SYSC 4907

Come and Chat (C&C): Fourth Year Engineering Progress Report Draft 1

Carleton University

Group Members:

Yunzhou Liu 101027110

Shizhong Shang 101115304

Zirui Qiao 101100225

Supervisor:

Lynn Marshall

03 November 2022

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

Contents

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

1. Introduction

1.1 Purpose

Instant messaging is an efficient way to deliver information, and today, instant messaging software has become an integral part of our lives. For some of the most famous instant messaging software, such as Discord, WhatsApp and Facebook Messenger, we enjoy the convenience they offer while occasionally complaining about their shortcomings. This has led us to take a keen interest in such software.
Based on the instant messaging software that we have studied, in this project, we will build an imitated, web-based instant messaging software called Come and Chat (C&C). This proposal will outline the proposed plan and objectives to complete a working prototype of the project in early January 2023.

1.2 Scope

The C&C will provide mechanisms for signing up/in users, searching users, adding/deleting friends, sending/receiving messages to/from friends, building group chats, recording chat history, and allowing users to post messages as their status, allowing users' friend to see users' status.
More functions could be added after the functions above are implemented. Customized parts are always considered after the procedures above.

2. Objectives

All requirements listed in each subsection of section 2 are in priority order, which means the requirement with the highest priority order in 2.1.1 is shown as the first requirement in section 2.1.1.

The subsections in section 3 are also in priority order; the subsections int front are the prerequisites of the following subsections. For example, a user cannot search for friends (2.1.2 - 1) before register in (2.1.1 - 1) the C&C system.

2.1 Functional Requirements

2.1.1 Identity Functions

1. Users provide email, nickname and password to sign up for the system.
2. Users provide their nickname and password to sign in.
3. Users can modify their profiles, including changing nickname, password, email, address, avatar, phone number, and birthday.
4. Users can log out of C&C.

2.1.2 Friend Functions

1. Users search for other users by nickname.
2. A user can apply to be a friend of another user.
3. A user can approve/reject a friend request from another user.
4. A user can view all his/her friends in a friend list.
5. A user can delete a friend.
6. A user can mute a friend so that he/her will not receive any message from the friend until unmuted or enters the chat room of the friend.
7. A user can hide a friend so that he/her will not be able to find the friend from friends list.
8. A user can unset a friend so that the friend is unmuted and unhidden.
9. A user can show all friends so that all friends, including hidden friends, are shown in friends list.
10. A user can search his/her friends by friend’s nickname.
11. A user can view a friend’s information including: avatar, nickname, email and statuses.

2.1.3 Group Functions

1. A user can create a group by providing a proper group name; group notice, avatar and group

initial members are optional.
2. A user in a group can increase the member of a group by inviting friends.
3. A user can view all groups he/her joined in a group list.
4. A user can leave a group.
5. A user can view all the members of the group he/her has joined.
6. Users can view and change the information of the groups they have joined.
7. A user can mute a group so that he/her will not receive any message from the group until unmuted or enters the chat room of the group.
8. A user can hide a group so that he/her will not be able to find the group from group list.
9. A user can unset a group so that the group is unmuted and unhidden.

2.1.4 Chat Functions

1. Chat messages can only be sent to/received from a user's friends and groups.
2. Users can send texts and images in chat room as a chat message.
3. Only the user themselves and the friend/group members can view messages in a chat room.
4. All users in a group chat can view messages from all users in the group chat.
5. A user can view the chatting histories of chats and group chats.
6. A friend receiving a chat message and a user sending a chat message should happen almost simultaneously.
7. A user will be able to know whether or not unread messages have been sent by a group or friend without entering a chat room.

2.1.5 Status Functions

1. A user can post text and image messages as his/her status.
2. A user can only view his/her own and friends' statuses.
3. A user can 'like'/'unlike' a status and view the amount of 'like's of a status.
4. A user can write text as comments to a status.
5. A user can view all comments of a status.
6. A user can delete his/her own statuses.

2.1.6 Administrator Functions

1. An administrator of C&C can log into C&C by providing the correct username and password.
2. An administrator of C&C can view statistics including:
   a) total number of users,
   b) number of online users,
   c) total number of statuses,
   d) total number of comments, and
   e) total number of chat messages.
3. An administrator can search a user by nickname or user id.
4. An administrator can block/unblock a user, and the blocked user cannot log in to the C&C system until unblocked.

2.1.7 System Scheduling

1. The system shall use the MVC model.

2.1.8 Optional Functions: Camp Grouping

1. There exist n camps in C&C set by the administrator
2. Every client must join 1 and only 1 camp during registration.
3. Clients within the same camp, if they are friends, can chat with each other without limits.
4. Chat messages sent from different camps will be encoded, which means one cannot chat with clients in other camps.

5. There are periodic events that allow 2 random camps to understand each other's messages over time. These events are triggered by time.
6. Clients cannot switch camp.

2.2 Page Requirements

1. The list shows all friends' chats, and group chats should exist the whole time after signing in.
2. Signing in/up page should be a separate page from the chatting pages.
3. All messages sent by a user should appear with their avatar simultaneously.

2.3 Design Constraints

1. Performance should be reasonably fast.
2. The system shall be usable at least on Chrome.
3. The system shall support PC and mobile.
4. The system must comply with the relevant privacy legislation.

2.4 Software Quality Attributes

1. Users can only see messages in chats between themselves and their friends and the group chat they joined.
2. Users can only view the history of messages that they can see.
3. The system can be remotely accessible to all users.
4. The system shall protect all users' information.

2.5 Other Requirements

1. The time plan can be divided into 2 parts, version 1 and version 2. Version 1 will be completed before Jan 22, 2023, and version 2 will be completed before Mar 14, 2023.
2. The hard deadline for the system is the end of April of next year for project completion.
3. One or many servers are required for this project. Specific requirements are still in process of investigation.

3. Changes to Objectives

In the section 2.1 - Functional requirements, the original subsections: 2.1.1 Identity and Relationship Functions and 2.1.2 Communication Functions are divided into 5 different subsections (2.1.1 - 2.1.5). The original vague requirements are described more clearly, making the entries more numerous. After intense discussion, we decided to focus on text and image chat messages and statuses for the time being; the C&C system is so complex that we thought we should wait until the basic functionality had been designed and implemented before looking at other type of chat messages and statuses. Optional functions are added as a new sub-section, where an original designed rule is described. These features are designed to improve the fun and originality of the software, as the software can function even without this part, so this part of the requirement is classified as an optional requirement.

A new subsection(2.1.6) for administrator functions is added in functional requirement section. We realized that as social software, administrators are needed to manage users' improper actions. Administrators should be able to block users who misbehave without violating the privacy of any user. This is why the only actions the administrator can perform on the user are searching, blocking and unblocking.

We would like to take this opportunity to observe users' behaviour more visually and without invading their privacy, and to produce statistics. This is the reason of adding requirement 2 in subsection 2.1.6.

To add original features and make the program more interesting, a new subsection(2.1.8) for Optional

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

functions is added in the functional requirement section. The sub-section describes the rule of Camp Grouping, which includes technical challenges such as identity creation, rights management, chat encryption and decryption of information and timed tasks. These functions are considered optional and will only be implemented after all other functions are implemented and tested.

In section 3.2, the only change is that the requirement of viewing all group member's avatar list while in a group chat room is removed. Since users can neither mute/hide/unset a non-friend user nor chat with the user, it is unnecessary to list all group member the whole time. This is the reason we remove the requirement.

The rest sections remain unchanged.

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

4. Design

The design methodology learned from SYSC 3120 (Software Requirements Engineering) was very useful in the design of this project. As a result, the design process followed the requirements analysis, modelling and specification learned in SYSC 3120.

4.1 UseCase Diagram

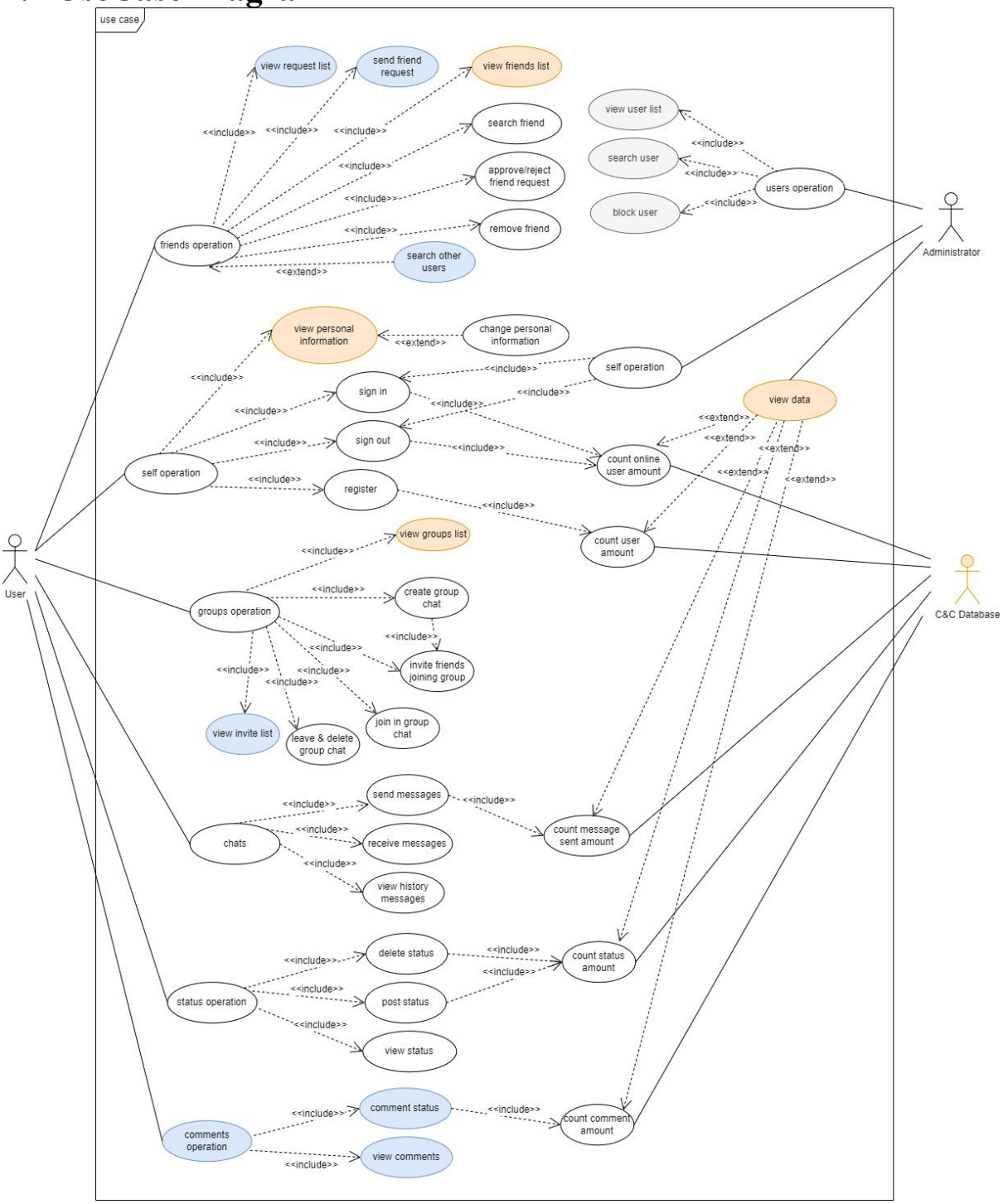

Figure 4.1.1: Use Case Diagram

In the use case diagram, we made three modifications. Use cases with white as the background colour are designed when they are created. Use cases with an orange background are added or modified during the first modification. Use cases with blue background are added or modified during the second modification. Use cases with grey background are added or modified in the third modification.

There are three objects in the use case diagram: user, administrator and database.

Users mainly have six operations. The first case is that the user operates the friend system. Users can add, delete and find users at will. The second situation is that users operate the self information system. Users can freely change their personal data, register their accounts, log in or log out of their accounts. The third situation is that users operate the group system. Users can create groups at will, invite friends to join their own groups, leave groups, and accept group invitations from friends. The fourth situation is that users can manage their own chat system. Users can send, receive and delete chat messages with their friends or groups. The fifth situation is that users can manage their own status system. Users can create, delete, and view previous statuses. The sixth case is that users can add their comments after their friends or their status.

The administrator mainly performs three operations in this software. First, the administrator can log in or out of the account freely. Secondly, the administrator has the right to view the list of all users and lock other users. Finally, the administrator can view the database information, such as the number of registrations and online accounts.

4.2 UseCase Descriptions

user case description 2st draft.docx

methodology

The use case description file records the information of each operation in the use case diagram in detail.

The use case description file records the information of each operation in the use case diagram in detail. Each operation is described by the following attributes.

Use case name
Brief description. A simple sentence that explains the meaning of the use case name.
Primary actor. The main roles involved in the use case
Secondary actor. Secondary actor participating in the use case
Precondition. You want to implement the precondition of the use case
Dependency. Derivative operation of use case
Generalisation
Basic flow. Basic process of realizing use cases
Specific alternative flow1. The response of the system when exceptional operations occur
Specific alternative flow2. The system's response when exceptional operations occur
Global alternative flows
Bounded alternative flows
Special requirements

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

4.3 Flow Diagrams

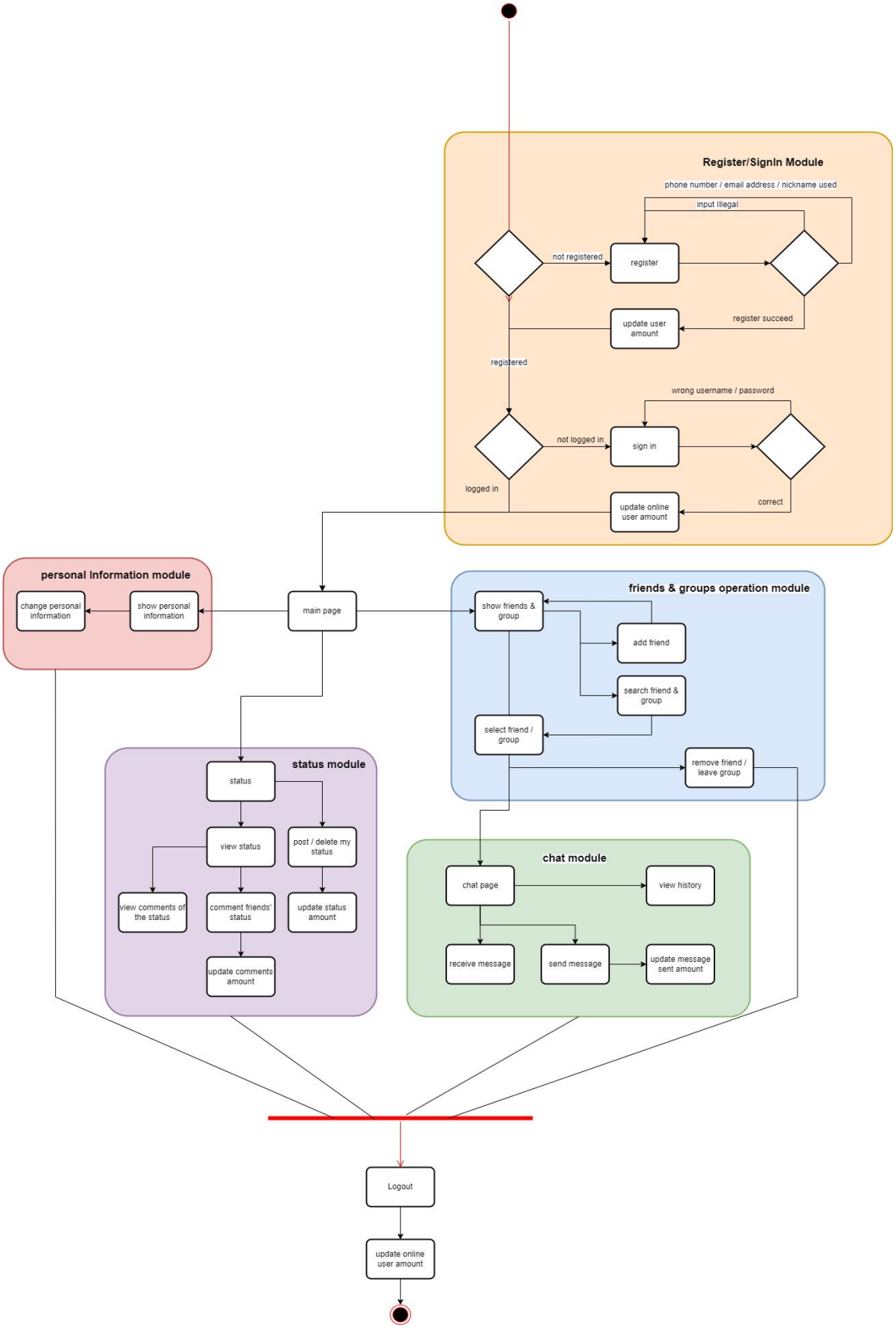

Figure 4.3: User Flow Diagram

Flow Diagram shows the operation of a user from logging in to logging out of the account under normal process. First, the user enters the website page to log in. If there is an account, the user will enter the main

page. If there is no account, the user will enter the registration page. The user enters the login page after successfully registering an account. Entering the correct user name and password will cause the user to enter the main page or enter the login page repeatedly after entering the wrong user name or password. After entering the homepage, users can choose to enter the personal information page, friends and groups page, chat page and status page. Users can enter the logout page to perform logout operation no matter which page they are on.

When the user enters the personal information page, the user can view and modify the user information.
When users enter the friends and groups page, they can add, delete, and view friends and groups.
When users enter the status page, they can add, delete, modify, and view the status.
When users enter the chat page, they can view past chat messages and send chat messages.

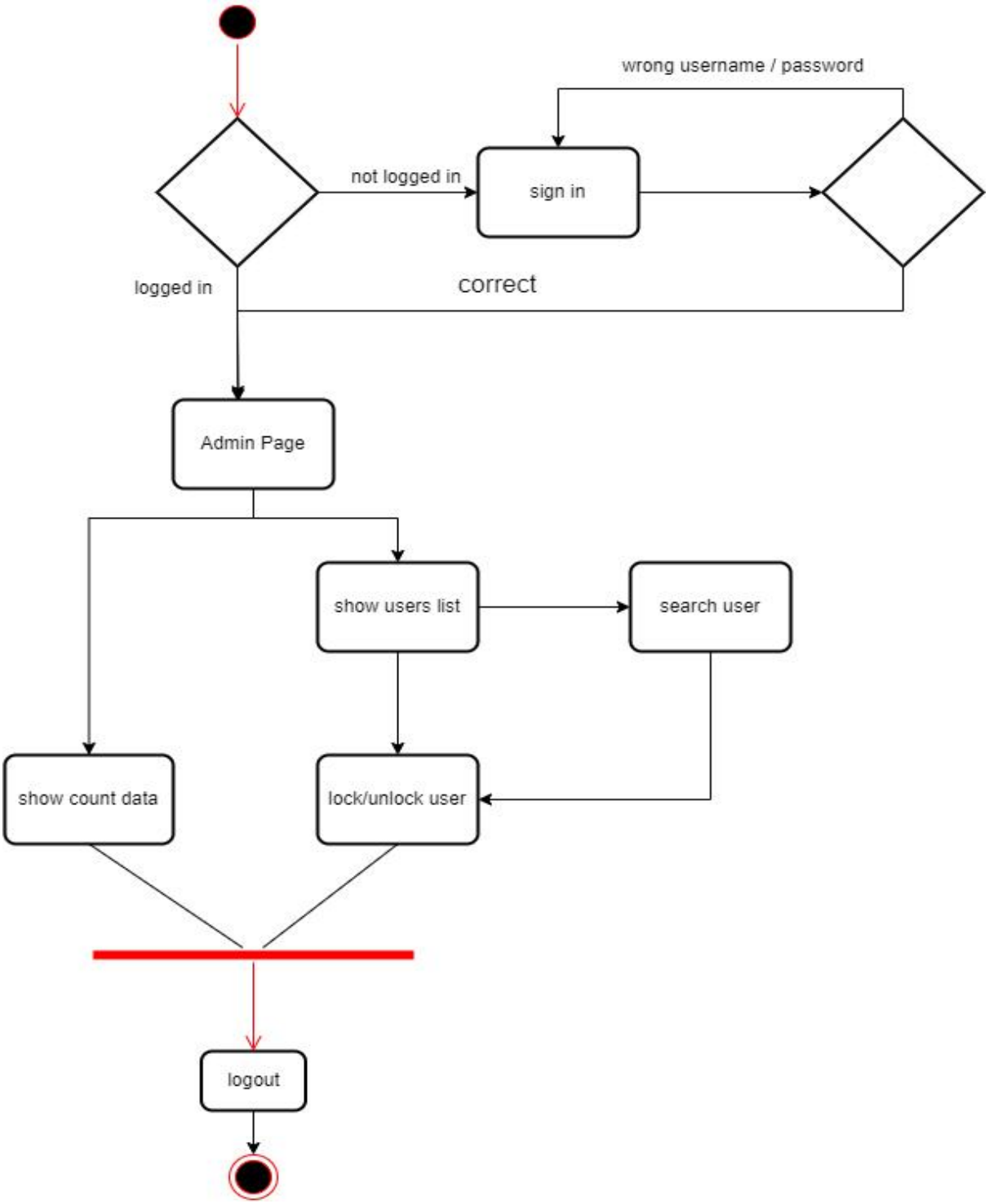

Figure 4.3: Administrator Flow Diagram

The above figure shows the administrator flow chart. The process of the administrator when logging in the page is the same as that of ordinary users. The difference is that the administrator will enter the administrator page after passing the login page. On the administrator page, the administrator can view the user's data, such as the number of online users, the total number of public registrations, and then enter the login page. The administrator can also select to display all users and block users, and then enter the login page.

4.4 Back-end

4.4.1 Module UML

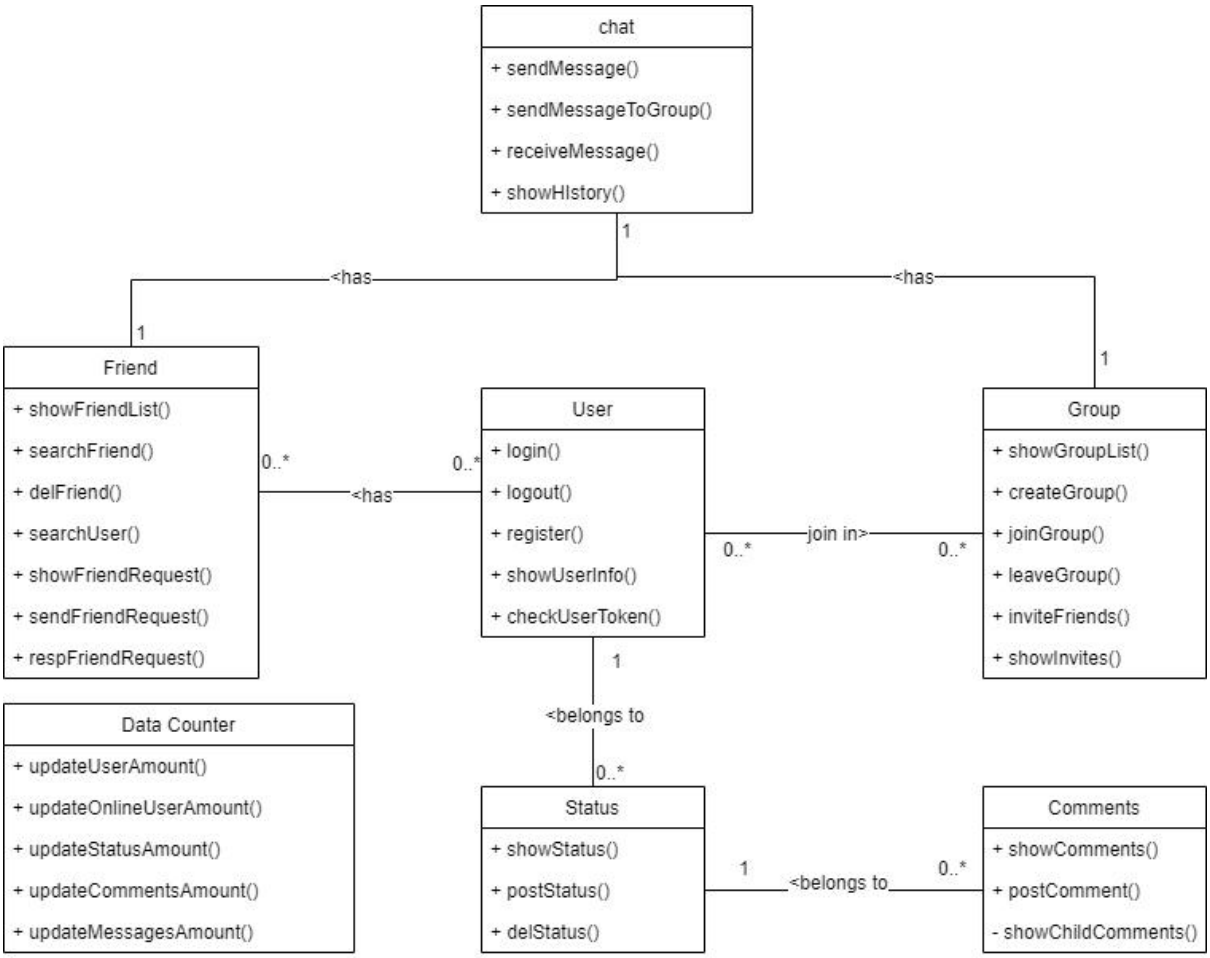

Figure 4.4.1: Back end simple module diagram

The figure above simply describes the composition of the back-end program. Each module corresponds to the function in the design. For example, in the user module, users can log in, log out, register, view personal information and other functions through the user module.

The user module represents the individual user, who can log in, log out, register, etc. The friend module represents the association between users. A user can have multiple friends. Users can modify, add and delete friends through the friend module. The group module represents another relationship between users. A group is similar to a container, which can contain multiple users. Users can create a new group, or join, leave or invite other users who are not in the group to join the group. The status module represents the comments published by users, and users can modify, publish and delete the status. The comment module is used to talk about the status. Users can publish comments after their or others' status. Chat module represents all information exchanges between users. The data counter module can count the data of other modules, such as the number of users and the number of status.

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

4.4.2 ER Diagram

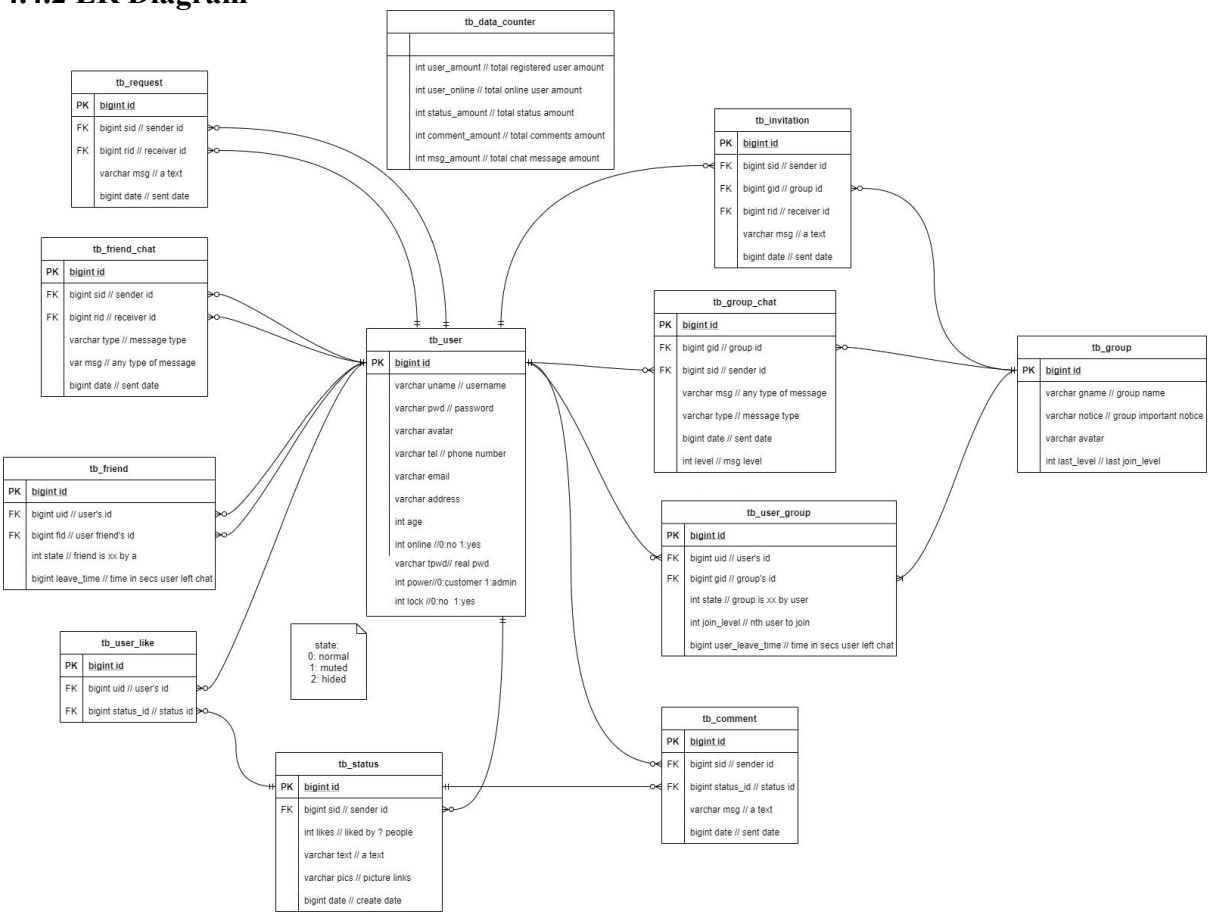

Figure 4.4.2: Database structure diagram

The above figure describes the names and corresponding attributes of all tables in the database. The description of each attribute is labelled by annotations.

There are 12 tables in the database.The tb_user stores various user information. The online attribute indicates whether the user is online. The pwd attribute represents the user's encrypted password. Tpwd represents the original password of the user. Lock indicates the status of the user's account and whether it is available. The tb_friend table stores friend relationships. The state attribute indicates whether the user is blocked by his friends in a friend relationship. The leave_time attribute indicates the time when the user last sent the message. tb_ friend_ Chat table stores chat information between users and friends. The type attribute indicates whether the user is sending a picture or text. The tb_ request table stores the invitation to add friends sent by users. The tb_ data_ counter table stores various information about the system. The tb_ invitation table stores the group joining invitations sent by users. The tb_group table stores all group information. The last_level attribute indicates the last member to join the group. The tb_ group_chat table stores the chat information of members in the group. The tb_ user_ group table stores group member information. The tb_ user_ like table stores users' likes of status. The tb_ status table stores all status information. The likes attribute indicates how many user tags like this status. tb_ Comment stores all comment information.

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

4.4.3 Backend UML

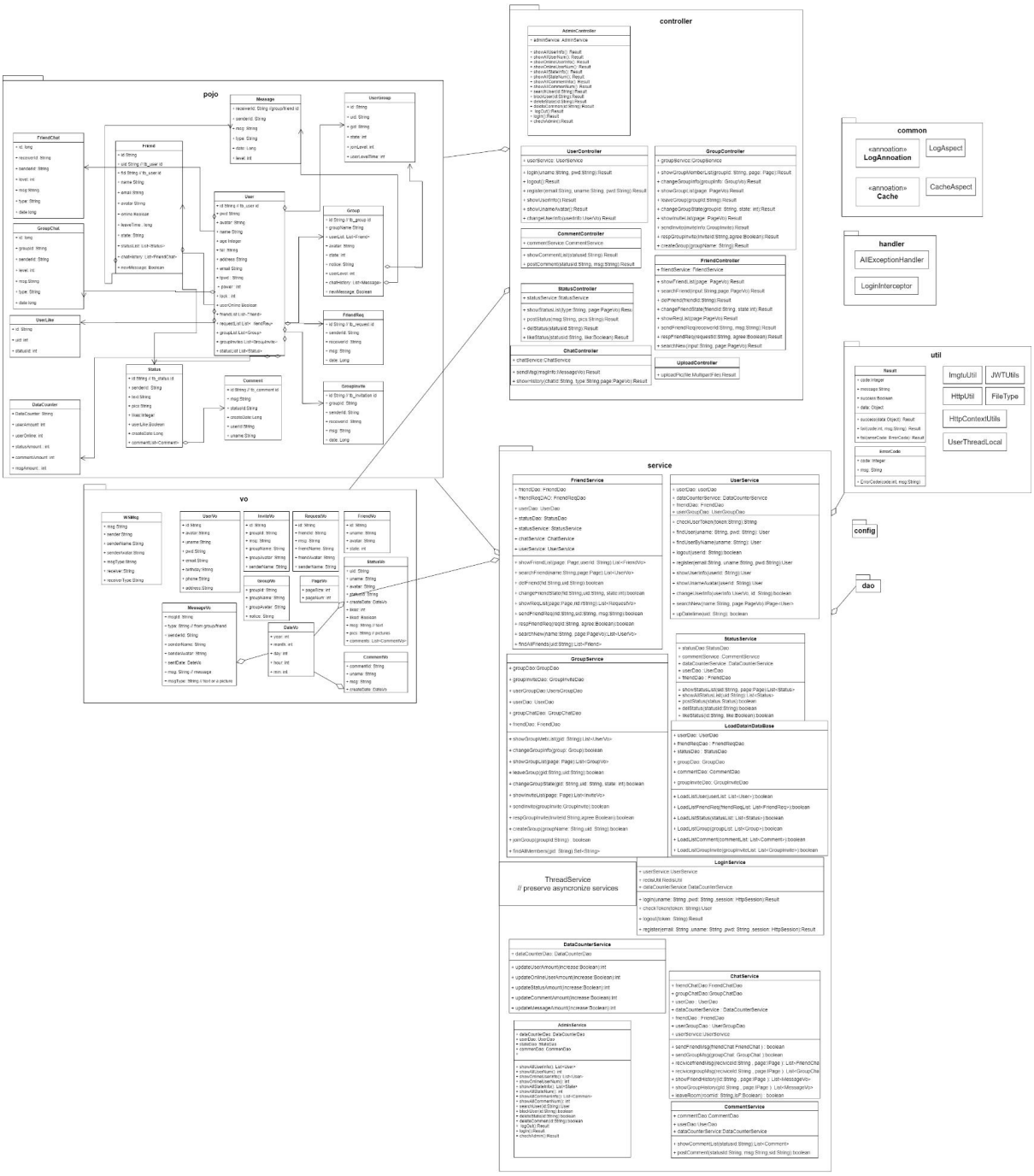

Figure 4.4.2: Detailed design drawing of back end

The figure above describes the specific design requirements of the control layer, service department, instance, tool layer and setting layer in the back-end.

There are 9 packages in the backend system. Each package contains some classes. Pojo packages represent entity classes. These entity classes correspond to the tables in the database. The user class contains user information, such as username and password. When the user performs login and other operations on the front end, the user class will be called to obtain information. The friend class contains the friend relationship between users. The friendchat class represents the entity class that carries information when users send information to friends. The friendreq class represents the class that hosts the request for adding friends when the user sends the request to an unknown user. In the group class, you can find the information of the group

created now. The usergroup class contains information about the members of each group. The groupchat class contains chat messages between members of each group. The groupinvite class is an invitation sent by a group member to other users who are not members of the group. The status class is the status information published by the user. The comment is the user's evaluation published after the status. The message class is a summary of different chat messages. The datacounter class is the statistics of all existing information.

Dao packages are developed through annotations. The function of dao is to contact the database and modify the records in the database through the classes in pojo.

Vo packet is the format of transmitting information to the front end. The classes in the pojo package carry information. These classes can be converted into classes in the vo package and then sent to the front end. The service package is the service layer. The classes in the package are concrete operations. The friendservice class calls multiple dao layer classes to operate the database. Finally, modify the friend table in the database. The userservice class calls userdao, frienddao, usergroupdao, and datacounterdao to perform login, logout, and other operations. The groupservice calls the dao layer to modify the group information. Statusservice calls dao to modify the status information. Loginservice mainly supports user login and logout operations. Commentservice modifies the information in the comment table. Chatservice is used to manage chat information of all users. Adminservice operates on behalf of the administrator.

The controller package is mainly oriented to the front end. The classes in the package implement different operations by calling the service layer. The admincontroller calls adminservice to perform various operations on the user, status, and comment. The usercontroller calls the userservice to implement the operation. Friendcontroller calls friendservice to implement the operation on friend. Chatcontroller operates the user's chat by calling chatservice. The commentcontroller calls commentservice to operate on the comment.

The config package is used to include various configuration classes. The structure rules of these configuration settings information. The util package contains different tool classes for other classes to use. The classes in the handler package set system errors.

In back-end development, we follow the software SOLID. SOLID is composed of five design principles: single responsibility principle, open/close principle, internal replacement principle, interface isolation principle and dependency inversion principle.

Single responsibility principle

A class is only responsible for completing one responsibility or function. In other words, instead of designing large and comprehensive classes, we should design classes with small granularity and single function. To put it another way, a class contains two or more unrelated functions, so we say that its responsibilities are not simple enough, and it should be divided into multiple classes with more single functions and finer granularity.

Opening and closing principle

Software entities (modules, classes, methods, etc.) should be "open to extension and closed to modification".

Adding a new function should be to extend code (add modules, classes, methods, etc.) on the basis of existing code, rather than modify existing code (modify modules, classes, methods, etc.).

Liskov Substitution Principle

The object of subtype/derived class can replace the object of base/parent class in the program, and ensure that the logical behaviour of the original program remains unchanged and the correctness is not damaged.

Interface Segregation Principle

The client should not be forced to rely on interfaces it does not need. The "client" can be understood as the caller or user of the interface.

Dependency Inversion Principle

The dependency inversion principle is also called the dependency inversion principle. This principle is similar to the inversion of control, and is mainly used to guide the design at the framework level. High level modules do not rely on low level modules, they all rely on the same abstract. Abstraction does not depend on specific implementation details, and specific implementation details depend on abstraction.[1]

4.5 Front-end

4.5.1 Prototype

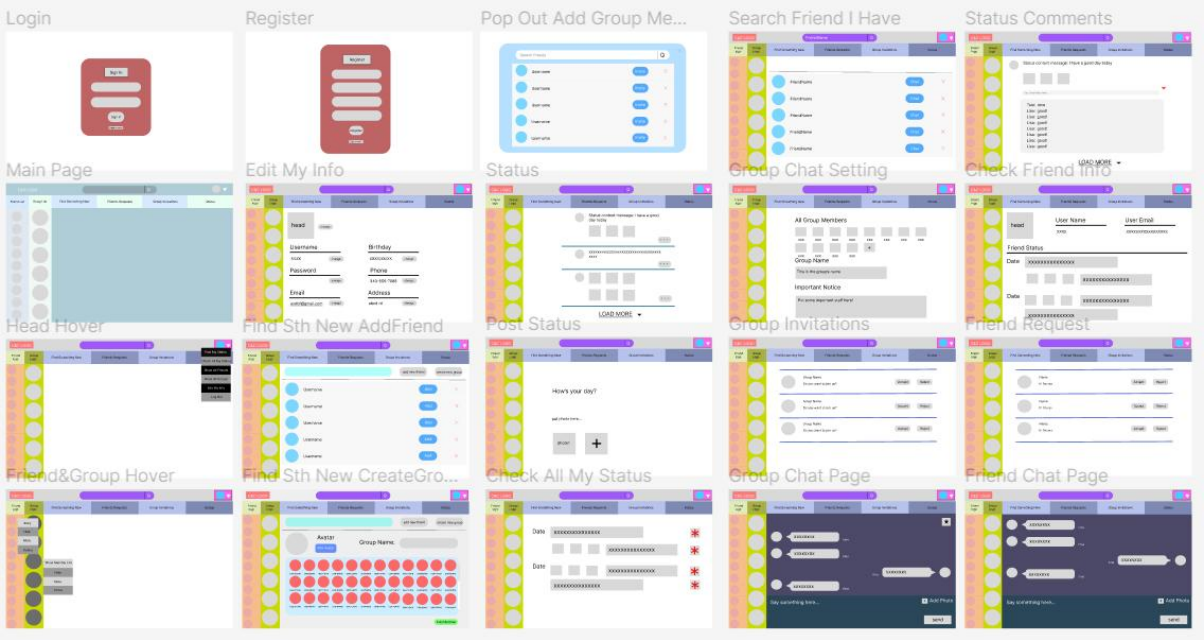

Figure 4.5.1.1: Overall Client Side Prototype C&C Project

The figure 4.5.1.1 above is the user side overall prototype for the C&C project, this figure shows all twenty pages that will be shown to the user while using the C&C system. There is a main page that has a big difference of color with the other pages is because the team is still considering about the color set of this system when it is shown to the users.

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

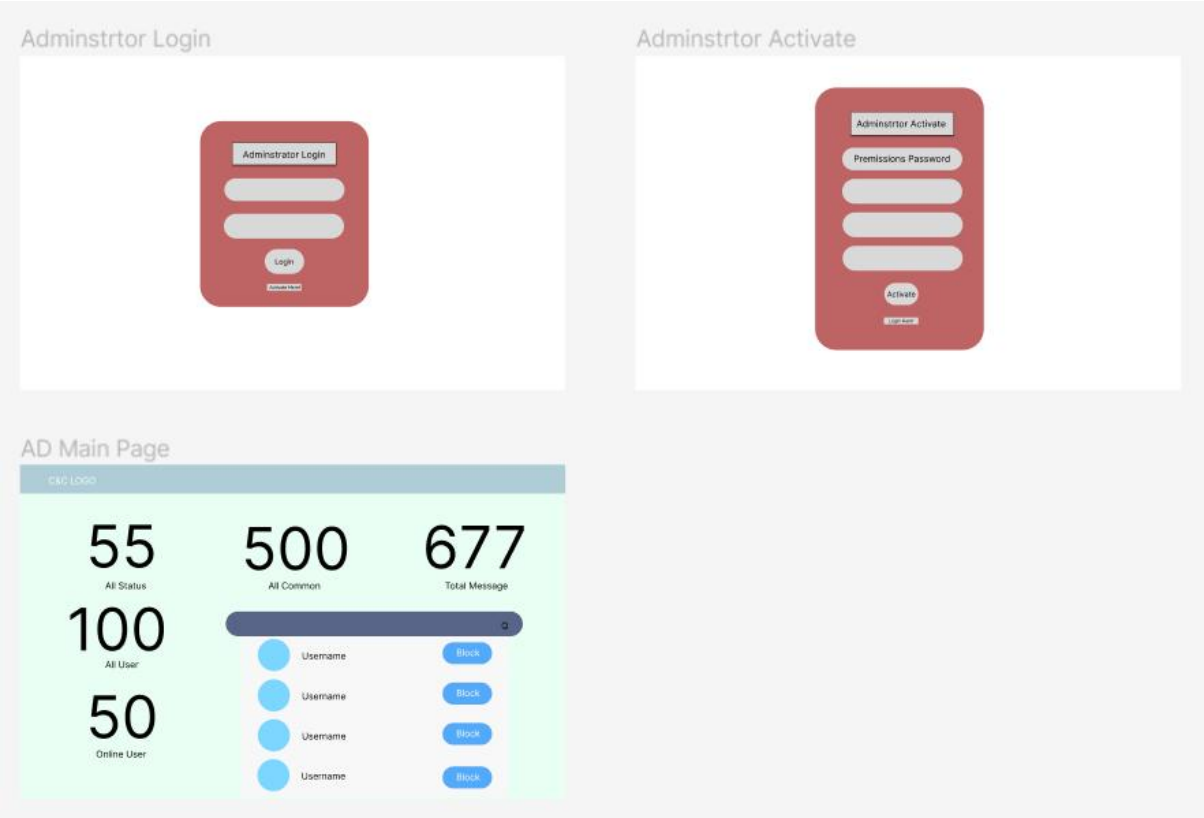

Figure 4.5.1.2: Overall Administrator Side Prototype C&C Project

The figure 4.5.1.2 above is the administrator side overall prototype for the C&C project, this figure shows all three pages that will be shown to the administrator while using the C&C system. The team is deciding about if new administrator is allowed to register or just fix it.

For more information on all the page structures, refer to 4.5.2 Page Structure Document

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

4.5.2 Page Structure Document

All the page designs are made in the website Figma[1], this design website is convenient to use and easy to make changes, so the team decide to use it as design support.

Figure 4.5.2.1 - “Login” page

“Login” page (Figur 4.5.2.1) is used to let users login to the system, user’s name and password are needed. On this page, the user can either choose to login by clicking the “Sign In “ button or if the user is new to the system, this page can jump to the “Register” page by clicking the “Register here!” button.

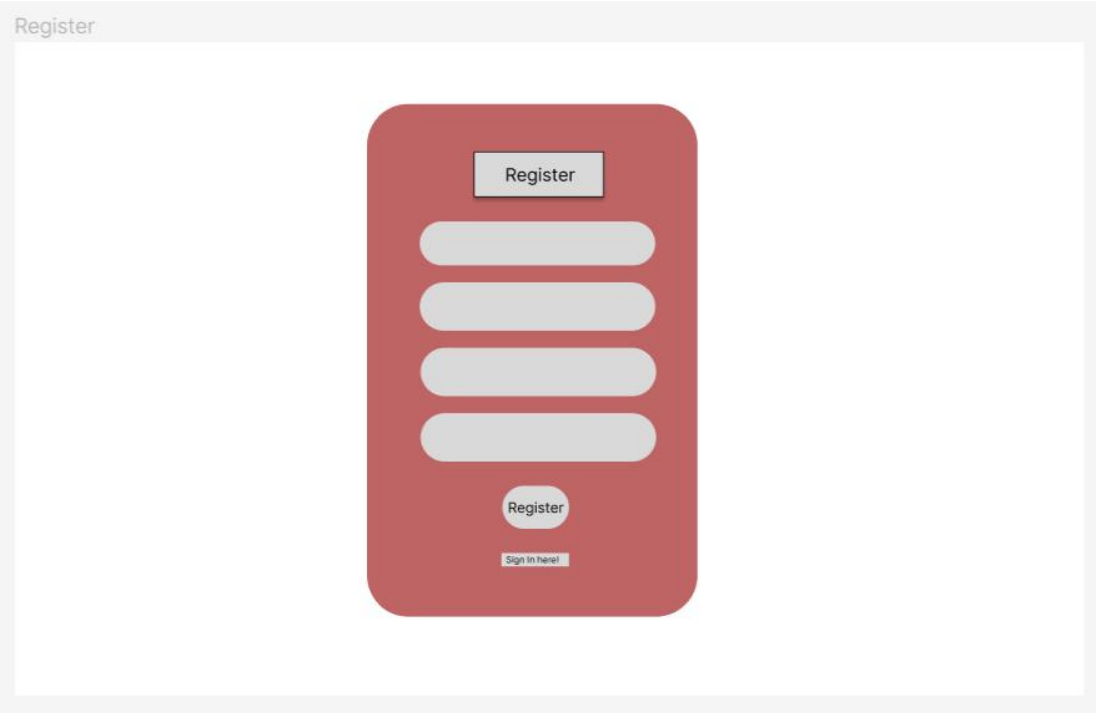

Figure 4.5.2.2 - “Register” page

"Register" page (Figure 4.5.2.2) is used for the registration of users. On this page, users need to create a username, provide their email address, and confirm their password to make sure the two passwords are the same. Users can click the "Register" button to register, or if the user already has an account, they can click "Sign In Here!" to jump to the "login" page.

Figure 4.5.2.3 - "Main" page

"Main" page (Figure 4.5.2.3) is the page that will be shown to users after they login to the system, it will show the C&C logo, the searching bar and the user avatar at the top, then it followed by six main parts of this website: friend list, group list, find something new, friends requests, group invitations and the status.

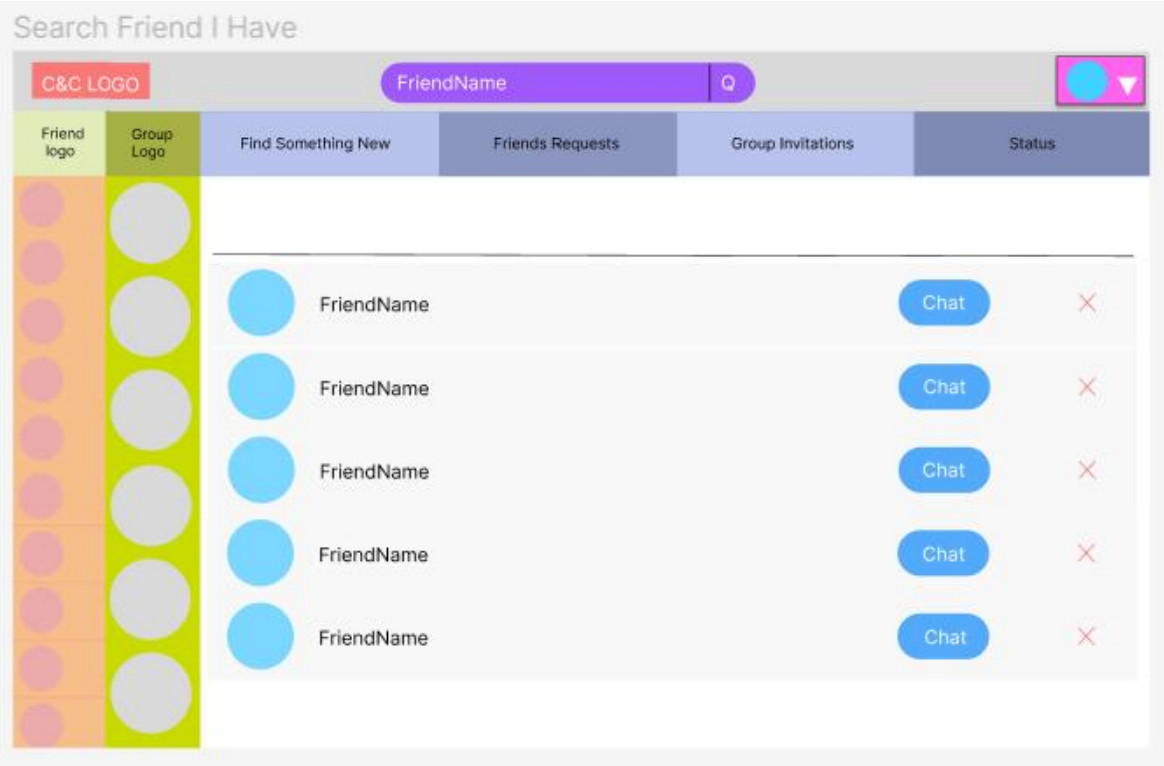

Figure 4.5.2.4 - "Search Friend I Have" page

"Search Friend I Have" page (Figure 4.5.2.4) is shown when the user searches a friend's name in the searching bar, the page will show the scroll component of all the similar friends' names, then the user can choose to chat with them.

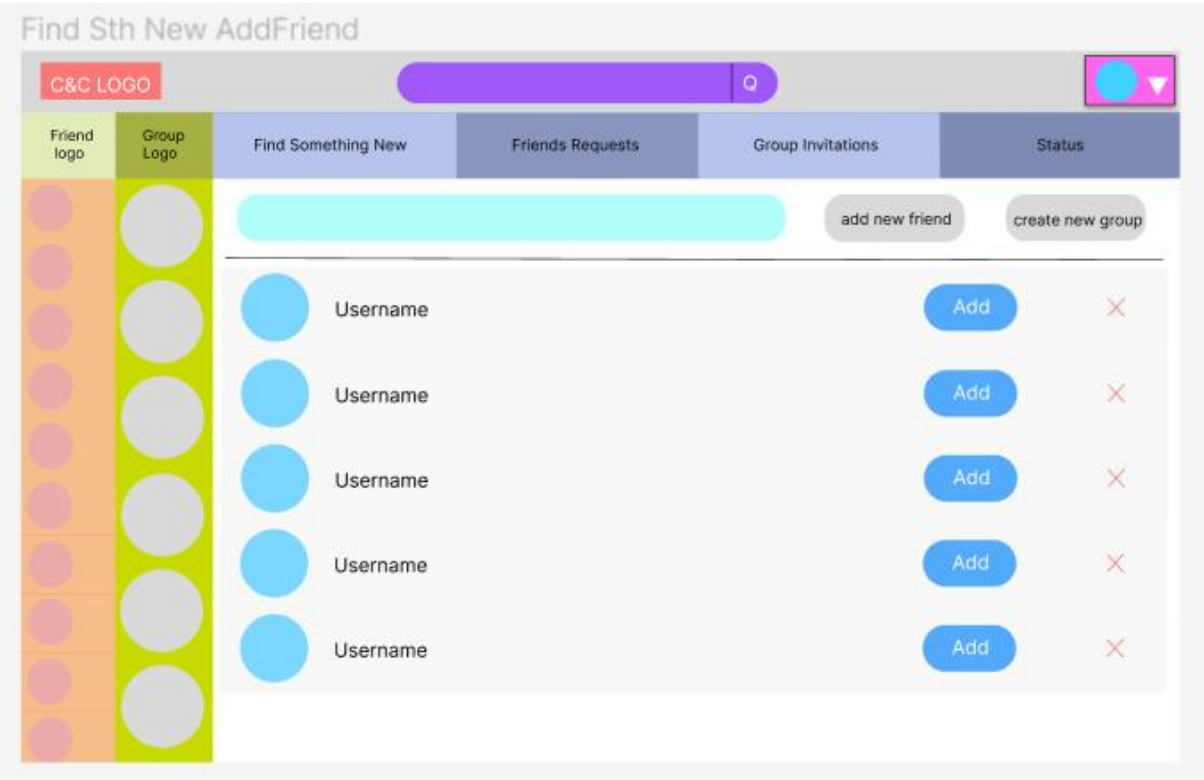

Figure 4.5.2.5 - "Find Something New Add Friend" page

"Find Something New Add Friend" page (Figure 4.5.2.5) is shown when the user wants to find a new friend, then can input the name in the search bar, then the C&C system will find all the similar names to

help the user find this person.

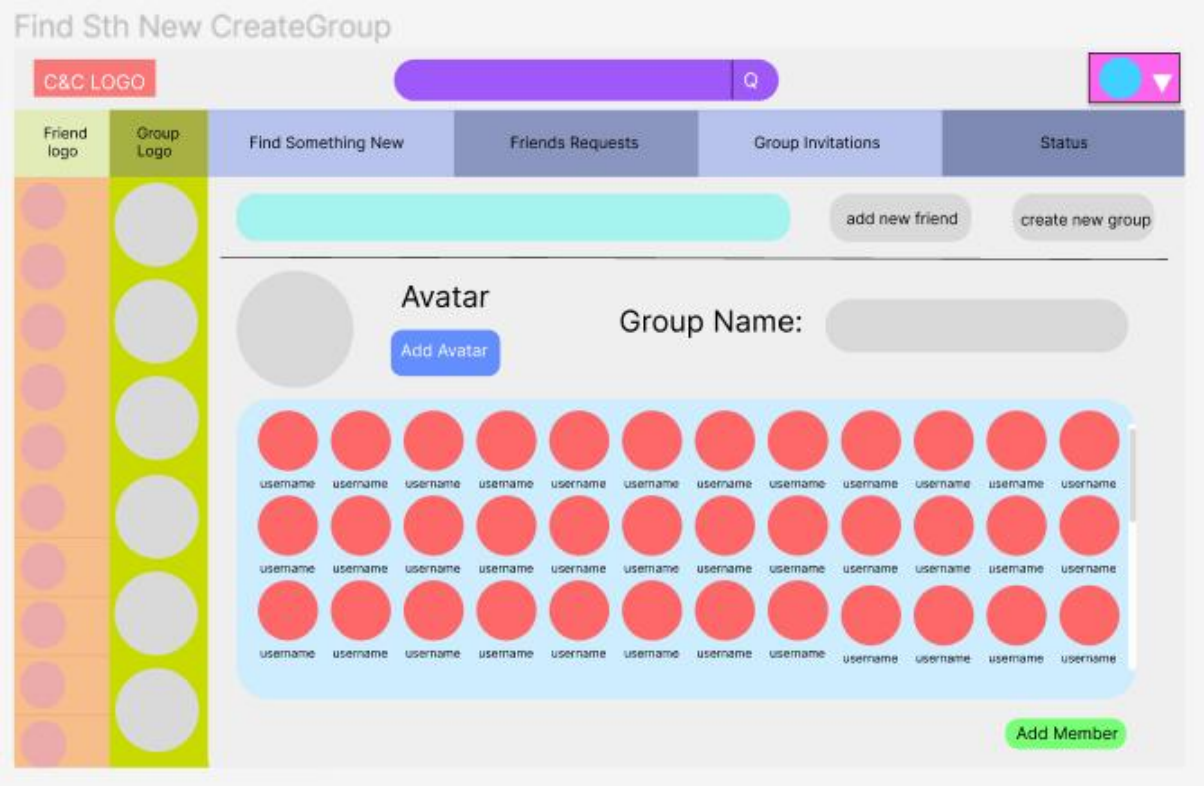

Figure 4.5.2.6 - “Find Something New Create Group” page

“Find Something New Create Group” page (Figure 4.5.2.6) is used when the user wants to create a new group, then the user can choose an avatar for the group and give a name to the group. By clicking the “Add Member” button, the selected friends will be shown in the scroll component, then the selected friends will get a group invitation.

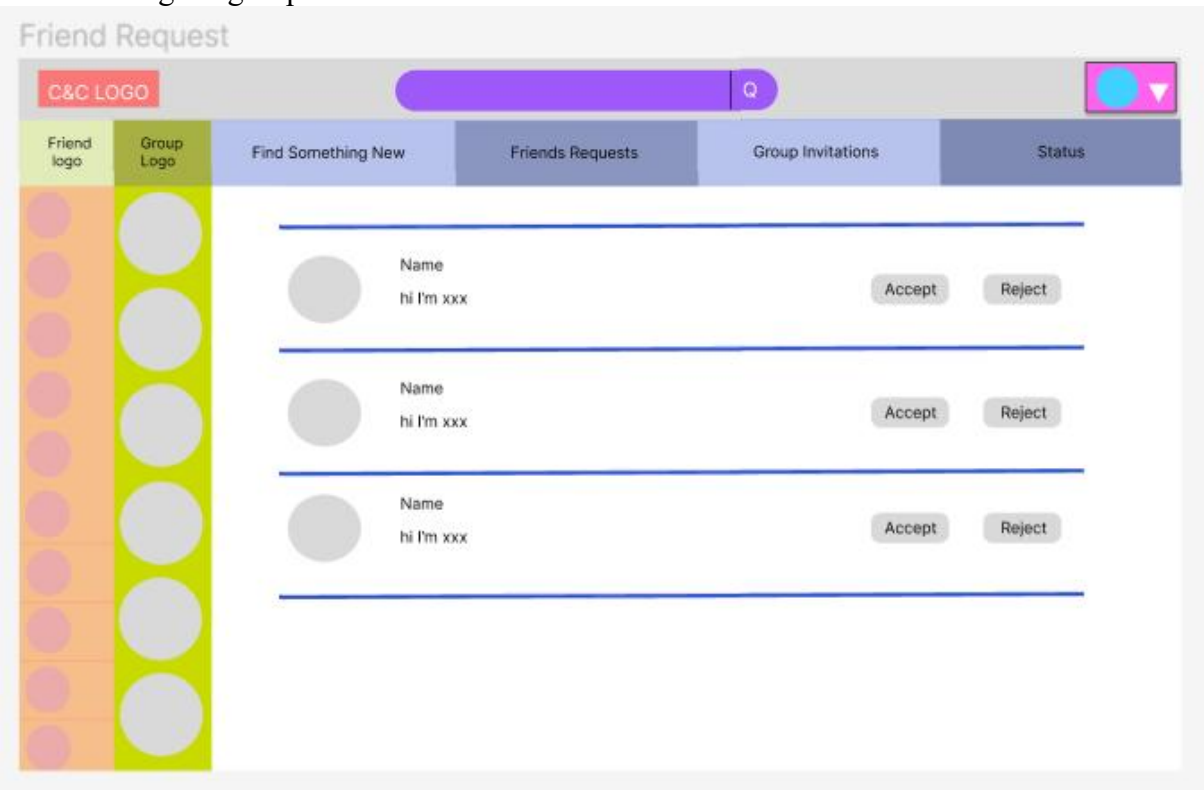

Figure 4.5.2.7 - “Friend Request” page

"Friend Request" page (Figure 4.5.2.7) is used for showing all the pending friends, the user can choose to accept or reject the friend request.

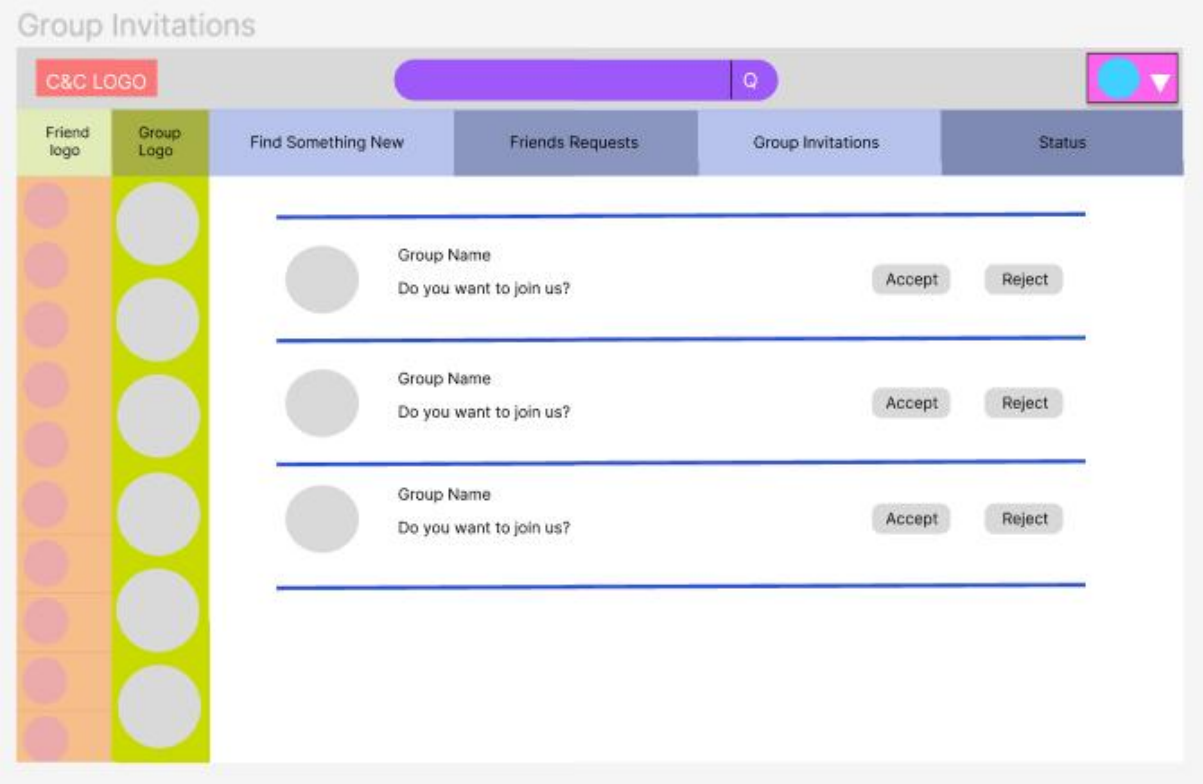

Figure 4.5.2.8 - "Group Invitations" page

"Group Invitations" page (Figure 4.5.2.8) is used for showing all the pending group invitations, the user can choose to accept or reject the group invitations.

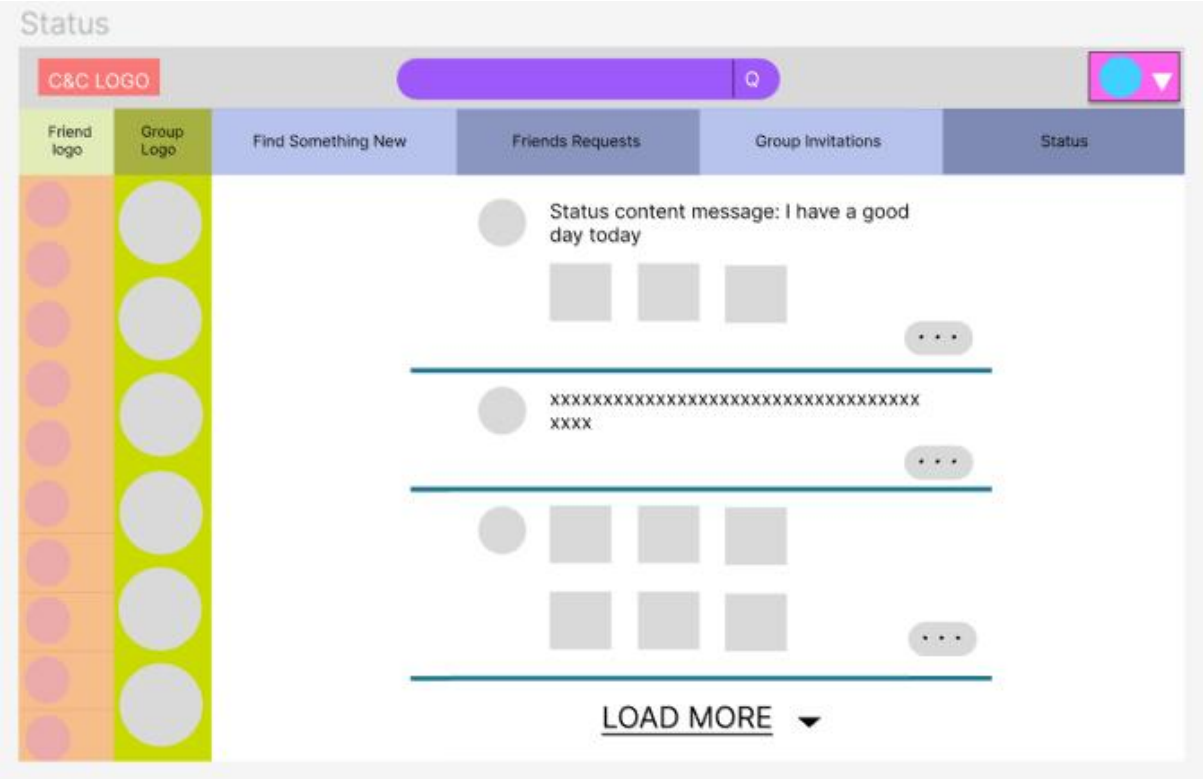

Figure 4.5.2.9 - "Status" page

"Status" page (Figure 4.5.2.9) is used to show all the statuses that the user can see, user can do a "like" action or comment on the status.

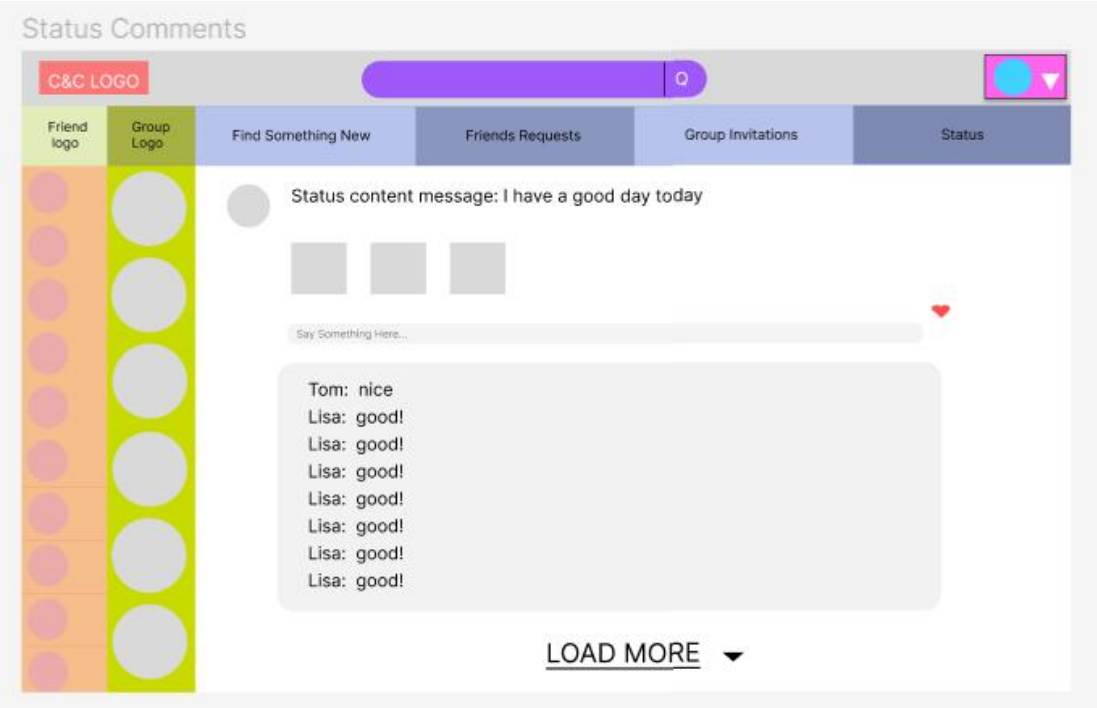

Figure 4.5.2.10 - "Status Comments" page

"Status Comments" page (Figure 4.5.2.10) can show all the comments in detail, the user can see comments from common friends.

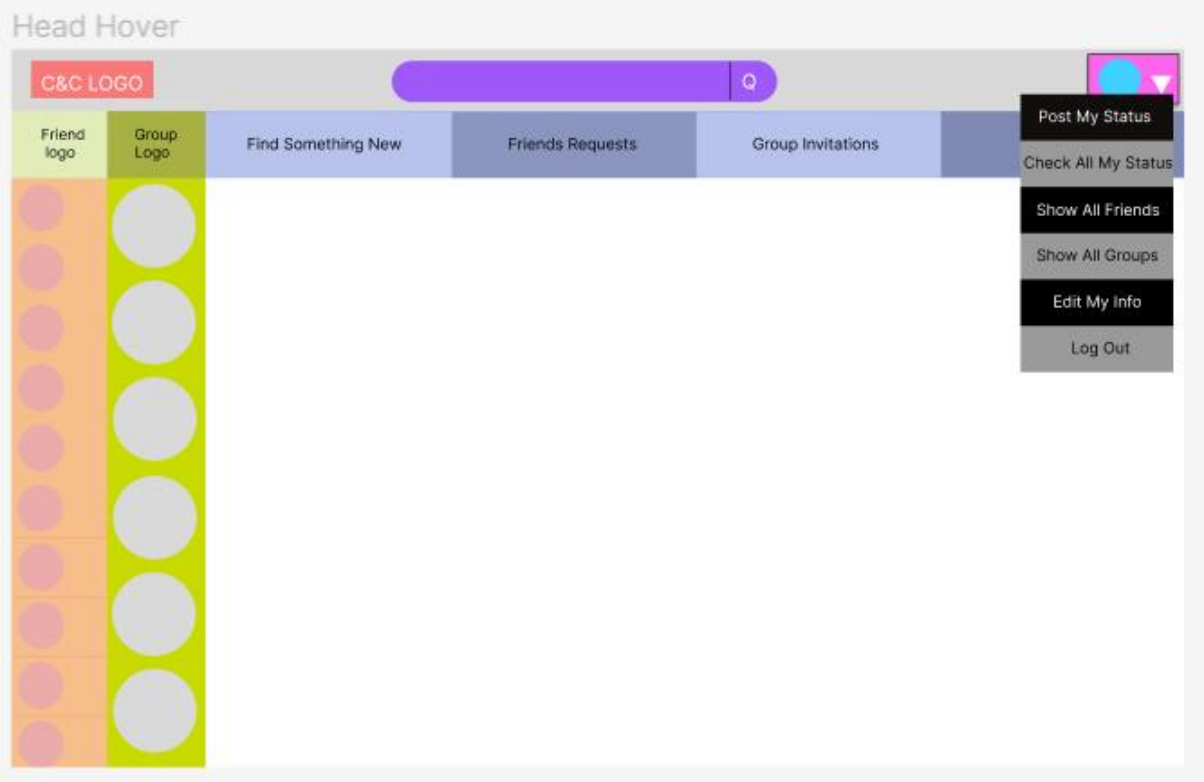

Figure 4.5.2.11 - "Head Hover" page

Head hover is shown in Figure 4.5.2.11, the head hover can post status, check users' own status, edit users' own info and log out. Besides that, the "show all friends" and "show all groups" can refresh

users' friend list and group list.

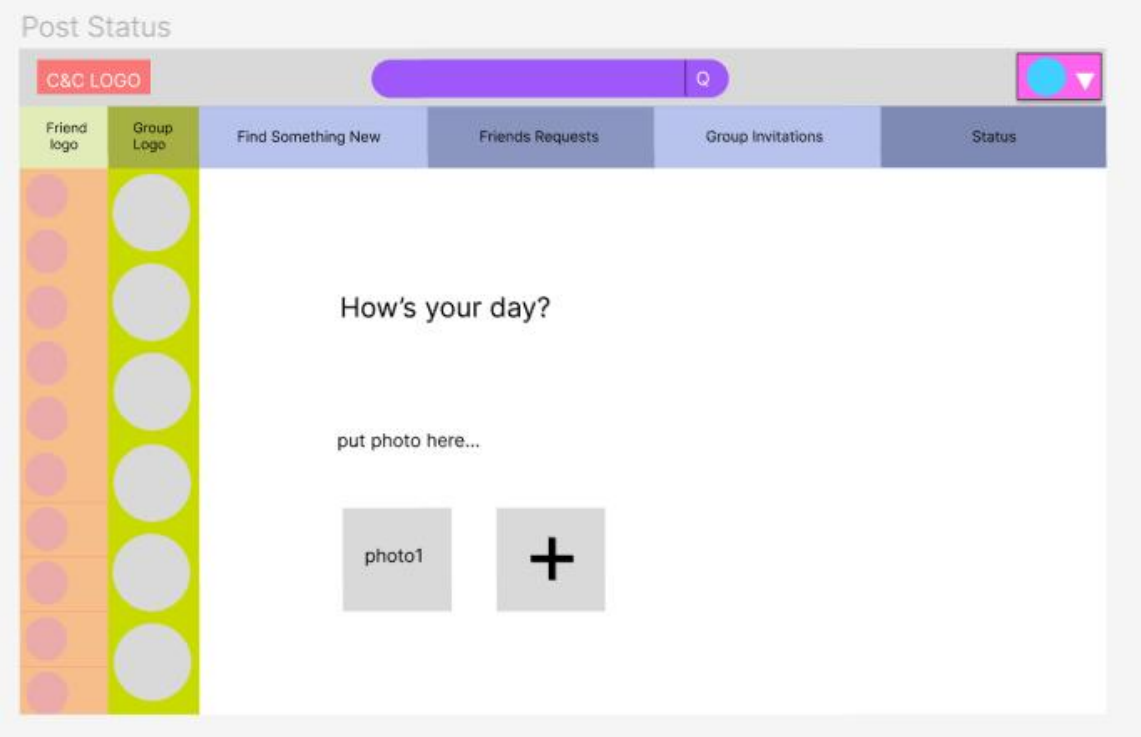

Figure 4.5.2.12 - "Post Status" page

Users can post their status through the "Post Status" page (Figure 4.5.2.12).

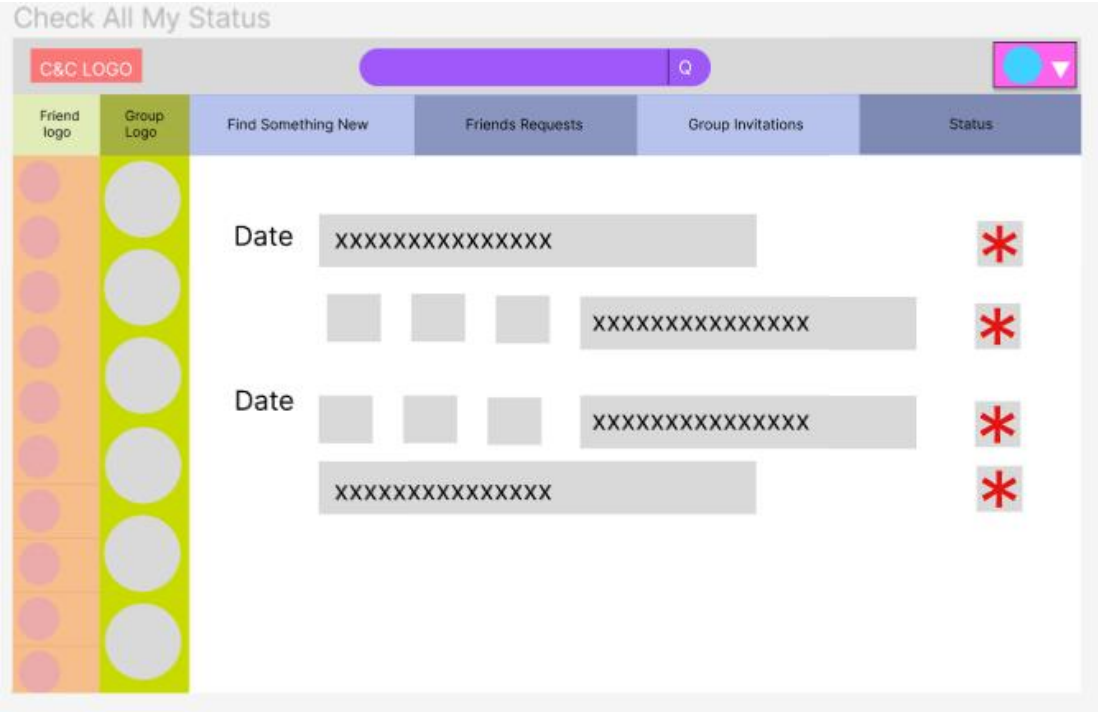

Figure 4.5.2.13 - "Check All My Status" page

Users can check all their past statuses or delete their statuses on "Check All My Status" page (Frigure 4.5.2.13), the status on this page will be shown with the post date.

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

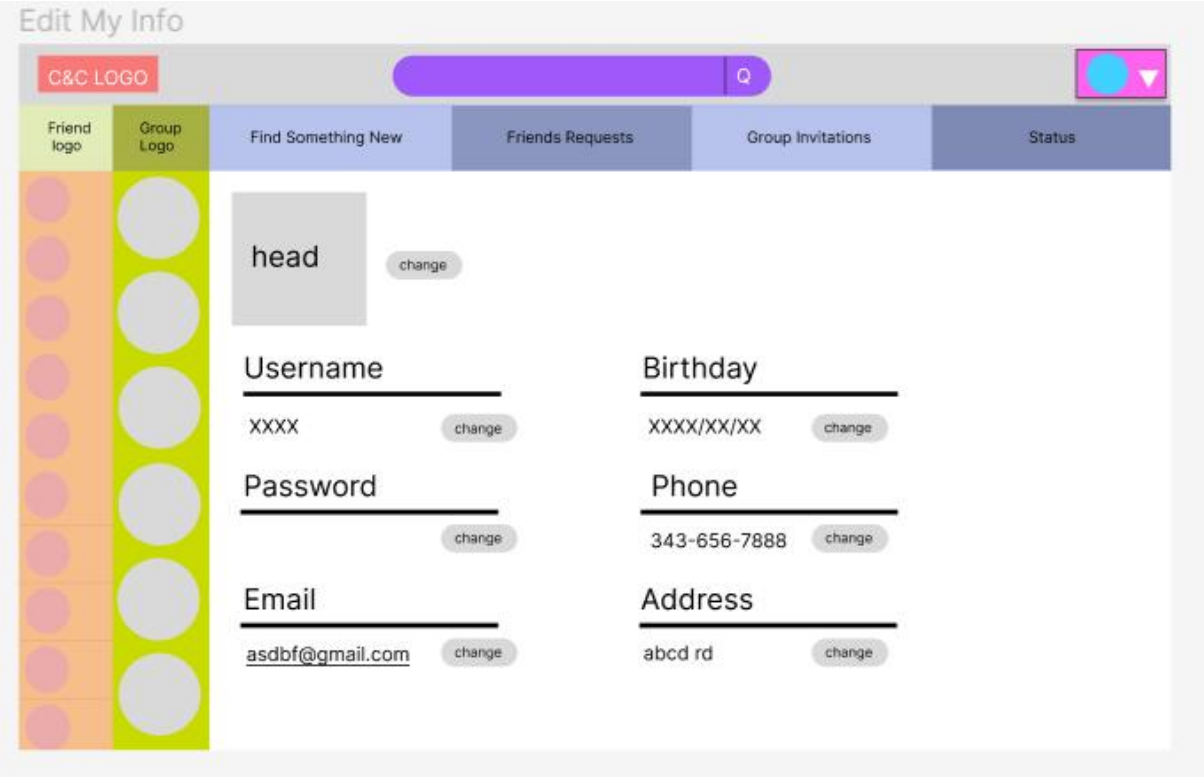

Figure 4.5.2.14 - "Edit My Info" page

Users can edit their personal information on "Edit My Info" page (Figure 4.5.2.14).

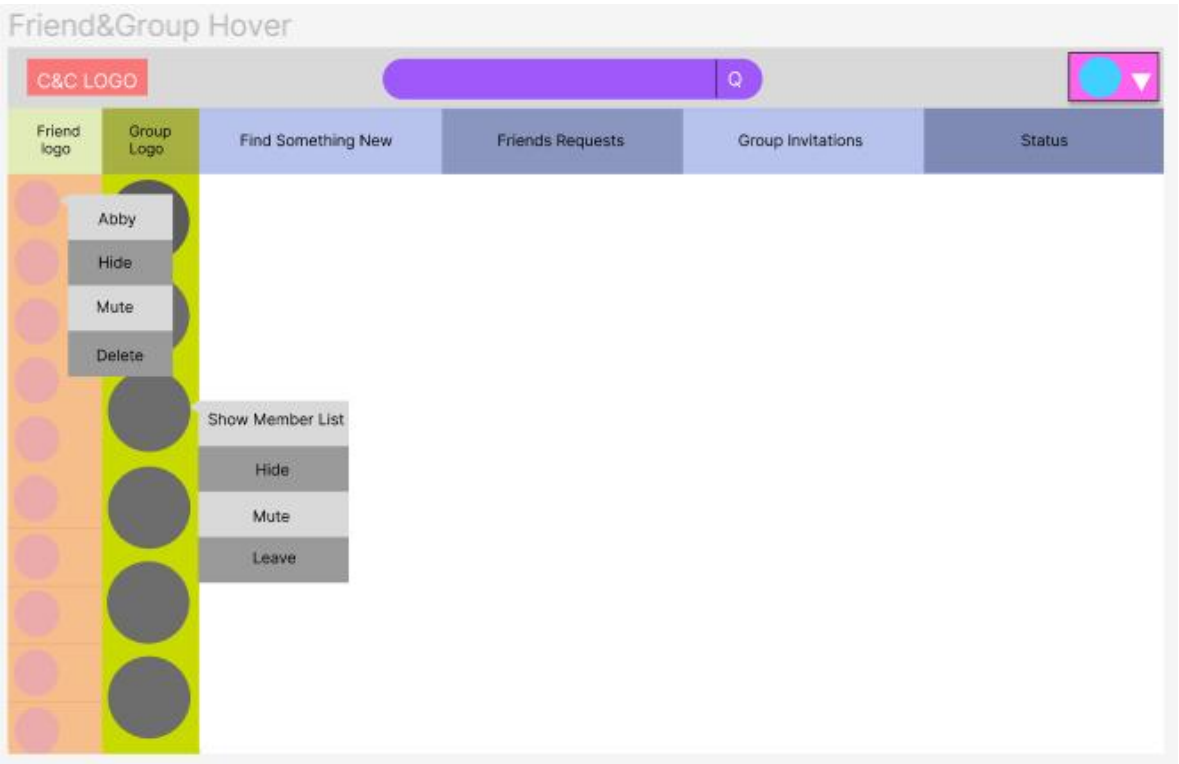

Figure 4.5.2.15 - "Friend and Group Hover" page

Friend and group hover are shown in Figure 4.5.2.15. The friend hover shows the friends' names on top, users can choose to hide, mute or delete this friend. Users can use the group hover to show the group member list, hide the group, mute the group or leave the group. After the user chooses to mute the friend or the group, the mute button will become unset. After the user chose to hide the friend or the group, they can use "show all friends" and "show all groups" in the head hover to show those friends or the group again.

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

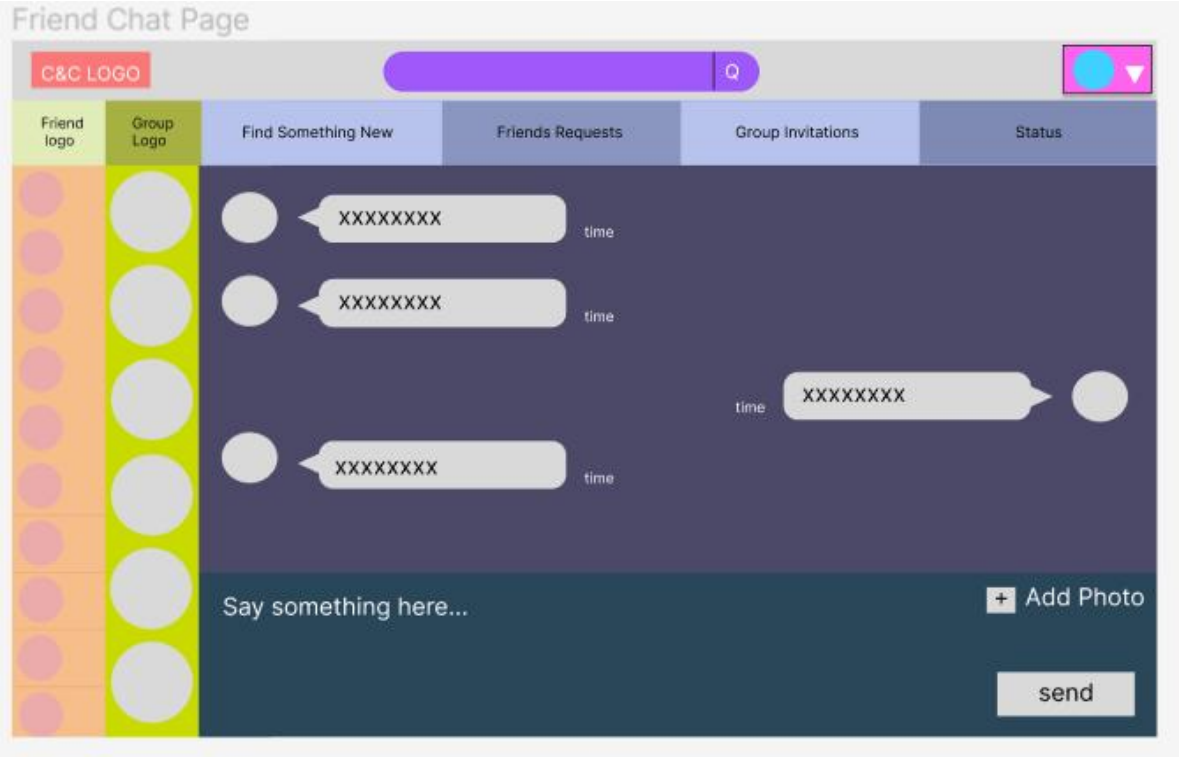

Figure 4.5.2.16 - “Friend Chat” page

“Friend Chat” page (Figure 4.5.2.16) support the users to chat with other people, and upload photoes while chatting. This page will also show the time that the message has been sent and the sender’s avatar. By clicking the friend’s avatar, user can get into this friend’s info page.

Figure 4.5.2.17 - “Check Friend Info” page

“Check Friend Info” page (Figure 4.5.2.17) show some of the friends’ information and their statuses with post dates.

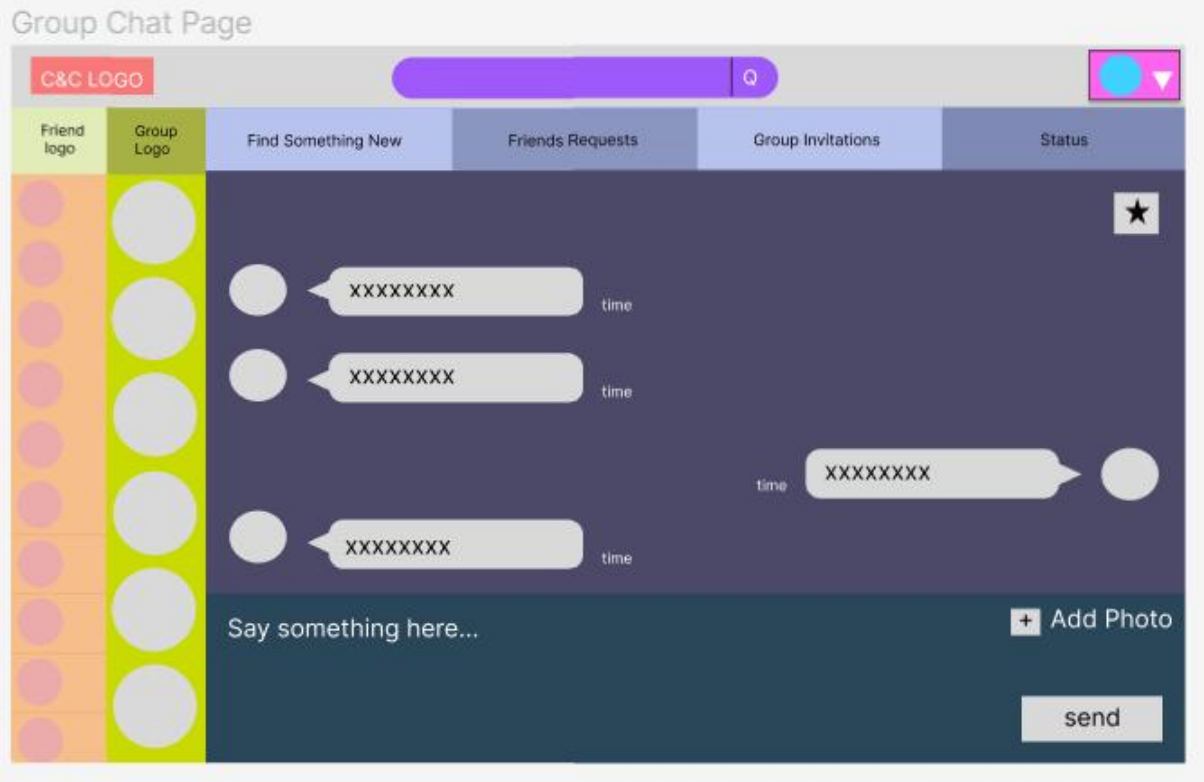

Figure 4.5.2.18 - “Group Chat” page

“Group Chat” page (Figure 4.5.2.18) is similar to the “Friend Chat” page, the difference is, this time the user cannot see personal info by clicking the avatar but they can click the star on the right corner to access to the group setting page.

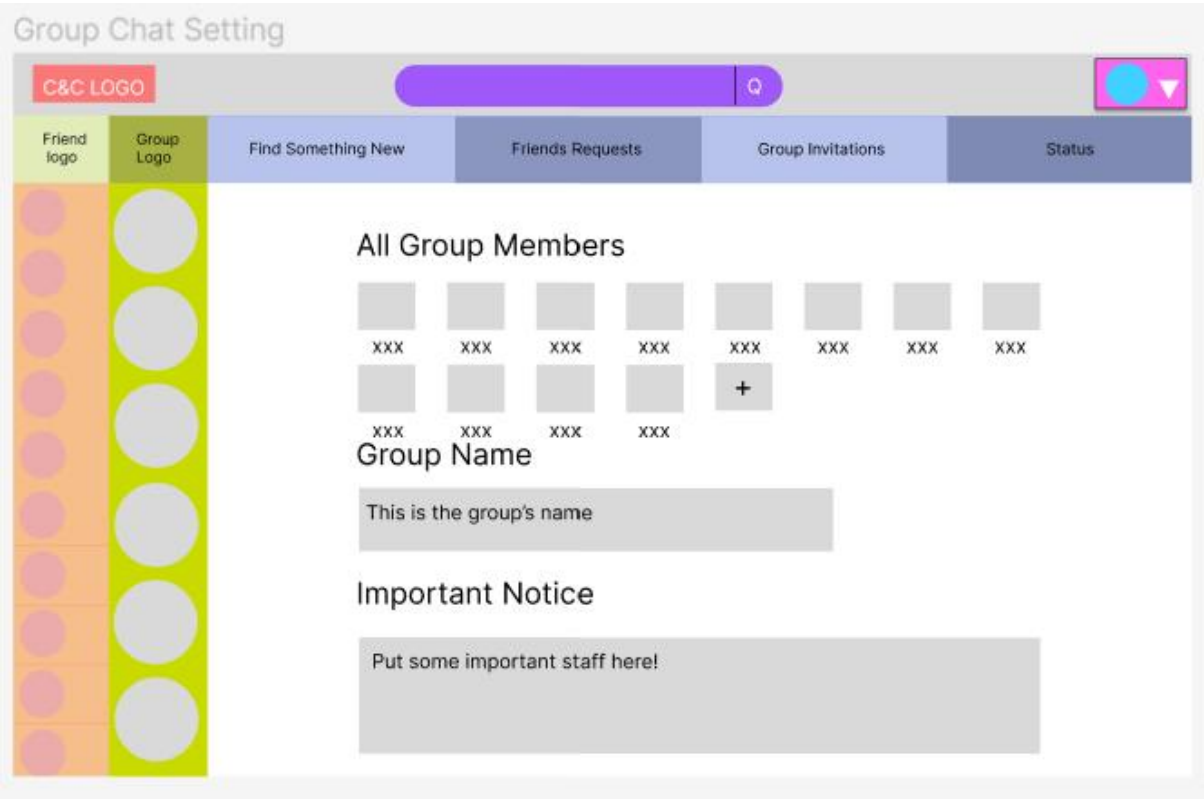

Figure 4.5.2.19 - “Group Chat Setting” page

“Group Chat Setting” page (Figure 4.5.2.19) shows all the group members, group name and important notice

for this group, on this page, users can invite their friends to the group, the little add icon will pop an invitation window.

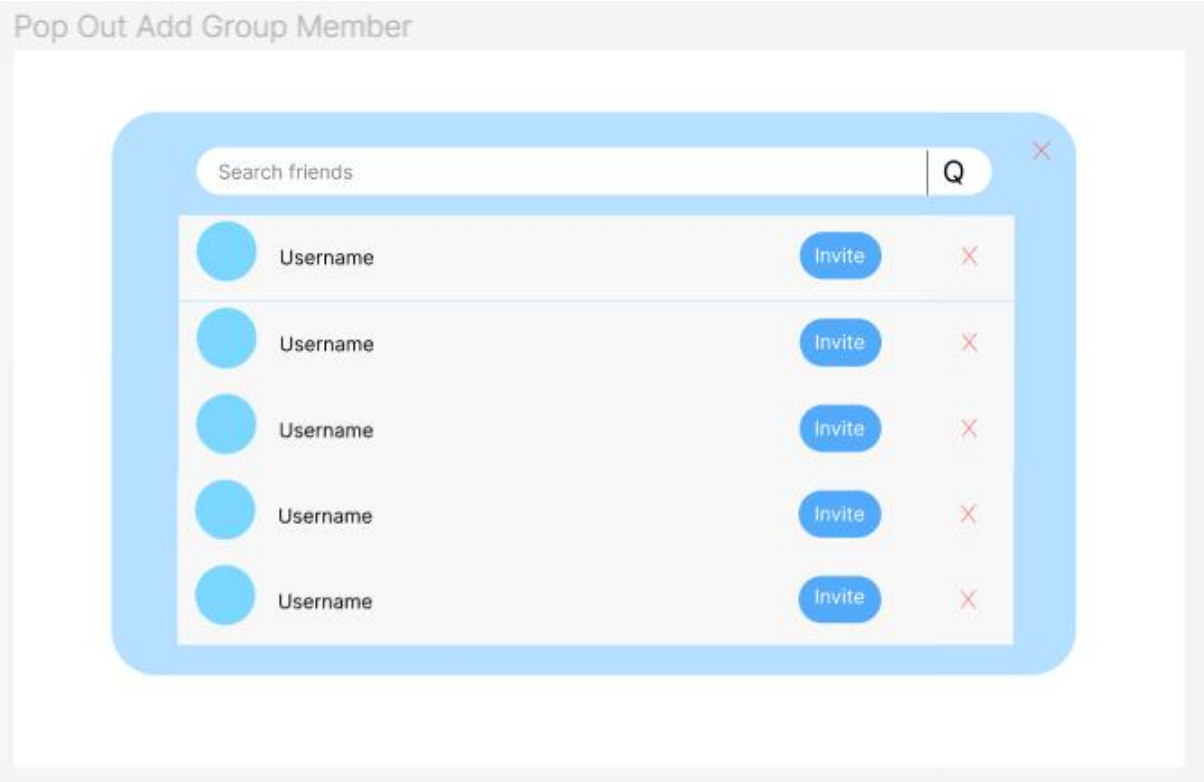

Figure 4.5.2.20 - “Pop Out Add Group Member” page

“Pop Out Add Group Member” page (Figure 4.5.2.20) has a scrolling page to show users’ friends that can be invited, and the users can search for their friends’ name to send the invitation.

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

4.5.3 UML

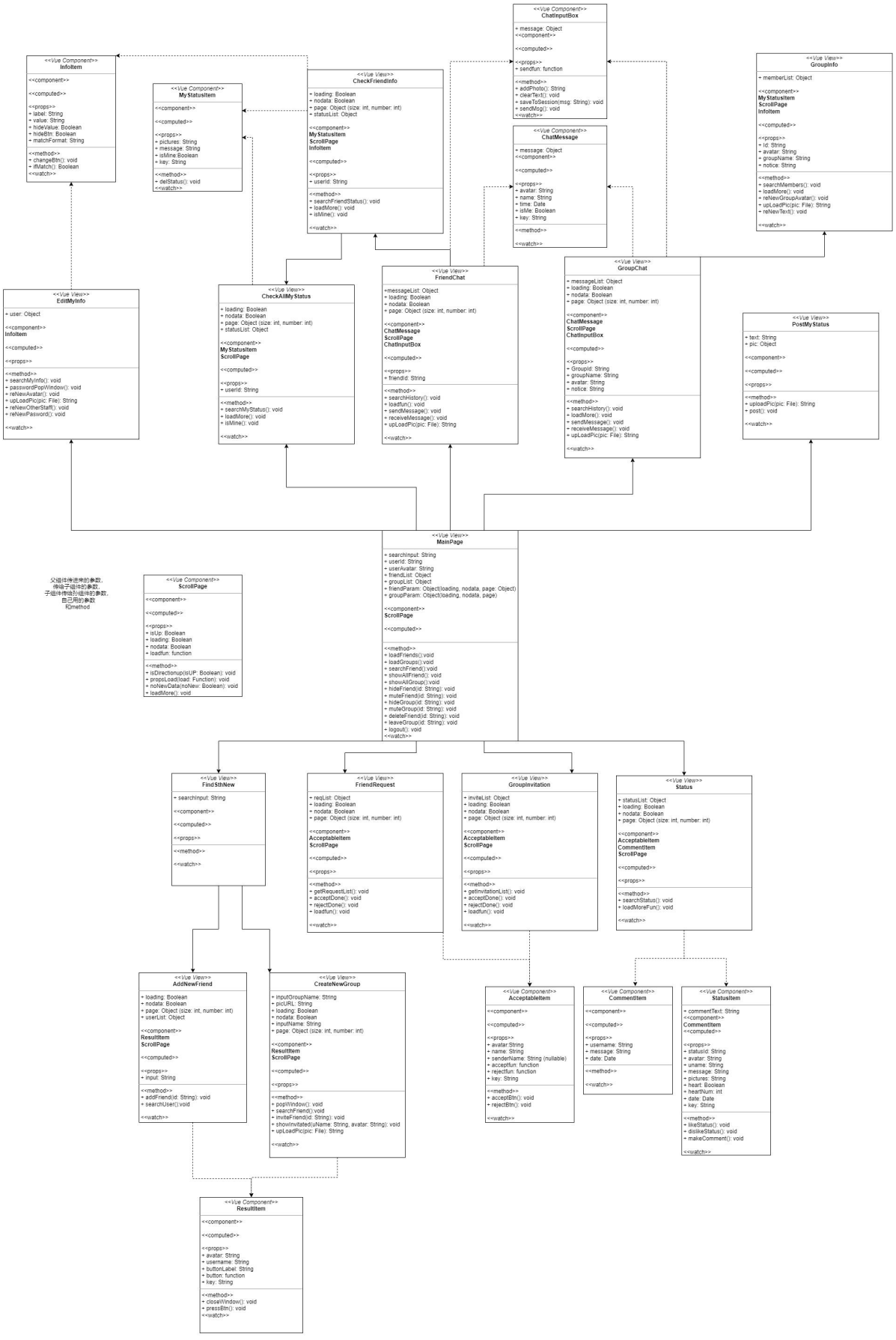

Figure 4.5.3.1 - Front-end UML

Depending on the page jumping rules described in page structure document (4.5.2), a simple front-end UML is designed. Arrows pointing directions explains the jumping sequences, for example, jumping from MainPage to EditMyInfo is possible.

Each box represents a class, and the very top few lines are the local variables of the class. The component part shows all reused classes in the current class. The props part shows all parameters given by the class's parent. The computed part lists all parameters that change instantly, and the method part shows all local functions. The watch part list all parameters that are monitored by the class.

Although, most front pages structures are designed well in the front-end UML, small changes will occur during implementation.

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

4.6 API Documents

The team has already designed the APIs that we can think about to build this C&C system, all the APIs are saved in the API document(api_v1.4.md). The explanation of a sample API is shown in (figure 4.6.1).

This API will be used in the user login page, and it is corresponding to the userController in the login() method. The request type of this sign in API is PUT, it will ask for the username and the password in a string from the backend, and the data that is returned will tell if the request succeeds, sending the error code and the succeed message back and returning the token of the data (figure 4.6.4).

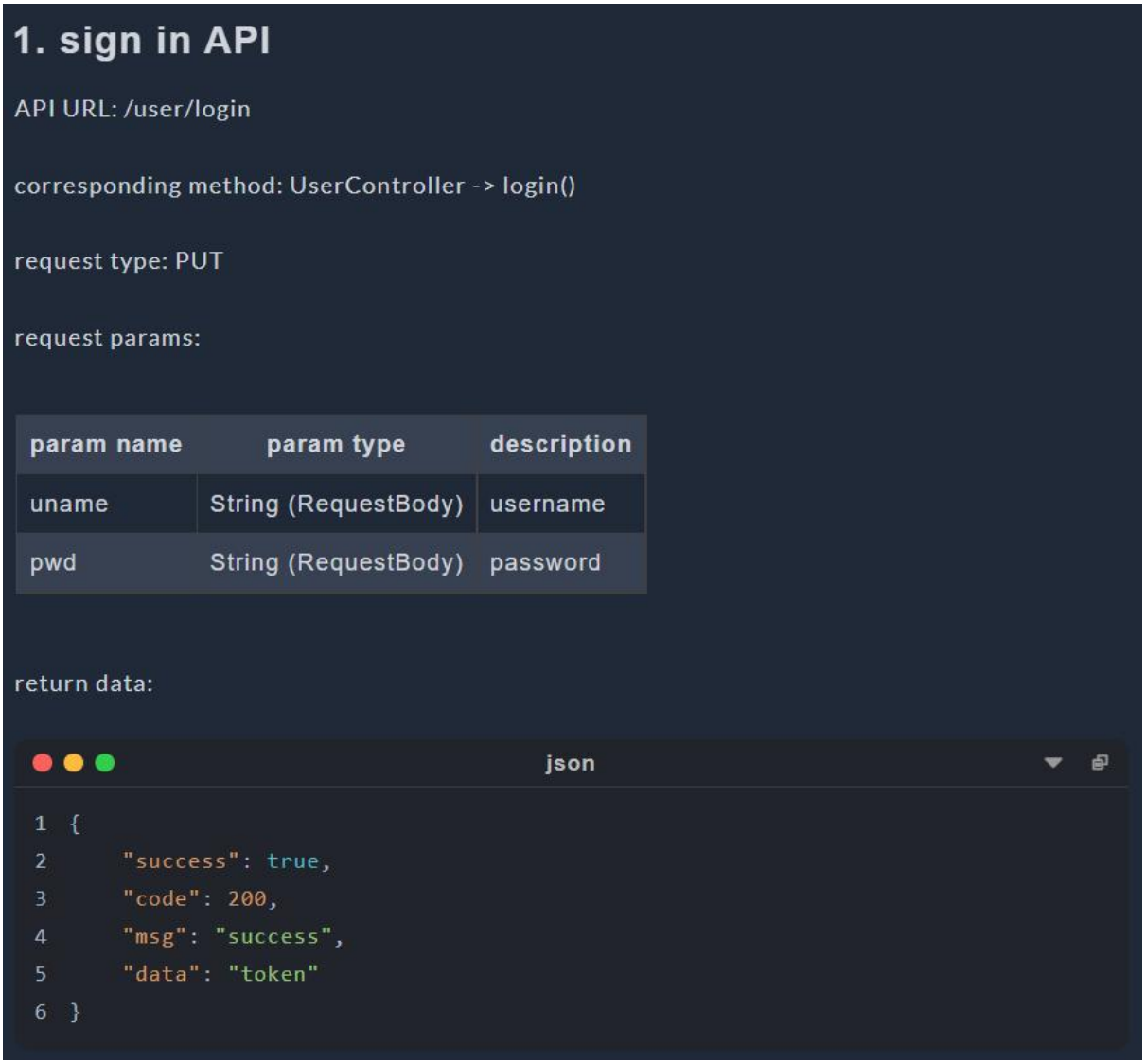

Figure 4.6.1 - “Sign In” API

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

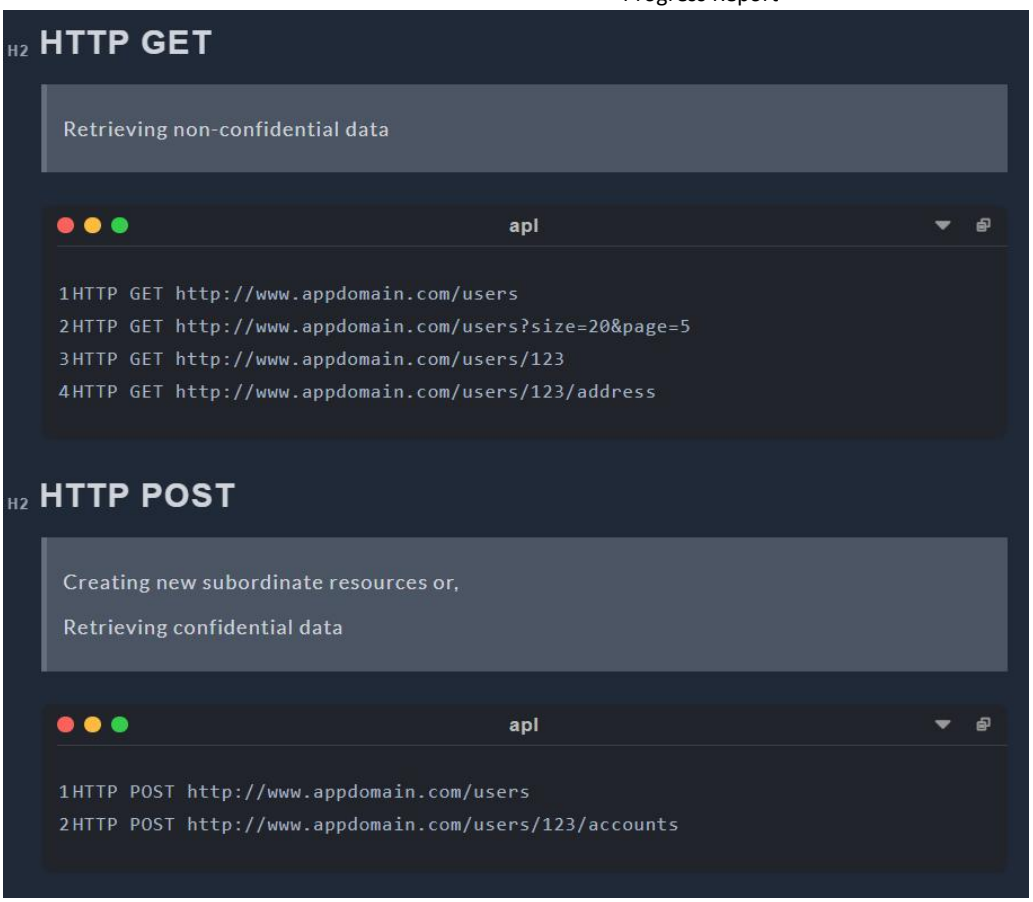

Figure 4.6.2 - HTTP Request Structure 1

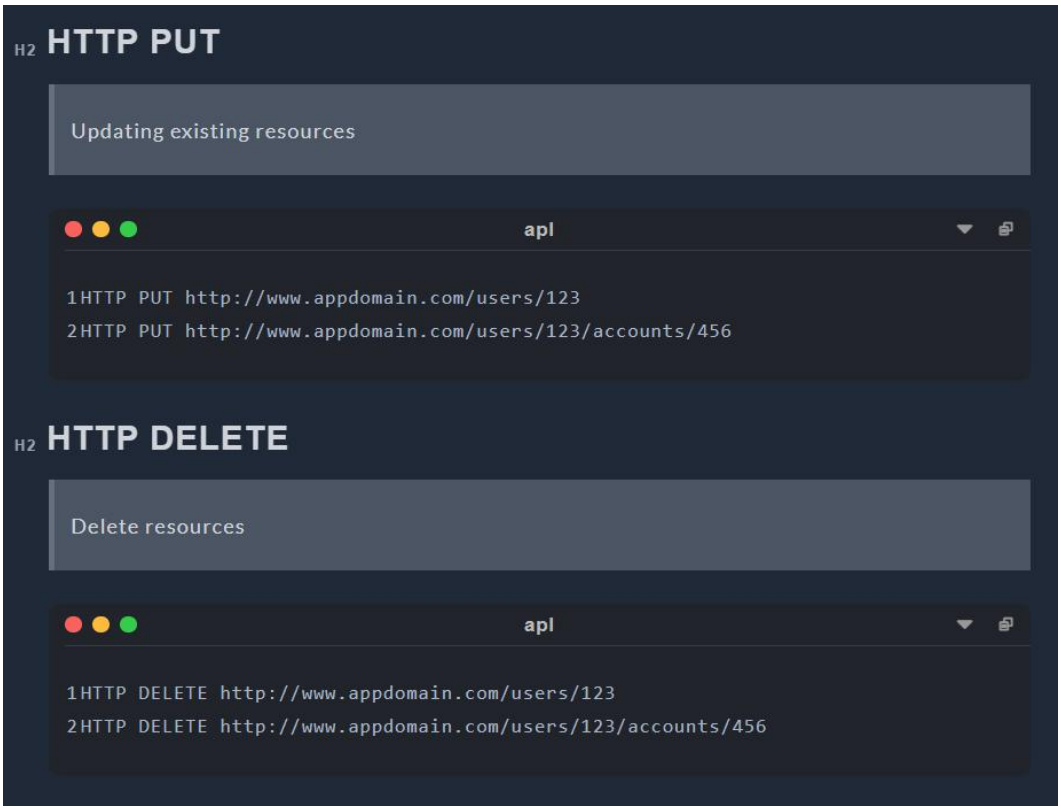

Figure 4.6.3 - HTTP Request Structure 2

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

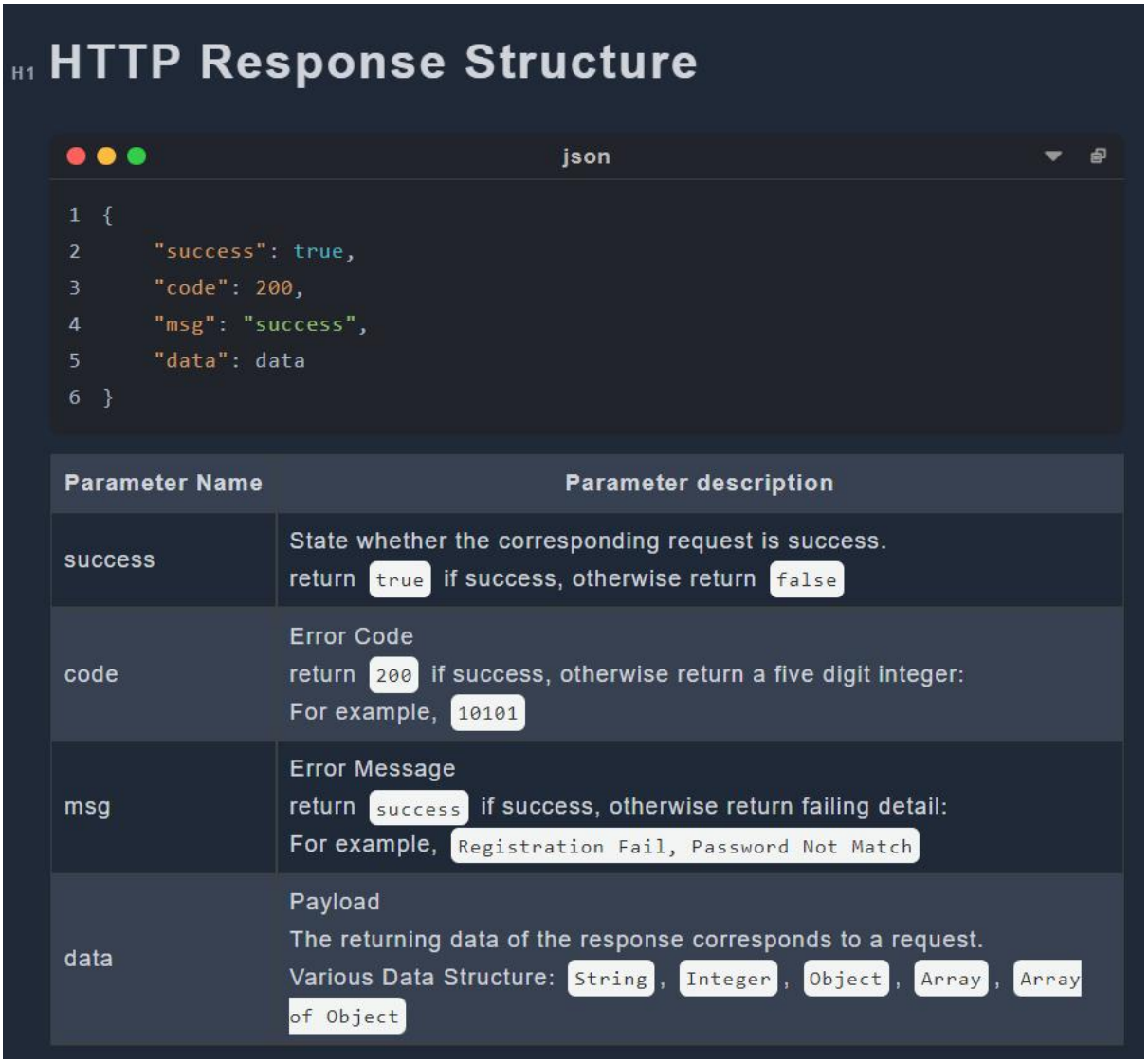

Figure 4.6.4 - Explanation of the returning data

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

4.7 Overall Project Structure

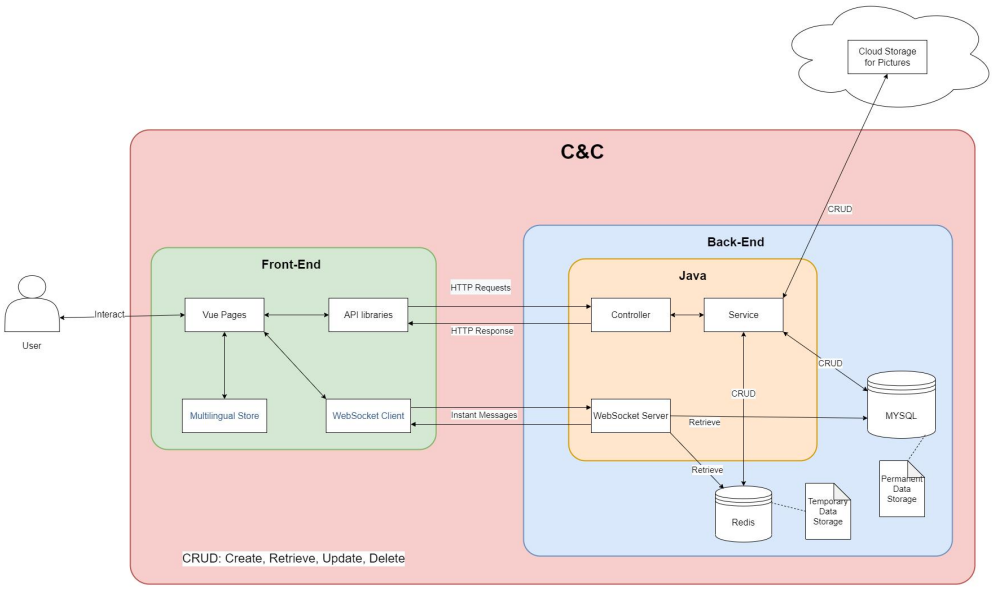

Figure 4.7.1: Overall Structure of C&C Project

An overall project structure is designed to describe the project in a more clear way.

Users can only interact with Vue pages in Front-End part. Vue pages handles page jumping by itself and provide multi-languages through the help of Multilingual Store. Most requests are sent to Back-End part through API libraries as HTTP requests. Once Back-End generate replies corresponding to requests, API libraries obtain and return the replies to Vue pages and display it to the user. Chat messages cannot be treated exactly as HTTP requests, therefore, WebSocket Client module will send and receive instant messages (WebSocket messages) to and from WebSocket Server in Back-End.

Java is the main software application in Back-End. The Controller module recognize all HTTP requests and invoke the corresponding service in Service module to handle the requests. Once service generate a result, Controller judge the result and reply the judging result to Front-End. MYSQL is a relational database which stores permanent data for C&C and Redis is a NoSql database which stores temporary data for C&C. The Cloud Storage on the top of the graph is a cloud database which can only store pictures, all pictures in C&C will be stored in the Cloud Storage module. Only Service module can perform all CRUD operations on the three databases, CRUD stands for Create, Retrieve, Update and Delete. The WebSocket Server module is only used to accept and reply to instant messages (WebSocket messages). To obtain necessary user data, the WebSocket Server can retrieve data from both Redis and MYSQL.

C&C comprise all elements in the graph except for User and Cloud Storage. The C&C portion is labelled in red.

5. Justification

5.1 Java

Specifically, the Java language has the following advantages:

1. Java is a pure object-oriented language. It can directly reflect the objects in real life, such as animals. Therefore, it is easier for developers to understand and write programs.

2. Java language can be used on multiple platforms. Java programs can be compiled on Windows platform, Linux, MacOs and other platforms. The compiled programs can run on other platforms.

3. Java provides a lot of built-in class libraries, which can simplify the programming work of developers and shorten the development time of projects.

4. It provides support for Web application development. For example, Servlets and JSPs can be used to develop Web applications.

5. The C++ language has removed features that are difficult to understand or easily confused, such as header files, pointers, structures, units, operator overloads, virtual basic classes, multiple inheritance, etc., making the program more rigorous and clean. [2]

Using java technology can help us reduce unnecessary repetitive work when making this project, and the database provided by java can help us reduce the working time.

5.2 Spring

Spring is an open source framework. One of the main advantages of the framework is its hierarchical architecture, which allows users to choose which component to use. Spring uses basic JavaBeans to do things that previously could only be done by EJBs. However, Spring's use is not limited to server-side development. From the perspective of simplicity, testability and loose coupling, any Java application can benefit from Spring.

1. Declaration of support
In Spring, we can get rid of the transaction management code and flexibly manage transactions in a declarative manner to improve development efficiency and quality.

2. Convenient program testing
In Spring, testing is no longer a complex operation, but a very simple thing. For example, Spring supports Junit4 and can easily test Spring programs through annotations.

3. Convenient integration of various excellent frameworks
Spring does not exclude various excellent open source frameworks. On the contrary, Spring can reduce the difficulty of using various frameworks. Spring provides direct support for various excellent frameworks.

4. Reduce the difficulty of using Java EE APIs
Spring provides an encapsulation layer for many Java EE APIs that are difficult to use. Through the simple encapsulation of Spring, the difficulty of using these Java EE APIs is greatly reduced. [3]

In this project, we use spring because spring is used by many people. Other users have written many spring based libraries and new technologies. These technologies can make it easier for us to deal with projects.

5.3 MYSQL

The main advantages of MySQL are as follows:

1. Speed: fast operation.

2. Price: MySQL is free for most individuals.

3. Easy to use: Compared with the setting and management of other large databases, it is less complex and easy to learn.

4. Portability: It can work on many different system platforms, such as Windows, Linux, UNIX, Mac OS, etc.

5. Rich interfaces: APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, Tcl and other languages are provided.

6. Security and connectivity: Host based authentication is allowed. When connecting to the server, all password transmissions are encrypted to ensure password security. Because MySQL is networked, it can be accessed anywhere on the Internet to improve the efficiency of data sharing. [4]

In this project, we use MySQL because it is a very popular and free technology. We can use MySQL to reduce our spending on projects and find rich learning resources.

5.4 Redis

A user is unlikely to send only one request after logging into C&C. Therefore, a lot of time is wasted frequently retrieving data from MYSQL. To speed up the generation of responses to requests, it is helpful for the C&C system to keep useful data for an extended period of time instead of retrieving it from MYSQL every time it needs it.

NoSQL Database solves this problem perfectly; they store temporary data in a RAM or cache and provide data when software needs them. Retrieving data from RAM (NoSQL Database) is a million times faster than the same operation from SSD/HDD (Relational Database). Therefore, to speed up the response, C&C stores frequently used data in the NoSQL database, for example, user tokens. There are two famous NoSQL databases: Redis and MongoDB. Redis is faster and uses fewer resources, but most of its APIs are atomic, which means it could take time to write customized APIs for C&C to use it. Redis stores data in the RAM only in the Key-Value pair and does not support other query languages. MongoDB is slower and uses more resources; while it owns APIs with various functions, it is undoubtedly more straightforward to use than Redis. MongoDB store data in Key-Value pair but allow multiple keys and support many query languages, even JSON[5].

Redis states that the MongoDB database needs Redis when the following requirements are required [6]:

- Query optimization (caching)
- Session management
- Real-time analytics
- High-speed data ingestion
- Message brokering and queues

The comparison between Redis and MongoDB is very similar to that between C and Python; both Redis and C have higher performance, while MongoDB and Python are easier for developers. C&C will not store complex data in RAM; the speed is the most relevant. Therefore, Redis is chosen. However, if C&C is a

huge sophisticated software and stores various types of data in RAM or cache, MongoDB is the better choice.

5.5 Vue3

Vue is a JavaScript framework for building user interfaces. It builds on top of standard HTML, CSS, and JavaScript and provides a declarative and component-based programming model that helps you efficiently develop user interfaces, be they simple or complex[7].

Vue is designed to be flexible and incrementally adoptable, depending on the use case, Vue can be used in different ways:

- Enhancing static HTML without a build step
- Embedding as Web Components on any page
- Single-Page Application (SPA)
- Fullstack / Server-Side Rendering (SSR)
- Jamstack / Static Site Generation (SSG)
- Targeting desktop, mobile, WebGL, and even the terminal

We choose it to build the C&C web-pages, because compared with the traditional HTML, Vue contains HTML, CSS and JavaScript together, so it decrease the possibility of using the wrong path while building the user interface. Also, Vue has a better structure than HTML, instead of having three files for a single page, Vue page contain all of HTML, CSS and JS codes in the same file, the size of the C&C system can be reduced significantly.

5.6 Element-Plus & Tailwind

Element-Plus and Tailwind will be used for designing the webpages, these two components will take the place of the traditional CSS while developing the webpage.

Element-Plus is a Vue 3.0 based component library for developers, it contains many built-in beautiful visual components that can be used to decorate the webpage. Since Vue 3.0 is being used to build the system, Element-Plus is suitable for design. It also provides many fancy webpage building functions that can be used directly[8].

Tailwind is an API for the design system, different from CSS that we need to use big blocks for each class, Tailwind can reduce those classes into one line of code. Each part of the webpage can have its own style, instead of collecting all the style parts and putting them together, the high flexibility of it also allows customized web-pages[9].

Although both Element-Plus and Tailwind simplify the process of decorating front pages dramatically, neither of them can replace CSS for the reason that some customized requirements cannot be proceeded by them. CSS is still an indispensable decorating tool for C&C project.

5.7 HTTP and WebSocket

HTTP represents for Hypertext Transfer Protocol which is an application-layer protocol for transmitting hypermedia documents, such as HTML[10] and JSON. Through it, data can be transferred between Front-End and Back-End easily. In C&C, most of the transferred data are in format of JSON[11]. However, only the client can send HTTP requests, not the server, and there must be an HTTP request before an HTTP

response.

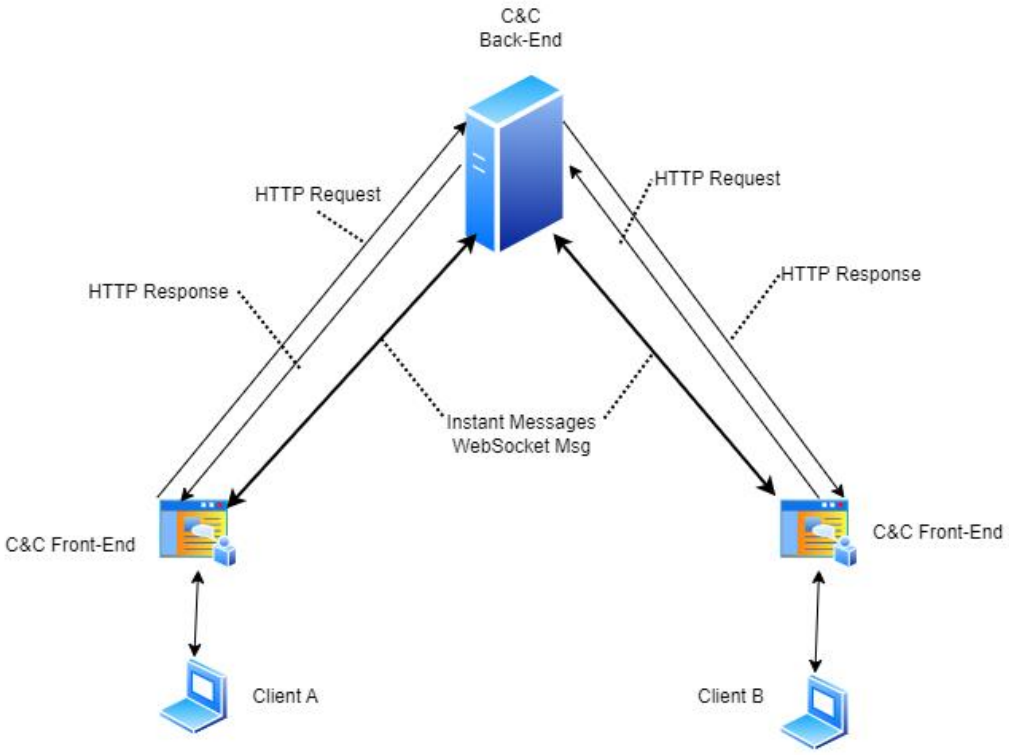

Figure 5.7.1: Data Flow Diagram of C&C

There is no direct connection between Client A and Client B users in the C&C system in Figure 5.7.1. The only way for A to see messages sent by B is for the server to send messages to A without being requested. However, servers cannot send unsolicited HTTP messages to clients, while client A needs to know messages sent by client B immediately. To be dumb, we could solve this problem by generating a set of HTTP requests to the server once every second. But this would seriously increase the pressure on the server, and a one-second delay is a long time for instant messaging, so such a solution doesn't work.

To solve the problem, there are two possible solutions according to our investigation: WebSocket and WebRTC.

WebSocket is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server[12]. WebRTC represents for Web Real-Time Communication which is a technology that enables Web applications and sites to capture and optionally stream audio and/or video media, as well as to exchange arbitrary data between browsers without requiring an intermediary[13]. Both technology allows clients connect with each others.

Briefly, WebSocket need a intermediary to create TCP connection among clients while WebRTC create p2p UDP connection (peer to peer) directly between two clients. UDP connection[14] could lose some data during transmittion while 100% transmission integrity guaranteed for TCP connection[15]. The guarantee of transmission integrity can take a lot of time, so that UDP transmission will be faster than TCP.

In terms of performance, WebRTC is much faster than WebSocket, as it connects directly to its peer using UDP, while WebSocket first connects to the server and then connects to another client from the server using TCP. Conversely, WebSocket has a clear advantage in terms of data integrity.

In C&C, a chat messages encoding requirement is described in sub-section 2.1.8-Optional Functions; C&C may need to control whether or not chat messages are encoded among users. A chat history viewing function is described in term 5 of sub-section 2.1.4-Chat Functions; C&C need to store all sent chat messages for each user. To do message encoding, decoding and recording, we cannot put the three steps in front-end for security reasons, back-end it the only choice, therefore, WebSocket is the better choice of C&C.

In the future, C&C may add video and voice calling capabilities, and WebRTC would be a good choice for implementing these features.

5.8 Restful API

To handle HTTP requests, it is essential to have a consistent architecture for APIs. Fortunately, many API architectures have been developed by predecessors.

According to out investigation, there are 3 main API architectures: REST, SOAP and RPC.

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer, is a set of guidelines for scalable, lightweight, and easy-to-use APIs[16]. The guidelines of REST are:

- Client-Server Separation: Separate client from server.
- Uniform Interface: Use specific HTTP format.
- Stateless: Each client-server interaction is independent of every other interaction.
- Layered system: Requests and responses must always be formatted the same way.
- Cacheable: Server responses should indicate whether a provided resource can be cached by the client and for how long.

The design of C&C which separates client and server is a guideline for REST. As the most famous API architecture, it perfect for our web-based instant messaging software — C&C.

SOAP stands for Simple Object Access Protocol which relies heavily on XML, and together with schemas, defines a very strongly typed messaging framework. SOAP strictly defines how messages should be sent and what must be included. This makes SOAP APIs more secure than REST APIs, although the rigid guidelines make them more code-heavy and harder to implement[17]. In C&C, various types of data structures are used in API transfer; therefore, despite its safety features, it is not a good choice for the C&C project.

The RPC (Remote Procedural Call) protocol is the most straightforward of the three architectures. Unlike REST and SOAP, which facilitate data transfer, RPC APIs invoke processes. In other words, they execute scripts on a server. RPC APIs are limited in security and capabilities, so it is unlikely to see them as often as REST or SOAP APIs on the web. However, it can be used for internal systems for making basic process requests, especially many at once[18]. As an internal system-specific API architecture, it is not a suitable choice for the API architecture of C&C.

In conclusion, REST API or Restful API is the best choice among the three API architectures.

6. Process

Up to now, we have completed all the design and preparation work related to the project. We have completed the use case diagram and design diagram of the front end and back end. We designed the basic and non basic functions of this project. We designed the api interface for front-end and back-end

communication. We have found and learned the technology needed to handle the project. Next, we will enter the formal production stage of the project.

7. Management Methodology

WaterFall model is selected for our project management. We spent tons of time on designing, it is unlikely for us to miss many important requirements. Therefore, the requirements for our project is tend to be explicit and constant. In our project, only one iteration is estimated to be proceed.

Rapid Prototyping model is an excellent model for software projects, it allows flexible design and development. Developers can make any changes in no either designing phase or developing phase, and the end user can “see” the system requirements as they are being gathered by the project team. For our project, there is no end user in our designing nor developing phase. The whole team designed the project together and discussed various aspects of the project, changes will occur, but not much. Another reason for not using the rapid prototyping model is that this model requires rapid iterations, so our team must put tons of time into the project. All three of us planned to graduate in 2023, and this model is not friendly for 4th-year students like us.

Spiral Model is very attractive for our team. It handles change through iteration and minimizes risk. Still, rapid iteration is not a good choice for our team. Doing risk analysis takes time and require a large team, 3 people team is never considered as a large team.
Rapid Application Development model is also a good model, it divides the whole project into several small projects and use waterfall model on each. The start of the second small project does not require a complete of the first small project, such developing model could work very fast. However, performing such a development process requires a very familiar team with the corresponding area, which has to be large enough to split into smaller groups for different small projects. Our team of three did not meet any of the above points, so we could not choose this model either.

The main idea of using the incremental model is to keep the whole team working; however, for a small group of only three team members dealing with such a complex C&C project, most of the work is already planned, and there is not much free time. Therefore, there is no need to choose incremental model despite its excellence.

In conclusion, the WaterFall model best suits our situation.

8. Upcoming Challenges

We are looking forward to finding a highly efficient way that can support the instance message sending such as WebSocket and apply it on C&C. At the same time, a cloud data storage needs to be found to store the photos. The API connection between the front-end and the back-end needs to be figured out.

Even though we already have a relatively complete plan for the whole system, we might face more unexpected difficulties and accidents while we are writing the code. To solve those future problems, we will do more research and learn more techniques as time passes.

9. Individual Contribution

9.1 Project Contribution

Name	Project Design Contribution
Yunzhou Liu	ERDiagram V1.0-V1.2 Admin Back-end UML
Shizhong Shang	User Page Structure Admin FlowDiagram Admin Front-end Page Admin Front-end UML
Zirui Qiao	UseCase Diagram V1.0-V1.3 User FlowDiagram V1.0-V1.2 User Back-end UML V1.0 API specification V1.0-V1.4 Overall Project Structure
Yunzhou&Zirui	Module UML V1.0
Shizhong&Zirui	User Page Design User Front-end UML
All Members Together	UseCase Description 1st, 2nd draft

Table 1.0 Individual Contribution in Project Design

9.2 Progress Report Contribution

Name	Progress Report Contribution
Yunzhou Liu	4.1 Design - UseCase Diagram 4.2 Design - UseCase Descriptions 4.3 Design - Flow Diagrams 4.4 Design - Back-end 5.1 Justification - Java8 5.2 Justification - Spring 5.3 Justification - MYSQL 6 Process 10 Conclusion
Shizhong Shang	4.5 Design - Front-end 4.6 Design - API Specifications 4.8 Design - Overall Sequence Diagrams 5.5 Justification - Vue3 5.6 Justification - Element-Plus & Tailwind 8 Upcoming Challenges
Zirui Qiao	1 Introduction 2 Objectives 3 Changes to Objectives 4.7 Overall Project Structure 5.4 Justification - Redis 5.7 Justification - WebSocket 5.8 Justification - Restful API 7 Management Methodology
Together	9 Individual Contributions 11 Reference Appendix 1. Definition, Acronyms, and Abbreviations

Table 1.1 Individual Contribution in Progress Report

10. Conclusion

In the progress report, we summarized our current work progress. We summarized the functional and nonfunctional requirements of the project. We designed the basic framework of the project, completed the front-end and back-end use case diagrams, flow charts, front-end vision pages, back-end program structure and database structure. We also designed the API interface for front-end and back-end communication. We have made it clear that the technologies needed to complete the project include java, spring, MySQL, Redis, HTTP, etc. In each technical description section, we briefly explained the advantages of these technologies and the reasons why we chose them. We also summarized the challenges to be faced in the future. We have clarified everyone's contribution in this report.

11. References

[1] Oloruntoba, S. (2021, November 30). Solid: The first 5 principles of object oriented design. DigitalOcean. Retrieved November 26, 2022, from https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

[2] Learn java. Dev.java: The Destination for Java Developers. (n.d.). Retrieved November 24, 2022, from https://dev.java/learn/

[3]Learn about spring. Spring. (n.d.). Retrieved November 24, 2022, from https://spring.io/learn

[4]Quickly develop cloud applications with mysql heatwave. Oracle Canada. (n.d.). Retrieved November 24, 2022, from https://www.oracle.com/ca-en/mysql/

[5]"MongoDB Vs. Redis Comparison: Pros And Cons," MongoDB, 2022. https://www.mongodb.com/compare/mongodb-vs-redis (accessed Nov. 19, 2022).

[6] "Redis vs MongoDB: Which is Better? MongoDB & Redis 101 | Redis," Redis, Sep. 19, 2022. https://redis.com/docs/why-your-mongodb-needs-redis/ (accessed Nov. 19, 2022).

[7]"Vue.js - The Progressive JavaScript Framework | Vue.js," *Vuejs.org*, 2014. https://vuejs.org/ (accessed Nov. 26, 2022).

[8]"A Vue 3 UI Framework | Element Plus," *Element-plus.org*, 2022. https://element-plus.org/en-US/#hybrid-layout (accessed Nov. 26, 2022).

[9]"Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.," *Tailwindcss.com*, Nov. 15, 2020. https://tailwindcss.com/ (accessed Nov. 26, 2022).

[10]"HTTP | MDN," *Mozilla.org*, Sep. 13, 2022. https://developer.mozilla.org/en-US/docs/Web/HTTP (accessed Nov. 19, 2022).

[11]"JSON," *Json.org*, 2022. https://www.json.org/json-en.html (accessed Nov. 19, 2022).

[12]"The WebSocket API (WebSockets) - Web APIs | MDN," *Mozilla.org*, Sep. 09, 2022. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (accessed Nov. 19, 2022).

[13]"WebRTC API - Web APIs | MDN," *Mozilla.org*, Oct. 30, 2022. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API (accessed Nov. 19, 2022).

[14]L. Rosencrance, G. Lawton, and C. Moozakis, "User Datagram Protocol (UDP)," *SearchNetworking*, 2021. https://www.techtarget.com/searchnetworking/definition/UDP-User-Datagram-Protocol#:~:text=User%20Datagram%20Protocol%20(UDP)%20is,provided%20by%20the%20receiving%20party. (accessed Nov. 19, 2022).

[15]B. Lutkevich, "Transmission Control Protocol (TCP)," *SearchNetworking*, 2021. https://www.techtarget.com/searchnetworking/definition/TCP#:~:text=Transmission%20Control%20Protocol%20(TCP)%20is,of%20data%20to%20each%20other. (accessed Nov. 19, 2022).

[16]J. Juviler, "REST APIs: How They Work and What You Need to Know," *Hubspot.com*, Aug. 30, 2022. https://blog.hubspot.com/website/what-is-rest-api (accessed Nov. 19, 2022).

[17]"SOAP vs REST APIs: Which Is Right For You? | SoapUI," *Soapui.org*, 2014. https://www.soapui.org/learn/api/soap-vs-rest-api/ (accessed Nov. 19, 2022).

[18]J. Juviler, "4 Types of APIs All Marketers Should Know," *Hubspot.com*, Aug. 26, 2021. https://blog.hubspot.com/website/types-of-apis#:~:text=There%20are%20four%20widely%20agreed,internal%20APIs%2C%20and%20composite%20APIs. (accessed Nov. 19, 2022).

Yunzhou Liu 101027110
Shizhong Shang 101115304
Zirui Qiao 101100225

Appendix 1. Source Documents

Use Case Diagram Version 1
Use Case Diagram Version 2
Use Case Diagram Version 3
Use Case Diagram Version 4
Use Case Description Version 1
Use Case Description Version 2
User FlowDiagram Version 1
User FlowDiagram Version 2
User FlowDiagram Version 3
Admin FlowDiagram Version 1
Module UML
ERDiagram Version 1
ERDiagram Version 2
ERDiagram Version 3
User Front Page Prototype
User Front Page Prototype Detail
User Front Page Structure
User Front-End UML
User Back-End UML Version 1
Admin Front Page Prototype
Admin Front-End UML
API Specification Document Version 1
API Specification Document Version 2
API Specification Document Version 3
Overall C&C Structure