization will be conducted.

Items	Date	Task	Status
1	October 2023	<ul style="list-style-type: none"><li>● Database design and related research</li></ul>	Completed
2	November 2023	<ul style="list-style-type: none"><li>● GUI design and related research</li><li>● GUI implementation</li><li>● Database setup</li></ul>	Completed
3	December 2023	<ul style="list-style-type: none"><li>● GUI implementation</li><li>● Real-time chatting feature implementation (for both server side and client side)</li><li>● Interactive quizzes creating and answering system implementation</li></ul>	In progress
4	January 2024		
5	February 2024		
6	March 2024	<ul style="list-style-type: none"><li>● System testing and debugging</li><li>● System optimization</li></ul>	Scheduled

*Table 1 : Schedule*

## **5. Conclusion**

This project aims to build an IM software for students so that they can consult their teachers more conveniently and effectively. Additionally, to facilitate student-teacher interactions during lessons and enhance students' learning efficiency, a quiz-creating and answering system will be included in the IM software to allow teachers to create interactive quizzes while students can respond to those quizzes during lessons. To achieve this goal, we have decided to develop this IM software in the form of a PWA as it can provide user experience like a native app and it is cross-platform. As of the time of writing of this interim report, the system architecture design, the database design and its setup, as well as most of the UI design and its implementation have been finished. Currently, we are working on all the functionalities on both the client side and the server side.

## Reference

- [1] L. Ceci, “Number of mobile phone messaging app users worldwide from 2019 to 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 Jan. 2, 2024).
- [2] B. Beth Hurst, R. Randall Wallace, and S. B. Nixon, “The Impact of Social Interaction on Student Learning,” *Reading Horizons: A Journal of Literacy and Language Arts*, vol. 52, no. 4, Oct. 2013.  
[https://scholarworks.wmich.edu/cgi/viewcontent.cgi?article=3105&context=reading\\_horizons](https://scholarworks.wmich.edu/cgi/viewcontent.cgi?article=3105&context=reading_horizons) (accessed Jan. 2, 2024).
- [3] E. Makarova, “Teacher-student interaction in the context of higher education”, [https://www.shs-conferences.org/articles/shsconf/pdf/2021/10/shsconf\\_dihelt2021\\_01041.pdf](https://www.shs-conferences.org/articles/shsconf/pdf/2021/10/shsconf_dihelt2021_01041.pdf) (accessed Jan. 2, 2024).
- [4] 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. (accessed Sep. 30, 2023)
- [5] 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)
- [6] React. Available: <https://react.dev/> (accessed Sep. 17, 2023).
- [7] “Virtual dom and Internals,” React, <https://legacy.reactjs.org/docs/faq-internals.html> (accessed Sep. 17, 2023)
- [8] Vite, <https://vitejs.dev/guide/> (accessed Sep. 24, 2023).

- [9] Node.js. Available: <https://nodejs.org/en> (accessed Sep. 11, 2023).
- [10] “NPM,” npm. Available: <https://www.npmjs.com/> (accessed Sep. 11, 2023).
- [11] Express. Available: <https://expressjs.com/> (accessed Sep. 11, 2023).
- [12] I. Fette and A. Melnikov, “The WebSocket Protocol,” RFC Editor. Available: <https://www.rfc-editor.org/rfc/rfc6455> (accessed Sep. 11, 2023).
- [13] “Introducing Json,” JSON. Available: <https://www.json.org/json-en.html> (accessed Sep. 11, 2023).
- [14] “JSON and Bson,” MongoDB. Available: <https://www.mongodb.com/json-and-bson> (accessed Sep. 11, 2023).
- [15] “Introduction: Testing library,” Testing Library RSS. Available: <https://testing-library.com/docs/> (accessed Sep. 24, 2023).
- [16] “MongoDB atlas database: Multi-cloud database service,” MongoDB. Available: <https://www.mongodb.com/atlas/database> (accessed Sep. 24, 2023).
- [17] V. Anthony Fu, Vitest. Available <https://vitest.dev/guide> (accessed Sep. 20, 2023).
- [18] “Vite-Jest,” GitHub. Available: <https://github.com/sodatea/vite-jest/tree/main/packages/vite-jest> (accessed Sep. 24, 2023).
- [19] “Introduction: Testing library,” Testing Library RSS. Available: <https://testing-library.com/docs/> (accessed Sep. 24, 2023).
- [20] “API tools,” Postman API Platform. Available: <https://www.postman.com/product/tools/> (accessed Sep. 24, 2023).