Chatting System in Companies

CS003 Project of Software Engineering

Hongbo, Wang
18098532-I011-0018
Yichu, Li
18098533-I011-0103
Jiayin, Li
18098537-I011-0095
Jingwen, Liu
18098537-I011-0070

January 3, 2022

Contents

1 Requirement

1.1 Problem

1. Due to the COVID-19 outbreak, many enterprises started to use the long distance to handle official business. For example, Google decided to keep this operation mode and allowed a part of staffs work in home. In fact, according to a study released by the U.S. Census, 5.2% of workers in the U.S. worked from home full time in 2017, a 3.3% increase since 2000. To put that in perspective, that's 3.7 million people working from home in 2000, increased to 6.5 million. An estimate by Upwork states that 1 in 4 Americans which is over 26% of the American workforce is expected to work remotely through 2021. Long distance working mode can cut costs and save time. Also, according to the survey, many employees would like to work from home. Thus, enterprises may need a chat system to keep communicating.

Figure 1: Statistics of working from home

2. The management groups of enterprises need an efficient communication channel to send messages or assign tasks to staffs. Thus, the enterprise should use the same chatting software to communicate.

3. The management groups should have the ability to manage accounts of their staffs. Thus, the chat system need realize unified management of accounts and provide management system to administrators of enterprises.

4. The company will have some confidential documents to be transmitted, and the security and reliability of the transmission media must be ensured.

5. For enterprises, file transfer system is an important part of communication and business. It requests the chat system should provide file transfer system to support working.

6. The departments or project teams need a convenient system to communicate with other members. The chat system should have provide group chatting service to users.

1.2 Background Information

1. Introduction: our software is called "company chatting system", which is used to provide efficient and encrypted chatting system for company. Our chat system will provide two major functions to enterprises, chatting and database management.

2. Glossary: much of the special terminology for this domain will be terminology related to communications, especially for Python socket technology.

3. Competing software: there are many popular chat systems applied in business, such as QQ and WeChat. WeChat and QQ are the giants of social messaging in China. They are equally popular and owned by a single company, Tencent. However, the main target user of these chat systems is ordinary user. Very few of them provide efficient functions to enterprises. Besides, they don't provide management system of accounts to management groups.

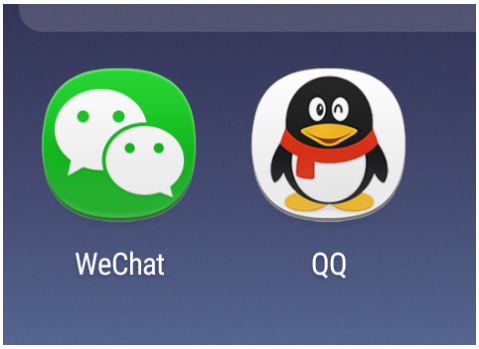


Figure 2: Logo of WeChat and QQ

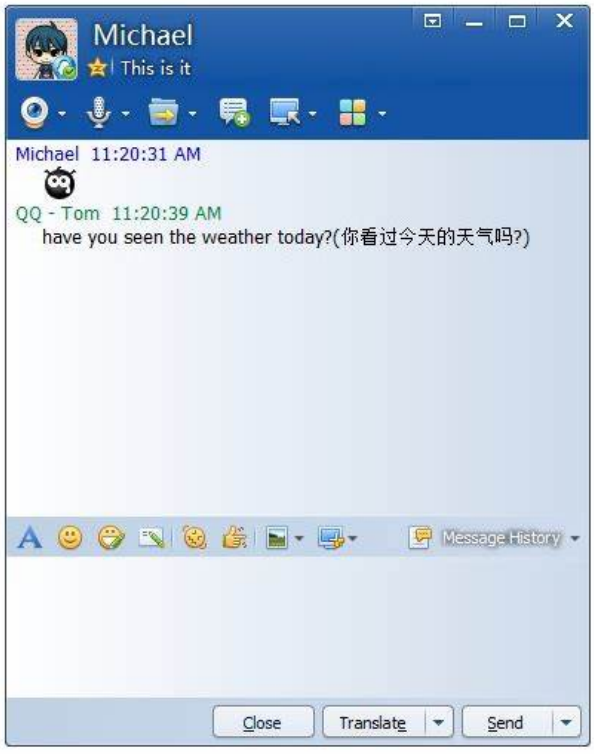


Figure 3: Chat system of QQ

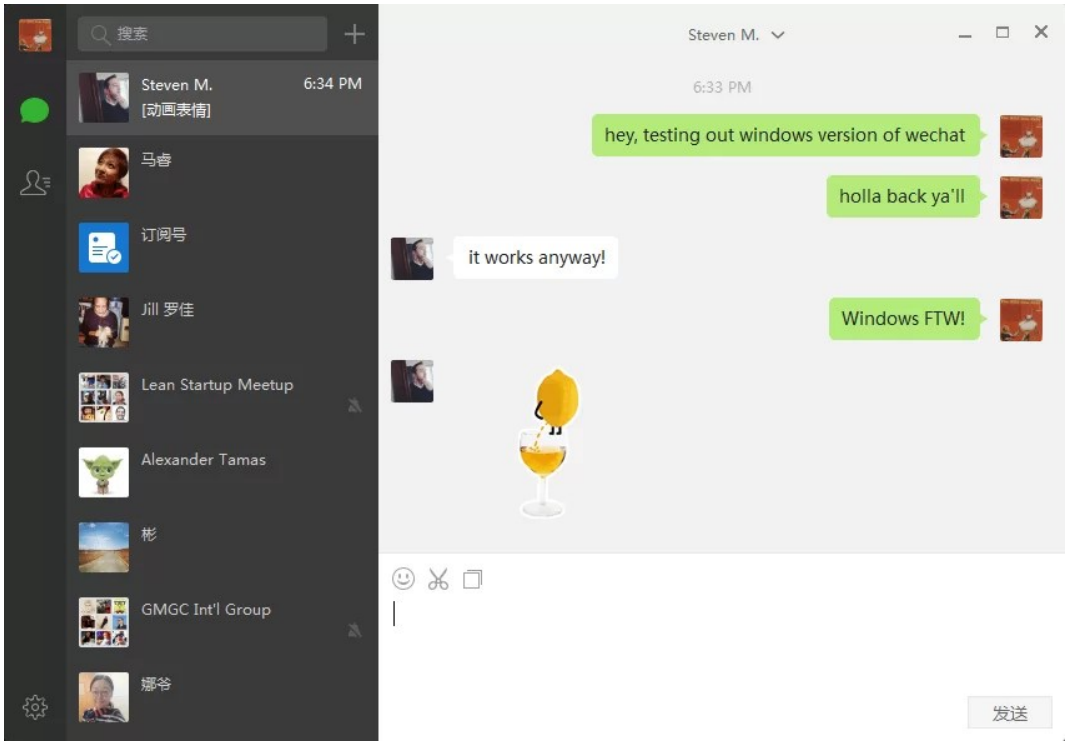


Figure 4: Chat window of WeChat

4. Customers and users: In this domain, our target users are mainly small and medium enterprises. Because the number of this kind of companies is increasing gradually, and there is a huge demand for online chatting.

1.3 Environments and System Models

It is very expensive and inconvenient for companies, especially for the small and medium ones to design their exclusive chatting software by themselves. As a result, there are a plenty of opportunities for us to march on this kind of market. In our requirement analysis, we have two subsystems in our system:

- Ordinary employee system: this system is suit for ordinary employee. This system has the ability to login and logout by username and password freely, chat (our software supports the transmission of text, pictures and files), access his/her chat history, upload and download pictures and files.
- Administrator system: this system is suit for administrator. Expect for all the functionalities of ordinary employee system, this system can also has the ability to manage the database information, including add new account, edit account and delete account.

1.4 Functional Requirements

1. Input: messages like texts, files, pictures by the sender.
2. Output: output the sent information to the receiver.
3. Users can log in freely with username and password.
4. Administrators can manage user information.
5. Support the transmission of text, pictures and files.

6. Use encryption algorithms to ensure the confidentiality of communication messages.
7. Users can freely modify their basic information.

1.5 Non-Functional Requirements

1.5.1 Quality Requirements

- Our software currently is only supports transmission within the local area network (LAN). As a result, the headquarter company cannot send message to one of its branch companies through our software.
- Response Time: less than 50ms, the encryption algorithm for message transmission would increase the delay to a certain extent.
- Capacity: the number of host computers depends on the bandwidth limitation of enterprises, which means that number of hosts communicating at the same time is limited.

1.5.2 Platform Requirements

- PC Platform: Windows, Linux, macOS
  - An operating system is considered to be the backbone of any system. Without an operating system, the user and system cannot interact. It acts as a mediator between both of these. We mainly have three kinds of operating systems, namely, Linux, MAC, and Windows. The reason is that, Windows, Linux, macOS are the three most popular operating systems in the world according to the survey below. To begin with, MAC is an OS that focuses on the graphical user interface and was developed by Apple, Inc, for their Macintosh systems. Microsoft developed the Windows operating system. It was developed so as to overcome the limitation of the MS-DOS operating system. Linux is UNIX like a source software and can use an operating system that provides full memory protection and multi-tasking operations.

Figure 5: Our three supported Operating Systems

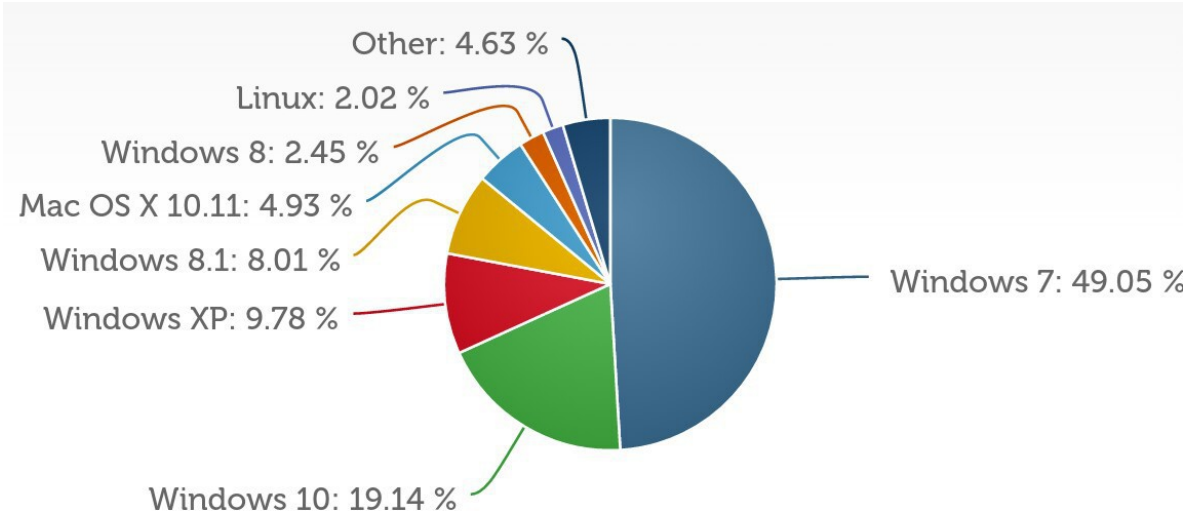


Figure 6: Indicating that Windows, MacOS and Linux are the three most popular operating systems in the world

- Programing language: Python
  - Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. When we choose our programming language, we considered many languages that we are familiar with such as C/C++, Java. We finally chose Python as our project's programing language because these reasons:

    1. Extensive Libraries

    Python downloads with an extensive library and contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually. This comes in handy, especially in projects.

    2. Improved Productivity

    The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

    3. Simple and Easy

    It is also quite easy to learn, understand, and code.

    4. Readable

    Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

    5. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

6. Free and Open-Source

Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

7. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

8. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

- Network equipment: Hub, Switch, Router, etc

  - Network equipment is used to combine, split, switch, boost, or direct packets of information along a computer or telecommunications network. This product area includes hubs, switches, routers, bridges, gateways, multiplexers, transceivers and firewalls. Here are three network equipment that we need companies to provide:

    Hubs provide a central location for attaching wires to workstations. There are two types: passive and active.

    Switches connect devices to host computers and allow large numbers of these devices to share a limited number of ports.

    Routers are protocol-dependent devices that connect sub-networks or divide a very large network into smaller sub-networks.

1.5.3 Process Requirements

- Project Plan: development lead times under 3 months

- Development Model: Scrum model in agile software development Scrum is a lightweight framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. There are many development models we can choose like the waterfall model, the phased-release model, the incremental model, the spiral model, we finally chose the Scrum model because this model has these advantages:

  Scrum helps to deliver the product the highest possible value
  Scrum can help teams carry out project deliveries in a fast and effective way
  Scrum makes sure that money and time are used efficiently
  Large and complex projects can be separated into practically manageable parts
  Improvements analyzed during the sprint review
  Scrum works well for dynamic and fast-moving project improvement
  Scrum meetings allow the team to have neat visibility

Scrum is agile as it takes and comprehends feedback given by customers and stakeholders
Brief meetings allow shifts to depend on feedback more practically
During daily scrum meetings the effort of each individual member is visible

Figure 7: Framework of our scrum model

2 Design

2.1 Purpose

Our goal is to develop an internal communication software for the company to get rid of the openness of file transmission based on the Internet, and hope that the software developed can meet the basic communication of the company's internal employees and ensure the safety and reliability of file transmission.

The company chat system is divided into three parts, namely the client, the server, and the database.

The transmission protocol of this software is TCP protocol, which realizes the transmission of information by using the socket architecture.

2.2 General Priorities

We divide the priorities of this system into 4 parts and arrange them in descending order of priority:

1. Security:

   Safety is the most important part. One is encrypted transmission. The information we send between employees is encrypted using encryption algorithms before sending to ensure the confidentiality of communications between employees. The purpose is to prevent middlemen from capturing data packets to understand the content of communication between employees.

   Secondly, chat history is not saved. In order to prevent criminals from using employee computers to obtain chat records through improper means, in the software design stage, the cache (chat records) is automatically cleared after the software is closed to ensure security.

   The server is only responsible for forwarding the message and will not be stored on the server's hard disk.

2. System response time:

   Response time is the second most important part. The response time of the system determines the degree of user satisfaction.The response time of this software includes 2 parts.

   The first part: the response time of the transmitted text message. When the user finishes writing the text, press Enter. At this time, the client sends the message to the server, and the server forwards the message content to the target user and displays it on the target user's computer. In this process, we controlled the response time within 40ms by reducing the time complexity of the program (the two computers are connected through a wireless LAN, and the distance is about 20m).

   The second part: file transfer response time. Since this software is a chat software based on LAN, the transmission bandwidth can be much higher than the Internet. We set the upload speed (user uploads files from the client to the server list) as 20MB/s, set the download speed (the user downloads files from the server's hard disk to the client) as 100MB/s. In order to ensure the balance between server load and good customer experience.

3. Simple operation:

   Easy to operate means that people of any age can easily learn to use this chat software. By realizing a beautiful chat interface, easy-to-understand button icon settings, and combining icons that people are accustomed to using in life, it meets the convenience of user groups and achieves the purpose of flexible use of this software. It is suitable for all ages.

4. Portability:

Portability is through understanding the commonality of mainstream operating systems, and finally determining the use of python development, that is, to meet the compatibility and portability of mainstream systems such as Windows, Linux, and macOS.

2.3 Outline of the Design

2.3.1 Project Architecture

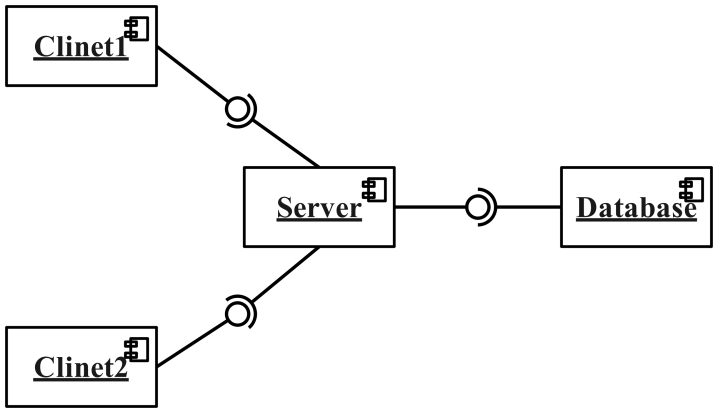


Figure 8: The distributed system of chatting room

This is a distributed system diagram of the company's chat system. It is divided into 3 parts, namely client, server and database. The role of the client is to initiate a connection to obtain services such as logging in and sending information, the role of the server is to wait and then process the connection, and the role of the database is to match employee information and other services

2.3.2 Client Architecture Details

2.3.2.1 Staff

Functions:

1. Login: Establish a communication connection with the server and verify that the account password is correct.

2. Send SMS: After writing the SMS, send it to the server for forwarding operation.

3. Send emoticons: select the corresponding emoticons, and the server will forward them after sending.

4. Send picture: Drag and drop to send the picture, the server will forward the picture after sending.

5. Send files to users: Send files by dragging and dropping, and the server will forward the files after sending.

6. Upload a file to the server: Click the file button to upload the specified file to the server, and the entire company can download it.

7. Download the file to the client: click the file button to download the specified file to the local.

8. Change personal information: Click on your avatar to open the personal information form, and the server will save and send it after the change is completed.

9. View other people's information: Click on the person's profile picture to open the designated employee information table to view the employee's basic information.

10. Change the font size: By clicking the font size button, you can change the text size of the interface to achieve a good interactive interface.

11. Search for employees: You can use the search button to search for employees whether they exist or are online.

12. Exit: Click the close button in the upper right corner to exit the chat software.

User Interface:

1. Interface: Create a user-friendly interface.

2. Avatar: Display the employee's personal avatar on the software to facilitate identification and search by others.

3. Chat bubbles: Use chat bubbles to make the software interface look more beautiful and tidy.

2.3.2.2 Admin

Functions:

1. Login: Establish a communication connection with the server and verify that the account password is correct.

2. Add employee: Add the user name and password of the employee through the administrator UI interface.

3. Delete employee: delete the user name and password of the employee through the administrator UI interface.

4. Modify employee: through the administrator UI interface, modify the employee's user name and password.

5. Delete all employees: delete all employees in the database through the administrator UI interface.

6. Refresh the database: refresh the database interface through the administrator UI interface.

2.3.3 Server Architecture Details

Functions:

1. Match user name and password: Receive client information, and match the user name and password according to the database to see if they are correct.

2. Forward SMS: Receive the sender's SMS and forward it to the recipient.

3. Forward emoticon package: receive the sender's emoticon package information and forward it to the receiver.

4. Forward the picture: Receive the picture information from the sender and forward the information to the receiver.

5. Receive uploaded files: Receive the sent files and save them to the server.

6. Send the downloaded file: Send the server-side file to the recipient.

7. Change personal information: Receive personal information changed by employees and update the employee information database.

8. Refresh the list in real time: When an employee logs in, all clients refresh the online list in real time.

9. Real-time face refresh: When an employee changes his avatar, all clients refresh the employee's avatar in real time.

10. Send database: send the database to the administrator client.

2.3.4 Database Architecture Details

2.3.4.1 Administrator Database

Functions:

1. Store user name: store the user name in the database as the login user name.

2. Store password: store the password in the database as the login password.

3. Add creation time: save the creation time in the database as a note.

4. Add employee function: execute the request of adding employee information sent by the server.

5. Delete employee function: execute the request to delete employee information sent by the server.

6. Update information function: execute the request to update the account and password sent by the server.

2.3.4.2 Employee Personal Information Database

Functions:

1. Store user name: store the user name in the database as employee personal information.

2. Store position information: store the position as employee personal information in the database.

3. Store phone information: store the phone as employee personal information in the database.

4. Store mailbox information: store the mailbox as employee personal information in the database.

5. Store avatar information: store avatars in the database as employee personal information.

6. Update employee personal information: execute the request to update employee information sent by the server.

2.4 Major Design Issues

There is not enough budget. In the development process, it is impossible to build a high-performance server without sufficient budget. We can only use a laptop as a server, and you can barely use it. After going online, there may be some bugs in server compatibility.

Few test physical machines. There are not enough physical machines to check the compatibility of the software, and it may not be compatible with some lower versions of Windows and Linux operating systems.

2.5 Other details of the design

The algorithm used by the software is the DES encryption algorithm, which is a symmetric cipher developed by IBM in 1972. There are three entry parameters in the software: key, data, mode. key is the key for encryption and decryption, data is the data for encryption and decryption, and mode is the working mode. The advantage of DES is that the key is short, the encryption process is simple, and the encryption and decryption speed is fast. It is suitable for occasions where large amounts of data are encrypted.

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration

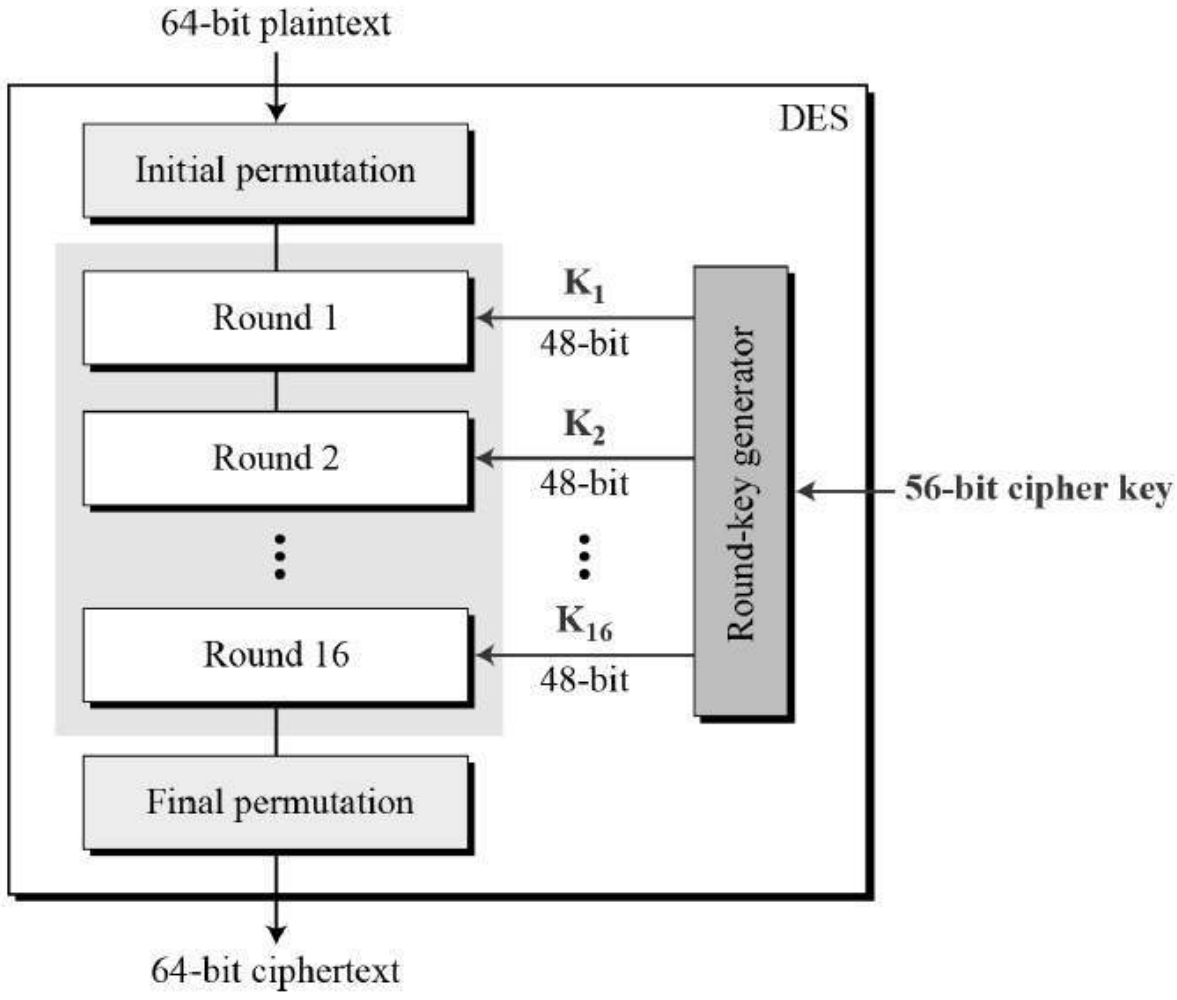


Figure 9: Schematic diagram of DES

2.6 Class Diagram

2.6.1 Server Class Diagram

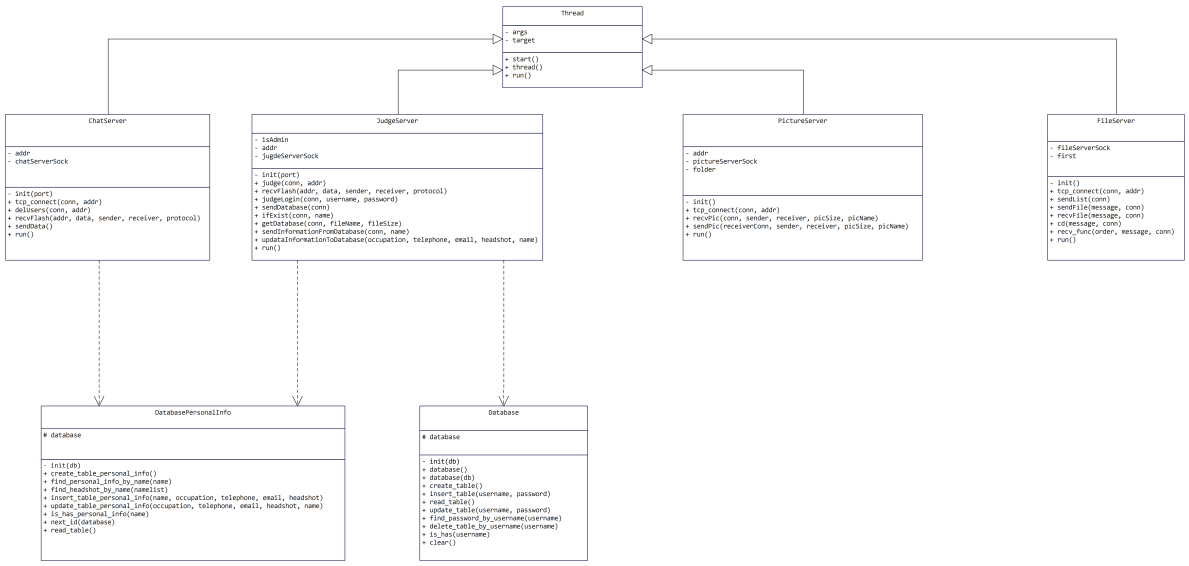


Figure 10: The class diagram of server

This is the class diagram on the server side. There are 7 classes in total.

Among them, ChatServer, JudgeServer, PictureServer, and FileServer use multi-threading technology, so these classes all inherit the methods in the Thread class.

Both ChatServer and JudgeServer need to rely on functions in the DatabasePersonalInfo class.

JudgeServer also needs to rely on functions in the Database class.

1. ChatServer: Realize the functions of Message, emoticon package sending, online list real-time refresh and so on.

2. JudgeServer: Realize the functions of receiving and sending whether the username and password match, and real-time refreshing of the avatar.

3. Picture server: realize functions such as sending pictures by employees.

4. File server: realize the functions of staff uploading and downloading files.

5. DatabasePersonalInfo: Realize functions such as storing employee personal information in the database.

6. Database: realize functions such as adding employee information, matching user names and passwords.

2.6.2 Client Class Diagram

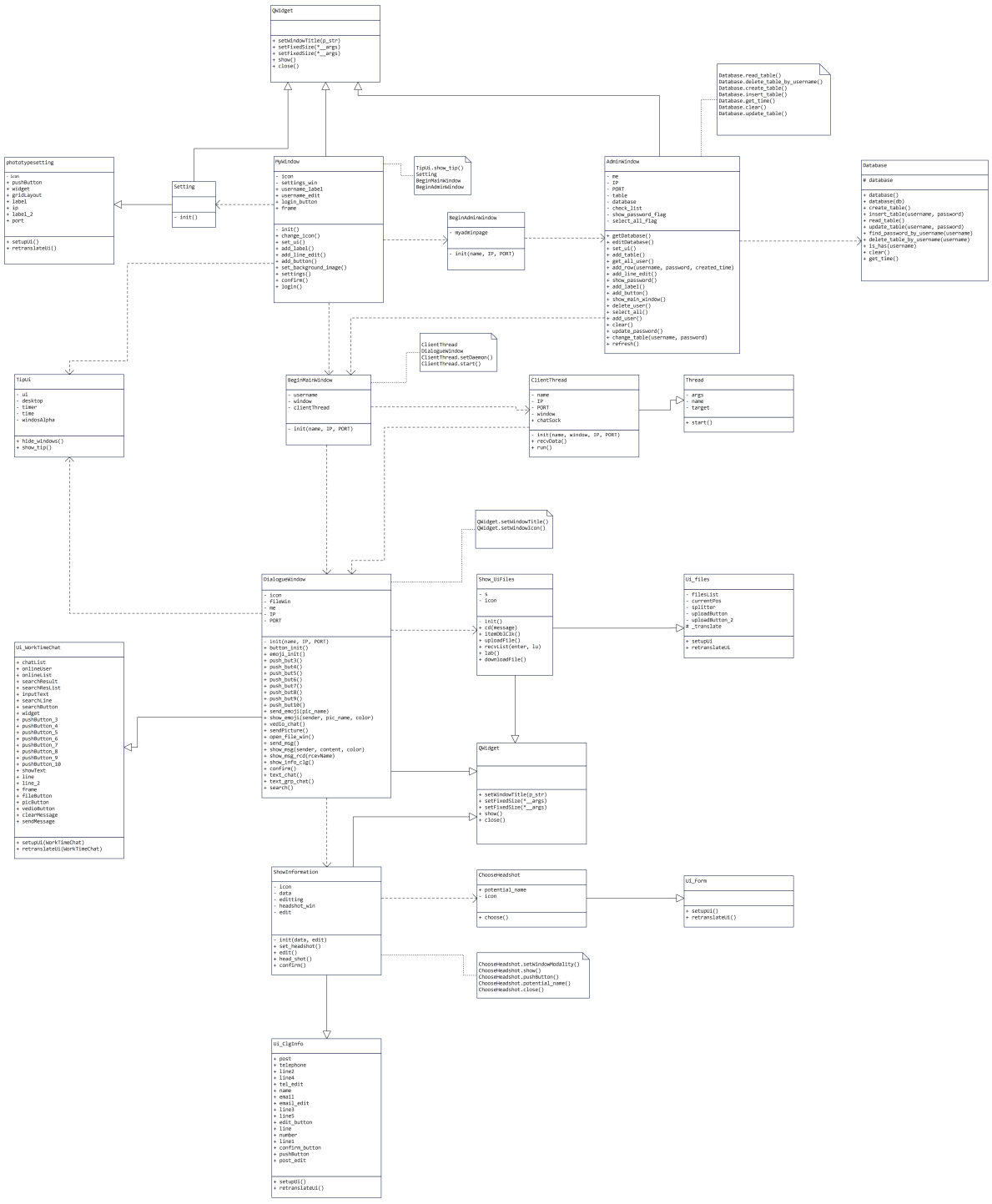


Figure 11: The class diagram of client

This is the class diagram of the client, there are 19 classes in total. Among them, Setting, MyWindow, and AdminWindow all use the functions of PyQt to create the client interface, so these classes all inherit the methods in the QWidget class.

Phototypesetting uses some fixed setting methods, so it inherits the setting class.

ClientThread uses multi-threading technology to monitor the information sent by the server in real time, so it needs to inherit the Thread class.

DialogueWindow, ShowUiFile, ShowInfomation create methods on the user interface, so they inherit the Qwidget class.

DialogueWindow inherits the UiWorkTimeChat class, and its purpose is to initialize the position and color of some buttons.

ShowUiFile inherits UiForm, the purpose is to realize the initialization of the basic settings related to the avatar.

ShowInformation inherits the UiClgInfo class, the purpose is to realize the initialization of basic employee information.

MyWindow needs to rely on the functions of the Setting class, BeginAdminWindow class, and BeginMainWindow class

BeginAdminWindow needs to rely on the function of AdminWindow AdminWindow needs to rely on the Database class and the BeginMainWindow class to implement functions.

The BeginMainWindow class needs to rely on the functions of the DialogueWindow class and the ClientThread class.

1. Qwidget: a function library that can create independent windows
2. Typesetting: Realize the setting of avatar
3. Setting: play sub-category to specify the avatar in detail
4. MyWindow: Realize the settings of the login interface
5. BeginAdminWindow: call the transition class of AdminWindow
6. AdminWindow: The interface for the administrator to manipulate data
7. Database: a class that implements related data operations
8. TipUi: Realize some auxiliary functions of the employee chat interface
9. BeginMainWindow: the transitional class that calls the employee chat interface
10. ClientThread: A class that implements list refresh and receiving server information
11. Thread: a function library about multithreading
12. UiWorkTimeChat: Realize setting information such as emoticon package location, UI style, etc.
13. DialogueWindow: Realize the main interface for user chat
14. ShowUiFiles: realize file sending and receiving function
15. Uifiles: Implement UI settings for sending files
16. ShowInformation: realized the function of displaying the received message to the chat interface
17. ChooseHeadshot: Realize the UI design of choosing the avatar
18. UiForm: Help ChooseHeadshot to implement UI design
19. UiClgInfo: UI design and underlying implementation of employee personal information

2.7 Sequence Diagram

2.7.1 Staff Sequence Diagram

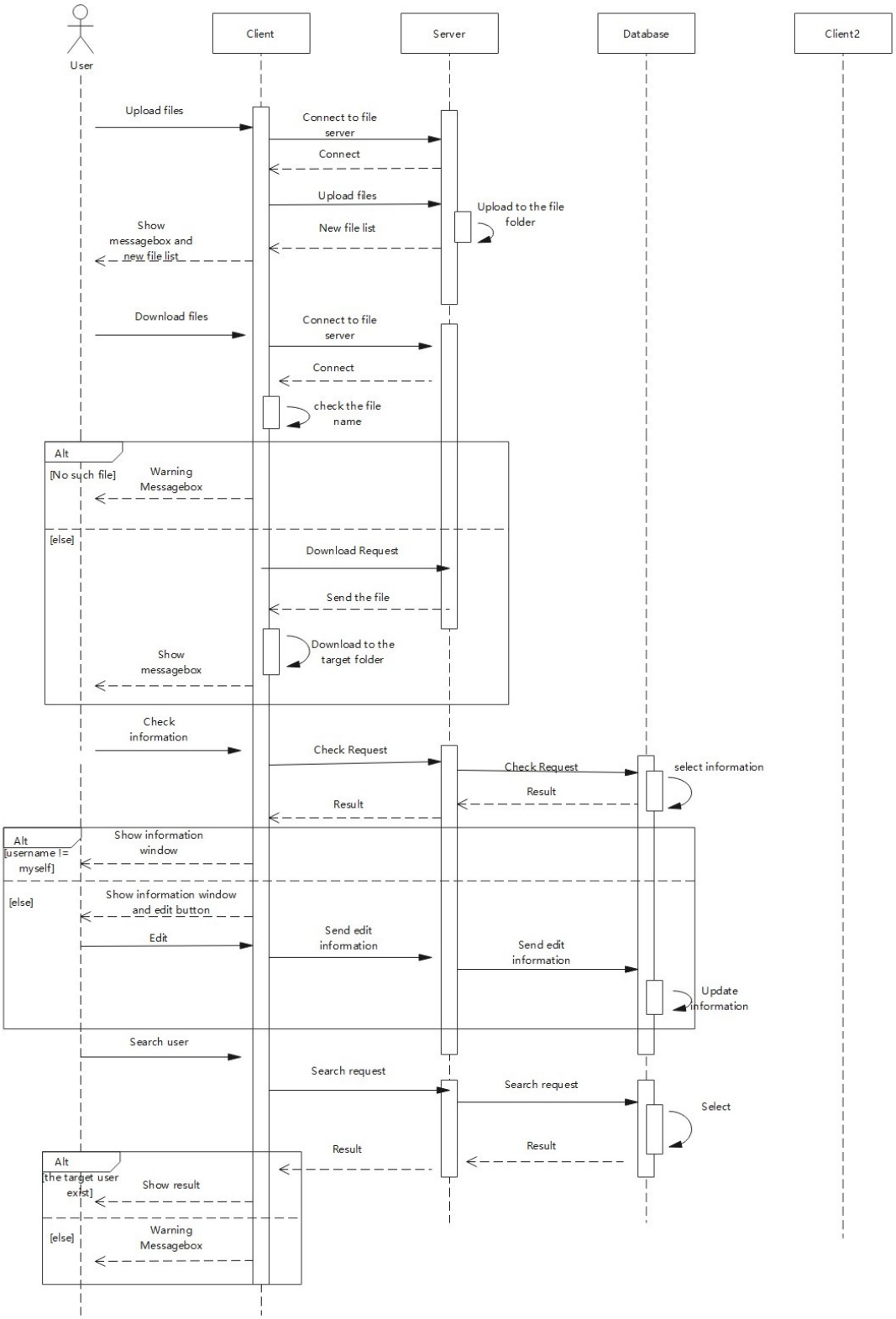


Figure 12: The sequence diagram of user

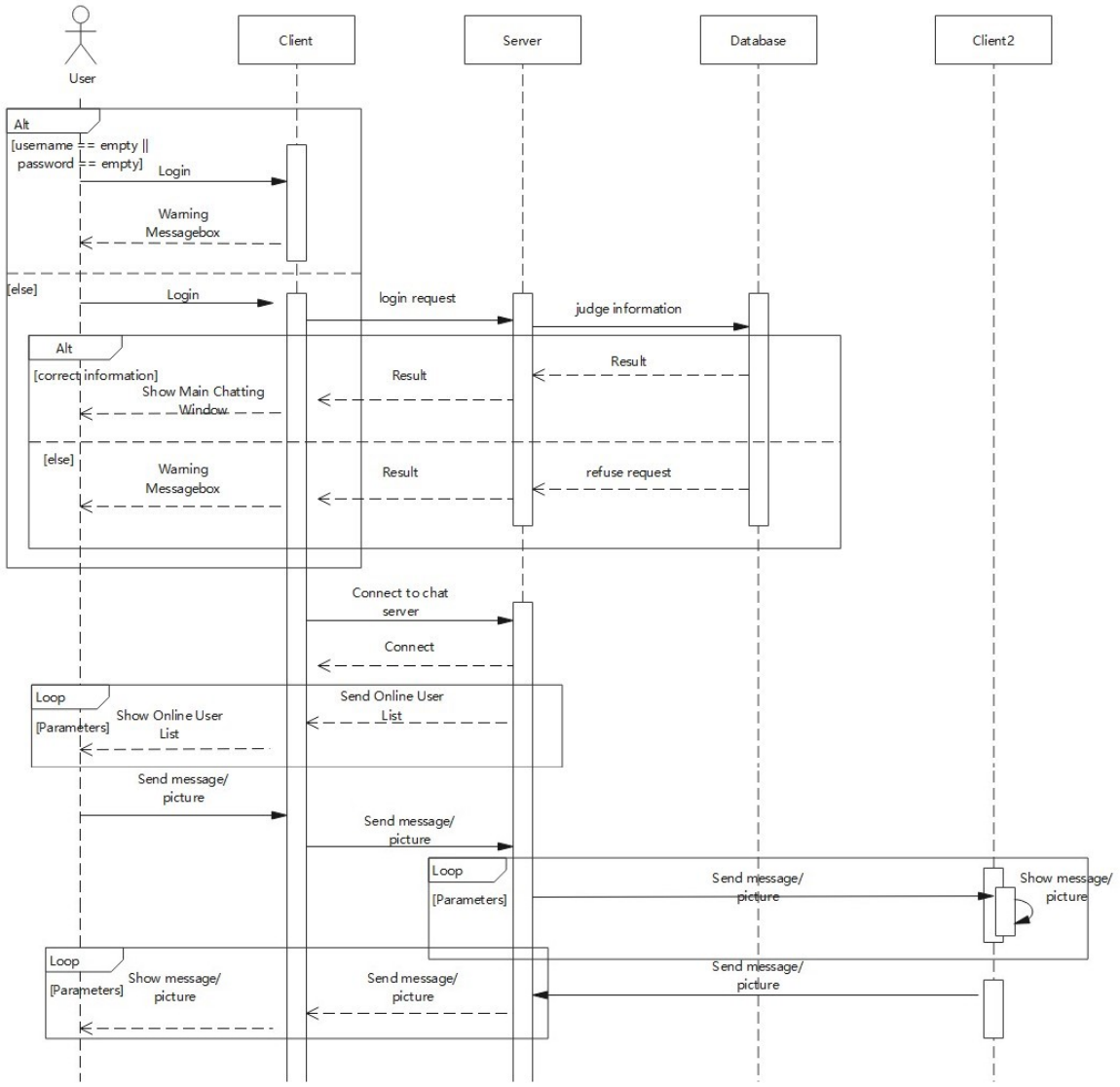


Figure 13: The sequence diagram of user

These diagrams are the process of users chatting. First, when user login into the system, the login request will be sent from client to server. Then, server will send and check the username and password. If the information is wrong, it will show the warning message box. Users can use main window chat with others, upload and download files, check information of others and edit information of herself or himself. All requests will send from client to the server. If user want to check information or edit data, the request will send from server to database and database will send back information or save new data.

When user chat or send pictures or files to other user, this request will be sent from client to server, then, server will send it to the other client that the user chat with.

2.7.2 Admin sequence Diagram

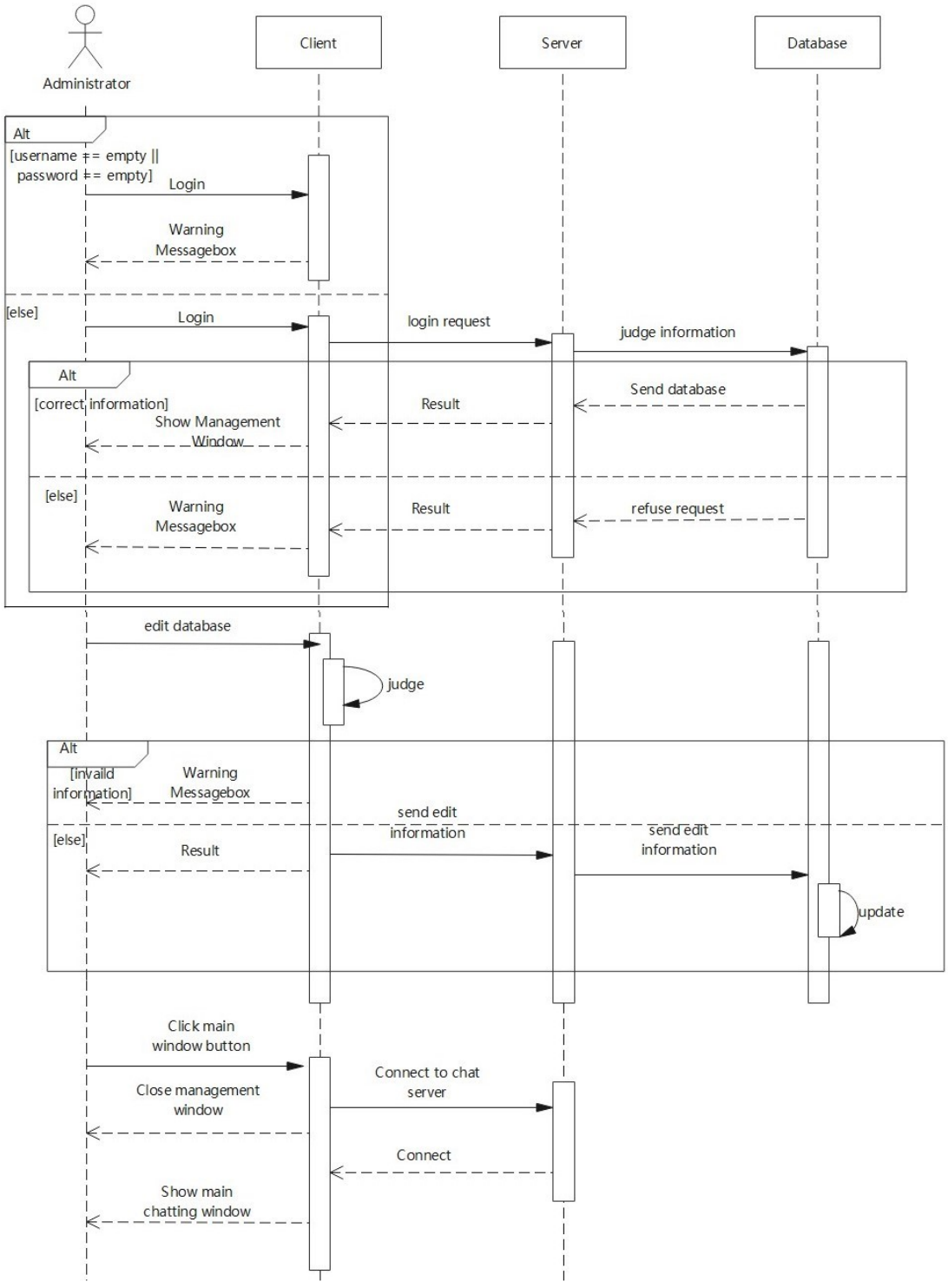


Figure 14: The sequence diagram of admin

This diagram is the process of administrator management system. First, when administrator login into the system, the login request will be sent from client to server. Then, server will send and check the username and password. If the account is the administrator, it will show the management window. If the information is wrong, it will show the warning message box.

Then, administrator can create, edit and delete accounts in the window. The client will send request to server, server will send and save the data in the database.

When the administrator clicks the “main window” button, it will show the main chat window.

2.8 Use Case Diagram

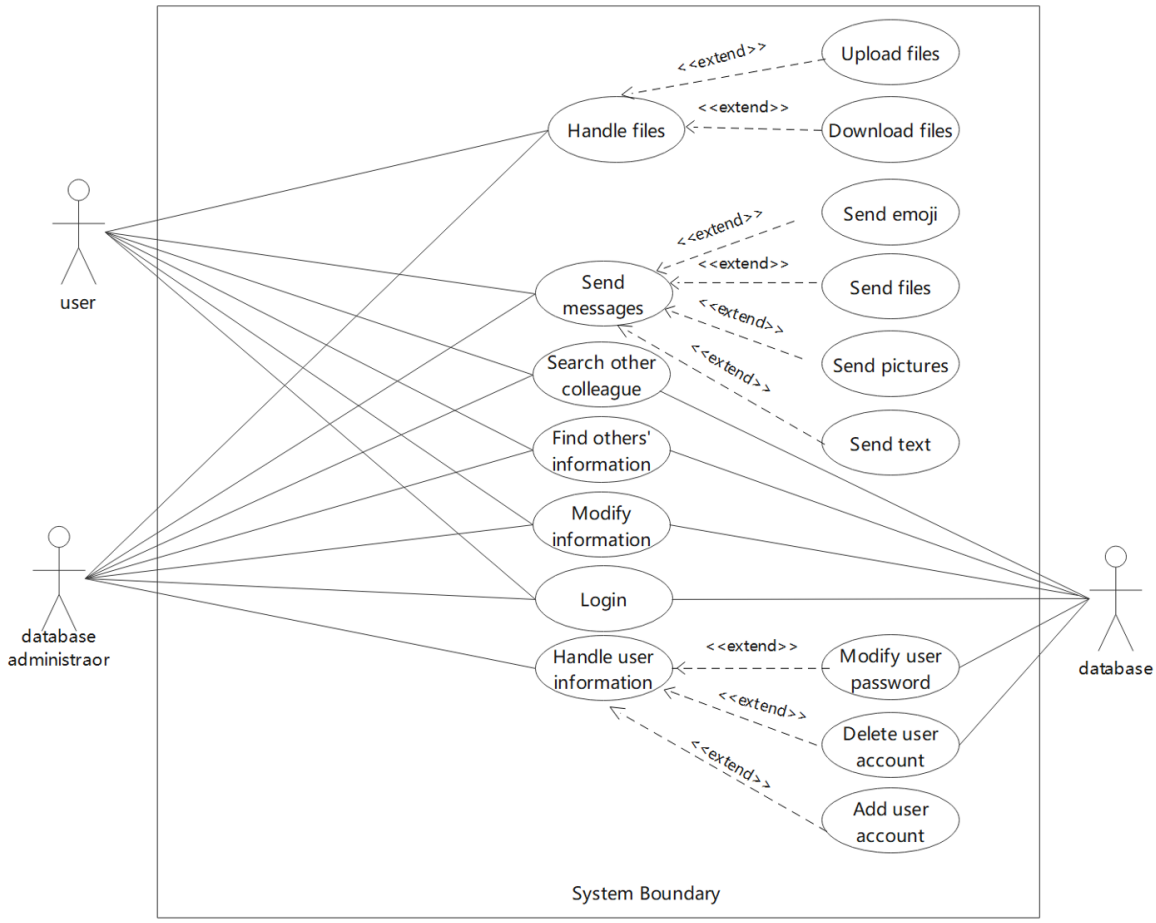


Figure 15: The user case diagram

There are two characters, user and database administraor. The user can login,send text message, emoji, files and pictures, upload and download files, search other colleagues and see their information and modify self information. The database administraor can not only do what normal user can do but can also handle user information, including add / delete user account and modify user’s password. Besides, handling user information, finding others’ information, searching other colleagues, modifying self information and login will make the client has interactions with database.

2.9 Flow Chart

2.9.1 The flow of login

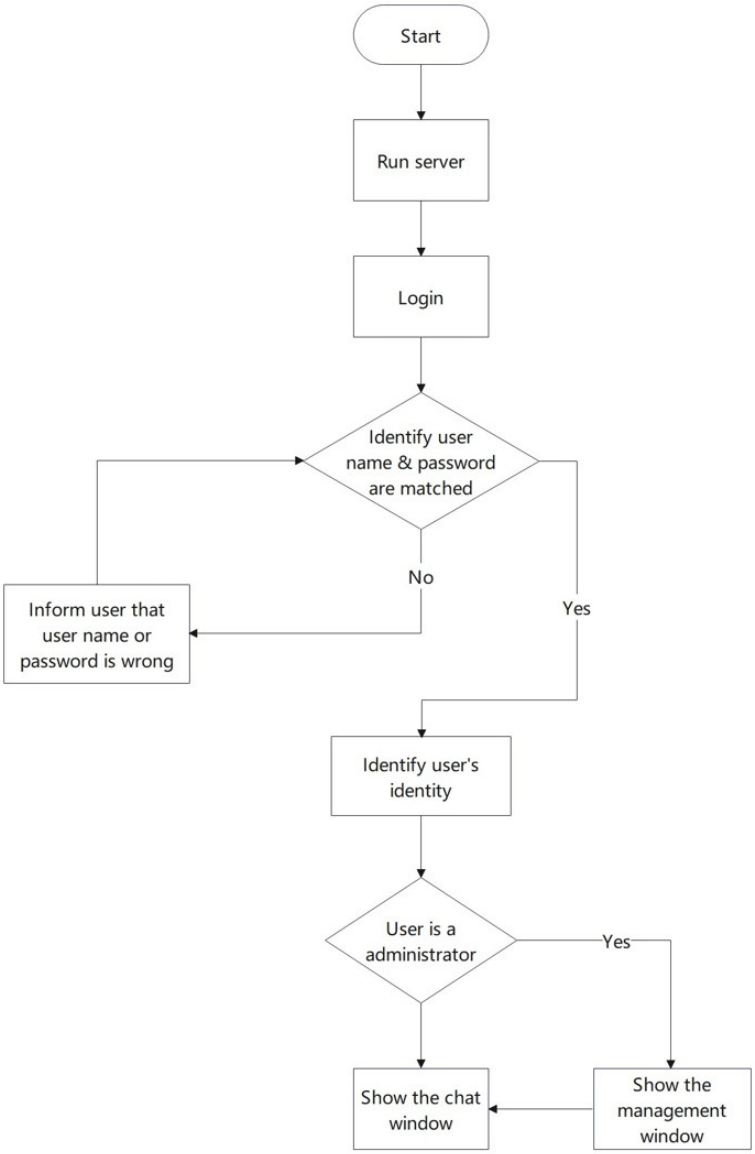


Figure 16: The flow chart of login

This flow chart explains the process of login system. The server needs run before any user can login. After server runs successfully, users can enter the login system. Then, they need to enter their username and password. After that, system will determine whether the name and password follow the standard format and the pair of username and password are matched according to our database. If not, the system will give a warning window and ask the user to enter again. Otherwise, the user is allowed to login. Then, our system will determine the identity of the user. If the user is an administrator, system will show the management window, else the user is a regular user and show the chat window.

2.9.2 The flow of sending files

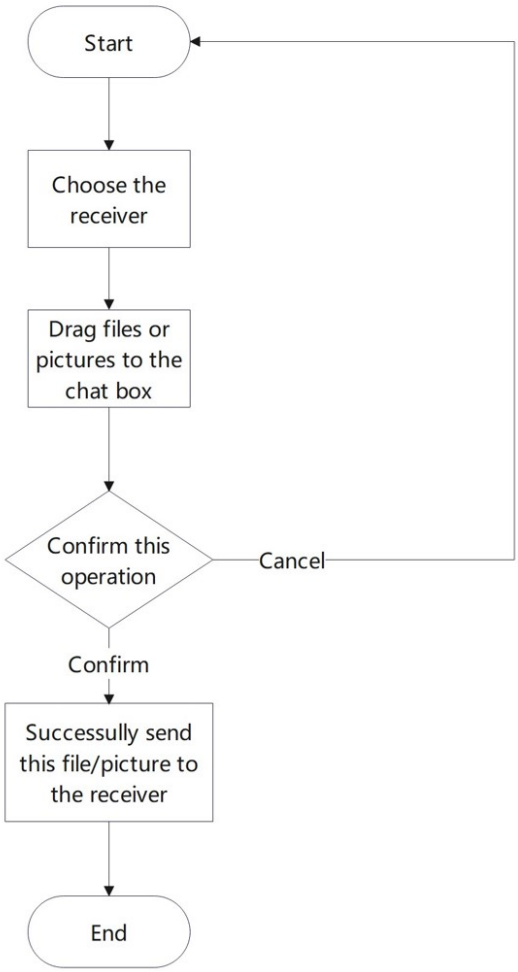


Figure 17: The flow chart of sending files

The user needs to choose a specific receiver before sending any file. Then, the user can choose to drag files or pictures to the chat box. Then, our system will pop a window to confirm the operation of sending files or pictures. If user cancels the operation, the operation will be terminated and return to the beginning. Otherwise, the user confirms the operation and then our system will help the user to send the files or pictures to the target user.

2.9.3 The flow of editing personal information

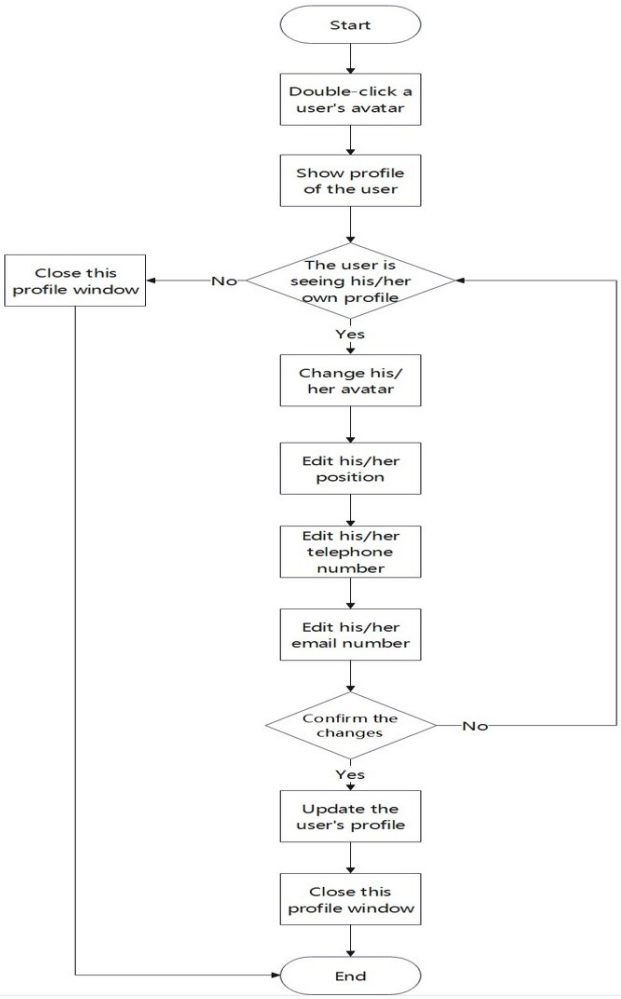


Figure 18: The flow chart of editing personal information

Before user see someone's profile, the user needs to double-click the user's avatar. Then our system will show the profile of the user. Then, if the system finds it is not the user's own profile, the user cannot change the profile. The only thing he/she can do is to see the user's profile window until he/she close the window. Otherwise, if the user is seeing his/her own profile, he/she can change his/her avatar, edit his/her position, telephone number or email number. Then, if the user confirms the changes, the information would be updated and uploaded to the database.

3 Development

3.1 Product Introduction

Our product is an enterprise-facing chat system (including server, client and database). Users can use our system to chat or send files and pictures to others in a safe network environment. Besides, administrators can manage accounts of their employees.

We tried to provide more convenient communication services to companies to enhance their operation effectiveness. This system can be a platform for communication and work for employees and leaders.

3.2 Development Method

Our project uses agile development method "scrum model"(3.2).

This agile model can divide a project into many parts and every part will be completed in different sprints.

In every sprint, we can complete a usable program and test the demo.

This model can effectively make sure that the program will be completed step by step and totally completed on time.

Every team member will have his or her own position. It will improve our team efficiency.

Figure 19: Scrum Model

3.3 Team Members

Our team has four members and every member has his or her special duties.

1.Product Owner: Li Jiayin

Design the product roadmap and backlogs

Decide the developing period and tasks in every sprint

Keep clear objectives of the project

Organize meetings to keep communicating

Functions of upload and download files

Functions of online users list and search

2.Scrum Master: Li Yichu

Ensure that the development process fits the scrum method

Communicate with the product owner to confirm tasks of every sprint

Assign detailed tasks to team members

Test program and fix bugs

Login system and management system

Develop the database

3.Developer and UI Designer: Liu Jingwen

UI design

Functions of chatting

Functions of user information

Functions of emoji and pictures

Functions of file transfer

Test program and fix bugs

4.Developer and Server Architect: Wang Hongbo

Server architecture

Develop database

Design transport protocol

Functions of file transfer

Functions of pictures

Functions of management system

3.4 Product Roadmap

We divided our project into 3 sprints and designed 3 milestones. Before we started our project, we discussed requirements of customers and designed many epic stories to ensure the requirements. Then, we divided these epic stories in different milestones.

The distribution of epic stories depended on following roles:

1. Priority of necessary functions
2. Balance of workloads
3. Enough time

Finally, all epic stories were divided into 3 milestones.

In milestone 1 (10.11 – 10.31), we will complete all necessary functions of a chat system (such as login, chat, upload and download files)

In milestone 2 (11.1 – 11.14), we will complete our database and a management system for administrator. Besides, users can edit their personal information and check information of others.

In milestone 3 (11.15 – 12.5), we will develop our UI and detailed functions.

Every milestone will have a sprint to complete its requirements.

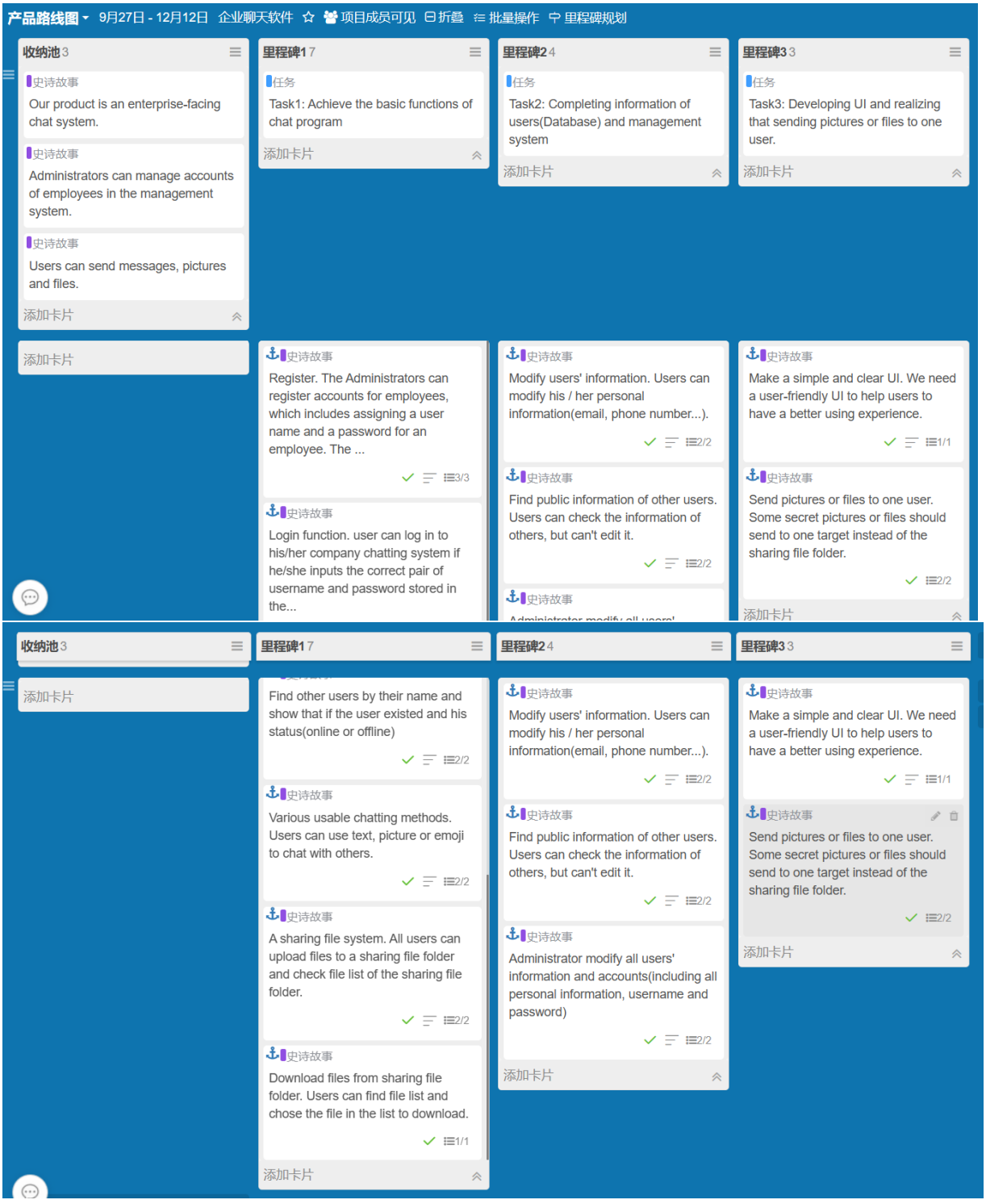


Figure 20: Product Roadmap

3.5 Milestone

We designed 3 milestones to complete different kinds of requirements of users. Every milestone will spend 2-3 weeks to complete. Besides, we added lots of user stories to clear and definite the requirements of epic stories.

3.5.1 Milestone 1 – Basic Functions

In this milestone, we have 6 epic stories and 7 user stories. All basic functions of the chat system are divided in this milestone.

Duration: 10.11 – 10.31

Introduction of Epic Stories:

- Login: Login function. user can login to his/her company chatting system if he/she inputs the correct pair of username and password stored in the database. Then, according to the identity (regular user or administrator), the system can display different functions.

- Register: The Administrators can register accounts for employees, which includes assigning a user name and a password for an employee. The data would be sent to the database after the administrators' registration.

- Search: Find other users by their name and show that if the user existed and his status(online or offline) Chat: Various usable chatting methods. Users can use text, picture or emoji to chat with others.

- Upload A sharing file system. All users can upload files to a sharing file folder and check file list of the sharing file folder.

- Download: Download files from sharing file folder. Users can find file list and chose the file in the list to download.

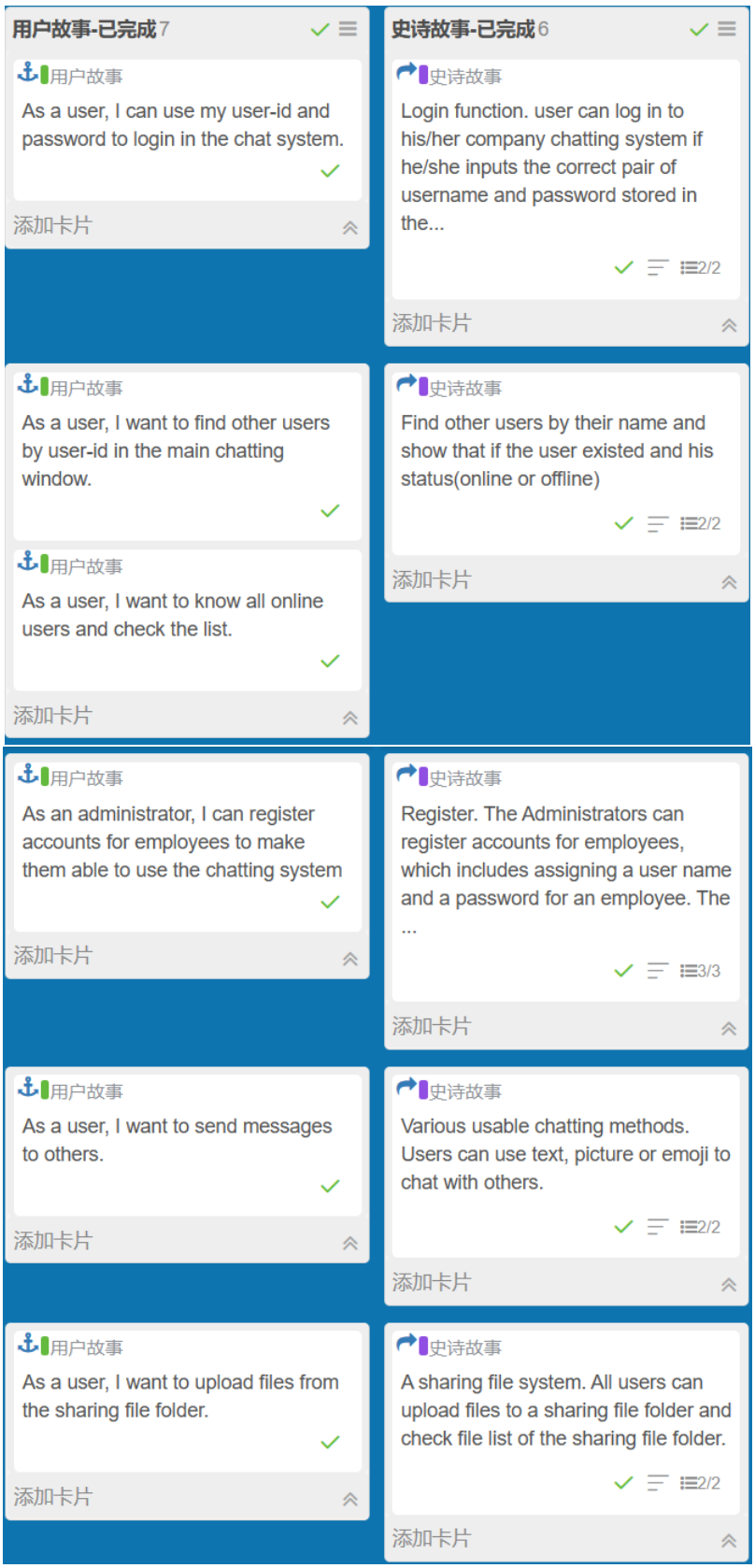


Figure 21: leangoo of Milestone 1 1/2

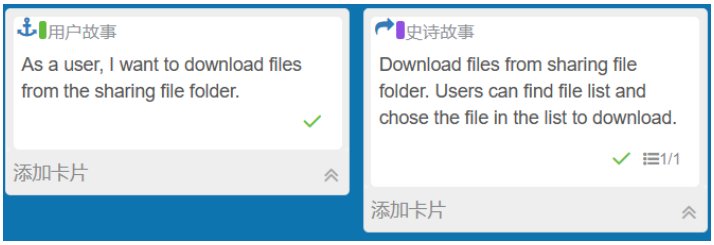


Figure 22: leangoo of Milestone 1 2/2

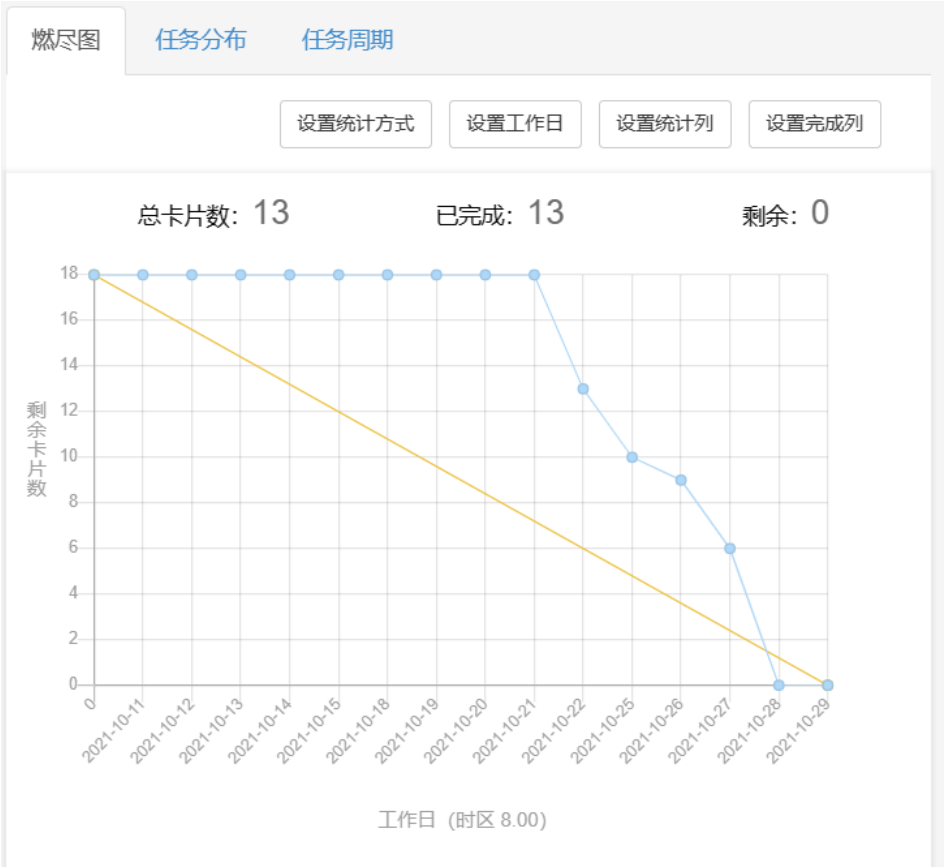


Figure 23: Burn-Down Chart of Milestone 1

3.5.2 Milestone 2 – Management System and Database

In this milestone, we have 3 epic stories and 5 user stories. We will complete the database (adding the information of users) and start to program the management system for administrators.

Duration: 11.1 – 11.14

Introduction of Epic Stories:

- User Information: Modify users' information. Users can modify his / her personal information (email, phone number...).
- Check Information: Find public information of other users. Users can check the information of others, but can't edit it.

- Management System: Administrator modify all users' information and accounts(including all personal information, username and password)

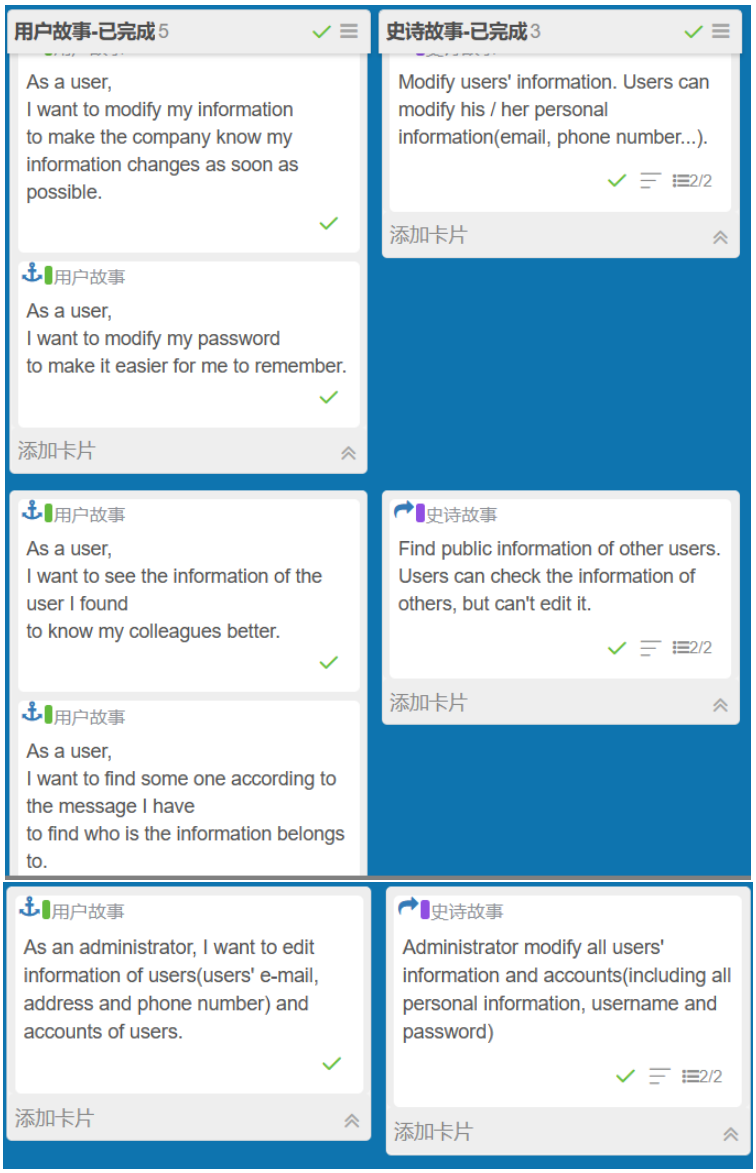


Figure 24: leangoo of Milestone 2

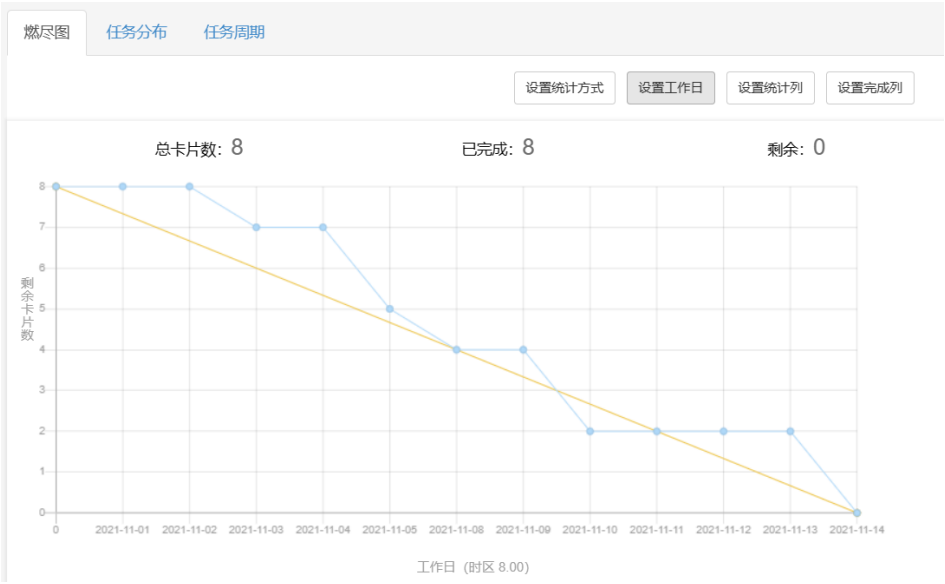


Figure 25: Burn-Down Chart of Milestone 2

3.5.3 Milestone 3 – Advanced Functions

In this milestone, we have 2 epic stories and 4 user stories. We will develop the details of functions and design a better UI. Besides, we try to optimize the transmission functions of files and pictures.

Duration: 11.15 – 12.5

Introduction of Epic Stories:

- UI: Make a simple and clear UI. We need a user-friendly UI to help users to have a better using experience.
- Transmission: Send pictures or files to one user. Some secret pictures or files should send to one target instead of the sharing file folder.

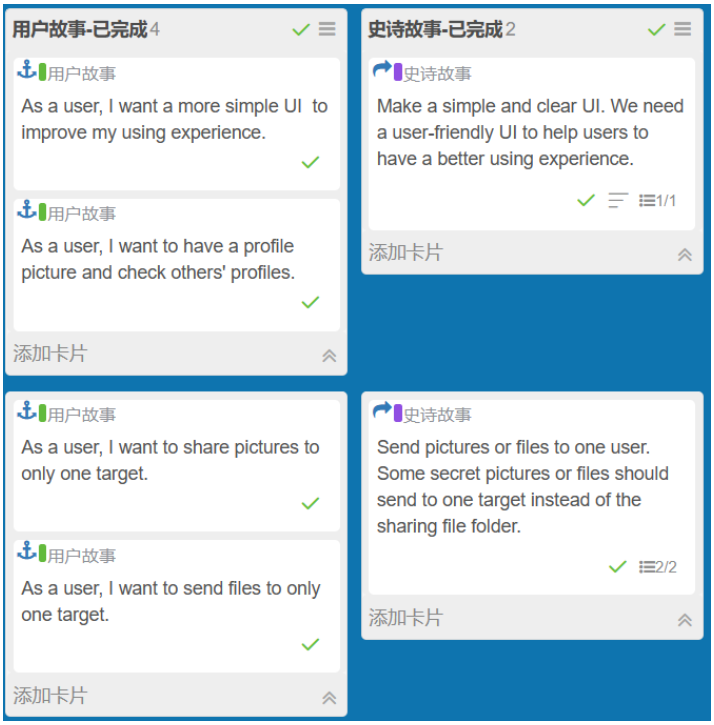


Figure 26: leangoo of Milestone 3

Figure 27: Burn-Down Chart of Milestone 3

3.6 Sprint

We divided user stories into 3 sprints. Every sprint we will complete a milestone. We designed lots of tasks for user stories and assigned these tasks to every team member.

At the beginning of sprints, we will have a sprint planning meeting to discuss detailed functions we should complete in this sprint.

During the sprint, we will have daily stand-up meeting every 2-3 days to communicate the process of tasks and discuss other things.

At the end of sprints, there will have a feedback meeting to show the demo and sum up the achievements and shortcomings of the project.

Finally, we will have a retrospective meeting to reflect the performance of this sprint and improve it in the next sprint.

3.6.1 Sprint 1 - Basic Functions

Duration: 10.15 - 10.29

3.6.1.1 Sprint Planning

In this sprint, we have 7 user stories and 8 tasks. We assessed the importance of every function and designed following tasks.

Tasks:

- Login:
  - Task 1: Complete the login function. Users can use their IDs and password to login into the chat program.
  - Task 2: If the user doesn't have an account in the database, the user can go to the registration page from the login page.
- Register:
  - Complete the register function. Users can create new accounts and save their IDs and password in the database.
- Search:
  - Complete the user search function. Users can be searched by their user-id and their information will be shown.
- Chat:
  - Complete the sending message function.
- Upload:
  - Uploading files from the sharing file folder in the server.
- Download:
  - Downloading files from the sharing file folder in the server.
- User List:
  - Showing the online user list on the main chatting window of client.

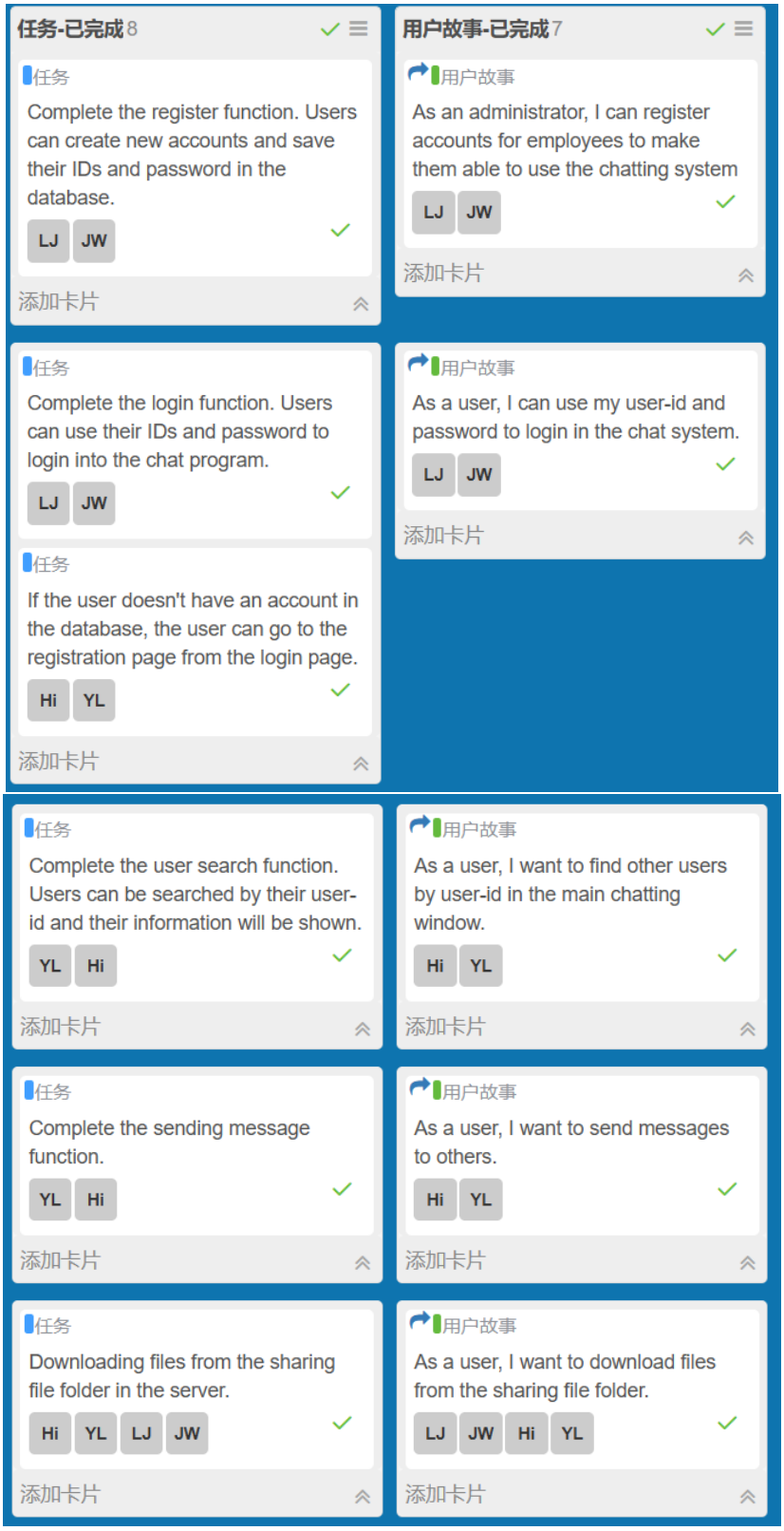


Figure 28: leangoo of Sprint 1 1/2

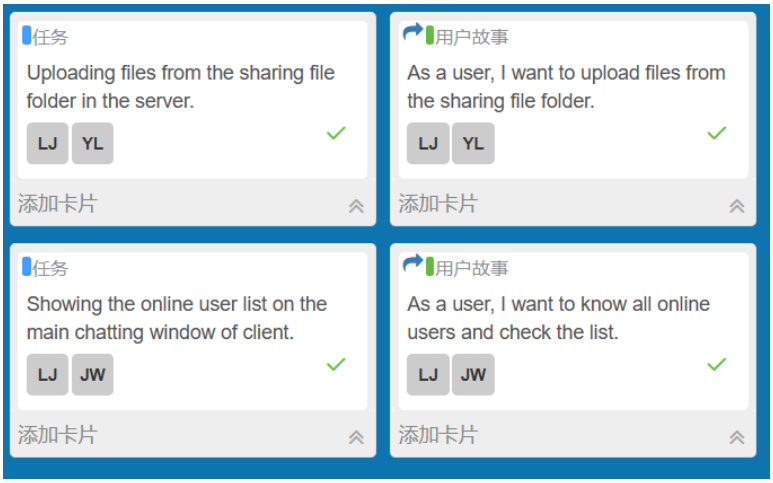


Figure 29: leangoo of Sprint 1 2/2

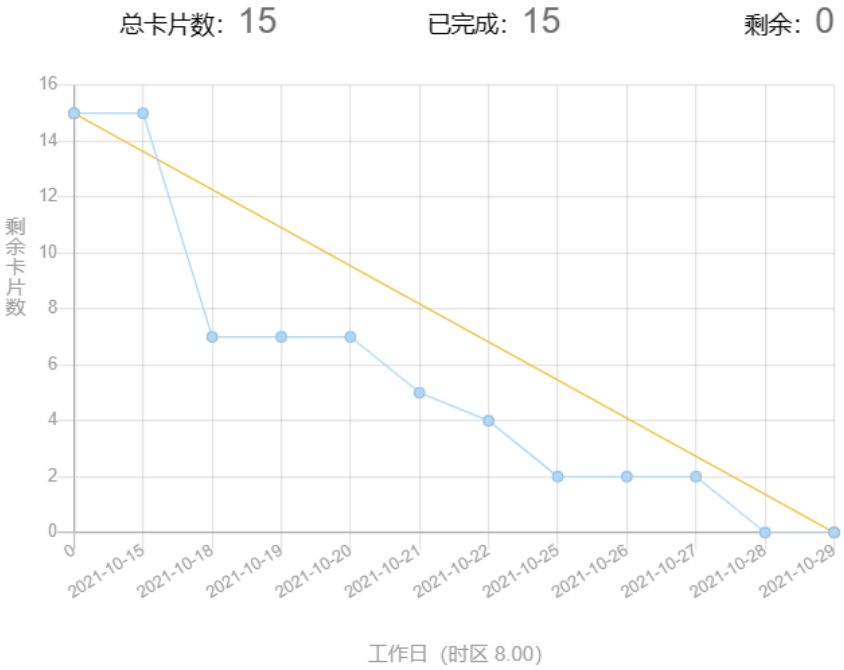


Figure 30: Burn-Down Chart of Sprint 1

3.6.1.2 Daily Stand-up Meetings

We discussed how to build our server and database and decided the structure of UI.

1. We will use four servers (judge server, chat server, picture server, file server) to connect with clients to achieve different functions. (10.15 – 10.20)

2. The database will only store username and password of users in the sprint. (10.21 – 10.24)

3. We need 3 UI pages. (Login, main window, management window) The different rights of normal users and the administrator. (10.25 -10.29)

3.6.1.3 Demo

We tested the demo of sprint1 (login, chat, upload and download files……). We successfully completed all requirements of user stories and made a simple UI to test our project.

The demo of sprint 1 can realize login, chat, check user list, upload and download functions.

The weakness is that our UI is a rather crude one, it need improve in the sprint 3. Besides, the program will become unresponsive sometimes.

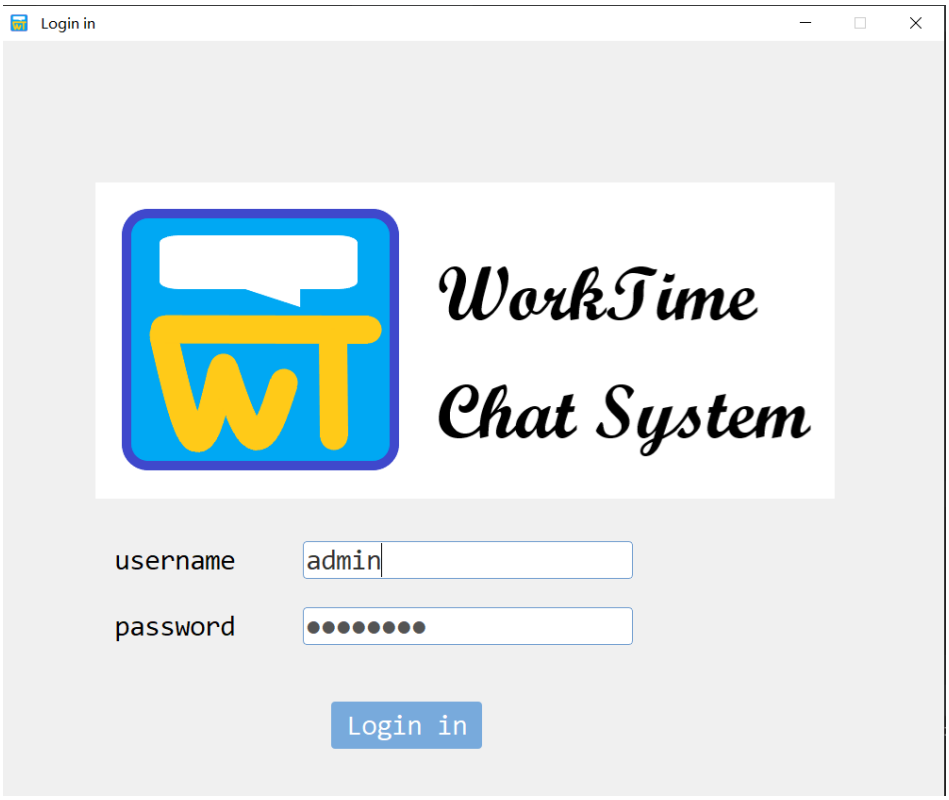


Figure 31: Login window

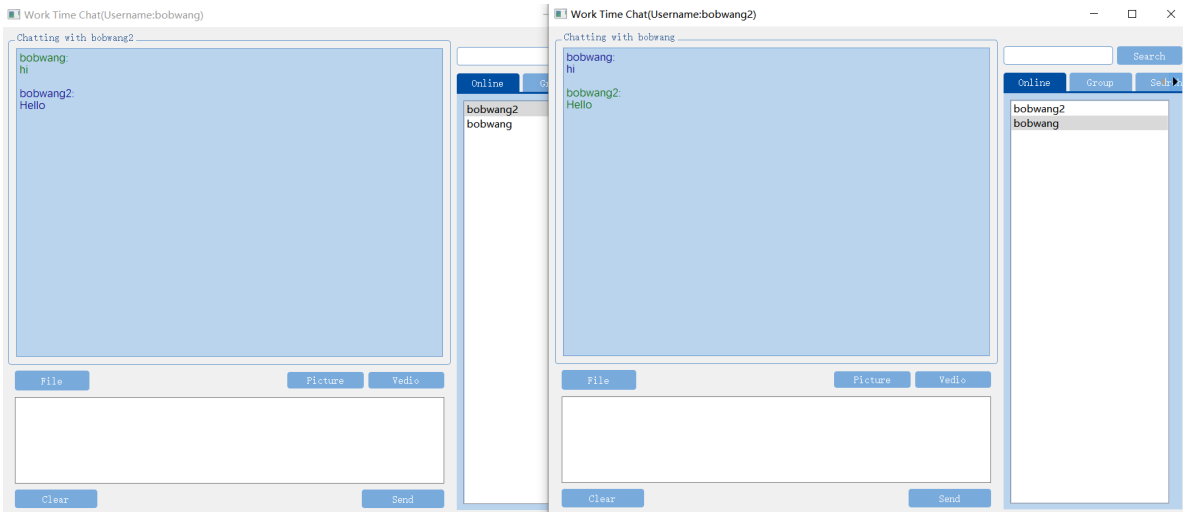


Figure 32: Chatting window

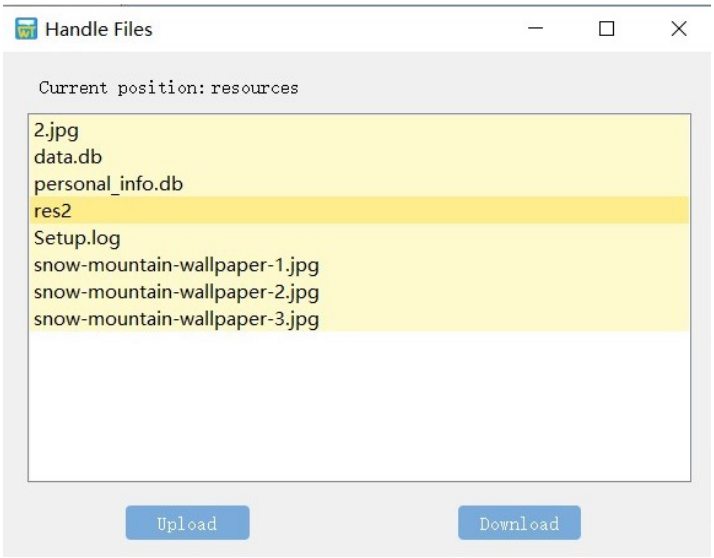


Figure 33: Handle Files

3.6.1.4 Retrospective

We need more stand meetings to keep communicating and perfect our database.

Stop editing UI of login window and main window. The development of UI will start in sprint3.

Then, starting to prepare the tasks of the next sprint.

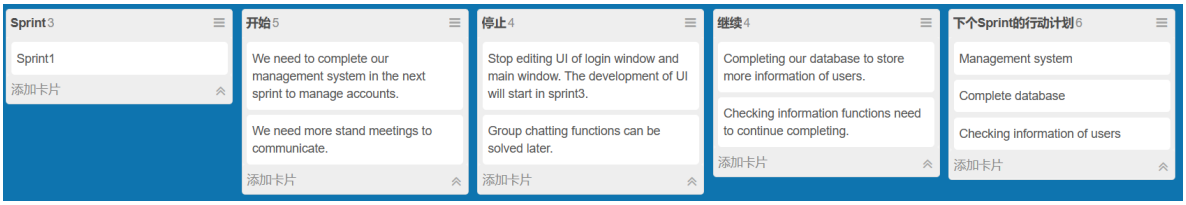


Figure 34: Retrospective of Sprint 1

3.6.2 Sprint 2 - Manage database

Duration: 11.1 – 11.14

3.6.2.1 Sprint Planning

In this sprint, we have 5 user stories and 5 tasks. We will perfect our database and complete the management system. Users can edit and check information and administrator can manage all accounts of users by management system.

Tasks:

- Modify Information:
  - Task 1: Realize the password modify function, which should ask the user to input the password again before modifying the password.
  - Task 2: Realize the information modify function which including phone number, gender, address, e-mail and so on.

- Task 3: Realize the function that administrators can edit database.

- Check Information:

- Realize the function that a user can see information about his or her colleagues.

- Search:

- Realize the function that a user can search for information about a colleague according to the phone number, address, e-mail and so on.

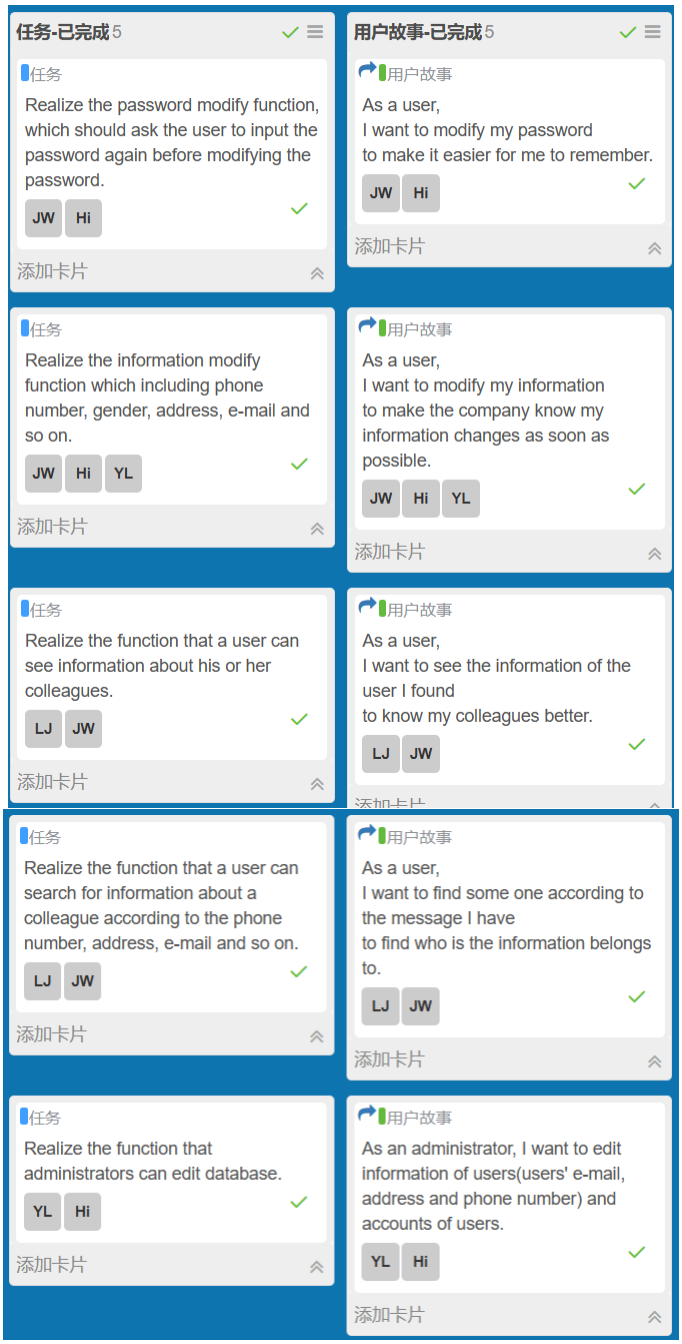


Figure 35: leangoo of Sprint 2

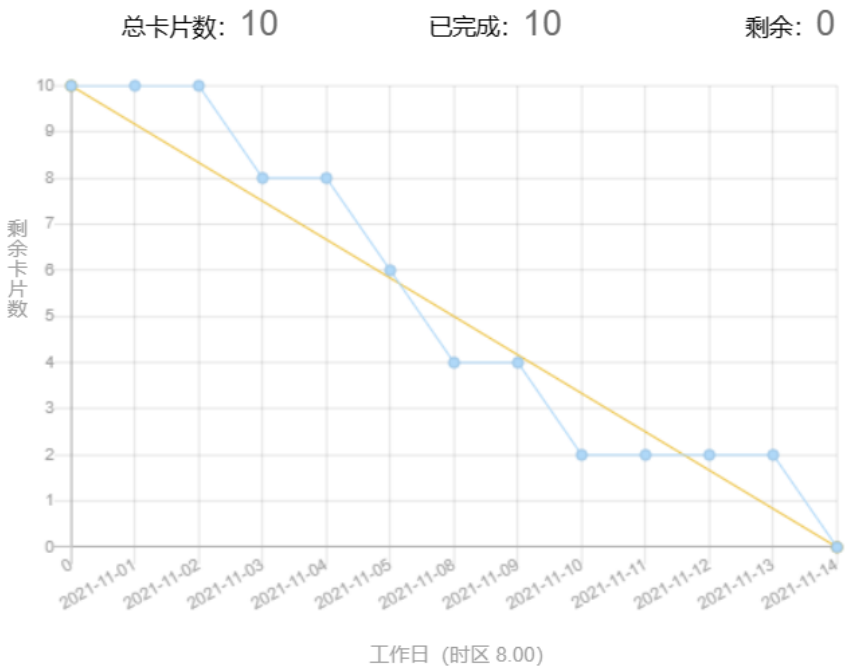


Figure 36: Burn-Down Chart of Sprint 2

3.6.2.2 Daily Stand-up Meetings

We discussed how many information should be stored in the database and how to transfer the data of database between server and client. Then, we need design a UI of management system.

1. Our database should include staff ID, position, head sculpture, phone number and e-mail. (11.1 – 11.7)
2. We should design protocols to send information to the client or receive new data from the client. (11.8 – 11.9)
3. The conditions of editing information. (11.10 – 11.11)
4. Management system UI (11.12 – 11.14)

3.6.2.3 Demo

We tested the demo of sprint2 (management system and user information). We successfully completed all requirements of user stories and improve protocols of transport.

The demo of sprint2 can edit user information and check information of other users. Administrator can use management system to manage accounts of users.

The management system can create, edit and delete accounts.

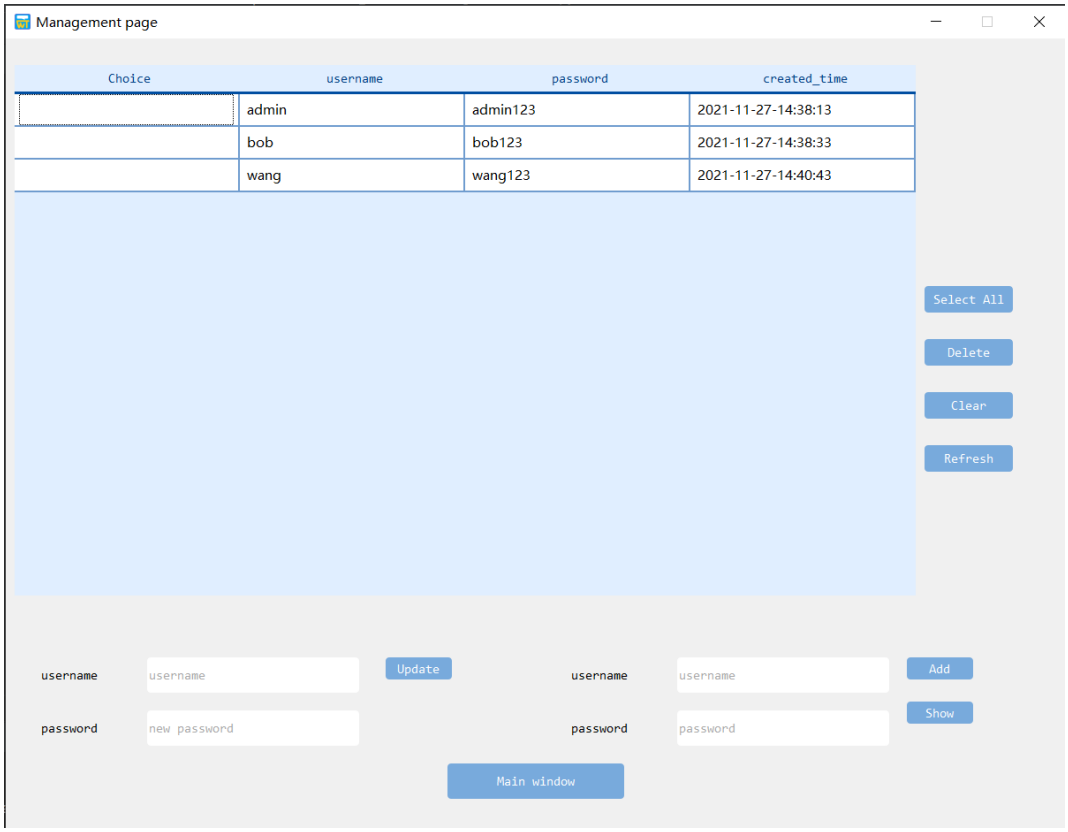


Figure 37: Management Window

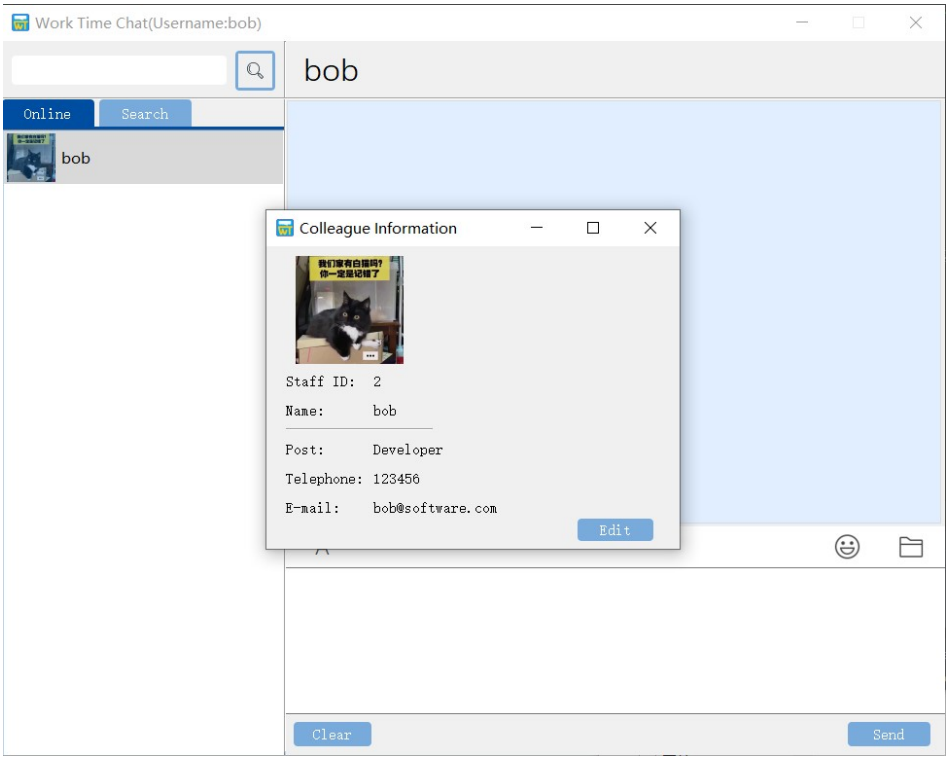


Figure 38: Check Information

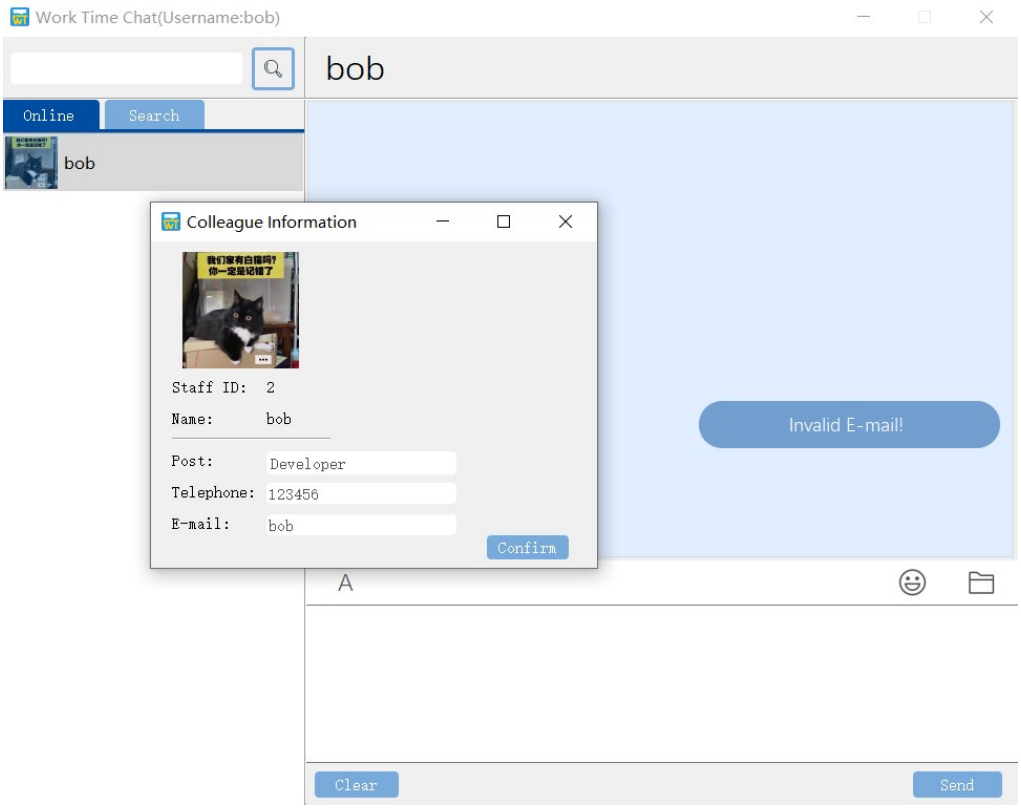


Figure 39: Edit Information

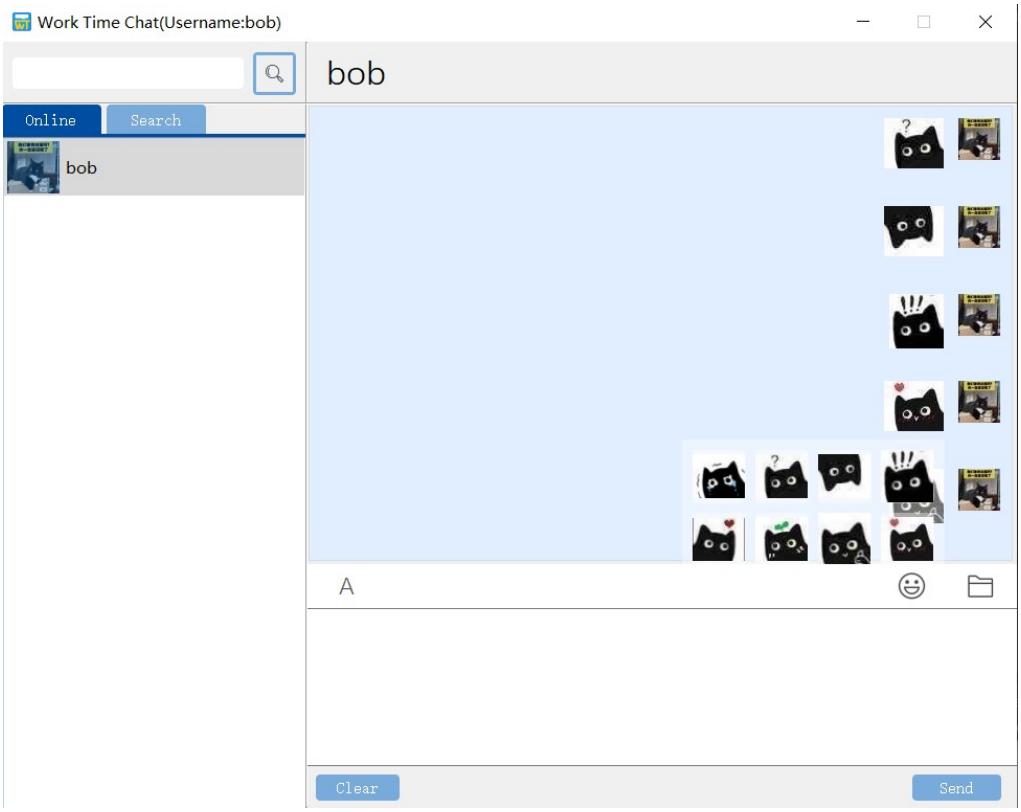


Figure 40: Emoji

3.6.2.4 Retrospective

We need a better UI and a more reasonable plan in the next sprint.

Database is completed. Stop editing our database.

Keep communicating and meetings.

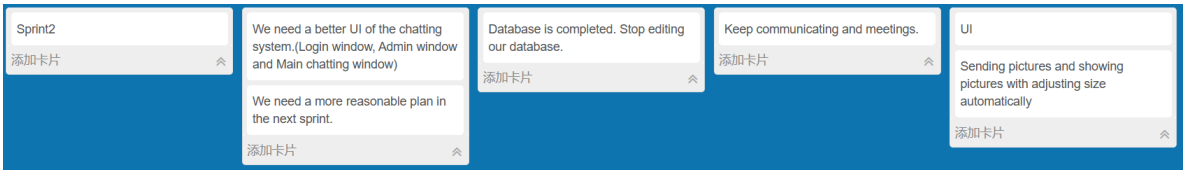


Figure 41: Retrospective of Sprint 2

3.6.3 Sprint 3 - Advanced Functions

Duration: 11.15 – 12.5

3.6.3.1 Sprint Planning

In this sprint, we have 4 user stories and 4 tasks. We will design a better UI for our project, improve details of the system and realize files and pictures transport from user to user.

Tasks:

- Transport:
  - Task 1: Realizing the function that users can send file to one target user.
  - Task 2: Users can send pictures to one target user and show pictures on the chatting window.
- UI:
  - The better UI which is simple and nice-looking to improve user experience. (Login window, Admin window, Chatting window)
- Detail:
  - We should provide some profile pictures and show them on the chatting page.

3.6.3.2 Daily Stand-up Meetings

We discussed the appearance of UI and the transport functions.

1. If users can edit custom UI or form of fonts? They can only edit fonts. (11.15 – 11.18)
2. Transport functions can support dragging upload. (11.19 – 11.20)
3. Users can upload multiple files at the same time. (11.21 – 11.24)
4. Discuss and try to add new chat functions. (11.25 – 11.30)
5. Analyze bugs (12.1 – 12.5)

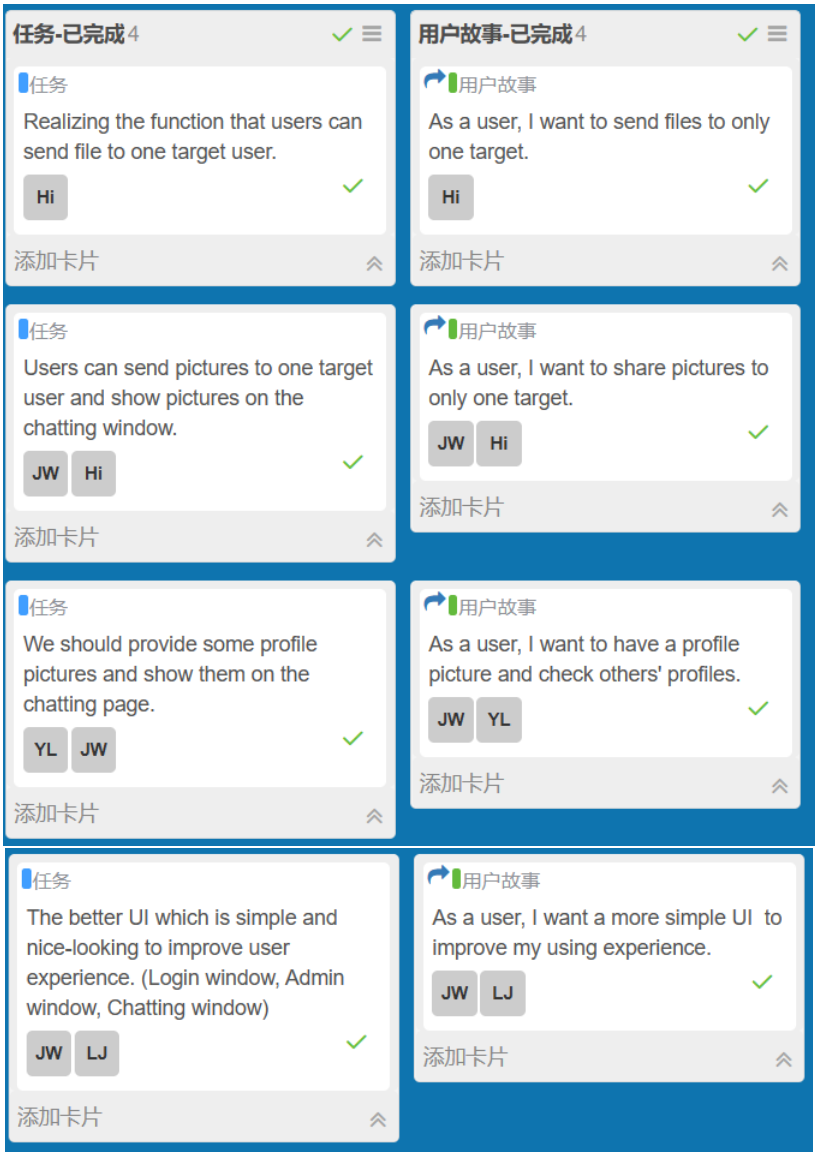


Figure 42: leangoo of Sprint 3

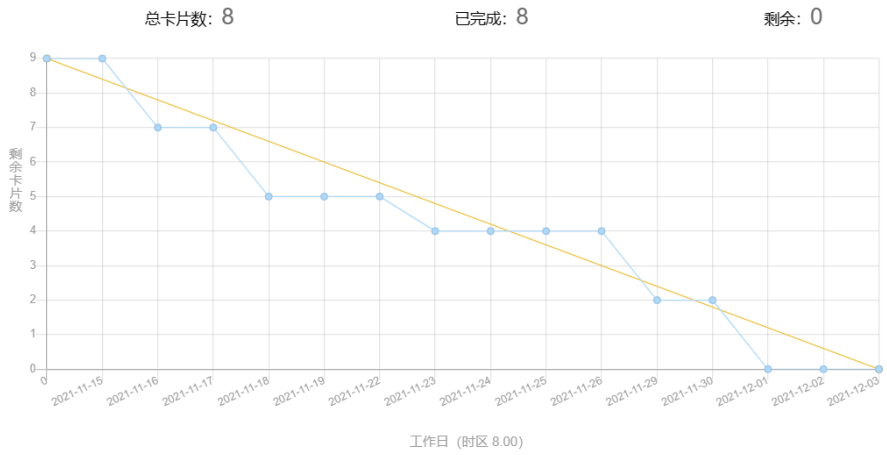


Figure 43: Burn-Down Chart of Sprint 3

3.6.3.3 Demo

We tested the demo of sprint3 (new UI and transport functions). We successfully completed all requirements of user stories and designed a better UI and a powerful transport function.

Our new UI will show the head sculptures of users and support custom fonts.

This transport function can support dragging upload and multiple files upload. If users send pictures to others, the chat window will show the picture in both clients. When users click the picture, they can check the larger image.

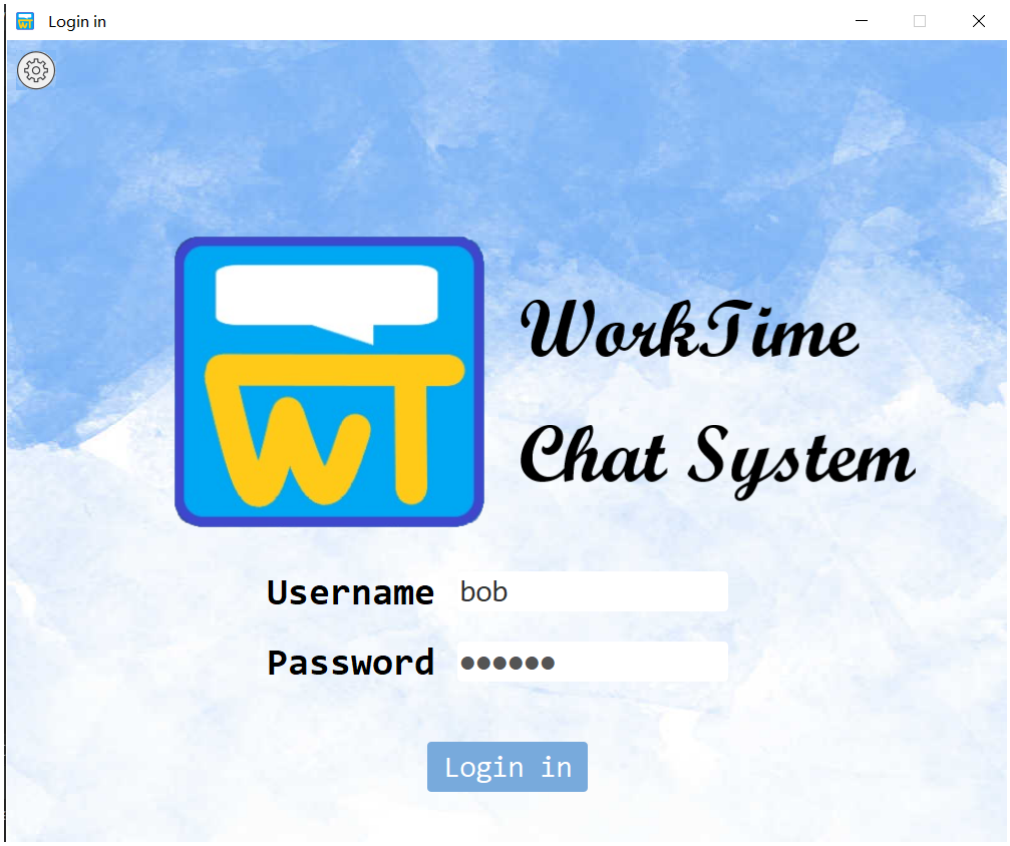


Figure 44: New Login Window

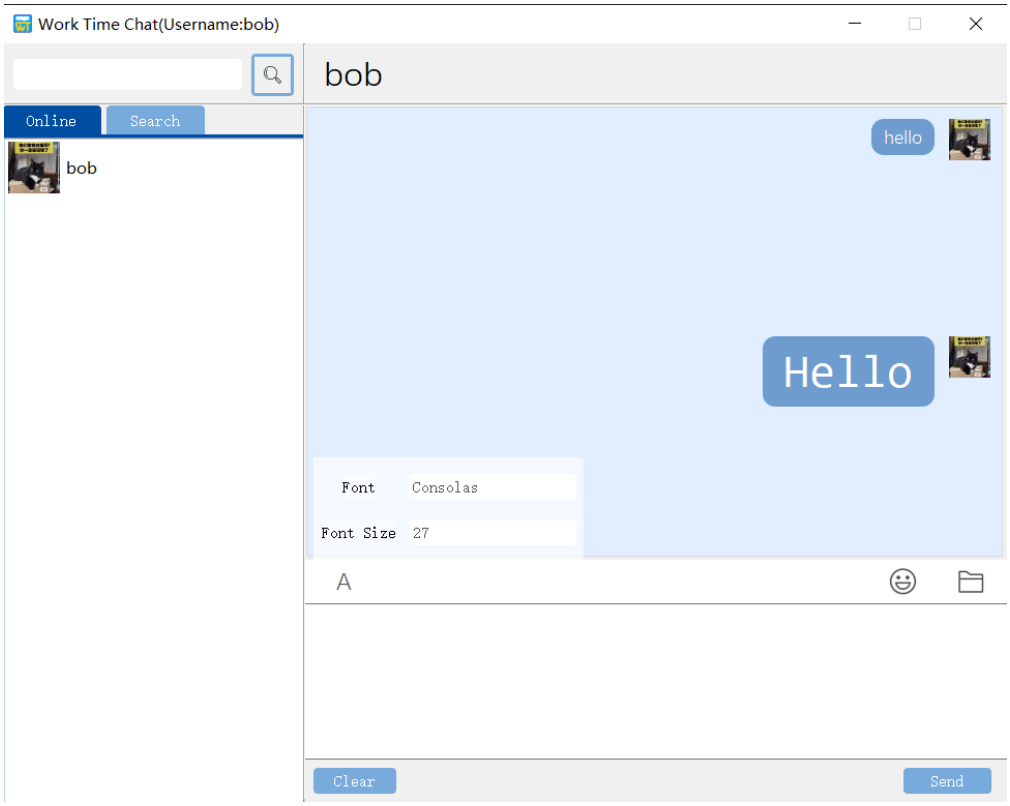


Figure 45: New Chat Window

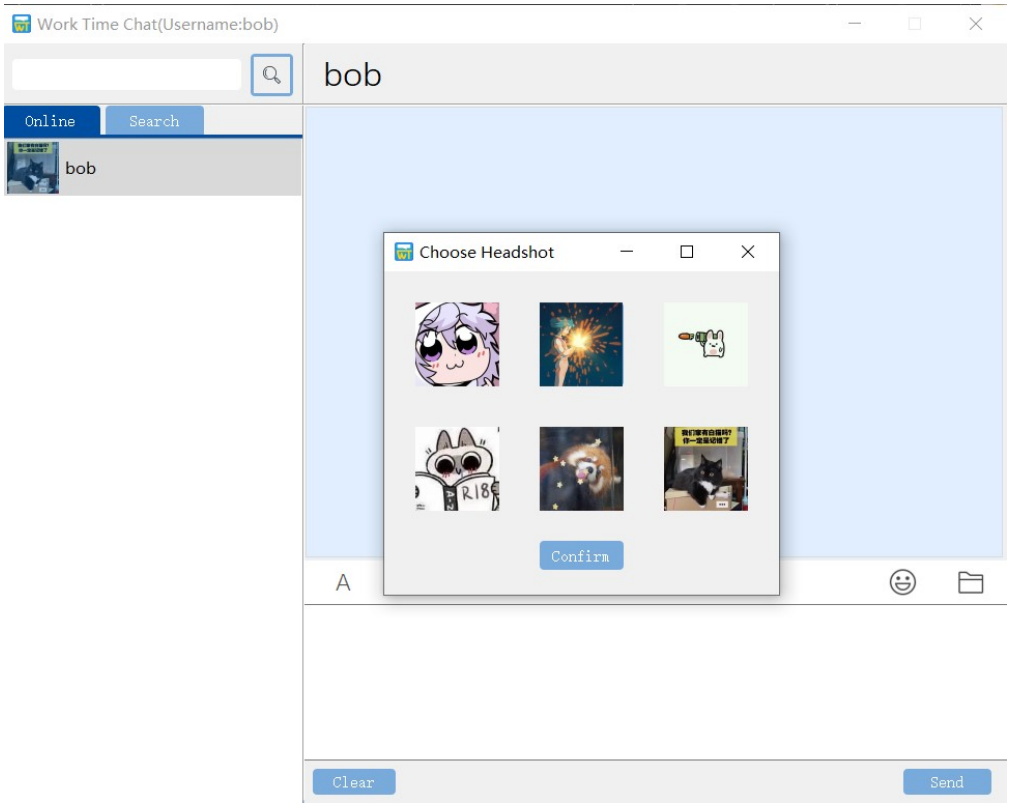


Figure 46: Headshot

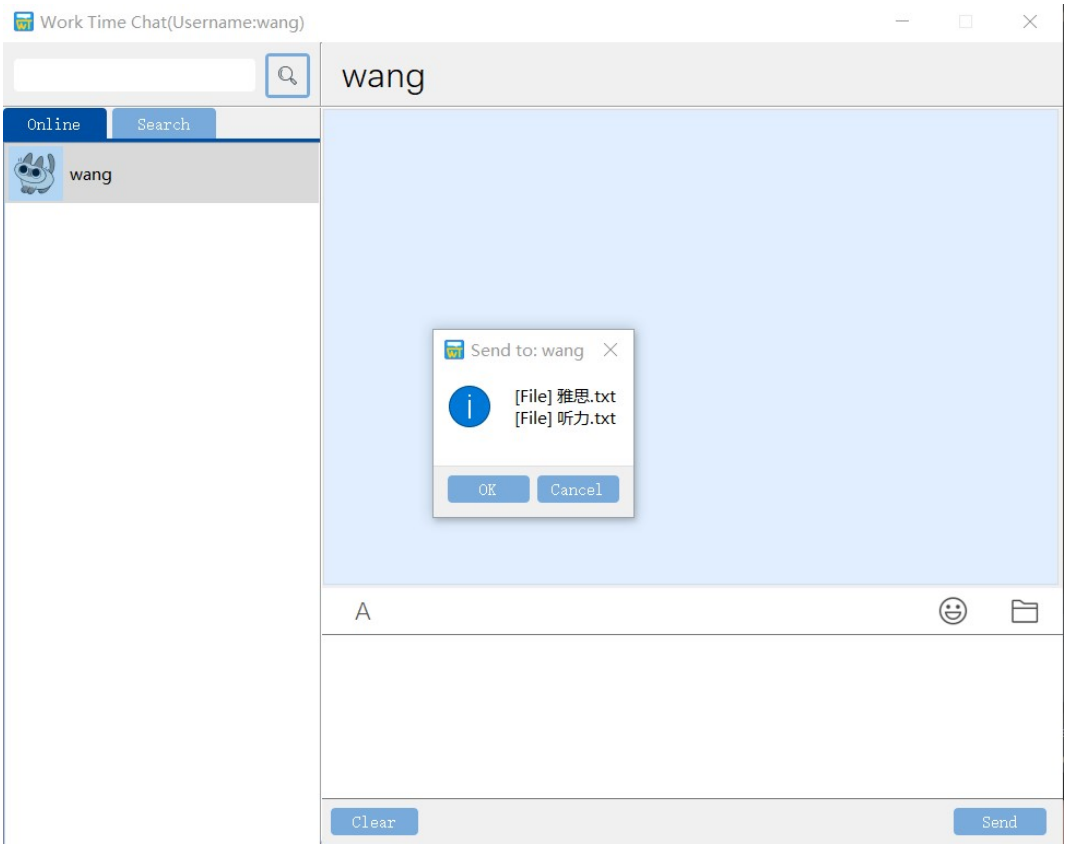


Figure 47: Send Files to Others

3.6.3.4 Retrospective

We totally completed our project and fix many bugs. We start to test the whole system and write the document.

We will begin black-box testing and white-box testing in the remaining time.

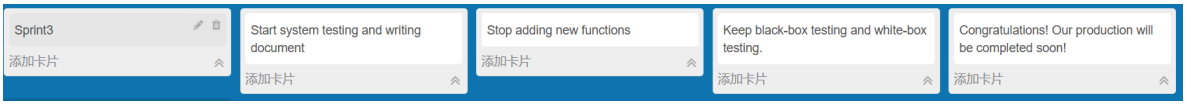


Figure 48: Retrospective of Sprint 3

4 Testing

Software testing is the act of examining the artifacts and the behavior of the software under test by validation and verification. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Testing is important because:

- Software testing is really required to point out the defects and errors that were made during the development phases.
- Ensuring that customers consider our group to be reliable and maintaining their satisfaction with the application is critical.
- Ensuring the quality of the product is very important. High-quality products delivered to customers help gain their confidence.
- Testing is necessary in order to provide customers with facilities, such as delivering high-quality products or software applications that require lower maintenance costs, resulting in more accurate, consistent and reliable results.
- The effective performance of a software application or product needs to be tested.
- It is important to ensure that the application does not cause any failures, as it can be very expensive in the future or later stages of development.

We divided the testing section into three sub-headings, they are Black-box Testing, White-box Testing, Unit Testing and Automated Testing. In Black-box Testing, we use five methods to perform the Black-box Testing: the equivalence partition method, boundary value analysis method, decision table method, causality diagram method and fault speculate method. There are static testing and dynamic testing in White-box Testing. In static White-box Testing, we introduce our desktop checking method and tools, and our code review and walkthrough log. In dynamic White-box Testing, we did all the logical coverage, including statement coverage, decision coverage, condition coverage, decision / condition coverage, combination coverage and path coverage. In the path coverage, we use the control flow graph (CFG) to calculate the cyclomatic complexity and the independent path. For the Unit Testing and the Automated Testing, we use the unittest and airtest software to perform, respectively.

The figure below(4.1.1) can understand our testing overall steps more clearly.

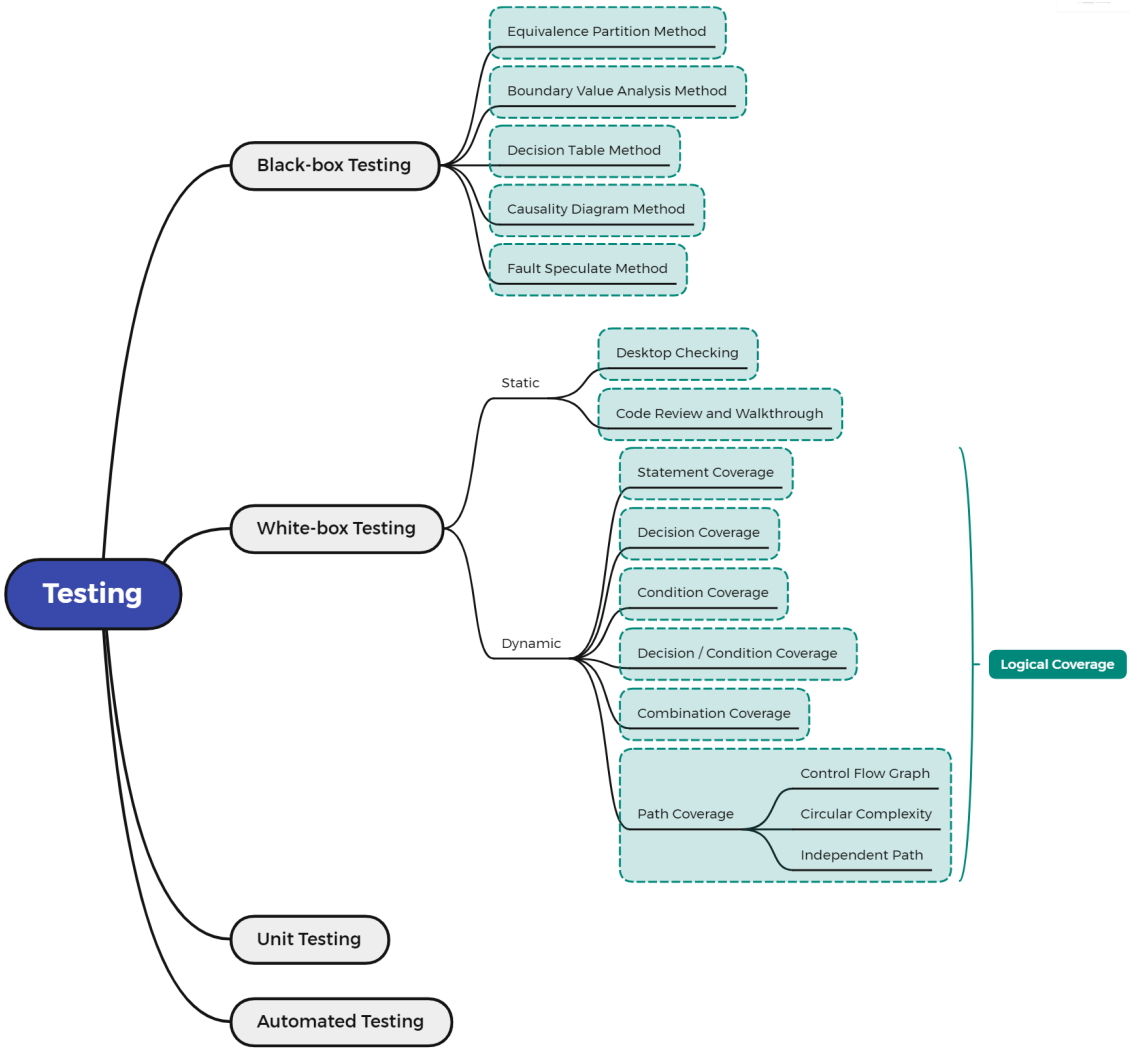


Figure 49: Our testing over all steps

4.1 Black-box Testing

In this testing, we mainly use the login system as an example to conduct our Black-box Testing. Total five methods are used in this part: the equivalence partition method, boundary value analysis method, decision table method, cause and effect diagram method and fault speculate method. We use the login function(4.1) as our testing example of most part of black-box testing.

```python
def login(self):

    """implement the login function"""
    print("Login: ", IP, PORT)
    username = self.username_edit.text()
    password = self.password_edit.text()
    s1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    result = re.compile('^[a-zA-Z0-9_]+$')
    password_length = re.compile('^[a-zA-Z0-9_]{6,18}$')
    success = 1
    try:
        s1.connect((IP, PORT + 3))
    except socket.error:
        TipUi.show_tip('Socket error\nFail to conncet!',)
        success = -1

    if success == 1:
        if not username:
            TipUi.show_tip('Name type error\nUsername Empty!')
            s1.close()
            success = -1
        elif not password:
            TipUi.show_tip('Password type error\nPassword Empty!')
            s1.close()
            success = -1
        elif not result.match(username):
            TipUi.show_tip('Username: illegal characters\nAccept English Letters, Numbers and Underline')
            s1.close()
            success = -1
        elif not result.match(password):
            TipUi.show_tip('Password: illegal characters\nAccept English Letters, Numbers and Underline')
            s1.close()
            success = -1
        elif not password_length.match(password):
            TipUi.show_tip('Password: illegal length\nAccept Length: 6-18')
            s1.close()
            success = -1

    if success == 1:
        print('username: ' + username)
        print('password :' + password)
        pid = 1  # Protocol 1 is used for send username and password
        head = {
            'protocol': pid,
            'username': username,
            'password': password,
        }
        head_info = json.dumps(head)  # Convert dictionary to string
        head_info_len = struct.pack('i', len(head_info))  # Pack the length of the string
        s1.send(head_info_len)  # send the length of head_info
        s1.send(head_info.encode('utf-8'))  # server: send4

        result = s1.recv(buffSize)
        success = int(result.decode())
        if success == -1:
            TipUi.show_tip('Password and Username is error')

    if success == 1:
        if username == 'admin':  # go into the admin window if the user is admin
            self.admin = AdminWindow(username, IP, PORT)
            self.admin.show()
            self.close()

        else:
            self.stuff = DialogueWindow(username, IP, PORT)
            self.stuff.show()
            self.close()

    s1.close()
```

Figure 50: Login code

4.1.1 Equivalence Partition Method

Equivalence Partitioning Method is also known as Equivalence class partitioning (ECP). It is a software testing technique or black-box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed. In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is checked, then for this input conditions, Equivalence class represents or describes set of valid or invalid states. In the login system, the input conditions include the user name and the password. We set some rules for the user name and password, for example, user name and password can only contain uppercase and lowercase letters, numbers and underline, the length of password is limited to 6-18. According to these rules for these two types of input, we create the table as below:

Input conditions	Valid equivalence classes	Invalid equivalence classes
User name	Equal or larger than one character (1)	Empty (5)
	Contain only uppercase and lowercase letters, numbers and underline (2)	Other special characters except letters, numbers and underline (6)
		Chinese or other foreign characters (7)
Password	Character length: 6 to 18 (3)	Empty (8)
		Character length is not empty but less than 6 (9)
		Character length is larger than 18 (10)
	Contain only uppercase and lowercase letters, numbers and underline (4)	Other special characters except letters, numbers and underline (11)
		Chinese or other foreign characters (12)

Figure 51: Equivalence class table

You can see the valid equivalence classes and the invalid equivalence classes leverage user name and password. According to the table, we design the test cases as below:

Test cases	Input		Covered equivalence classes	Expected output
	User name	Password		
1	Admin_123	Admin123	(1) (2) (3) (4)	Standard output
2	Admin_123	NULL	(1) (2) (8)	Illegal output
3	Admin_123	cat	(1) (2) (4) (9)	Illegal output
4	Admin_123	Who_cares_about_your_password	(1) (2) (10)	Illegal output
5	Admin_123	P@ssw0rd	(1) (2) (3) (11)	Illegal output
6	Admin_123	管理员password	(1) (2) (3) (12)	Illegal output
7	NULL	Admin123	(3) (4) (5)	Illegal output
8	NULL	NULL	(5) (8)	Illegal output
9	NULL	cat	(5) (9)	Illegal output
10	NULL	Who_cares_about_your_password	(5) (10)	Illegal output
11	NULL	P@ssw0rd	(3) (5) (11)	Illegal output
12	NULL	管理员password	(3) (5) (12)	Illegal output
13	@dmin123	Admin123	(3) (4) (6)	Illegal output
14	@dmin123	NULL	(6) (8)	Illegal output
15	@dmin123	cat	(4) (6) (9)	Illegal output
16	@dmin123	Who_cares_about_your_password	(4) (6) (10)	Illegal output
17	@admin123	P@ssw0rd	(3) (6) (11)	Illegal output
18	@admin123	管理员password	(3) (6) (12)	Illegal output
19	管理员id	Admin123	(3) (4) (7)	Illegal output
20	管理员id	NULL	(7) (8)	Illegal output
21	管理员id	cat	(4) (7) (9)	Illegal output
22	管理员id	Who_cares_about_your_password	(4) (7) (10)	Illegal output
23	管理员id	P@ssw0rd	(3) (7) (11)	Illegal output
24	管理员id	管理员password	(3) (7) (12)	Illegal output

Figure 52: Testing Cases

These test cases are based on the equivalence class table that we have already designed. There are 24 test cases that cover all the equivalence class, includes all the invalid class. As a result, our test cases are both strong and robust.

4.1.2 Boundary Value Analysis Method

Boundary value analysis is one of the widely used case design technique for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.

Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.

Boundary values are those that contain the upper and lower limit of a variable. In the login system, the minimum length for user name is 1, as a result, we cannot input empty as the name. The length for password is limited to 6-18 by our software. Thus, the length for password can be 5, 6, 17, 18, 19, as the boundary values. Also, the character can be used as boundary value. For example, 'a' is the smallest character in our login system and 'z' is the largest. By using these approaches, we design our testing cases as below:

Test cases	Input		Expected output
	User name	Password	
1	NULL	Admin123	Illegal output
2	a	Admin123	Standard output
3	A	Admin123	Standard output
4	z	Admin123	Standard output
5	admin	admin	Illegal output
6	admin	Admin1	Standard output
7	admin	AttentionIsYouNeed	Standard output
8	admin	AttentionIsYouNeed1	Illegal output

Figure 53: Boundary value analysis of login interface

4.1.3 Decision Table Method

Decision Table Testing is a Black Box test design technique (behavioral or behavior-based technique), used where different combinations of test input conditions result in different outcomes. When a system has complex business rules, then the decision table testing technique helps in identifying the correct test cases. Decision tables are a concise visual representation for specifying which actions to perform depending on given conditions. They are algorithms whose output is a set of actions. The information expressed in decision tables could also be represented as decision trees or in a programming language as a series of if-then-else and switch-case statements.

We use login system as an example, in this system, we use 4 conditions and 2 outcomes; there are 16 condition stubs in total. Before simplified, the table is shown as below:

Rule / Condition stub	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C1: Username is NULL?	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
C2: Username only contain uppercase and lowercase letters, numbers and underline?	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
C3: Password character length: 6 to 18?	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T
C4: Password only contain uppercase and lowercase letters, numbers and underline?	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T
A1: Standard output								√								
A2: Illegal output	√	√	√	√	√	√	√		√	√	√	√	√	√	√	√

Figure 54: Decision table - Login(before simplification)

After we tried to simplify it, the updated table is shown as below:

Condition \ Rule stud	1	2	3	4	5
C1: Username is NULL?	F	F	F	F	T
C2: Username only contain uppercase and lowercase letters, numbers and underline?	F	T	-	-	-
C3: Password character length: 6 to 18?	-	T	F	-	-
C4: Password only contain uppercase and lowercase letters, numbers and underline?	-	T	-	F	-
A1: Standard output		√			
A2: Illegal output	√		√	√	√

Figure 55: Decision table - Login(after simplification)

```
def confirm(self):
    name = self.me
    headshot = self.showClgInfo.data[5]
    post = self.showClgInfo.post_edit.currentText()  # 新增
    telephone = self.showClgInfo.tel_edit.text()
    email = self.showClgInfo.email_edit.text()
    if not telephone.isdigit():
        return

    if not re.match(r"(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)", email, 0):
        return

    updataInfoSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    updataInfoSock.connect((self.IP, self.PORT + 3))  # PORT = 50007
    pid = 6  # Protocol 6 is used for
    head = {
        'protocol': pid,
        'name': name,
        'occupation': post,
        'headshot': headshot,
        'telephone': telephone,
        'email': email,
    }
    head_info = json.dumps(head)  # 将字典转换成字符串
    head_info_len = struct.pack('i', len(head_info))  # 将字符串的长度打包
    updataInfoSock.send(head_info_len)  # 发送head_info的长度  # server: send3
    updataInfoSock.send(head_info.encode('utf-8'))  # server: send4

    updataInfoSock.close()
```

Figure 56: Code - Information modify

Rule / Condition Stub	1	2	3	4
C1: Is this the information page of yourself?	F	T	T	T
C2: The phone number is consisted of numbers?	-	F	-	T
C3: The email has "@" and "." ?	-	-	F	T
A1: You can't edit the information	✓			
A2: Invalid information		✓	✓	
A3: Successful edit				✓

Figure 57: Decision table - Information modify

4.1.4 Cause and Effect Diagram Method

Cause-effect graph comes under the black box testing technique which underlines the relationship between a given result and all the factors affecting the result. It is used to write dynamic test cases. The dynamic test cases are used when code works dynamically based on user input.

In our login system, we need to analysis at first. We should analysis the reason, the intermediate state and the result of our login system. Then, the table is designed as below:

Reason	c1: The username entered complies with the rules
	c2: The username is NULL or contains illegal character
	c3: The password entered complies with the rules
	c4: The password entered does not meet the rules
Intermediate state	11: Enter the standard username and password
	12: Enter an illegal username or password
Result	e1: The input is legal and the pair of name and password can send to database for later verification
	e2: A pop-up window prompts that the input is illegal

Figure 58: Analyze cause, intermediate state and result

And then, according to the analyzing, we can create the cause and effect diagram as below:

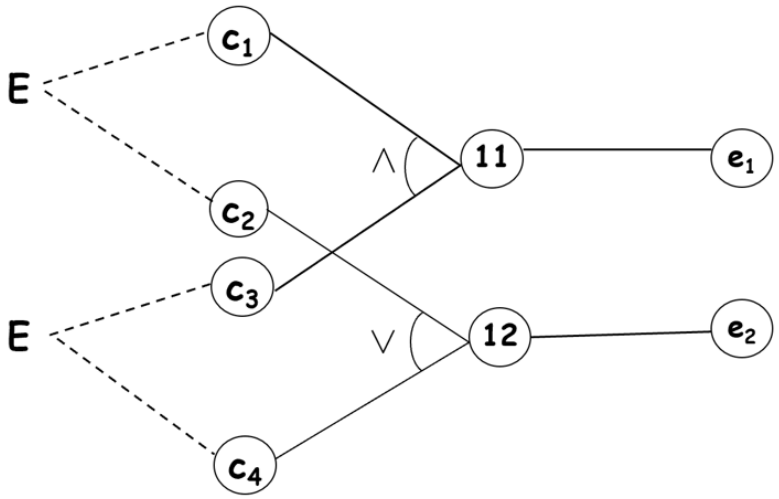


Figure 59: Cause and effect diagram

And then, we can create the decision table as below:

No.		1	2	3	4
Reason	c1	1	1	0	0
	c2	0	0	1	1
	c3	1	0	1	0
	c4	0	1	1	1
Intermediate State	11	1	0	0	0
	12	0	1	1	1
Result	e1	1	0	0	0
	e2	0	1	1	1

Figure 60: Decision table

According to the table before, we can design the testing cases:

Test cases	Input		Expected output
	User name	Password	
1	Admin123	Admin123	e1: legal
2	a	Admin123	e2: illegal
3	Admin123	p	e2: illegal
4	a	p	e2: illegal

Figure 61: Testing Cases

4.1.5 Fault Speculate Method

By using our empirical speculation, we can predict some possible error. Then we designed some testing cases for them.

Login System

Test cases	Input		Expected output
	User name	Password	
1	NULL	NULL	Illegal output
2	a	a	Illegal output
3	AttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeed	Admin123	Standard output
4	_	Admin123	Standard output

Figure 62: Fault speculate- Login

Personal Information Modification System

Test cases	Input	Expected output
	e-mail	
1	测试@test.com	Warning Message "Invalid email"
2	NULL	Warning Message "Invalid phone number"
3	AttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeedAttentionIsAllYouNeed@gmail.com	Successfully edit

Figure 63: Fault speculate- Information modify

4.2 White-box Testing

White box testing is an approach that allows testers to inspect and verify the inner workings of a software system—its code, infrastructure, and integrations with external systems. White box testing is an essential part of automated build processes in a modern Continuous Integration/Continuous Delivery (CI/CD) development pipeline.

There are static testing and dynamic testing in White-box Testing. In static White-box Testing, we have desktop checking, code review and walkthrough. In dynamic White-box Testing, we did all the logical coverage, including statement coverage, decision coverage, condition coverage, decision / condition coverage, combination coverage and path coverage.

4.2.1 Desktop Checking

After we finished a module of the project, we will use code comments to improve the readability of our project. Since our project is wrote by Python, there are several rules for our group: comments start with "#" is for something difficult in one or several lines of code. Each function and module should have its summarization by using " "' summarization is in there "' ". In our project, code comments have about 30% of our overall code. It is for later develop and maintain.

Before we submit our code to the project, we need to check the code style, structure, error, warnings in the code. For better finishing our project, our group members use an automated tools for the process of desktop checking, Pylint. Pylint is a source-code, bug and quality checker for the Python programming language. It is named following a common convention in Python of a "py" prefix, and a nod to the C programming lint program. It follows the style recommended by PEP 8, the Python style guide. As a result, we can check whether our code follow the correct PEP8 style.

There is an example of our usage of Pylint. We want to check our login system that stored in our "lonin.py" file. We use the Pylint tool to check before any edition, the result looks as below:

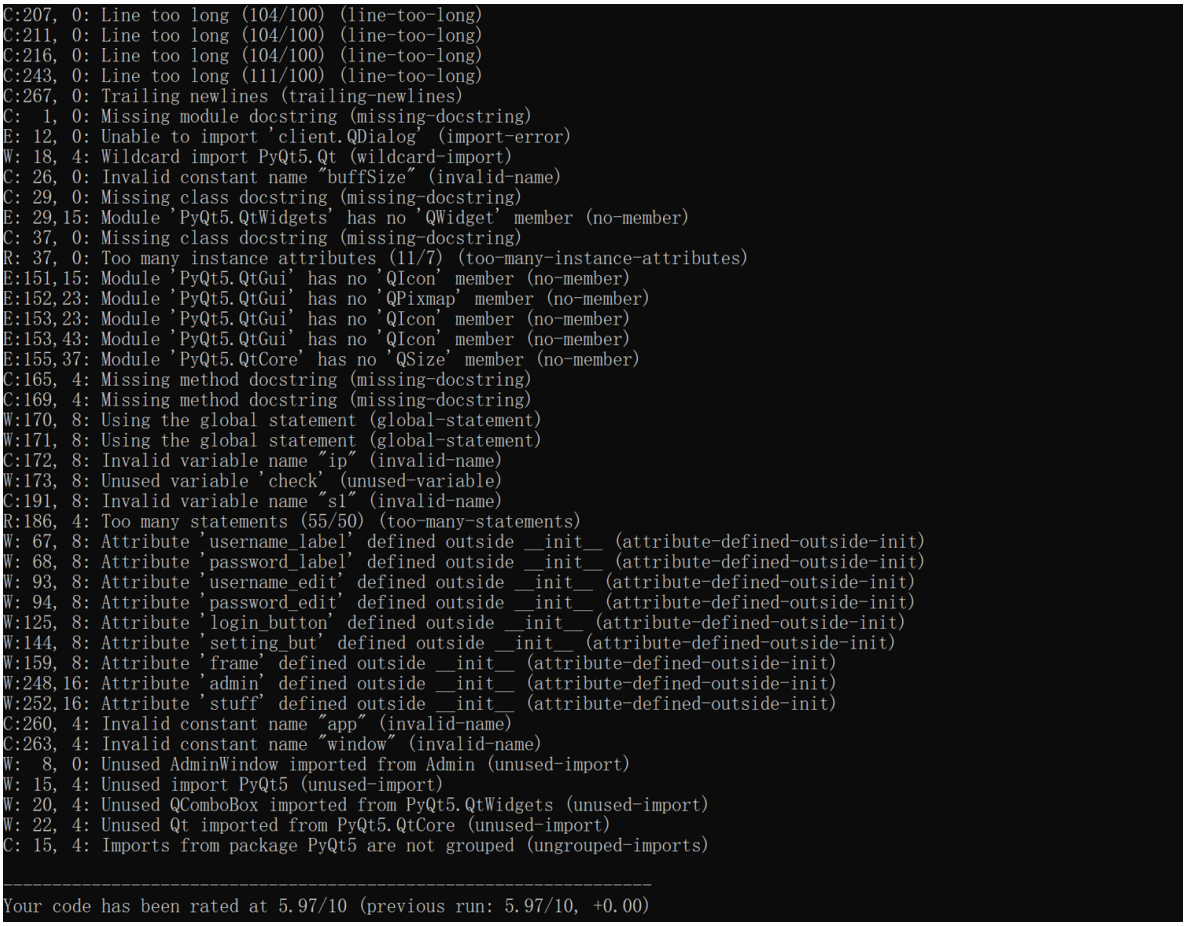


Figure 64: pylint - before modifying

In this figure, we can find that there are a plenty of warnings and style errors in this module. The tool rate our code as 5.97/10.

After modification, we can eliminate most of errors and warnings in this code, the later result is as below:

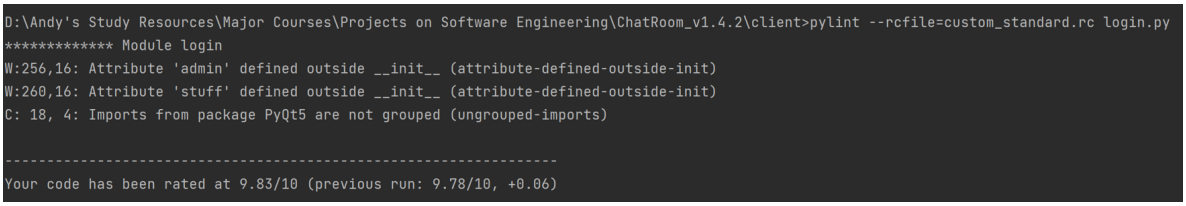


Figure 65: pylint - after modifying

Now, most of errors and warnings in this code are gone. The tool rate our code as 9.78/10. Thus we can pass this module because this code largely follow the PEP8 style.

4.2.2 Code Review and Walkthrough

We often discuss our code and record log on our daily stand-up meeting. We are responsible for checking other member's code and give some suggestions if necessary. We also record our code problems and discuss them twice a week during our practice class. We also assign the problems to one or several members to fix them. All of these should be recorded on our log. There are several code problems examples on our log as below:

The login window does not close after user login

```
Found date: 2021/10/07

Discovered member(s): Yichu Li, Hongbo Wang

status: already fixed

Fixed date: 2021/10/17

Fixed member(s): Yichu Li

Comment: This window will not close automatically, even the window create a new
window. So we should use close() to close it after login window create the other
window.
```

Figure 66: The login window doesn't close after user login

SQL injection attack

```
Found date: 2021/10/22

Discovered member(s): Hongbo Wang

status: already fixed

Fixed date: 2021/10/26

Fixed member(s): Yichu Li

Comment: Through the character input control of the input interface, the simple
database threat problem is solved
```

Figure 67: SQL injection attack

Transmit file name wrongly. When client upload a file which name include space, the server will only get the name before space.

```
Found date: 2021/10/15
Discovered member(s): Jingwen Liu
status: already fixed
Fixed date: 2021/10/16
Fixed member(s): Jingwen Liu
Reason:  name = message.split()[1]
Solution:  name = message.split()[1]
    should be changed into
    name = message.split(" ", 1)[1]
```

Figure 68: Transmit file name wrongly

space and enter can be sent

```
Found date: 2021/10/27
Discovered member(s): Jingwen Liu
status: already fixed
Fixed date: 2021/10/29
Fixed member(s): Jingwen Liu
Reason: space and enter are not detected to be invaild characters
Solution: .rstrip() userd after getting text in inputText
```

Figure 69: illegal string can be sent

4.2.3 Statement Coverage

Statement coverage is the weakest logic coverage. It only requires the design of enough test cases so that every statement in the program is executed once. For the code of the confirm IP address in the login system is as below:

```python
def confirm(self):
    global IP
    global PORT
    ip = self.settings_win.ip.text()
    check = ip.split(".")
    if not re.match(r"^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}"
                    r"(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$", ip, 0):
        TipUi.show_tip('Wrong format for ip address! ')
        return
    port = self.settings_win.port.text()
    if int(port) < 1024 or int(port) > 65535:
        TipUi.show_tip('Wrong format for port! ')
        return
    IP = ip
    PORT = int(port)
    self.settings_win.close()
```

Figure 70: Code - IP setting function

We can use this function as an example. The flow chart is as below:

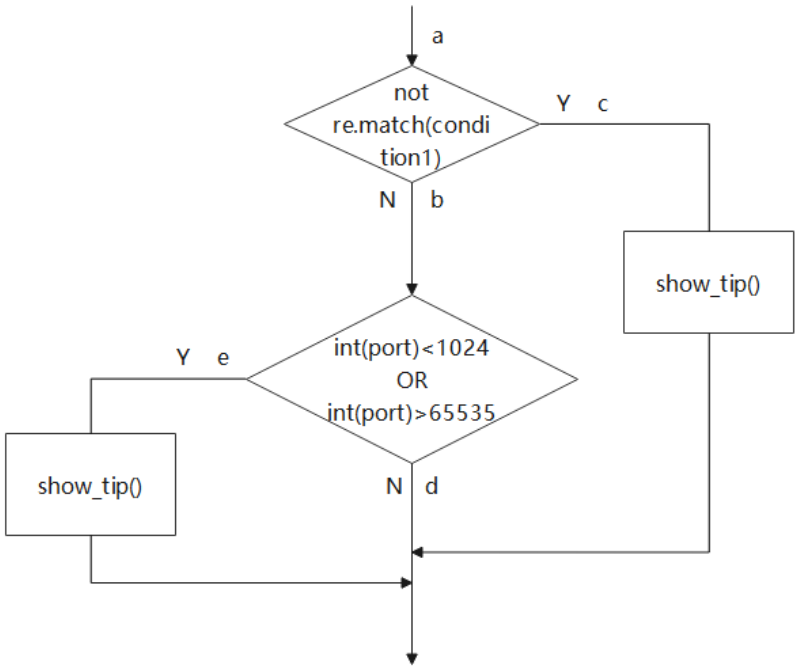


Figure 71: IP setting

Since the statement coverage is the weakest logic coverage, we can use only one testing case to cover this coverage as below:

Test case	Input		Path
	IP address	Port	
1	127.0.0.1	10	abe

Figure 72: Statement coverage

Statement coverage can ensure that every statement in the program is executed, but it cannot find errors in logical operations in the judgment, that is, it is not a sufficient test method.

4.2.4 Decision Coverage

Decision coverage requires the design of enough test cases so that each decision expression in the program under test is executed once for "true" and once for "false". So that every branch of the program is passed at least once. For the same module as below, we can use 3 testing cases to cover this coverage as below:

Test case	Input		Path
	IP address	Port	
1	127.0.0.1	10	abe
2	11a	8000	ac
3	127.0.0.1	8000	abd

Figure 73: Decision coverage

"Decision coverage" is stricter than "statement coverage" because if every branch has been executed, then every statement has also been executed. However, "decision coverage" is still not enough. The function of decision coverage is that it makes each decision obtain every possible result at least once.

4.2.5 Condition Coverage

In the actual program code, a decision usually contains several conditions. The purpose of condition coverage is to design several test cases. After executing the program under test, the possible values of each condition in each judgment must be satisfied at least once. For the confirm IP address' format module, we can use 3 testing cases to cover this coverage as below:

Test case	Input		Path	Covered condition
	IP address	Port		
1	127.0.0.1	10	abe	T1, T2, -T3
2	11a	8000	ac	-T1
3	127.0.0.1	8000	abd	T1, T2, -T2, T3

Figure 74: Conditional coverage

"Condition coverage" is usually stronger than "decision coverage" because it makes two different results for each condition in a decision, and decision coverage does not guarantee this.

4.2.6 Decision / Condition Coverage

Decision / condition coverage is actually a method of combining decision coverage and condition coverage, that is: design enough test cases so that all possible values of each condition in the judgment are satisfied at least once, and the possible results of each judgment also appeared at least once. In the same IP address confirm format module, we can also use 3 testing cases to confirm this coverage:

Test case	Input		Path	Covered condition
	IP address	Port		
1	127.0.0.1	10	abe	T1, T2, -T3
2	11a	8000	ac	-T1
3	127.0.0.1	8000	abd	T1, T2, -T2, T3

Figure 75: Conditional coverage

On the surface, decision / condition coverage tests the values of all conditions in each judgment, but in fact, when the compiler checks a logical expression containing multiple conditions, certain conditions in some cases will be Covered by other conditions. Therefore, decision / condition coverage may not be able to completely detect errors in logical expressions.

4.2.7 Combination coverage

The purpose of combination coverage is to enable the designed test case to cover all possible combinations of condition values for each judgment. At this time, we use another module, file download module as an example. At first, we need to mark the condition value combination in the downloadFile() function:

```
def downloadFile(self):
    item = self.filesList.currentItem()
    if item is not None:
        defaultName = item.text()
    else:
        return
    if '.' not in defaultName:
        QMessageBox.information(self, 'Warning', 'Download folder is not permitted! You can download fileself.s. ', QMessageBox.Ok)
        return
    fileDir, filetype = QFileDialog.getSaveFileName(self, "Save the file", defaultName,
                                                    "All Files (*);;Text Files (*.txt);;Microsoft Edge PDF Document (*.pdf);;"
                                                    "Microsoft Word Document (*.docx);;Microsoft PowerPoint Document (*.ppt);;"
                                                    "Microsoft Excel Worksheet (*.xlsx)")
    fileName = fileDir.split('/')[-1]
    message = 'get ' + defaultName
    if fileName:
        self.s.send(message.encode())
        head_struct=self.s.recv(4)
        head_len = struct.unpack('i', head_struct)[0]
        headdata = self.s.recv(head_len)
        head_dir = json.loads(headdata.decode('utf-8'))
        filesize_b = head_dir['filesize_bytes']
        filename = head_dir['filename']
        print(filesize_b)
        print(filename)
        recv_len = 0
        old = time.time()
        f = open(fileDir, 'wb')
```

Figure 76: Code - file download 1/2

```
        buffsize = 1024
        while recv_len < filesize_b:
            if filesize_b - recv_len > buffsize:
                recv_mesg = self.s.recv(buffsize)
                f.write(recv_mesg)
                recv_len += len(recv_mesg)
            else:
                recv_mesg = self.s.recv(filesize_b - recv_len)
                recv_len += len(recv_mesg)
                f.write(recv_mesg)
        print(recv_len, filesize_b)
        now = time.time()
        stamp = int(now - old)
        print('总共用时%ds' % stamp)
        QMessageBox.information(self, 'Message', 'Download compeleted! ', QMessageBox.Ok)
        f.close()
            ...
    else:
        QMessageBox.information(self, 'Error', 'File name is needed! ', QMessageBox.Ok)
    return
```

Figure 77: Code - file download 2/2

Condition	Value
Don't select files	Mark -T1, the first branch to be judged to be false
Select files	Mark T1, the first branch to be judged to be true
Don't select the download destination folder.	Mark -T2, the second decision to take the false branch
Select the download destination folder	Mark T2, the second branch is judged to be true
Valid file name	Mark T3, the third decision is the true branch
Invalid file name	Mark -T3, the third branch is judged to be false
File acceptance is complete	Mark T4, the fourth branch is judged to be false
The document has not been received	Mark T4, the fourth branch is judged to be true
The file has not been received, and the file size is less than 1024 bytes	Mark T4, -T5, the fifth branch is judged to be false
The file has not been accepted, and the file size is greater than 1024 bytes	Mark T4, T5, and the fifth judgment is the true branch

Figure 78: File download

After that, we can design the testing cases as below:

Case	path	coverage conditions	coverage number
Do not choose any file	ac	-T1	1
Do not choose located folder	abe	T1、T2	2、4
invalid file name	abdg	T1、-T2、-T3	2、3、6
file size is 0	abdfi	T1、-T2、T3、-T4	2、3、5、7
file size<buffer size	abdfhk	T1、-T2、T3、T4、-T5	2、3、5、8、9
file size>buffer size	abdfhj	T1、-T2、T3、T4、T5	2、3、5、8、10

Figure 79: Use cases of file download interface

4.2.8 Path Coverage

Only when every path in the program has been tested can the program be fully tested. The purpose of path coverage is to enable the designed test case to cover all possible paths in the program under test.

4.2.8.1 Login Function Example 1 We use the login function(4.1) as the testing example.

- Control Flow Graph
  - A control flow graph (CFG) is a representation, using graph notation, of all paths that might be traversed through a program during its execution. According to the code, we can draw the

control flow graph.

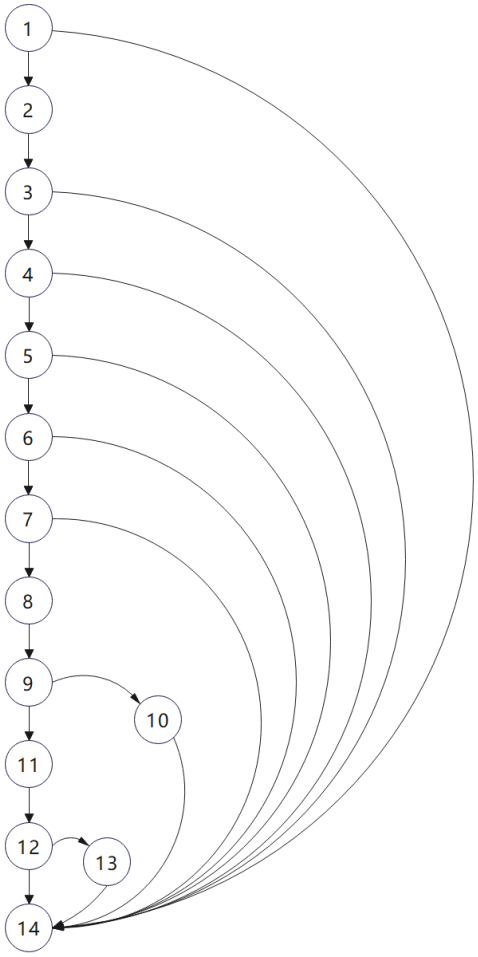


Figure 80: Control flow diagram

- Cyclomatic Complexity
  - By using the control flow graph above, we can calculate the cyclomatic complexity of this control flow graph. The cyclomatic complexity V(G)=9.
- Independent Path
  - The number of independent paths of this function is 9 as the cyclomatic complexity. Now we list all the independent path.

    Path 1: 1→14
    Path 2: 1→2→3→14
    Path 3: 1→2→3→4→14
    Path 4: 1→2→3→4→5→14
    Path 5: 1→2→3→4→5→6→14
    Path 6: 1→2→3→4→5→6→7→14
    Path 7: 1→2→3→4→5→6→7→8→9→10→14
    Path 8: 1→2→3→4→5→6→7→8→9→11→12→14
    Path 9: 1→2→3→4→5→6→7→8→9→11→12→13→14

- Testing Cases
  - According to the 9 independent paths above, we can design the testing cases as below:

Test cases	Input data	Expected output
Path 1	User name: Admin	Socket error: Fail to Connect
	Password: admin_123	
	Server: shutdown	
Path 2	User name: NULL	Name type error: Username Empty!
	Password: admin_123	
	Server: running	
Path 3	User name: Admin	Password type error: Password Empty!
	Password: NULL	
	Server: running	
Path 4	User name: 管理员	Username error: illegal characters, Accept English Letters, Numbers and Underline
	Password: admin_123	
	Server: running	
Path 5	User name: Admin	Password error: illegal characters, Accept English Letters, Numbers and Underline
	Password: admin密码@#￥	
	Server: running	
Path 6	User name: Admin	Password error: illegal length, Accept Length: 6-18
	Password: who_cares_about_your_password	
	Server: running	
Path 7	User name: admin	Password or username is wrong
	Password (wrong): 1234567	
	Server: running	
Path 8	User name: Admin	Enter the administrator interface
	Password: admin123	
	Server: running	
Path 9	User name: user1	Enter the regular user interface
	Password: user123	
	Server: running	

Figure 81: Test cases

- Graph matrix
  - Then, we can draw the graph matrix based on the above control flow graph:

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	1	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	1	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	1	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	1	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 82: Graph matrix

4.2.8.2 File Download Example 2

We use the file download function(4.2.7) as the testing example.

- Control Flow Graph
  - According to the code, we can draw the control flow graph.

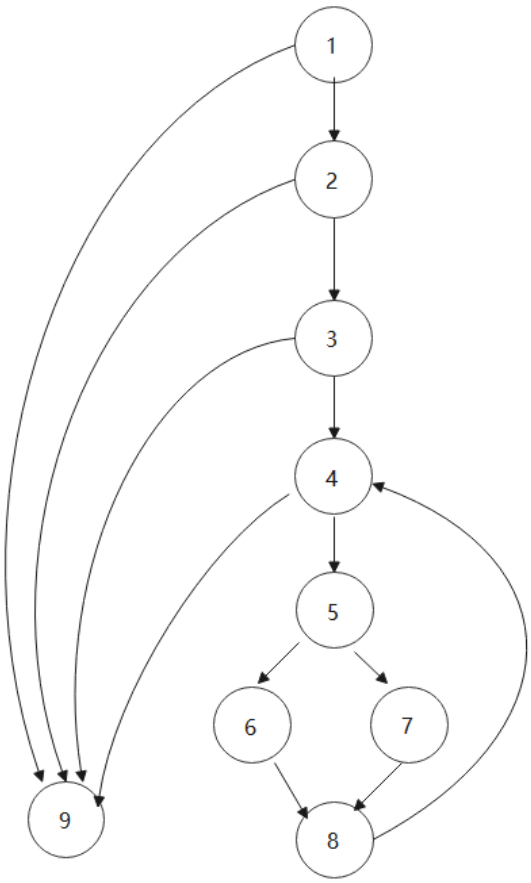


Figure 83: Control flow diagram

- Cyclomatic Complexity

- By using the control flow graph above, we can calculate the cyclomatic complexity of this control flow graph. The cyclomatic complexity V(G)=6.

- Independent Path

- The number of independent paths of this function is 6 as the cyclomatic complexity. Now we list all the independent path.

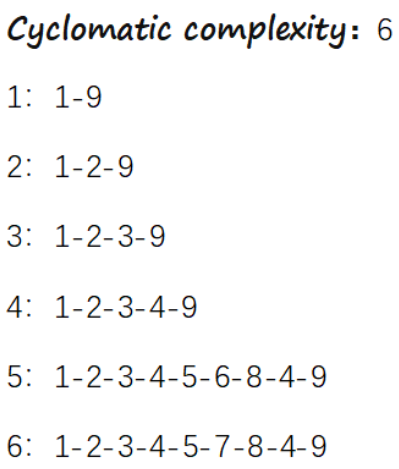


Figure 84: Cyclomatic complexity and the independent path

- Testing Cases

- According to the 6 independent paths above, we can design the testing cases as below:

Test Cases	Input	Output
Path 1	Do not choose any file	No reaction
Path 2	Do not choose located folder	Warning window
Path 3	invalid file name	Warning window
Path 4	file size is 0	Completed and show completed window
Path 5	file size<buffer size	Completed and show completed window
Path 6	file size>buffer size	Completed and show completed window

Figure 85: Test cases

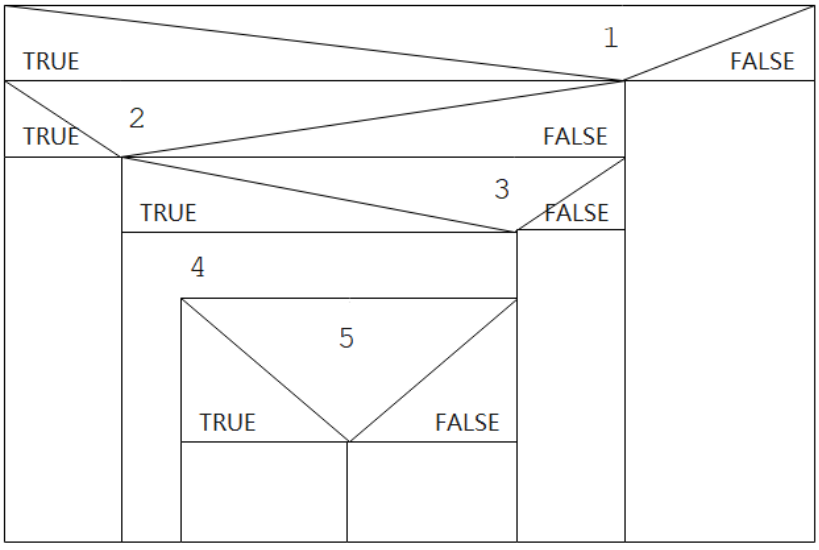
- Graph matrix

- Then, we can draw the graph matrix based on the above control flow graph:

	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	1
3	0	0	0	1	0	0	0	0	1
4	0	0	0	0	1	0	0	0	1
5	0	0	0	0	0	1	1	0	0
6	0	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	1	0
8	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

Figure 86: Graph matrix

- NS Diagram
  - An NS diagram in computer programming is a graphical design representation for structured programming. According to the code, we can draw the NS diagram:




```
1. if item is not None
2. if '.' not in defaultName
3. if fileName
4. while recv_len < filesize_b
5. if filesize_b - recv_len > buffsize
```

Figure 87: NS diagram

4.2.8.3 File Upload Example 3

We use the file upload function as the testing example.

```
    def uploadFile(self):
        fileList, filetype = QFileDialog.getOpenFileNames(self, "Choose files", "C:/", "All Files (*);;Text
Files (*.txt);;""Microsoft Edge PDF Document (*.pdf);;""Microsoft Word Document (*.docx);;""Microsoft
PowerPoint Document (*.ppt);;""Microsoft Excel Worksheet (*.xlsx)")
        # execute only when file exists
        if fileList:
            for file in fileList:
                name = file.split('/')[-1]
                message = 'put ' + name
                print(message)
                self.s.send(message.encode())
                # fileName = r'./' + file
                fileSize = os.path.getsize(file)
                dirc = {
                    'filename': name,
                    'filesize_bytes': fileSize,
                }
                head_info = json.dumps(dirc)
                head_info_len = struct.pack('i', len(head_info))
                self.s.send(head_info_len)
                self.s.send(head_info.encode('utf-8'))
                print(fileSize)
                with open(file, 'rb') as f:
                    while True:
                        print("Start sending1")
                        a = f.read()
                        if not a:
                            break
                        self.s.sendall(a)
                    f.close()
                    print("Send over, waiting")
                    while True:
                        end=self.s.recv(1024)
                        if end == 'EOF'.encode():
                            print("Server recv over")
                            break
                     # time.sleep(0.1)
                     # self.s.send('EOF'.encode())

            QMessageBox.information(self, 'Message', 'Upload compeleted! ', QMessageBox.Ok )
        self.cd('cd same')
        self.lab()
        return
```

Figure 88: Code - upload file

- Control Flow Graph
  - According to the code, we can draw the control flow graph.

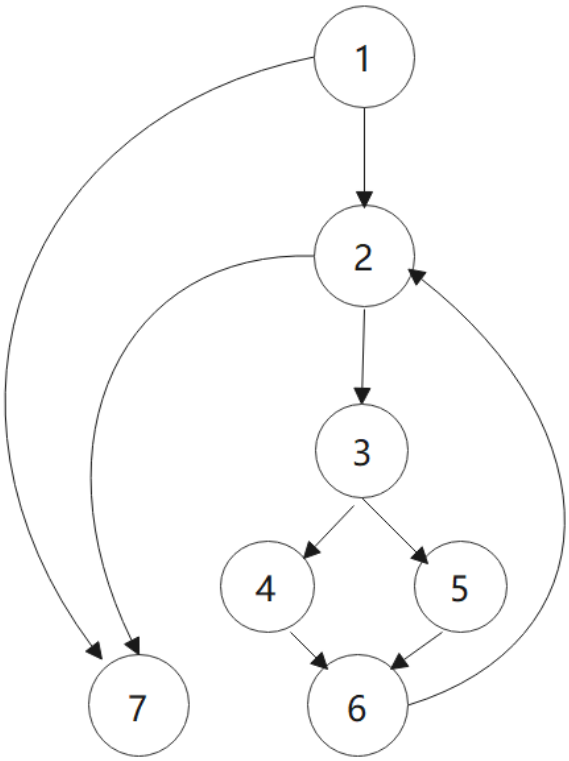


Figure 89: Control flow diagram

- Cyclomatic Complexity
  - By using the control flow graph above, we can calculate the cyclomatic complexity of this control flow graph. The cyclomatic complexity V(G)=4.
- Independent Path
  - The number of independent paths of this function is 4 as the cyclomatic complexity. Now we list all the independent path.

Cyclomatic complexity: 4

1: 1-7

2: 1-2-7

3: 1-2-3-4-6-2-7

4: 1-2-3-5-6-2-7

Figure 90: Cyclomatic complexity and the independent path

- Testing Cases
  - According to the 4 independent paths above, we can design the testing cases as below:

Test Cases	Input	Output
Path1	Do not choose any file	No reaction
Path2	Upload list is empty	Completed and show completed window
Path3	Sending completed without EOF response	Completed and show completed window
Path4	Sending completed with EOF response	Completed and show completed window

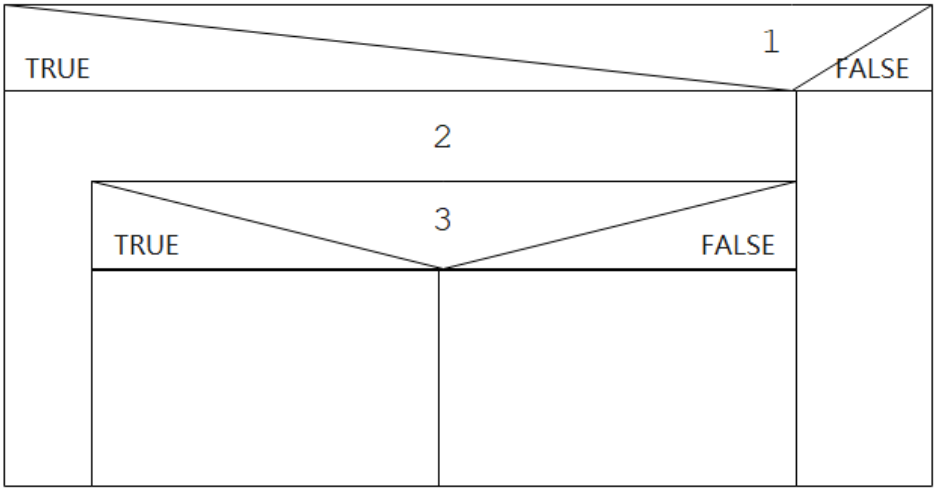
Figure 91: Test cases

- Graph matrix
  - Then, we can draw the graph matrix based on the above control flow graph:

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	1
2	0	0	1	0	0	0	1
3	0	0	0	1	1	0	0
4	0	0	0	0	0	1	0
5	0	0	0	0	0	1	0
6	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0

Figure 92: Graph matrix

- NS Diagram
  - According to the code, we can draw the NS diagram:




```
1. if fileList
2. for file in fileList
3. if end == 'EOF'.encode()
```

Figure 93: NS diagram

4.3 Unit Testing

The tool used for unit testing is Unittest, which provides solutions for creating test cases, test suites, and batch execution.

Use discover() during testing. This function is to recurse from the specified directory to its subdirectories, find all test modules and return a TestSuite containing their objects, and then load the only test file that matches the pattern.

```python
# get all test case
def get_server_allcase():
    discover = unittest.defaultTestLoader.discover(serverCase_path, pattern="Test_*.py")
    suite = unittest.TestSuite()
    suite.addTest(discover)
    return suite
```

Figure 94: Using discover() in unit testing

Use setup as the driving module, and perform piling operations for each unit that needs to be tested.

```python
class unittest_Database(unittest.TestCase):

    def setUp(self):
        self.database = Database('./data.db')

    def test_database1(self):
        self.database.database()

    def test_database2(self):
        self.database.database('data.db')

    def test_create_table(self):
        self.database.create_table()

    def test_insert_table(self):
        self.database.insert_table('bob', 'bob123')

    def test_update_table(self):
        self.database.update_table('bob', 'bob123')

    def test_find_password_by_username(self):
        self.database.find_password_by_username('bob')

    def test_delete_table_by_username(self):
        self.database.delete_table_by_username('bob')

    def test_is_has(self):
        self.database.is_has('bob')
```

Figure 95: driver and stub

Unit test results:

```
test_push_but8 (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_push_but9 (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_scroll_bar_set (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_search (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_sendPicture (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_send_emoji (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_send_msg (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_show_emoji (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_show_info_clg (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_show_msg (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_show_msg_rcd (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_text_chat (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_text_grp_chat (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_video_chat (Test_ChattingMainWindow.unittest_DialogueWindow) ... ok
test_add_button (Test_Login.unittest_MyWindow) ... ok
test_add_label (Test_Login.unittest_MyWindow) ... ok
test_add_line_edit (Test_Login.unittest_MyWindow) ... ok
test_change_icon (Test_Login.unittest_MyWindow) ... ok
test_confirm (Test_Login.unittest_MyWindow) ... ok
test_login (Test_Login.unittest_MyWindow) ... ok
test_set_background_image (Test_Login.unittest_MyWindow) ... ok
test_set_ui (Test_Login.unittest_MyWindow) ... ok
test_settings (Test_Login.unittest_MyWindow) ... ok


----------------------------------------------------------------------
Ran 39 tests in 0.610s

OK
```

Figure 96: The server-side unit test passed

```
test_recvFlash (Test_server.unittest_JudgeServer) ... ok
test_sendDatabase (Test_server.unittest_JudgeServer) ... ok
test_sendInformationFromDatabase (Test_server.unittest_JudgeServer) ... ok
test_updataInformationToDatabase (Test_server.unittest_JudgeServer) ... ok
test_recvPic (Test_server.unittest_PictureServer) ... ok
test_sendPic (Test_server.unittest_PictureServer) ... ok
test_tcp_connect (Test_server.unittest_PictureServer) ... ok

======================================================================
ERROR: test_database1 (Test_Database.unittest_Database)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\PycharmProjects\unittest2\testFolder\server\Test_Database.py", line 11, in test_database1
    self.database.database()
TypeError: 'str' object is not callable

======================================================================
ERROR: test_database2 (Test_Database.unittest_Database)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\PycharmProjects\unittest2\testFolder\server\Test_Database.py", line 14, in test_database2
    self.database.database('data.db')
TypeError: 'str' object is not callable

----------------------------------------------------------------------
Ran 38 tests in 0.326s

FAILED (errors=2, skipped=7)
```

Figure 97: Problems found during the client unit test

4.4 Automated Testing

Automated testing is the process of transforming human-driven testing behaviors into machine execution. Usually, after the test case is designed and passed the review, the tester executes the test step by step according to the procedure described in the test case, and compares the actual result with the expected result. In this process, manpower, time or hardware resources are saved, and the test efficiency is improved.

The automated testing software used by this software is AirTest, a UI automated testing tool produced by NetEase based on image recognition and poco control recognition. The framework of Airtest is an image recognition framework developed by the NetEase team. This framework is a novel graphical scripting language Sikuli. The principle of Sikuli's framework is that computer users do not need to write code line by line, but use screenshots and screenshots to form an automated test program.

4.4.1 Some basic functions of automated testing chat software

1. First, open the automated testing tool-AirTest, write a test script, and use the software to monitor the desktop, the purpose is to realize the picture search of the script screenshot. Before the test process, you also need to log in to another client (user name: wang, password: wang123), the purpose is to automatically test whether the message sent by the other party can be received.

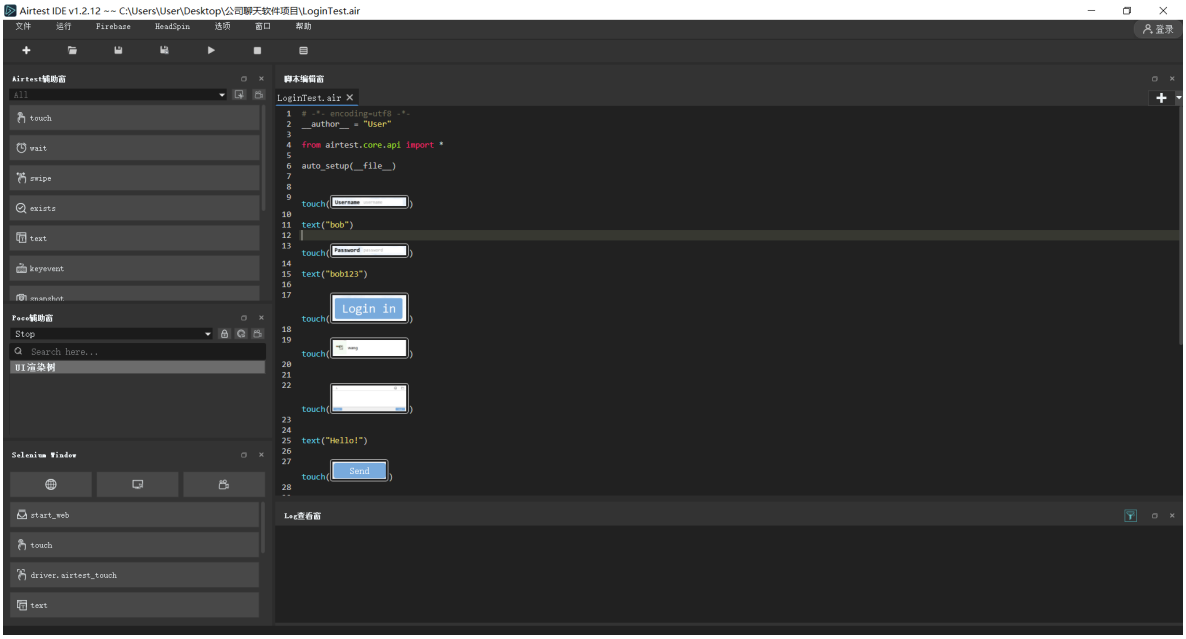


Figure 98: Prepare the automated testing software

2. Click the start button to start the automated test. We can see that AirTest will automatically identify the location of the Username and Password, and automatically enter the account and password, and finally click the Login in button.

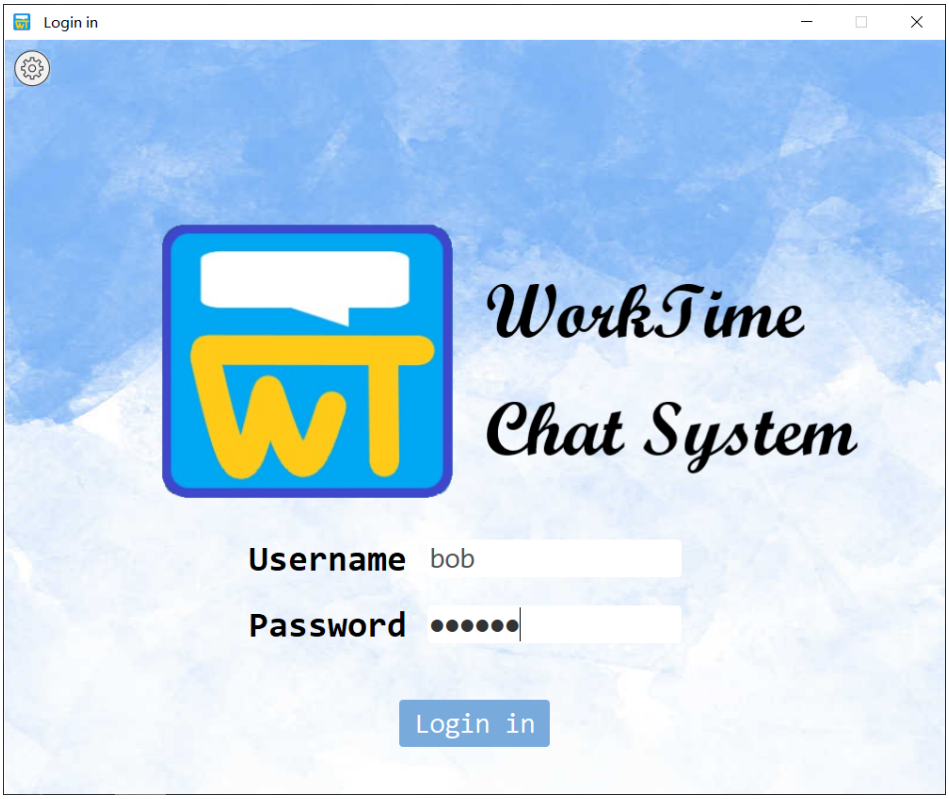


Figure 99: Automated test login function

3. The chat software will jump to the main chat interface. Then find the user wang's according to the picture recognition and click it, and then enter the text "Hello" entered by the script in the dialog box. Finally, the send button is recognized. After clicking, the message is successfully sent to wang's client, and finally the function of the emoji package is tested.

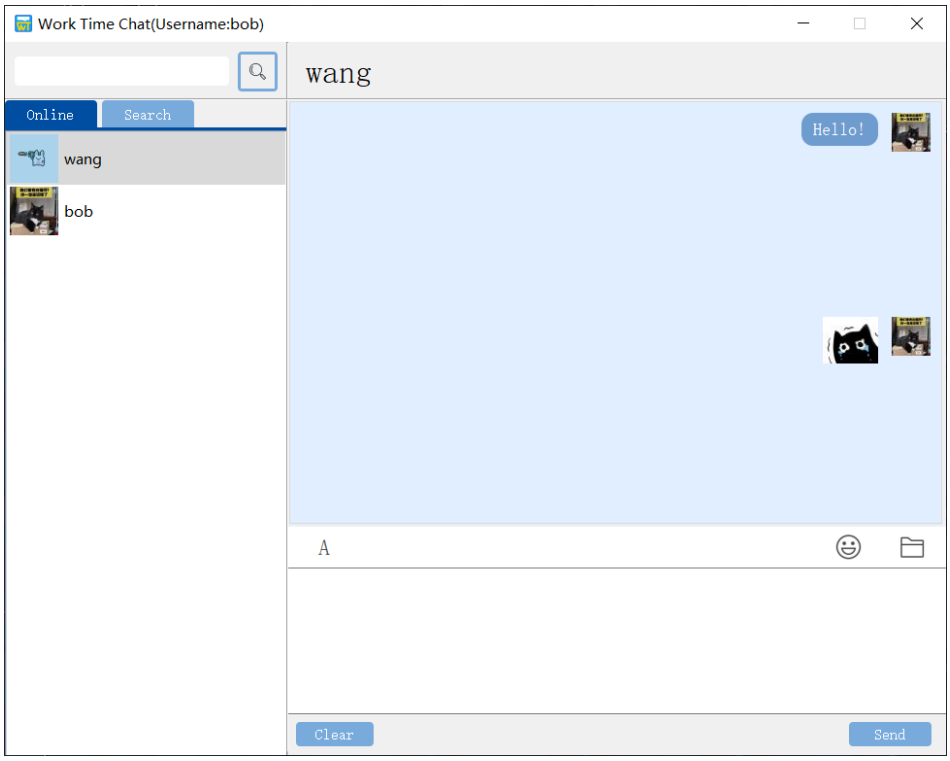


Figure 100: Automated testing of chat and emoji functions

4. The automated testing software will also test the function of searching for users, click the Search button, then enter admin in the input box, and click the button next to it, and find that the administrator is not online now.

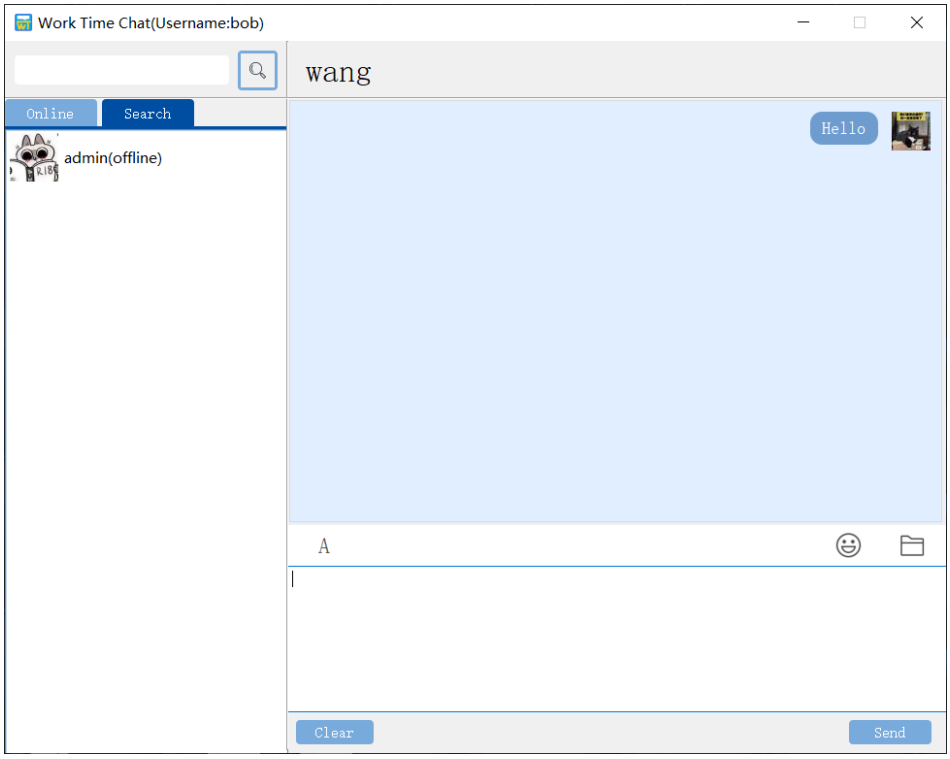


Figure 101: Automated test search function

5. Eventually an automated test report will be generated.

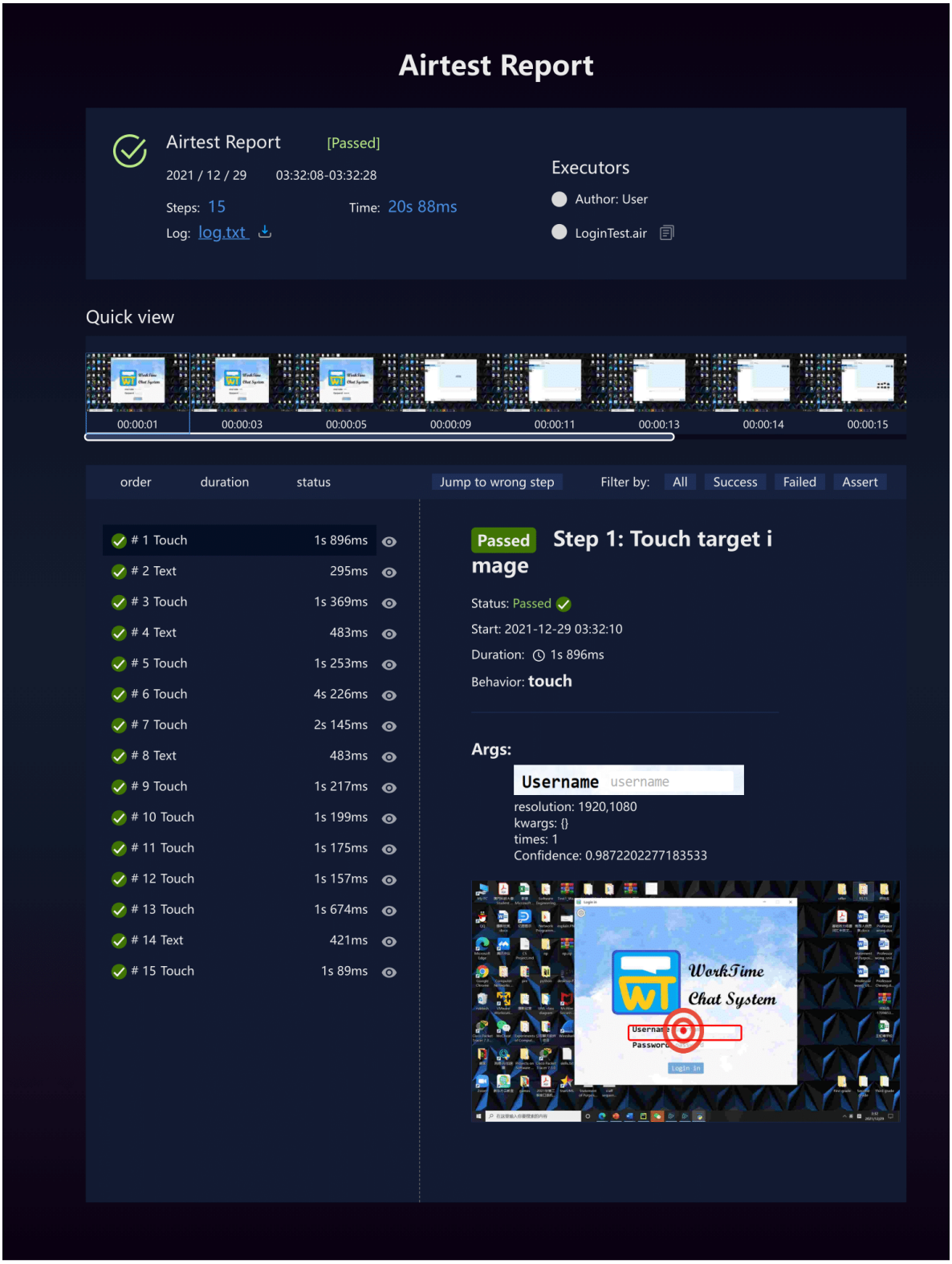


Figure 102: Automated test report

5 Management and Quality Guarantee

Management is a general phrase used to describe a category of computer software designed to help streamline the complexity of large projects and tasks as well as facilitate team collaboration and project reporting. Most management solutions can also handle resource and employee management, schedule coordination, task assignment, budgeting, time and risk analysis, and more. Our management includes the introduction of our team model, PERT chart, Gantt and earned value chart of our process.

A quality guarantee is an assurance of quality and customer satisfaction issued by a company and offered primarily to paying customers who have purchased products or services from the company. In our software, we have the quality guarantee by using CMMI.

5.1 Management

5.1.1 Team Model

A software engineering team can be organized in many different ways, like egoless, chief programmer and chief hierarchy. We prefer to become an egoless team to finish this software engineering project.

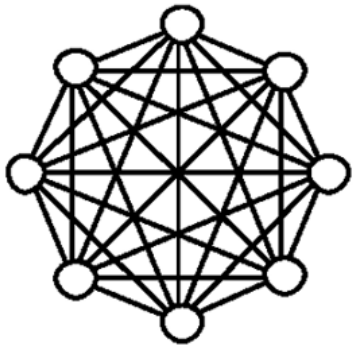


Figure 103: Team model

In our team, all members are equal, and everyone works hard for a common goal. Egoless team leads to greater productivity and reduces resistance to finding errors in the program. This experience of doing project together as a team enhances our sense of cooperation and communication.

5.1.2 PERT Diagram

A PERT chart is a visual project management tool used to map out and track the tasks and timelines. The name PERT is an acronym for Project (or Program) Evaluation and Review Technique.

Our PERT chart is shown as below:

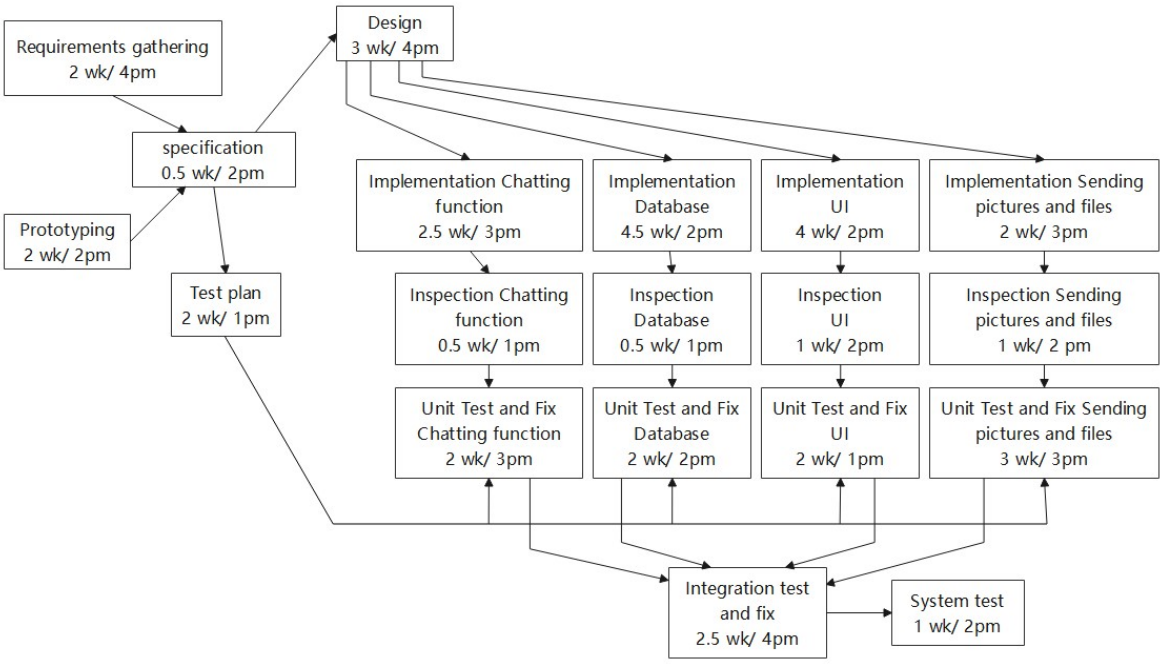
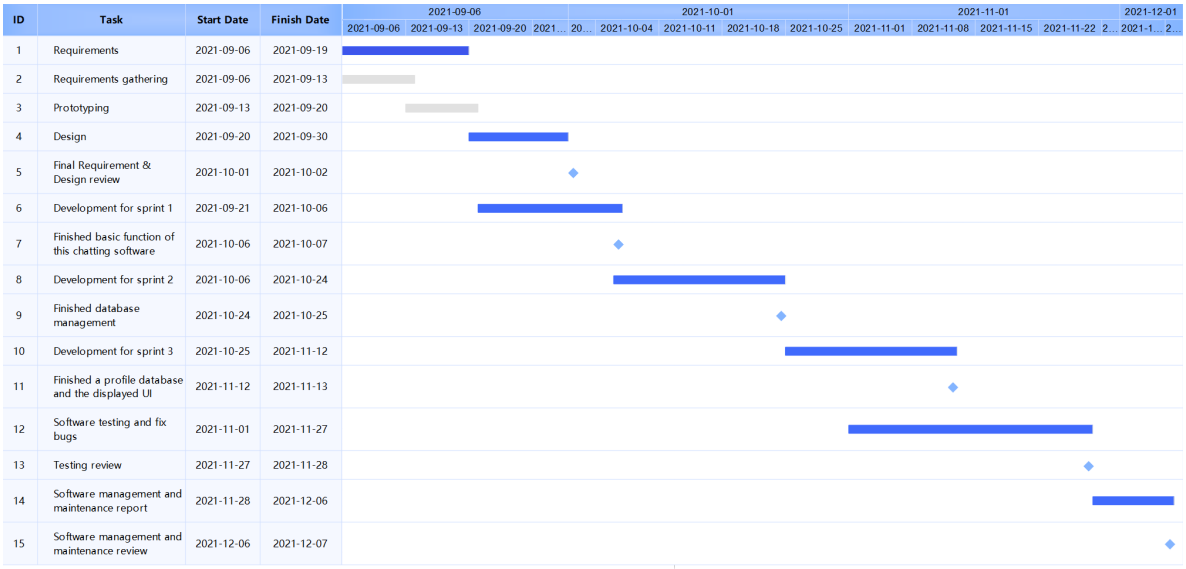


Figure 104: PERT Diagram

This PERT records our complete process of doing this software, includes the total spent time and the workload.

5.1.3 Gantt Chart

Gantt charts are similar to PERT charts in that they offer a graphical view of a project's tasks, schedule, and timelines. A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. Our Gantt chart of doing this project is drawn as below:



ID	Task	Start Date	Finish Date
1	Requirements	2021-09-06	2021-09-19
2	Requirements gathering	2021-09-06	2021-09-13
3	Prototyping	2021-09-13	2021-09-20
4	Design	2021-09-20	2021-09-30
5	Final Requirement & Design review	2021-10-01	2021-10-02
6	Development for sprint 1	2021-09-21	2021-10-06
7	Finished basic function of this chatting software	2021-10-06	2021-10-07
8	Development for sprint 2	2021-10-06	2021-10-24
9	Finished database management	2021-10-24	2021-10-25
10	Development for sprint 3	2021-10-25	2021-11-12
11	Finished a profile database and the displayed UI	2021-11-12	2021-11-13
12	Software testing and fix bugs	2021-11-01	2021-11-27
13	Testing review	2021-11-27	2021-11-28
14	Software management and maintenance report	2021-11-28	2021-12-06
15	Software management and maintenance review	2021-12-06	2021-12-07

Figure 105: Gantt chart

We record the timeline of our process clearly. These dark blue bars are the main top-level tasks and these grey bars are the subtasks. The diamonds are our milestones.

5.1.4 Earned Value Chart

Earned value is the amount of work completed, measured according to the budgeted effort that the work was supposed to consume. It is also called the budgeted cost of work performed. As each task is completed, the number of person-months originally planned for that task is added to the earned value of the project. Earned Value Management is a technique that helps Project stakeholders to measure project performance. Ultimately, this will also help in forecasting the project resources to complete the project. Our earned value chart has three curves including the budgeted cost of work schedules, earned value and the actual cost of work performed. Our earned value chart is shown as below:

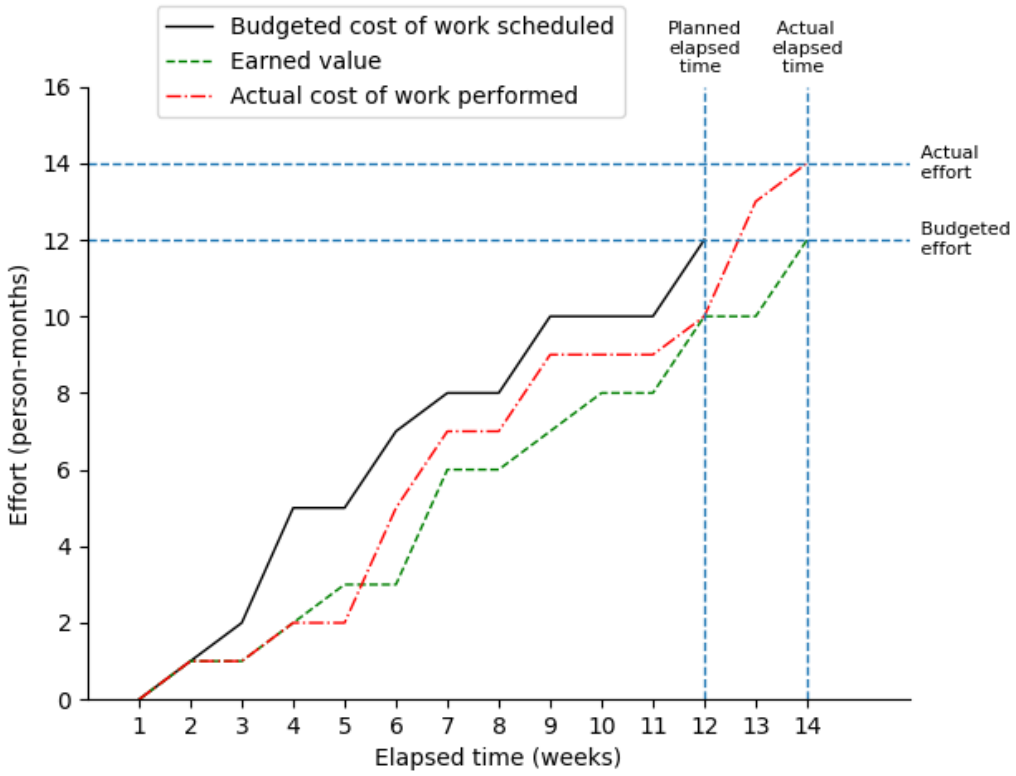


Figure 106: Earned value chart

5.2 Quality Guarantee

5.2.1 CMMI

The Capability Maturity Model Integration, or CMMI, is a process model that provides a clear definition of what an organization should do to promote behaviors that lead to improved performance. With five "Maturity Levels" or three "Capability Levels," the CMMI defines the most important elements that are required to build great products, or deliver great services, and wraps them all up in a comprehensive model.

As our group's evaluation, our project is CMMI Maturity Level 3 – Defined. We believe that we have a set of "organization-wide standards" to "provide guidance across projects and programs." By now our group understand our shortcomings and way to deal with them to improve their processes.