DESIGN IMPLEMENTATION DOCUMENTATION

for

FlowChat

Version 1.0

Prepared by : Group A7

LAM Tsz Kit (1155194085)
NG Chun Faat (1155194596)
CHU Ming Kong (1155194293)
CHOI Ho Yan (1155194468)
CHAN Yuet Xing (1155194086)

Submitted to : Dr. LAM Tak Kei

Course CSCI3100
Department of Computer Science and Engineering

March 11, 2025

Contents

1 Document Revision History

Version	Revised By	Revision Date	Comments
0.1	LAM Tsz Kit, NG Chun Faat, CHU Ming Kong, CHOI Ho Yan, CHAN Yuet Xing	1 March 2025	Initial Draft
0.2	LAM Tsz Kit, NG Chun Faat	3 March 2025	Completed Guidelines and Assumptions, System Architecture
0.3	CHAN Yuet Xing	6 March 2025	Completed database schema, workflow diagram, API designs and specifications for Direct Message
0.4	LAM Tsz Kit	7 March 2025	Completed database schema, workflow diagram, API designs and specifications for Account Management, Discussion Forum
0.5	NG Chun Faat, CHU Ming Kong, CHOI Ho Yan	9 March 2025	Completed Introduction, User Interface and Frontend Component Design
1.0	LAM Tsz Kit, NG Chun Faat, CHU Ming Kong, CHOI Ho Yan, CHAN Yuet Xing	10 March 2025	Reviewed and Finalized

2 Introduction

This document outlines the development considerations, guidelines, technical specification and design for FlowChat, an online forum web application. The app's primary features include a user-generated forum spanning diverse topics, networking capabilities with following and messaging functionalities, and user profile and management systems.

Readers may skip to their section of interest. The software development model and architecture are detailed in the *Guidelines and Assumptions*, and *System Architecture* sections. The design and high-level details of major visual components are covered in *User Interface and Frontend Component Design* section. The database structure, component communication interfaces, and backend logic are documented in the *Data Model Design*, *Application Programming Interface Design* and *Application Programming Interface Specification* sections.

3 Guidelines and Assumptions

3.1 Software Development Guidelines

According to the Software Requirements Specification, the software project can be divided into five main features, including Account Management, Administrative Management, Personal Profile, Discussion Forum, and Direct Message.

All developers should follow the incremental model in the software development process, where an incremental refers to one main feature. Each incremental period adopts a simple waterfall model involving design specification, implementation, testing and delivery. All incremental periods are designed to be sequential, according to the level of importance and the software order flow. The tentative development schedule is shown below:

Date	Feature	Task
10 Feb — 16 Feb	# 0 System Architecture	Initialization
17 Feb — 23 Feb	# 1 Account Management	Design and Implementation
24 Feb — 02 Mar	# 1 Account Management	Implementation and Testing
03 Mar — 09 Mar	# 4 Discussion Forum # 5 Direct Message	Design
10 Mar — 16 Mar	# 4 Discussion Forum	Implementation
17 Mar — 23 Mar	# 4 Discussion Forum	Implementation and Testing
24 Mar — 30 Mar	# 5 Direct Message	Implementation
31 Mar — 06 Apr	# 5 Direct Message	Implementation and Testing
07 Apr — 13 Apr	# 2 Administrative Management # 3 Personal Profile	Design
14 Mar — 20 Apr	# 2 Administrative Management # 3 Personal Profile	Implementation and Testing
21 Mar — 01 May	All	Integration Testing

All developers should pay attention to the issue boxes on GitHub KanBan. All ready-to-pick-up jobs will be assigned to some developers and available as issue boxes in the "Ready" section. Developers should move the issue box they are working on to the "In Progress" section during development. After development, the issue box should be placed in the "In Review" section for deployment and testing. After that, the issue box should be in the "Done" section.

3.2 Documentation Assumptions

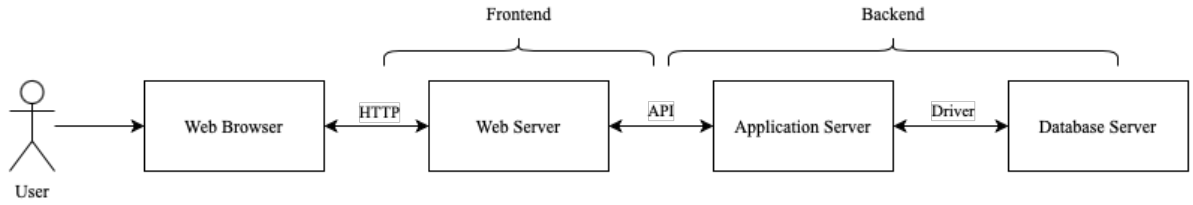
The first version of Design Implementation Documentation will focus on the system architecture and high-level design details of features including Account Management, Discussion Forum and Direct Message.

The design details of features for Administrative Management and Personal Profile will be added with the software development process. All content is subject to change.

All low-level design implementations are documented in code. Relevant information will be listed in the Appendix for references.

4 System Architecture

The system architecture is designed to be a Client-Server model. The overall structure is shown below:

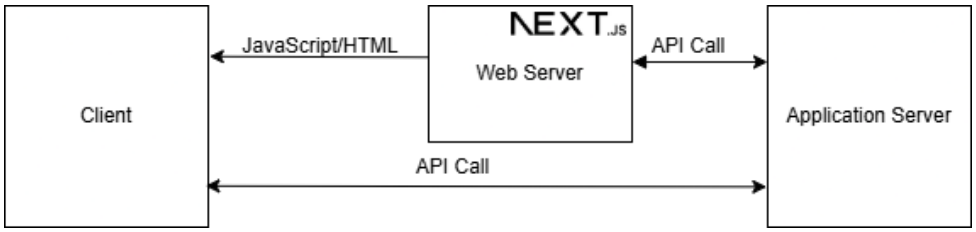


4.1 Web Server

The web server renders the application interface using React components, with Next.js enabling Client-Side Rendering (CSR), Server-Side Rendering (SSR), routing and data fetching. TypeScript is used as the primary development language to ensure type safety and correctness.

Deployed on Microsoft Azure, the server defaults to CSR for most components to reduce server workload. In CSR, the client compiles the web page locally using JavaScript sent from the server. For user-generated content like posts and comments, SSR is used to pre-compile pages on the server, delivering fully-rendered HTML to the client for faster load times and improved Search Engine Optimization.

Data requests to the application server are primarily initiated on the client side while server-rendered components handle data requests on the server side. These requests are facilitated through application programming interfaces provided by the application server, ensuring efficient data flow between client and server.



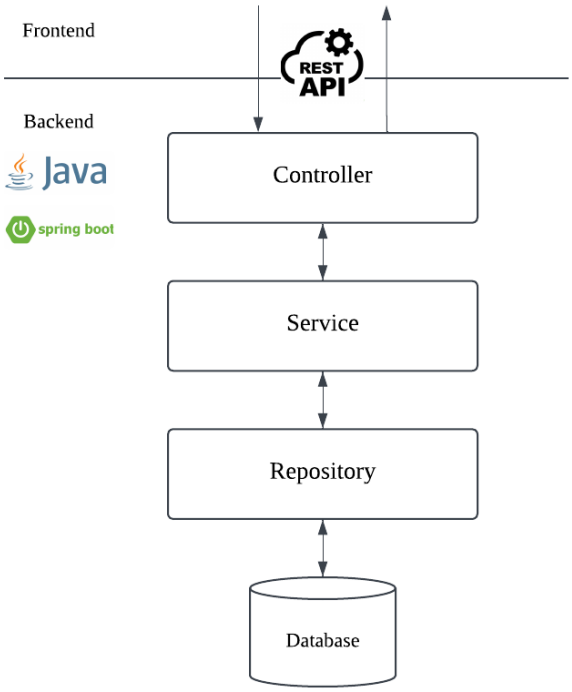
4.2 Application Server

The application server is responsible for writing application programming interfaces (APIs) to provide dynamic content such as processing business logic on data and interacting with the database server, to the web server.

The provided dynamic content is built with a microservices architecture, in which a functional feature is divided into several independent components that run each application logic as a service, to achieve loose coupling and enhance software scalability. The microservices use RESTful APIs to communicate with frontend, in order to provide standard HTTP methods and uniform interface, and process stateless requests.

The application server is written in Java and deployed to Microsoft Azure. To adopt the microservices architecture and RESTful API, the Spring Boot framework is used to provide configuration initialization and program structure, which is a layered structure:

- Controller: Responsible for defining the HTTP methods (GET, POST, PUT, DELETE), Uniform Resource Identifiers (URIs), query parameters and request body of APIs. Wraps data into the response body in JSON format and returns it to the frontend.
- Service: Responsible for applying business logic on data processing.
- Repository: Responsible for defining SQL and query data type. The data returns from the database can be directly matched to Java data types and objects, which are defined in the Model Class.



4.3 Database Server

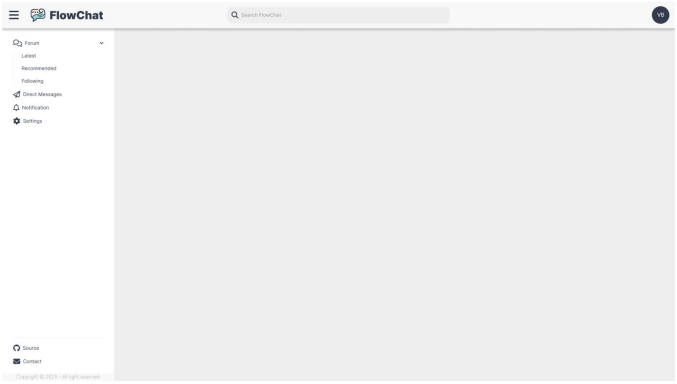
The database server is responsible for storing data.

The database type is SQL database and the server is hosted through Microsoft Azure in East Asia. The connection between application and database server is established using Java DataBase Connectivity (JDBC) driver with SQL authentication. The maximum storage is 2GB.

5 User Interface and Frontend Component Design

The User Interface (UI) and Frontend Component Design for FlowChat ensure a seamless experience using React and Next.js. The design adheres to a modular layout, ensuring consistency across pages while maintaining responsiveness for various screen sizes, with an optimal resolution of 1920x1080. The primary designs are completed using Figma. DaisyUI with a light color scheme is used to enhance accessibility and aesthetics.

5.1 Layout (Deployed)



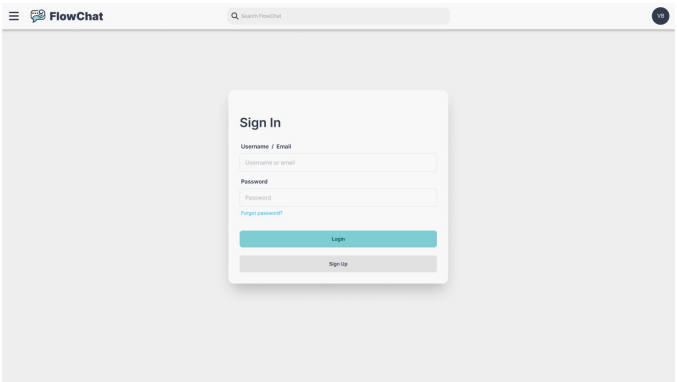
5.1.1 Navbar

The Navbar facilitates navigation across FlowChat's core features. Users may utilize the search bar (accepts keywords, maximum 100 characters) to locate posts or usernames. The logo redirects to the homepage, the profile icon accesses the user profile, and the menu icon toggles the Drawer. Navigation is unrestricted for ease of use while the search bar requires a log-in session key.

5.1.2 Drawer

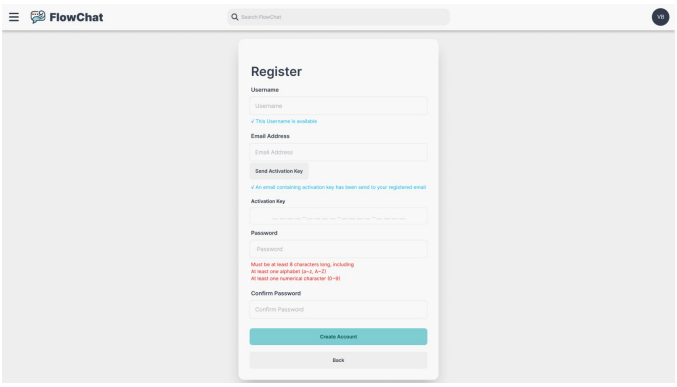
The Drawer provides additional navigation options for FlowChat users. It includes access to forums ("Latest" shows the newest posts, "Recommended" displays posts based on user interests, "Following" lists posts from followed users), messages, notifications, and settings.

5.2 Sign-In (Deployed)



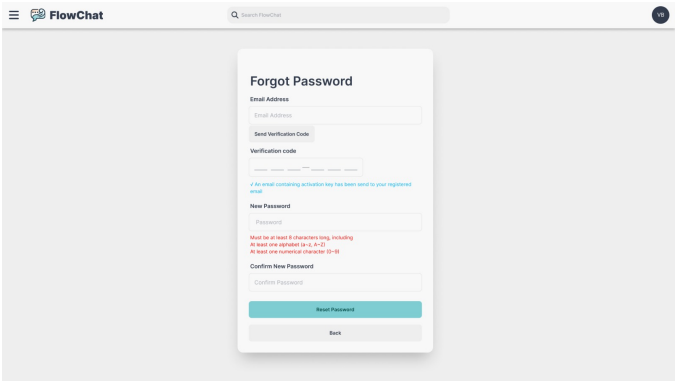
The Sign-In page enables user authentication for FlowChat access. Users must enter a username/email (maximum 50 characters) and password (minimum 8 characters, including letters and numbers). Select "Login" to proceed, "Sign Up" to register, or "Forgot Password" for recovery assistance, ensuring secure platform entry.

5.3 Register Account (Deployed)



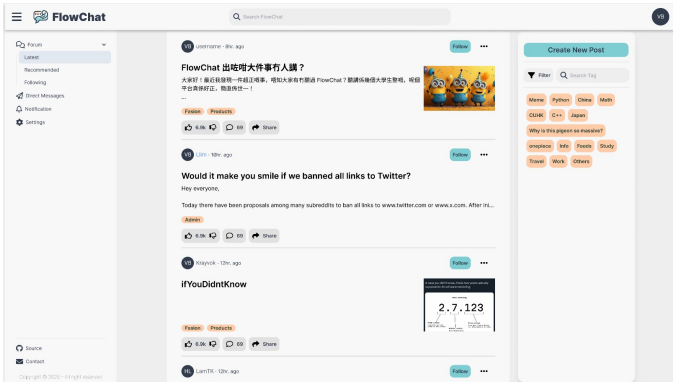
The Register Account page allows new users to join FlowChat. Users must provide a username (1-50 characters, unique), email (valid format), activation key (16 characters), and password (minimum 8 characters, including letters and numbers). Click "Send Activation Key" to receive the key, then "Create Account" to register.

5.4 Forgot Password (Deployed)



The Forgot Password page assists users in recovering their FlowChat account. Users must input their registered email (valid format), a verification code (6 digits), and a new password (minimum 8 characters, including letters and numbers). Click "Send Verification Code," then "Reset Password" to restore access.

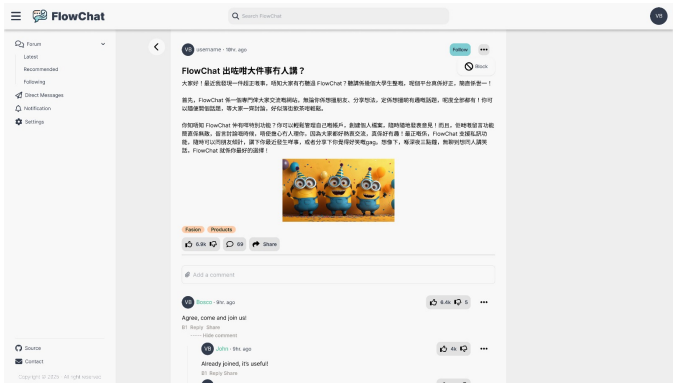
5.5 Post Preview List



The Forum page displays the latest posts in chronological order for FlowChat users. In each post preview, the user can follow the original poster, like, dislike and share. The tags in the bottom left help to identify

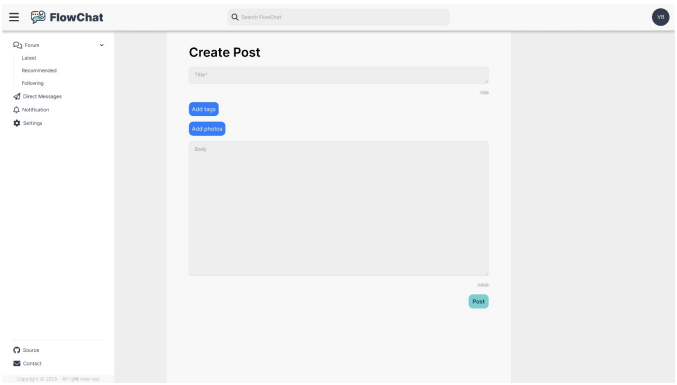
interests. Click the top-right menu of the post to hide one particular post. On the right-hand side, users can click to "Create New Post" or filter relevant posts by tags. Click in the post review to see the post details.

5.6 Post Details



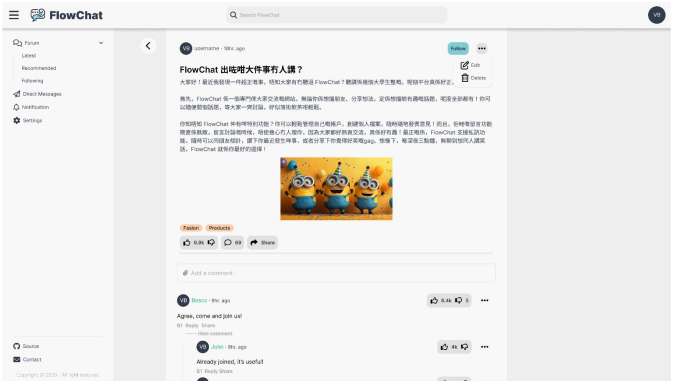
The Post Details page allows users to access individual posts in detail, including full post content and the comment section. In the comment section, users can leave text input (maximum 500 characters) and/or images (maximum 5MB) for opinions to the post, and further reply to comments. Users can like, dislike, or share comments to engage with the content. Use the upper-left arrow to return to the Home Page.

5.7 Create Post



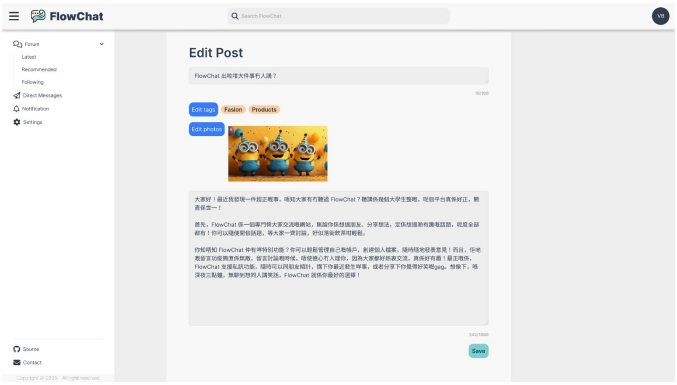
The Create Post page enables users to publish new content on FlowChat. Users must enter a title (maximum 100 characters), and content body (maximum 1,000 characters) with images (maximum 5MB). Click "Add Tags" to include relevant tags, then "Post" to submit the content.

5.8 My Post



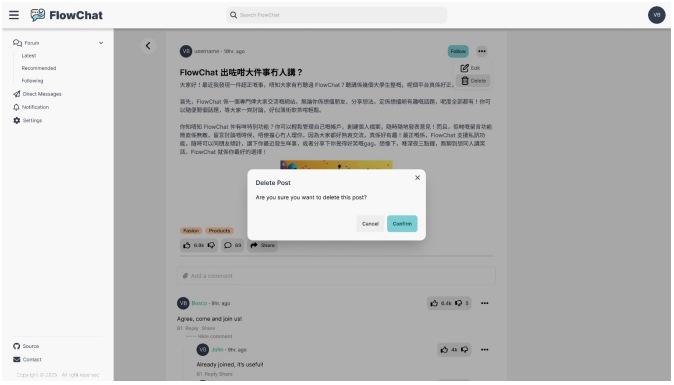
The My Post page allows users to review their published content. Users can click the top-right three-dot menu, select "Edit" to modify the content or "Delete" to delete the post.

5.9 Edit Post



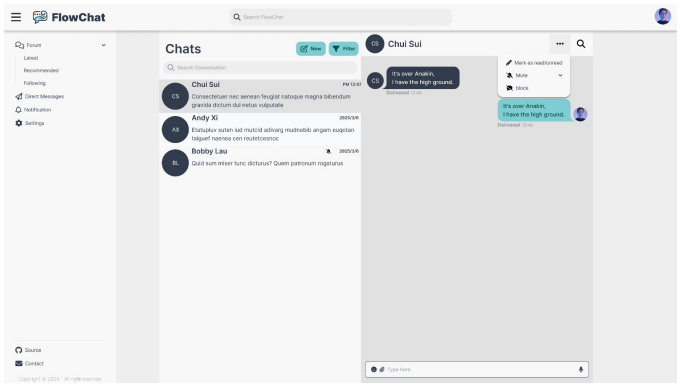
The Edit Post page enables users to modify their existing posts on FlowChat. Users can update the title (maximum 100 characters) and body (maximum 10,000 characters). Click "Add Tags" to revise tags or "Add Photos" to update images (maximum 5MB), then click "Post" to save changes.

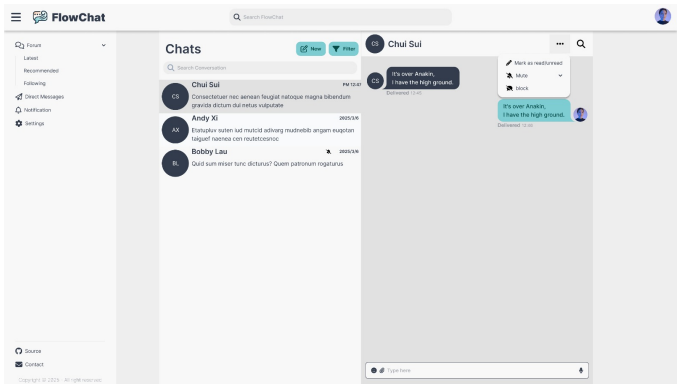
5.10 Delete Post

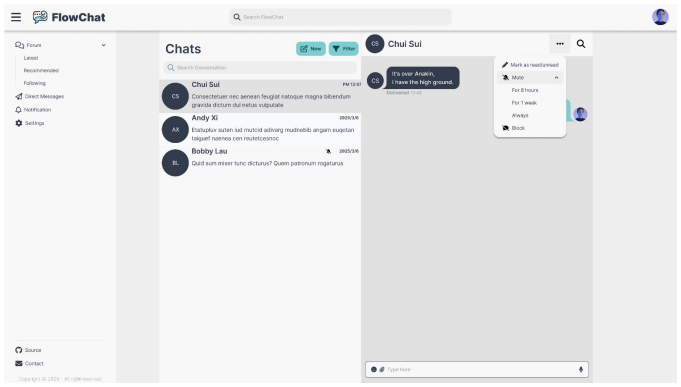


The Delete Post pop-up window allows users to remove their published content. A confirmation dialog appears with the message "Are you sure you want to delete this post?" Users can click "Cancel" to abort or "Confirm" to proceed with deletion. Users cannot undo this operation and recover deleted posts.

5.11 Direct Messages Page







The Direct Messages page enables private communication between FlowChat users. The left panel displays a list of conversations with usernames and message previews. Select a conversation to view messages in the right panel, where timestamps and delivery status are shown. Users can type messages (maximum 1,000 characters) in the input field at the bottom, send media (maximum 5MB), or record and send voice messages to continue the conversation.

6 Data Model Design

Data models are containers that format data for frontend-backend communication, processing, and storage.

6.1 Communication Model

The data format in the application programming interfaces is the JSON format. When requesting and responsing resources from the backend, it is used in the request body and response body.

In the request body, there is no general data model. All JSON keys must be set according to the specifications of each API.

In the response body, the general JSON format always starts with two pairs of keys/values "message" and "data". For each successful API response, the "message" indicates the work done, and the "data" is the actual information returned from the application server. See the following example:

```
{
    "message": "Password is reset",
    "data": {
        "username": edwinlamtk,
        "isSuccess": true
    }
}
```

For each expected exception that occurred in the application server, the boolean value with the key "isXXX" is set to false, and the values of other keys are set to null. See the following example:

```
{
    "message": "Key is not available",
    "data": {
        "isSuccess": false,
        "username": null
    }
}
```

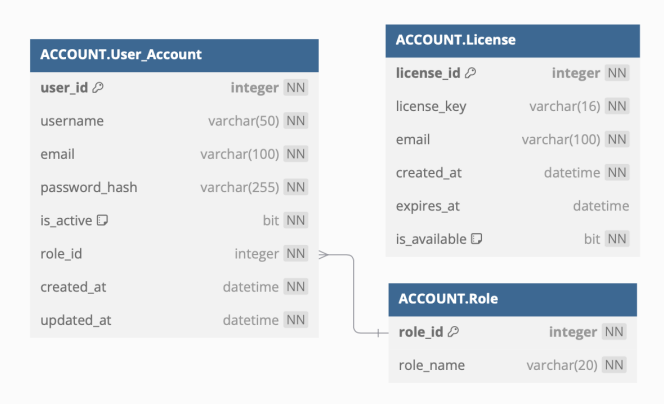
For each unexpected exception that occurred in the application server, the "message" is replaced by Java Exception Message, and the value of "data" is set to null. See the following example:

```
{
    "message": "Fail: java.lang.ArithmeticException: / by zero",
    "data": null
}
```

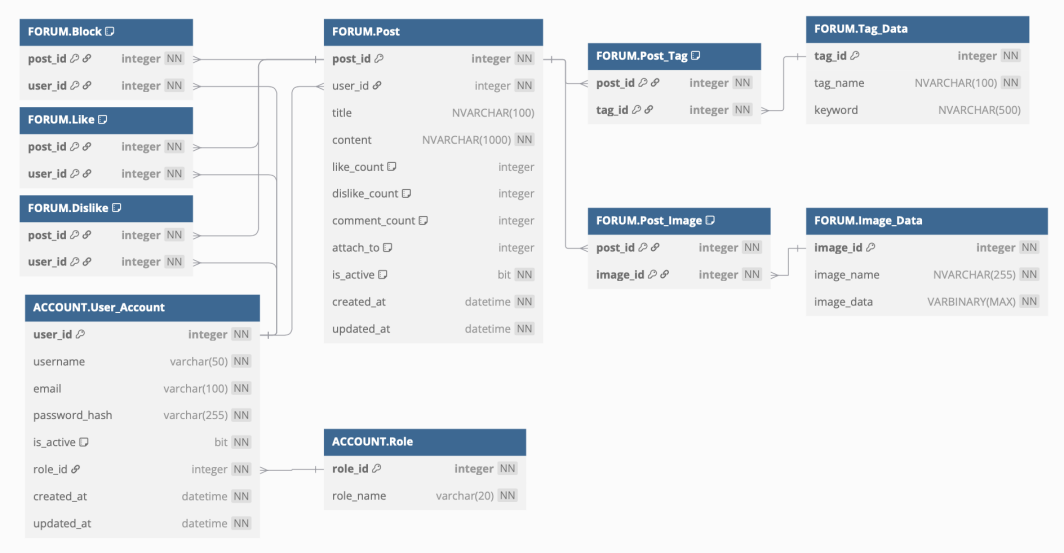
6.2 Database Schema

The database schema describes the data attributes, data types and the entity relationships in the database.

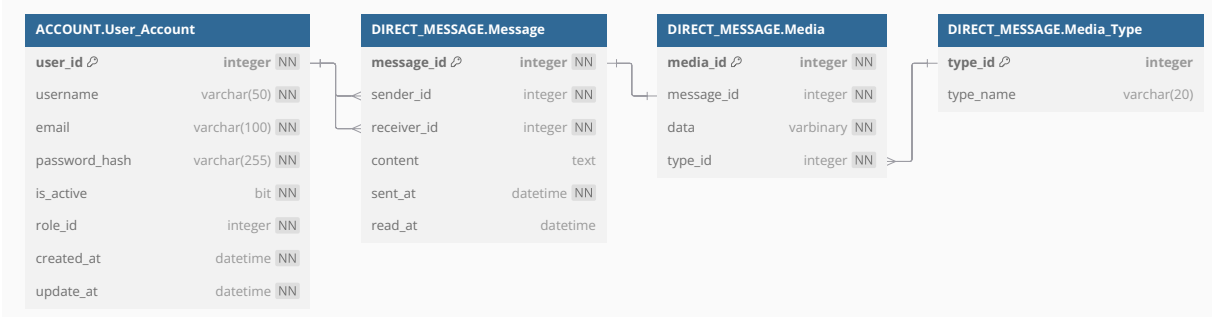
6.2.1 *Account Management (Built)*



6.2.2 *Discussion Forum*



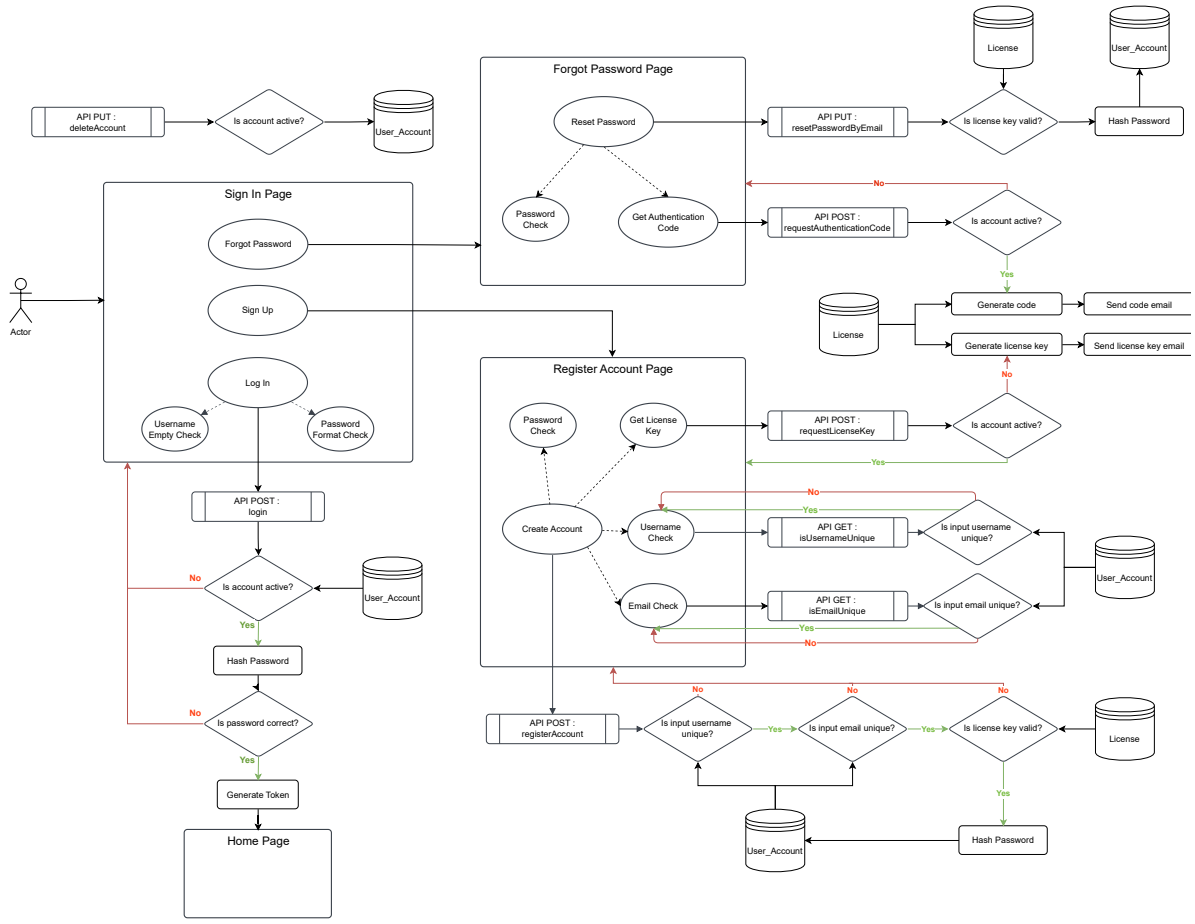
6.2.3 *Direct Message*



7 Application Programming Interface Design

Application programming interfaces (APIs) are the essential communication channel for frontend applications to communicate with backend applications. It specifies the application data input, expected data flows and exceptions, and data output. In the following, the overall data flowchart diagrams and each API design corresponding to feature requirements are presented. For detailed specification and expected exceptions, please see Section 9.

7.1 Account Management (Deployed)



The general API path for this functional feature is,

`https://flowchatbackend.azurewebsites.net/api/Account`

There are total 8 APIs for Account Management:

- Sign in to FlowChat (R1.2 Log-in Function, R1.3 Log-out Function)
  POST : `https://flowchatbackend.azurewebsites.net/api/Account/login`
- Check if an username is unique (R1.1 Account Sign-up)
  GET : `https://flowchatbackend.azurewebsites.net/api/Account/isUsernameUnique`
- Check if an email is unique (R1.1 Account Sign-up)
  GET : `https://flowchatbackend.azurewebsites.net/api/Account/isEmailUnique`
- Request a license key for account activation (R1.1 Account Sign-up, R1.6 License Management)
  POST : `https://flowchatbackend.azurewebsites.net/api/Account/requestLicenseKey`
- Register a new account (R1.1 Account Sign-up, R1.6 License Management)
  POST : `https://flowchatbackend.azurewebsites.net/api/Account/registerAccount`
- Request an authentication code for password reset (R1.4 Password Reset)
  POST : `https://flowchatbackend.azurewebsites.net/api/Account/requestAuthenticationCode`

- Reset password of a user account (R1.4 Password Reset)

  PUT : `https://flowchatbackend.azurewebsites.net/api/Account/resetPasswordByEmail`

- Delete a user account (R1.5 Account Deletion)

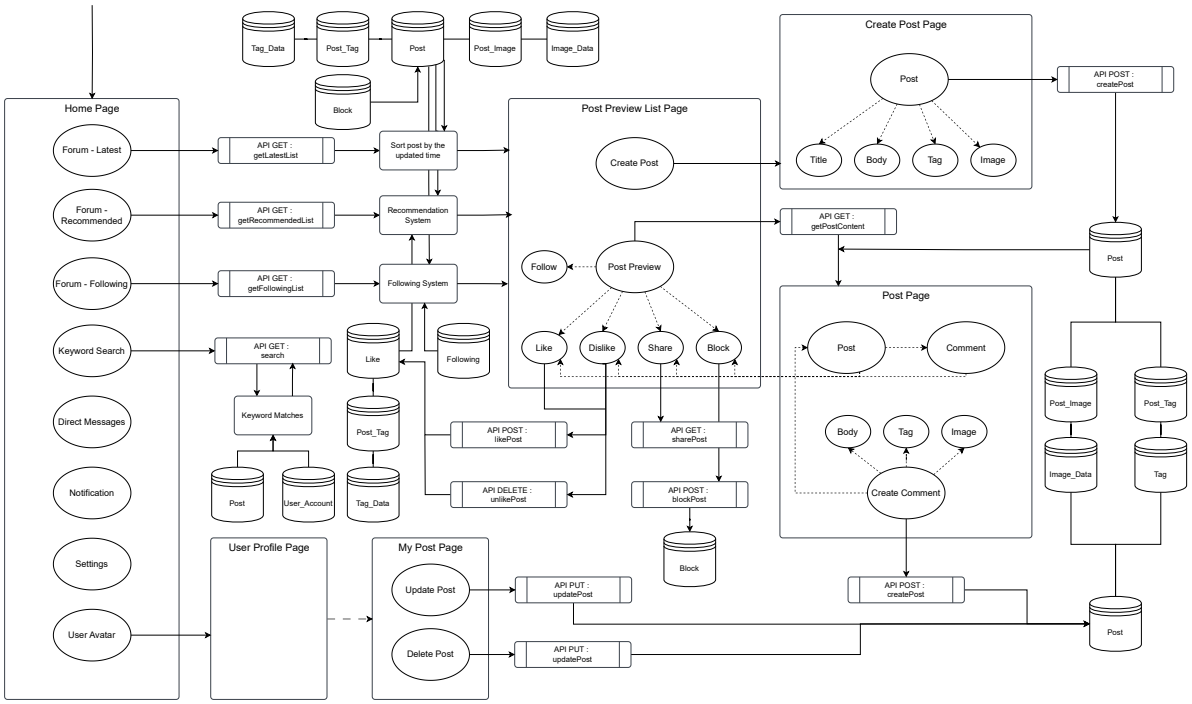
  PUT : `https://flowchatbackend.azurewebsites.net/api/Account/deleteAccount`

After logging into FlowChat, API services of all remaining functional features require an authorization header in the request, which contains a JSON Web Token (JWT) as shown in the following:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.
    eyJyb2xlIjoidXNlciIsImlkIjo4LCJzdWIiOiJlZHdpbjcifQ.--
    j1jfbCu0CN4QfuuuVzNEZp8tSjKTcSrTS7etIyh50
```

HTTP 200 OK will be returned if the request has a valid JWT, otherwise, HTTP 401 Unauthorized will be returned.

7.2 Discussion Forum



The general API path for this functional feature is,

`https://flowchatbackend.azurewebsites.net/api/Forum`

There are total 12 APIs for Discussion Forum:

- Get the latest list of post previews for a user (R4.1 Post View)

  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/getLatestList`

- Get the recommended list of post previews for a user (R4.1 Post View, R4.9 Recommendation System)

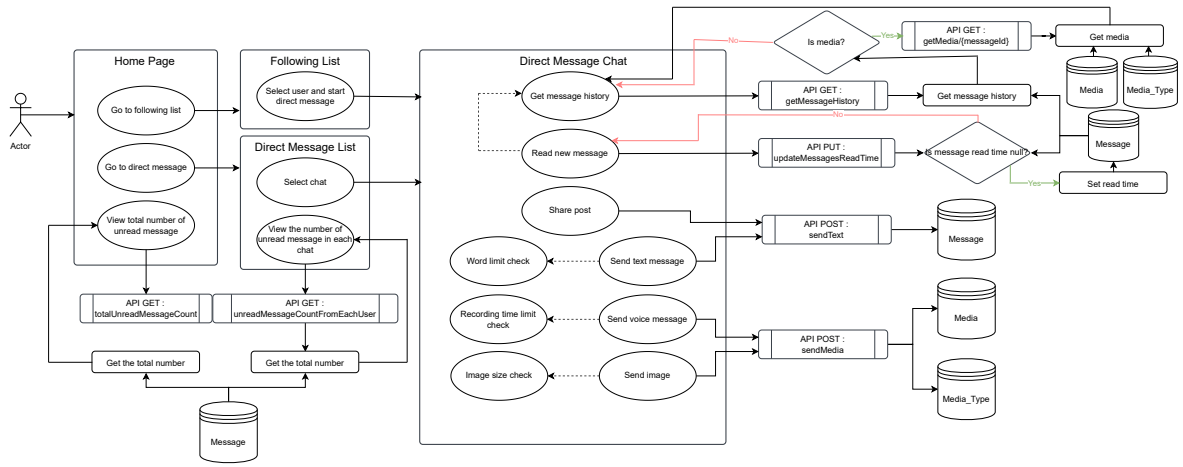
  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/getRecommendedList`

- Get the following list of post previews for a user (R4.1 Post View, R3.3 Following System)

  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/getFollowingList`

- Get all information of the post by a post ID (R4.1 Post View)
  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/getPostContent`
- Create a new post or comment (R4.2 Post Creation, R4.6 Comment)
  POST : `https://flowchatbackend.azurewebsites.net/api/Forum/createPost`
- Update a post or comment created by the user (R4.3 Post Update, R4.6 Comment)
  PUT : `https://flowchatbackend.azurewebsites.net/api/Forum/updatePost`
- Delete a post or comment created by the user (R4.4 Post Deletion, R4.6 Comment)
  PUT : `https://flowchatbackend.azurewebsites.net/api/Forum/deletePost`
- Block a post or comment created by other user (R4.5 Post Blocking, R4.6 Comment)
  POST : `https://flowchatbackend.azurewebsites.net/api/Forum/blockPost`
- Like and dislike a post or comment (R4.7 Like-Dislike Function, R4.6 Comment)
  POST : `https://flowchatbackend.azurewebsites.net/api/Forum/likePost`
- Un-like and un-dislike a post or comment (R4.7 Like-Dislike Function, R4.6 Comment)
  DELETE : `https://flowchatbackend.azurewebsites.net/api/Forum/unlikePost`
- Generate a share link for a post (R4.8 Share Function)
  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/sharePost`
- Search information by keywords (R4.10 Search Function)
  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/search`

7.3 Direct Message



The general API path for this functional feature is,

`https://flowchatbackend.azurewebsites.net/api/DirectMessage/`

There are total 7 APIs for Direct Message:

- Send a text message to the following users (R5.2 Text Message, R5.4 Post Share)
  POST : `https://flowchatbackend.azurewebsites.net/api/DirectMessage/sendText`

- Send an image or a voice message to the following users (R5.3 Image Share, R5.5 Voice Message)

  POST : `https://flowchatbackend.azurewebsites.net/api/DirectMessage/sendMedia`

- Get the message history with a following user (R5.2 Text Message, R5.3 Image Share, R5.4 Post Share, R5.5 Voice Message)

  GET : `https://flowchatbackend.azurewebsites.net/api/DirectMessage/getMessageHistory`

- Get single message, which may be image or voice message (R5.3 Image Share, R5.5 Voice Message)

  GET : `https://flowchatbackend.azurewebsites.net/api/DirectMessage/getMedia/{messageId}`

- Update the "read_at" time of the messages (R5.2 Text Message, R5.3 Image Share, R5.4 Post Share, R5.5 Voice Message)

  PUT : `https://flowchatbackend.azurewebsites.net/api/DirectMessage/updateMessagesReadTime`

- Get the count of total unread messages from all following users (R5.2 Text Message, R5.3 Image Share, R5.4 Post Share, R5.5 Voice Message)

  GET : `https://flowchatbackend.azurewebsites.net/api/DirectMessage/totalUnreadMessageCount`

- Get the count of unread messages from each following user (R5.2 Text Message, R5.3 Image Share, R5.4 Post Share, R5.5 Voice Message)

  GET : `https://flowchatbackend.azurewebsites.net/api/DirectMessage/unreadMessageCountFromEachUser`

8 Backend Component Design

In backend, the components are mainly controller, service, repository and data model Java classes. Components are managed using Object-Oriented Programming, in which the features of encapsulation and abstraction are highly adopted. It is suggested that not to define and couple any complex relationship of objects using the features of inheritance and polymorphism.

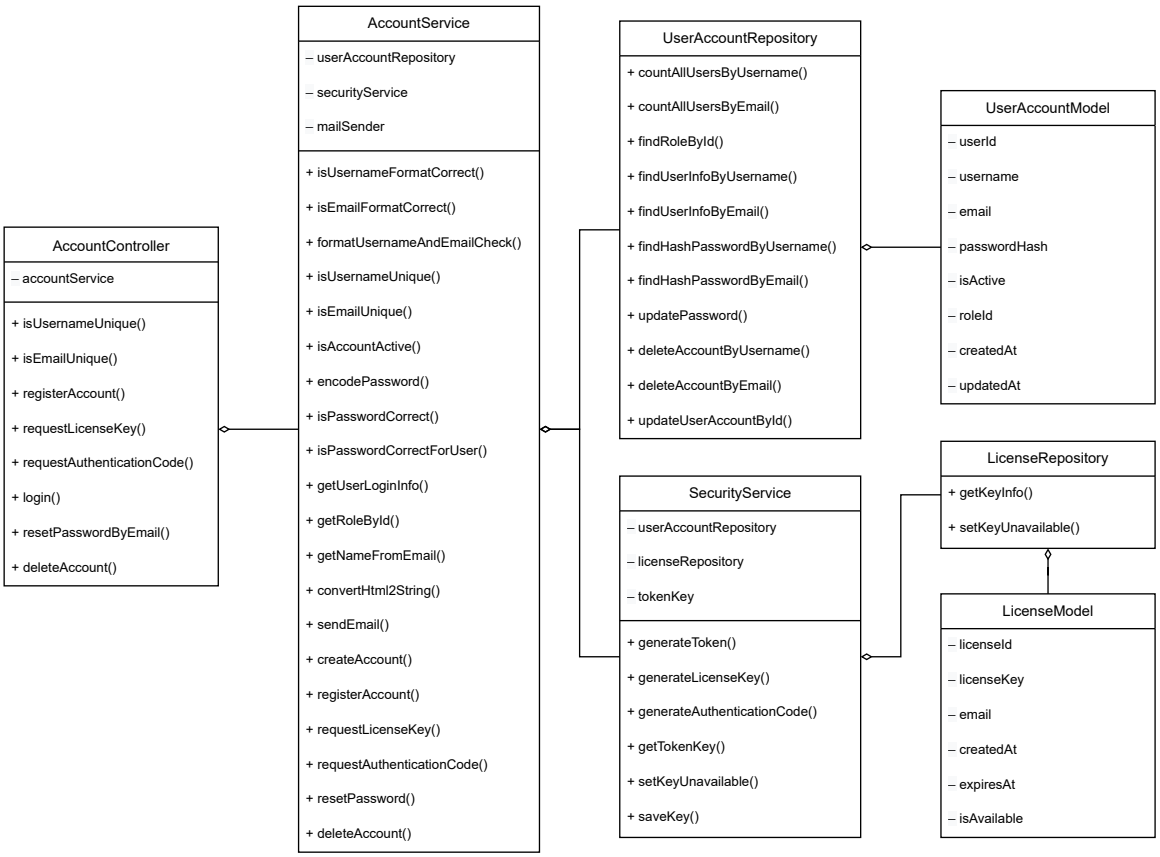
Data model classes are defined with object attributes matched with database entity attributes. All the access modifiers of attributes should be "private" and encapsulated by auto getter and setter methods.

Repository classes are interfaces extending from JpaRepository, allowing executions of SQL commands by calling the abstraction methods.

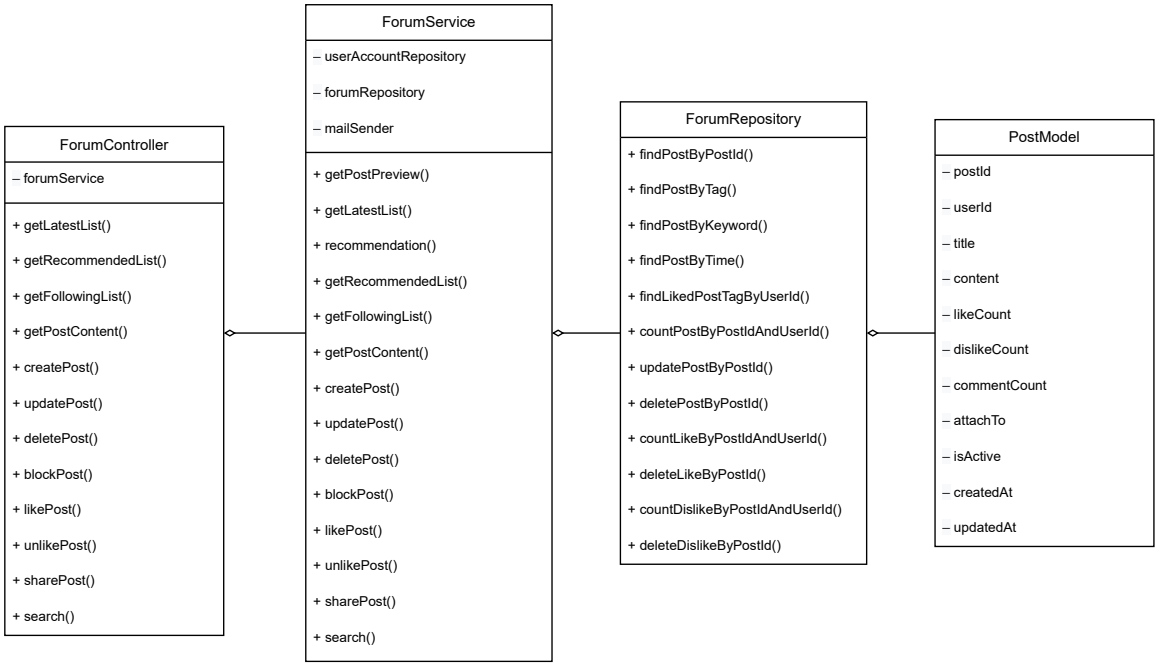
Controller classes are responsible for converting data in JSON format to Java Objects, passing them to service classes. The processed Java Objects in service classes return back to controller classes for converting back to JSON format. With the Inversion of Control principle in the Spring framework, the loosely coupled object relationship can be further achieved by injecting the dependency of service classes to controller classes.

The low level component designs for each functional feature are described by the following class diagrams. The detailed specification of component functions are written in the JavaDoc Comments. When the functions are finalized, they will be listed in the Appendix for references.

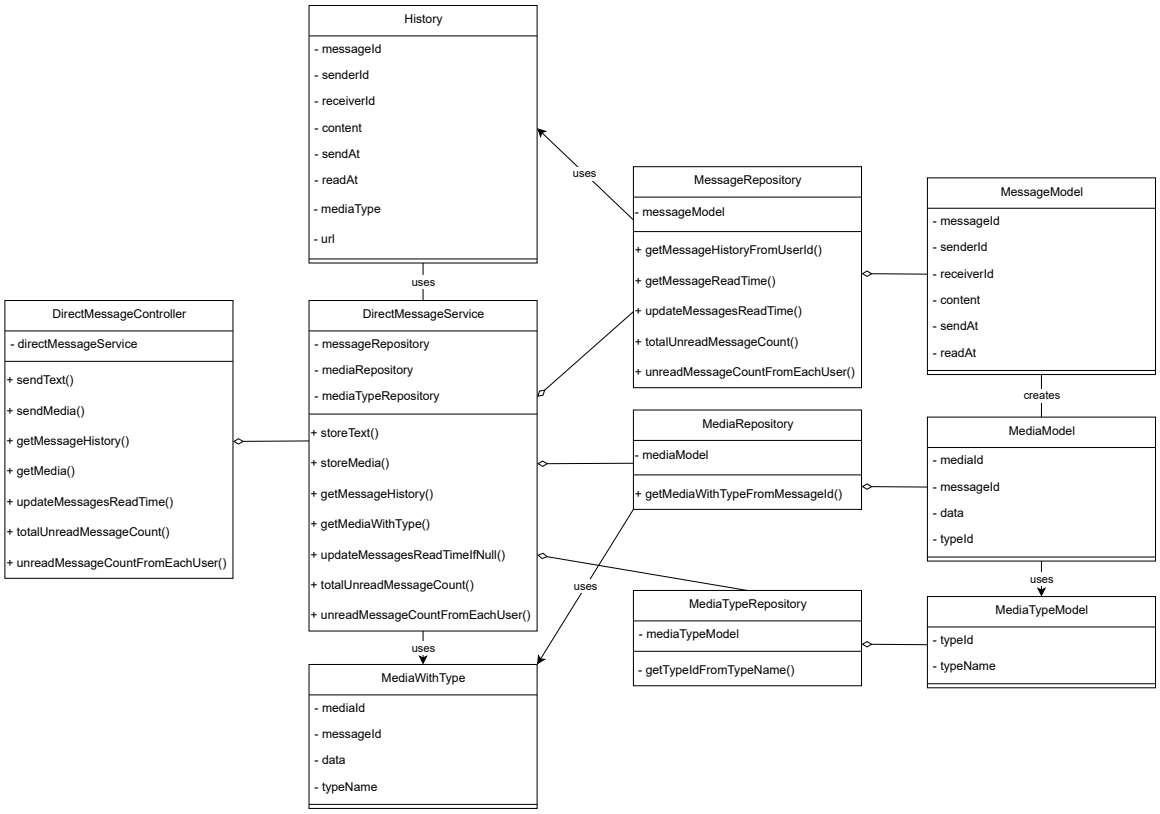
8.1 Account Management



8.2 Discussion Forum



8.3 Direct Message



9 Appendix I: Application Programming Interface Specification

9.1 Account Management (Deployed)

9.1.1 Sign In

- Sign in to FlowChat

  Description: Check if the input data format is correct, the account is active and the password is correct. After checking, generate a JWT session token. The token will be discarded in frontend when an user signs out.

  POST : `https://flowchatbackend.azurewebsites.net/api/Account/login`

  Request Body: set one of the "username" and "email" to be null

```
{
    "username": "edwinlamtk",
    "email": null,
    "password": "edwinlamtk"
}
```

  Response Body:

```
{
    "message": "Account is active and password is correct",
    "data": {
        "isAccountActive": true,
        "isPasswordCorrect": true,
        "user": {
            "roles": "user",
            "id": 25,
            "token": "ey...",
            "username": "edwinlamtk"
        }
    }
}
```

9.1.2 Register Account

- Check if a username is unique

  Description: Ensure that the username is not repeated when creating a new account.

  GET : `https://flowchatbackend.azurewebsites.net/api/Account/isUsernameUnique`
  Query Parameters: `username`

  Response Body:

```
{
    "message":"The username is unique",
    "data": {
        "isUsernameUnique": true
    }
}
```

```
{
    "message":"The username is not unique",
```

```
    "data": {
        "isUsernameUnique": false
    }
}
```

- Check if an email is unique

  Description: Ensure that the email is not repeated when creating a new account.

  GET : `https://flowchatbackend.azurewebsites.net/api/Account/isEmailUnique`
  Query Parameters: `email`

  Response Body:

```
{
    "message":"The email is unique",
    "data": {
        "isEmailUnique": true
    }
}
```

```
{
    "message":"The email is not unique",
    "data": {
        "isEmailUnique": false
    }
}
```

- Request a license key for account activation

  Description: Generate a new 16-char, 1-week long license key, save it in the database ACCOUNT.License, and send it to the user through HTML email.

  POST : `https://flowchatbackend.azurewebsites.net/api/Account/requestLicenseKey`

  Request Body:

```
{
    "email":"edwinlamtk@gmail.com"
}
```

  Response Body:

```
{
    "message": "A new license key is generated and sent",
    "data": {
        "isSuccess": true
    }
}
```

- Register a new account

  Description: Check if the input data format is correct, the account is inactive and the license key is valid. After checking, save the user information in the database ACCOUNT.User_Account.

  POST : `https://flowchatbackend.azurewebsites.net/api/Account/registerAccount`

  Request Body:

```
{
    "username": "edwinlamtk",
    "email": "edwinlamtk@gmail.com",
    "password": "edwinlamtk",
    "licenseKey": "N1SM9K4AZF90F1PU"
}
```

Response Body:

```
{
    "message": "A new account is created",
    "data": {
        "user": {
            "role": "user",
            "id": 14,
            "username": "edwinlamtk"
        },
        "isSuccess": true
    }
}
```

9.1.3 Forgot Password

- Request an authentication code for password reset

  Description: Generate a new 6-digit, 5-min long license key, save it in the database ACCOUNT.License, and send it to the user through HTML email.

  POST : `https://flowchatbackend.azurewebsites.net/api/Account/requestAuthenticationCode`

  Request Body:

```
{
    "email":"edwinlamtk@gmail.com"
}
```

  Response Body:

```
{
    "message": "A new authentication code is generated and sent",
    "data": {
        "isSuccess": true
    }
}
```

- Reset password of a user account

  Description: Update a new password of a user account with the given email in the database ACCOUNT.User_Account.

  PUT : `https://flowchatbackend.azurewebsites.net/api/Account/resetPasswordByEmail`

  Request Body:

```
{
    "email": "edwinlamtk@gmail.com",
```

```
    "password": "edwinlamtk",
    "authenticationCode": "250085"
}
```

Response Body:

```
{
    "message": "Password is reset",
    "data": {
        "username": edwinlamtk,
        "isSuccess": true
    }
}
```

9.1.4 Delete Account

- Delete a user account

  Description: Check if the input data format is correct, the account is active and the password is correct. After checking, change "is_active" of the user to false in the database ACCOUNT.User_Account.

  PUT : `https://flowchatbackend.azurewebsites.net/api/Account/deleteAccount`

  Request Body: set one of the "username" and "email" to be null

```
{
    "username": "edwinlamtk",
    "email": null,
    "password": "edwinlamtk"
}
```

Response Body:

```
{
    "message": "Account is deleted",
    "data": {
        "isSuccess": true
    }
}
```

9.1.5 Expected Exceptions

The list of expected exceptions:

Message	Description
Invalid username format	The given username is null, empty or contains ";".
Invalid email format	The given email is null, contains ";" or " ", or does not contains "@".
Username is not unique	The given username is same as other usernames in the database ACCOUNT.User_Account.
Email is not unique	The given email is same as other emails in the database ACCOUNT.User_Account.
Too many parameters	Login API and Delete Account API require one of two "username", "email" keys, but given two.

Account is active	The user account with the given username or email is active in the database ACCOUNT.User_Account.
Account is not active	The user account with the given username or email does not exist or is not active in the database ACCOUNT.User_Account.
Account is active but password is not correct	The given password does not match with the user password set in the database ACCOUNT.User_Account.
Key Type not match	The length of given license key is not 16, or the length of given authentication code is not 6.
Key not match	The given license key or authentication code does not match with any key in the database ACCOUNT.License.
Key is expired	The current time of the given license key or authentication code exceeds its expired time at the first time.
Key is not available	The given license key or authentication code is used or had already been expired.

9.2 Discussion Forum

9.2.1 Post Preview List

- Get the latest list of post previews for a user

  Description: Get the required number of posts from the database FORUM.Post, sorted by the post updated time.

  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/getLatestList`
  Query Parameters: `postCountFrom`, `postCountTo`, `userId`

  Response Body:

```
{
    "message": "The latest post preview list is returned",
    "data": {
        "isSuccess": true,
        "posts": [
            {
                "postId": "xxx",
                "username": "xxx",
                "title": "xxx",
                "description": "xxx",
                "image": "xxx",
                "tag": "xxx",
                "likeCount": "xxx",
                "dislikeCount": "xxx",
                "commentCount": "xxx",
                "updatedAt": "xxx"
            }
        ]
    }
}
```

- Get the recommended list of post previews for a user

  Description: Get the required number of posts from the database FORUM.Post, filtered by the recommendation system of the user.

GET : `https://flowchatbackend.azurewebsites.net/api/Forum/getRecommendedList`
Query Parameters: `postCountFrom`, `postCountTo`, `userId`

Response Body:

```
{
    "message": "The recommended post preview list is returned",
    "data": {
        "isSuccess": true,
        "posts": [
            {
                "postId": "xxx",
                "username": "xxx",
                "title": "xxx",
                "description": "xxx",
                "image": null,
                "tag": "xxx",
                "likeCount": "xxx",
                "dislikeCount": "xxx",
                "commentCount": "xxx",
                "updatedAt": "xxx"
            }
        ]
    }
}
```

- Get the following list of post previews for a user

  Description: Get the required number of posts from the database FORUM.Post, filtered by the following system of the user.

  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/getFollowingList`
  Query Parameters: `postCountFrom`, `postCountTo`, `userId`

  Response Body:

```
{
    "message": "The following post preview list is returned",
    "data": {
        "isSuccess": true,
        "posts": [
            {
                "postId": "xxx",
                "username": "xxx",
                "title": "xxx",
                "description": "xxx",
                "image": null,
                "tag": "xxx",
                "likeCount": "xxx",
                "dislikeCount": "xxx",
                "commentCount": "xxx",
                "updatedAt": "xxx"
            }
        ]
    }
```

```
}
```

9.2.2 *Post and Comment*

- Get all information of the post by a post ID

  Description: Get the full post content and the attached comment content.

  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/getPostContent`
  Query Parameters: `postId`

  Response Body:

```
{
    "message": "The post content is returned",
    "data": {
        "isSuccess": true,
        "post": {
            "postId": "1",
            "username": "xxx",
            "title": "xxx",
            "content": "xxx",
            "image": "xxx",
            "tag": "xxx",
            "likeCount": "xxx",
            "dislikeCount": "xxx",
            "commentCount": "3",
            "updatedAt": "xxx",
            "comments": [
                {
                    "postId": "2",
                    "username": "xxx",
                    "content": "xxx",
                    "image": "xxx",
                    "tag": "xxx",
                    "likeCount": "xxx",
                    "dislikeCount": "xxx",
                    "commentCount": "0",
                    "updatedAt": "xxx",
                    "comments": null
                },
                {
                    "postId": "3",
                    "username": "xxx",
                    "content": "xxx",
                    "image": null,
                    "tag": "xxx",
                    "likeCount": "xxx",
                    "dislikeCount": "xxx",
                    "commentCount": "1",
                    "updatedAt": "xxx",
                    "comments": [
                    {
                        "postId": "4",
```

```
                        "username": "xxx",
                        "content": "xxx",
                        "image": "xxx",
                        "tag": "xxx",
                        "likeCount": "xxx",
                        "dislikeCount": "xxx",
                        "commentCount": "0",
                        "updatedAt": "xxx",
                        "comments": null
                    }
                ]
                }
            ]
        }
    }
}
```


- Create a new post or comment

  Description: For a root post, "attachTo" is set to 0. For any comment, "attachTo" is set to responding postId of the post or comment.

  POST : `https://flowchatbackend.azurewebsites.net/api/Forum/createPost`

  Request Body:


```
{
    "userId": "xxx",
    "title": "xxx",
    "content": "xxx",
    "tag": "xxx",
    "image": "xxx",
    "attachTo": "xxx"
}
```


  Response Body:


```
{
    "message": "A new post/comment is created",
    "data": {
        "isSuccess": true
    }
}
```


- Update a post or comment created by the user

  Description: For a root post, "attachTo" is set to 0. For any comment, "attachTo" is set to responding postId of the post or comment.

  PUT : `https://flowchatbackend.azurewebsites.net/api/Forum/updatePost`

  Request Body:


```
{
    "postId": "xxx",
    "userId": "xxx",
    "title": "xxx",
```

```
    "content": "xxx",
    "tag": "xxx",
    "image": "xxx",
    "attachTo": "xxx"
}
```

Response Body:

```
{
    "message": "The post/comment is updated",
    "data": {
        "isSuccess": true
    }
}
```

- Delete a post or comment created by the user

  Description: Check if the post is created by the user. After checking, change "is_active" of the post to false in the database FORUM.Post.

  PUT : `https://flowchatbackend.azurewebsites.net/api/Forum/deletePost`

  Request Body:

```
{
    "postId": "xxx",
    "userId": "xxx"
}
```

  Response Body:

```
{
    "message": "The post/comment is deleted",
    "data": {
        "isSuccess": true
    }
}
```

9.2.3 Post Comment Interaction

- Block a post or comment created by other user

  Description: Save the relationship of user ID and blocked post ID in the database FORUM.Block.

  POST : `https://flowchatbackend.azurewebsites.net/api/Forum/blockPost`

  Request Body:

```
{
    "postId": "xxx",
    "userId": "xxx"
}
```

  Response Body:

```
{
    "message": "The post/comment is blocked",
    "data": {
```

```
        "isSuccess": true
    }
}
```

- Like and dislike a post or comment

  Description: Save the relationship of user ID and liked / disliked post ID in the database FORUM.Like / FORUM.Dislike.

  POST : `https://flowchatbackend.azurewebsites.net/api/Forum/likePost`

  Request Body:

```
{
    "postId": "xxx",
    "userId": "xxx",
    "action": "like" / "dislike"
}
```

  Response Body:

```
{
    "message": "The post/comment is liked/disliked",
    "data": {
        "isSuccess": true
    }
}
```

- Un-like and un-dislike a post or comment

  Description: Delete the relationship of user ID and liked / disliked post ID in the database FORUM.Like / FORUM.Dislike.

  DELETE : `https://flowchatbackend.azurewebsites.net/api/Forum/unlikePost`

  Request Body:

```
{
    "postId": "xxx",
    "userId": "xxx",
    "action": "unlike" / "undislike"
}
```

  Response Body:

```
{
    "message": "The post/comment is un-liked/un-disliked",
    "data": {
        "isSuccess": true
    }
}
```

- Generate a share link for a post

  Description: Generate an URL for the post.

  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/sharePost`
  Query Parameters: `postId`

  Response Body:

```
{
    "message": "The share link of the post is returned",
    "data": {
        "isSuccess": true,
        "link": "xxx"
    }
}
```

- Search information by keywords

  Description: Return any relevant post preview list and the username list to the keywords.

  GET : `https://flowchatbackend.azurewebsites.net/api/Forum/search`
  Query Parameters: `keyword`

  Response Body:

```
{
    "message": "The search results are returned",
    "data": {
        "isSuccess": true,
        "posts": [
            {
                "postId": "xxx",
                "username": "xxx",
                "title": "xxx",
                "description": "xxx",
                "image": null,
                "tag": "xxx",
                "likeCount": "xxx",
                "dislikeCount": "xxx",
                "commentCount": "xxx",
                "updatedAt": "xxx"
            }
        ]
        "usernames": [
            "user1",
            "user2"
        ]
    }
}
```

9.2.4 Expected Exceptions

The list of expected exceptions:

Message	Description

9.3 Direct Message

9.3.1 Send Message

- Send a text message to the following users

Description: Store the text message in the database.

POST `https://flowchatbackend.azurewebsites.net/api/DirectMessage/sendText`

Request Body:

```
{
    "receiverId": 1,
    "textMessage": "How are you?"
}
```

Response Body:

```
{
    "message": "Message is sent",
    "data": {
        "isSuccess": true,
        "sentTime": "2025-03-04 10:00:00"
    }
}
```

- Send an image or a voice message to the following users

  Description: Store the media as binary in the database.

  POST `https://flowchatbackend.azurewebsites.net/api/DirectMessage/sendMedia`
  Query Params: `receiverId`, `file`

  Response Body:

```
{
    "message": "Message is sent",
    "data": {
        "isSuccess": true,
        "sentTime": "2025-03-04 10:00:00"
    }
}
```

9.3.2 Receive Message

- Get the message history with a following user

  Description: Get the limited history from the database with the given user id. Only messages before the date time of the given message id will be shown.

  GET `https://flowchatbackend.azurewebsites.net/api/DirectMessage/getMessageHistory`
  Query Params: `withUserId`, `limit`, `beforeMessageId`

  Response Body:

```
{
    "message": "Get history successfully",
    "data": {
        "history": [
            {
                "messageId": 1,
                "senderId": 2
```

```
                "receiverId": 1,
                "content": "How are you?",
                "sentAt": "2025-03-04 10:00:00",
                "readAt": null,
                "mediaType": null,
                "url": null
            },
            {
                "messageId": 2,
                "senderId": 2
                "receiverId": 1,
                "content": null,
                "sentAt": "2025-03-04 10:10:00",
                "readAt": null,
                "mediaType": "image/png",
                "url": "https://flowchatbackend.azurewebsites.net/
                    api/DirectMessage/getMedia/2"
            }
        ]
    }
}
```


- Get single message, which may be image or voice message

  Description: Get the required binary data from the database.

  GET `https://flowchatbackend.azurewebsites.net/api/DirectMessage/getMedia/{messageId}`
  Path Params: `{messageId}`

  Response Body: For the case which the message is an image:

```
Binary data of the image
```

  For the case which the message is voice message:

```
Binary data of the audio
```

9.3.3 Read Message

- Update the "read_at" time of the messages

  Description: Check if the read time of message is null. Update read time if true.

  PUT `https://flowchatbackend.azurewebsites.net/api/DirectMessage/updateMessagesReadTime`

  Request Body:

```
{
    "messagesList": [1, 3]
}
```

  Response Body:


```
{
    "message": "Messages read_at time are updated",
    "data": {
        "isSuccess": true
```

```
    }
}
```

9.3.4 Get Unread Message Count

- Get the count of total unread messages from all following users

  Description: Get the count of read time that is not null.

  GET `https://flowchatbackend.azurewebsites.net/api/DirectMessage/totalUnreadMessageCount`

  Response Body:

```
{
    "message": "Success",
    "data": {
        "totalUnreadMessageCount": 5
    }
}
```

- Get the count of unread messages from each following user

  Description: Get the count of read time that is not null for each following user.

  GET `https://flowchatbackend.azurewebsites.net/api/DirectMessage/unreadMessageCountFromEachUser`

  Response Body:

```
{
    "message": "Success",
    "data": {
        "unreadMessageCount": [
            {"withUserId":2, "count": 5},
            {"withUserId":3, "count": 1}
        ]
    }
}
```

9.3.5 Expected Exceptions

The list of expected exceptions:

Message	Description
Invalid user id	The given JWT is valid, but the user id is not authorized to perform the action.