Greedy Snake

Small Team Project

Group 9





Group 9

*Members*

Bo Sun bxs863

Gregory Field . gxf894

Hao Feng . hxf879

Lixuan Dai . lxd901

Shengdong Yan . sxy777

**Content**

[Introduction 1](#_Toc4154662)

[Project Planning 2](#_Toc4154663)

[Brief Description 3](#_Toc4154664)

[System Design 5](#_Toc4154665)

[Server 5](#_Toc4154666)

[Main Server 6](#_Toc4154667)

[Game Server 6](#_Toc4154668)

[Client 6](#_Toc4154669)

[Main Client 7](#_Toc4154670)

[Game Client 8](#_Toc4154671)

[Database 11](#_Toc4154672)

[Graphical User Interface 11](#_Toc4154673)

[Sign In 11](#_Toc4154674)

[Sign Up 12](#_Toc4154675)

[Chatroom 13](#_Toc4154676)

[Game GUI 14](#_Toc4154677)

[Test Plan 16](#_Toc4154678)

[Table 16](#_Toc4154679)

[Cases 17](#_Toc4154680)

[Sign up Successful 17](#_Toc4154681)

[Sign up Fail 17](#_Toc4154682)

[Log In succeed 18](#_Toc4154683)

[Log In fail 18](#_Toc4154684)

[Send Invitation 19](#_Toc4154685)

[Evaluation 20](#_Toc4154686)

[Appendix 21](#_Toc4154687)

[A. Class Diagram 21](#_Toc4154688)

[B. Minutes 22](#_Toc4154689)

[C. Reference 25](#_Toc4154690)

[D. Statement of Contribution 25](#_Toc4154691)

# Introduction

Online gaming and chat rooms have been used together from as early as the late 1970’s, Plato were one of the pioneers in developing this technology. Using chat rooms for network games has been popular for as long as personal computers have been available. “By the late 1970s, it supported several thousand graphics terminals … in multi-user computing were developed on PLATO, including forums, message boards, online testing, e-mail, chat rooms, picture languages, instant messaging, remote screen sharing, and multiplayer games.” [1]. At the present day there are a variety of possibilities for people to communicate for network gaming such as voice chat and in-game messaging.

This project will use software development and project planning to design and implement a network system. The primary use of the system is to allow users to login to a chatroom and play games with other users over a local network. The following project will describe the process of how our group designed, implemented and tested our system.

# Project Planning

The project used a waterfall process model to carry out design, implementation and testing of the system. Each component of the system will go through these three phases and connected when they are complete. The components of the system will be assigned to members of the group and the progress will be discussed in weekly meetings. The allocation of tasks is described below.

*Allocation of Tasks*

*Servers*

*Main Server*

Bo Sun

*Game Server*

Bo Sun

*Clients*

*Chat Room Client*

Lixuan Dai

*Game Client*

Greg Field

*Game*

Shengdong Yan

*Database*

*Hao Feng*

*GUI*

*Lixuan Dai*

*Greg Field*

# Brief Description

This project aims to build an application using a client server architecture. Multiple users will be able to connect to the system and communicate through and messages and interactions through a local network. The system will use two servers to communicate between clients, the first server will handle user account creation, logging in and communication for the chatroom. The second server is used for handling communication between the clients for the game. The communication between the game transfers real time data about the players position and current score.

When the user uses the system for the first time, they will be presented with a login screen. If the user hasn’t previously created an account they will be required to do so. A user can create an account by clicking the sign-up button. They will then be taken to the sign-up screen; they will be required to provide a username and password. There are two fields for the user to provide this information, while the user is entering the username the system checks the database to see if the username already exists. There is a message displayed next to the username entry field that informs the user whether the entered username is ok. After the user has entered this information, they will be required to check the terms and agreement checkbox and then click the sign-up button. The system then checks the database to see whether there is an account with the username that has been provided. If the user already exists a pop-up messages will be displayed and inform the user that the sign up has failed. Otherwise the pop-up message will inform the user that the sign-up has been successful. Upon successful account creation the system will store their username and password in a user information table, the system will also encrypt the password to make the account secure.

If the user has registered an account, they can simply login to the system by entering their username and password and clicking the sign in button. The system then checks the database to check whether the information provided matches the account information stored in the database. If the information is incorrect a pop-up message is shown and informs the user that the login has failed. If the information is correct then the user is taken to the chatroom screen.

The system uses several classes to handle messages, this handles all the possible cases that could occur while using the system. Messages are sent from the user based on certain interaction with the system such as sign-up, login or communication between clients.

The chatroom allows users to see other clients connected to the system and communicate with them. The connected clients are displayed as list on the left-hand side of the screen. If the user wants to communicate with another client, they must first click on the user from the active users list. The user can then send a message by typing into the message field and clicking the send message button. If the user wants to play the game with another user, they can then send them a game invitation. To send a game invitation there is game icon that will then be sent to the other user. A popup message will be sent to both users. The user who sends the invitation will be informed that the game invitation has been sent successfully. The user who receives the game invitation pop-up message will ask them whether they wish to accept the invitation. If the user declines the invitation the other user be informed that the game invitation has not been accepted. If the user accepts the invitation both users will be taken to the games waiting screen.

The waiting screen shows the user the games rules and the controls for playing the game. There are two buttons, one of which the user is able to click. The button displays ready and indicates the user to click when they are ready to start playing the game. If one of the users click the button, the button will change color to green to indicate to the other user that they are ready to being the game. The game will begin once both users have clicked the ready button.

We have designed a multiplayer game that has been inspired by the popular game “Snake” released on Nokia’s mobile phones in the early 2000’s. Our game rules slightly differ from original because our game has multiple users, this required for new game logic to be designed and implemented to the game. The new game logic will be described in the relevant section.

During the game the user is able to pause the game and the game timer countdown will be paused. The game can only continue when the user who has paused the game presses the space bar. If a user exits the game the other user will be shown the end game screen and informed that the game is over.

Once the game had ended the end game screen will be shown with a message informing the user the result of the game. The user will then be able to click the “ok, return” button to return to the chatroom.

# System Design

We have designed and implemented a game called “Greedy Snake” with a chat system. This game supports user sign-up and login so that our server can save users game data. After the user logs in, they will enter a chat page. They can talk with their friends and invite them to join a game. Once two players are ready to begin a game, the GUI will switch to the game interface. In the rest part of chapter Design, we will show the principle GUI of our system.

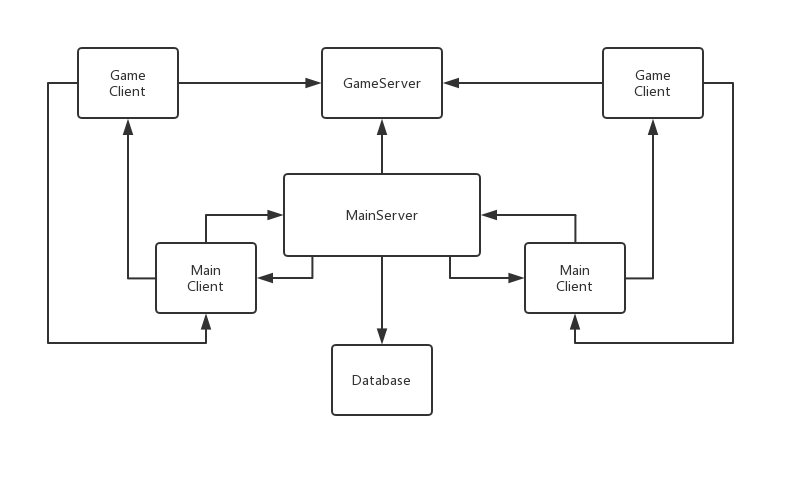


Fig 1 System Architecture

## Server

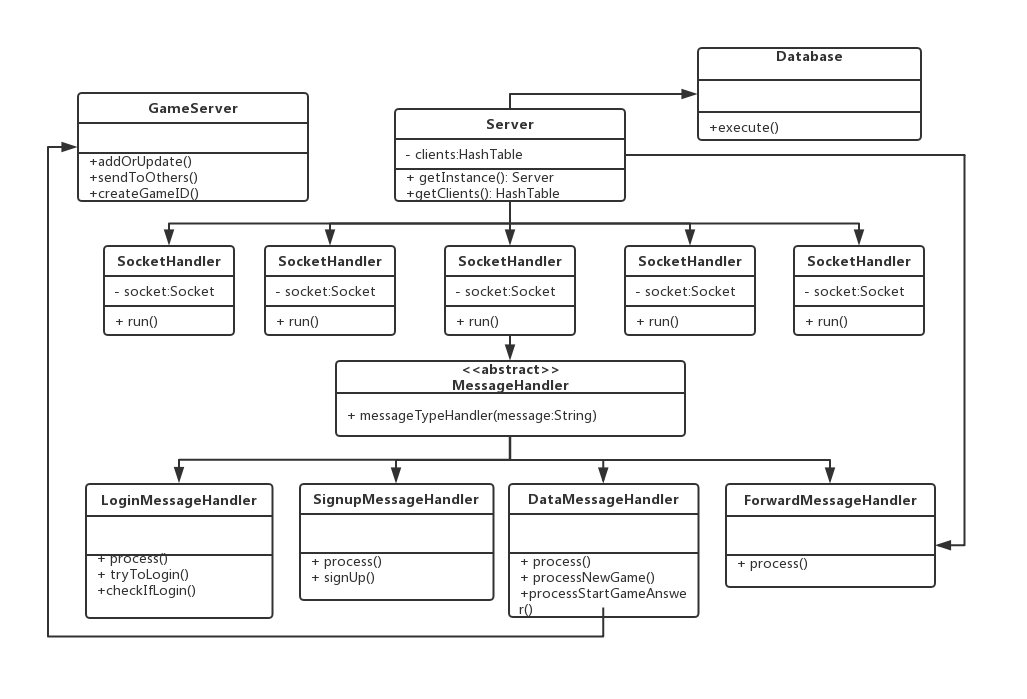


Fig 2 Server Architecture

Due to different limitations and capabilities of the Internet protocols, we have divided our server into 2 parts. The main server uses the TCP protocol, while the game server uses the UDP protocol. The server will broadcast its IP address to all clients when they request connect to server.

### Main Server

In order to provide a stable and continuous Internet service, the main server uses the TCP protocol. Every time a client establishes a connection, a new SocketHandler will be generated. Each SocketHandler has an abstract class of MessageHandler, which will be inherited by a specific MessageHandler. There are 4 kinds of MessageHandlers designed for our system, named LoginMessageHandler, SignupMessageHandler, DataMessageHandler and ForwardMessageHandler. All the MessageHandlers have the abilities to call the database.

LoginMessageHandler will process the request of logging in. It will check if the username exists in the database and if the password is correct. If all information is correct the system will allow users to enter. Otherwise, if there are any problems from the user’s operation a corresponding pop-up message will be displayed.

SignupHandler will process the request of signing up. It will check if the username exists in the database and if both passwords are consistent. If all information is correct, the system will allow users to enter. Otherwise, if there are any problems from the user’s operation a corresponding pop-up message will be displayed.

DataMessageHandler will process the request of creating a new game and forwarding the reply. If the user’s friend has agreed to the game, it will check if the user agrees to join a game. After all users agree to join the game, it will send a request to the game server to start a new game. If the user refuses to join the game, there will be a message sent back to the host.

ForwardMessageHandler is responsible to forward chat messages in the chatroom among clients.

### Game Server

Game server uses the UDP protocol to connect with others. This is a non-connect protocol, which allows the game to run smoothly and prevents a backlog of messages. There are three principle methods to manage game, named createGameID(), addOrUpdate() and sendToOthers(). The createGameID() method is used to create a new game, and save players’ information in the server if a player agrees to join the game. During the game, the addOrUpdate() method is used to receive information and save it in the server, while the sendToOthers() method is applied to forward game information to other players.

## Client

In order to maximize the efficiency, we divided our client into 2 parts, which are the Main Client and Game Client. We agreed that the Main Client will be used for managing the account and manages the messages in the Chatroom. The Game Client is designed for specifically to manage the game communication. By now, there has only been one game in our system. However, we would like to import a new game in our system in the future, we can easily combine new client for our system, due to our low coupling architecture.

### Main Client

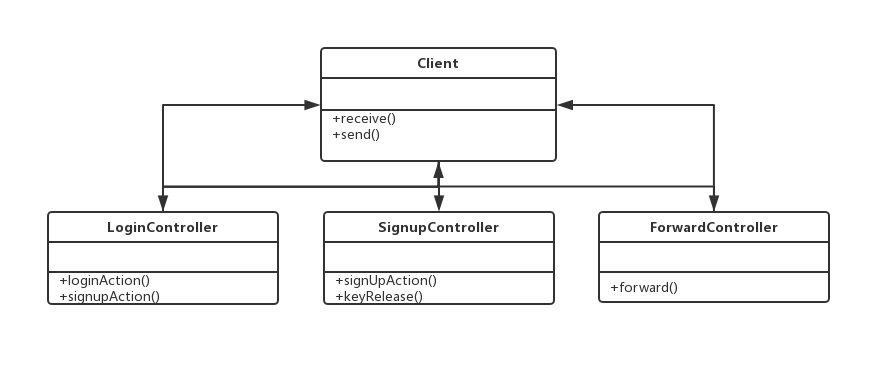


Fig 3 - Main Client Architecture

The main Client is divided into 3 sub-controllers, they are Login, Signup and Forward controllers. Each of them focuses on a specific function for the client. The client class is responsible to receive and send messages from the users and call a relevant controller. The LoginController is used to deal with the action of logging in. If the username exists and the password is correct, the system will allow users to enter, otherwise the system will remind to user to sign up. SignupController will handle the sign-up part in the system. If the new username does not exist in the database, the system will enroll new user information into the database. ForwardController is used to forward messages, including the chat message and game invitation.

### Game Client

#### Game General Design

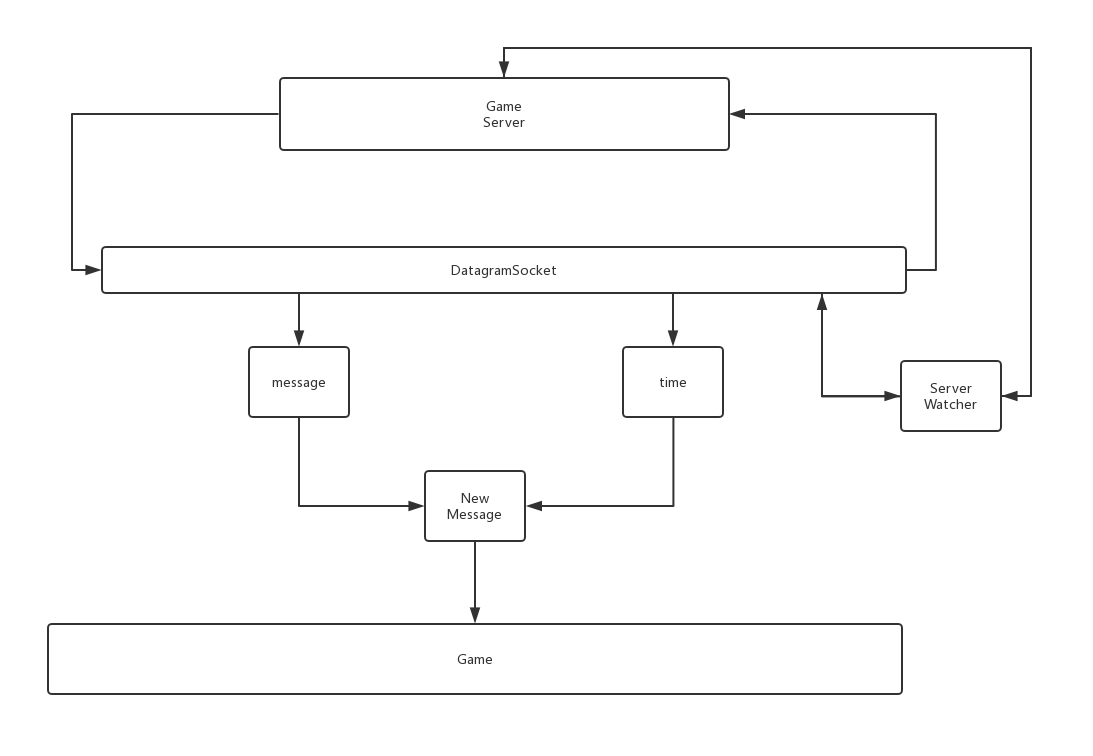


Fig 4 - Game Client Architecture

In the game client, there is a Server Watcher to help the Game Server and dataGramSocket set up the connection. Then they will continuously keep in touch. The dataGramSocket will combine the message from server and the time taken to generate new Message. Only the newest message will be sent to the games GUI and produce graphics to user.

#### Analysis of requirements

The game will have several essential requirements for the game to function as designed.

* The game will require a point system to keep track of each users score.
* The game will require user input to control the direction of their snake
* The game will require list data structure for the snake's body to store its position
* The game will require a method for collision detection for when the snake collides with another players snake or the edges of the game area
* The game will require a timer to countdown the duration of the game
* The game will require a “Node” for users to collect points
* The game will require random location generation for a new node
* The game will require for the snake's length to increase when a point has been collected
* The game will require for the position of a client snake to be sent to all clients
* The game will require for the score of a client to be sent to all clients
* The game will require a synchronized game state for the two play, for example the game will start only after both of them ready.

#### Game Rules

* The user collects points by colliding with the node
* If a user’s snake collides with the edges of the game area their score, the snake’s length and score will be reset and rebirth in the spawning position.
* If a user’s snake collides with another snake their score, the snake’s length and score will be reset and rebirth in the spawning position.
* The game ends when the game timer reaches zero, the games countdown timer duration is set for one and a half minute.
* The user with the highest score at the end of the game is the winner, and the final pane will be play,

#### User Controls

The direction the snake travels is dependent on the current direction. There are eight possible directions for the snake to travel. The snake is able to travel up, right, down, left and also combinations of these directions to move diagonally across the game board. To move diagonally the snakes’ current direction mustn’t be opposite to the new direction. For example, if the current direction is up it is not possible to start moving down.

The space key has been assigned for the user to pause the game. The game can be resumed by pressing the space key a second time. Only the user who has paused the game is allowed to resume the game. The user is also allowed to exit the game using the escape key at any time in the game.

#### Game Implement method

The key idea for our game is to initialize a timeline that will refresh the canvas at a constant frequency (80 milliseconds), before each refresh, the data of all objects will be updated according to the data read from the game server and then drawn onto the canvas, so the snakes can move dynamically. The snake moves by updating the position of the snakes X and Y co-ordinates. For example, if the snake is moving directly upwards the y co-ordinate will be subtracted from the current Y value.

Fig 5 Shows the game in the running state, the timer is displayed at the top center of the screen and shows how much time is left for the game. The user information is displayed in the top right and shows their username and current score. The blue and yellow oval shaped lines are both players. The white oval is the food that the players are aiming to eat.



Fig 5 - Game Running

#### Snake

The snake has been designed to allow the user to control the direction of the snake to move inside the game. The snake moves at a constant speed and aim to move towards the “food” to collect points. The snake consists of two classes, snake and the snake's body. The node is used for a X and Y position and is displayed as an oval. The snake class is used to control the snake's direction, length and game logic.

The isReachBorder() method checks whether the snakes x and y coordinates are within the game area. If they violate these conditions the method returns false and this is used to reset the snake's length and user score. The isCollisionWithSnake() method detects whether the snakes X and Y values are within the area of another snake’s body. These two methods are essential for the games rules to followed.

#### SnakeBody

The snake's body is a linkedlist of points, every point stores the X and Y values of a body element. Every time the snake moves a new first point will be generated for the head, and the last point will be removed from the tail. If the snake eats the node, the last point will not be removed to make the snake grow.

#### Node

The node contains X and Y coordinates and is displayed as an oval shape, which represents the food on the map. If the node is eaten by a snake, another new node will be generated on the map in a random location within the map.

#### TimeCounter

We use another thread to count the time (2minutes), when the time countdown reaches 0 the game will finish and the end menu will be shown.

## Database

The structure of our system database has been illustrated in Fig 6. By now, we have designed three tables user\_info, snake\_player and snake\_game for our system.

The table user\_info records the information of users. There have been two attributes username and password. The username is the primary key, which means a user cannot change their username after registration.

The table snake\_player counts the information of users’ snake game history. We store the sum number users played and record how many times they won and drew.

The players information is saved in table snake\_game.

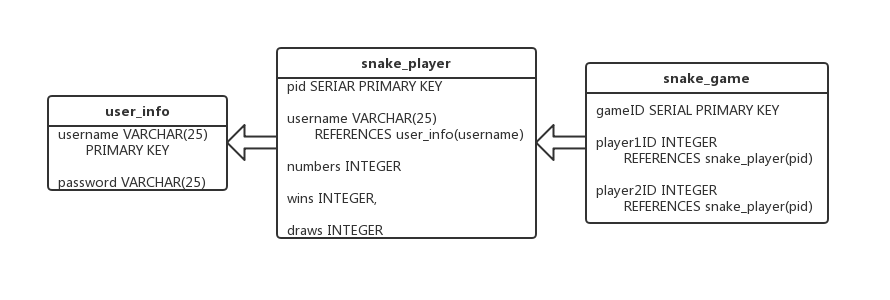


Fig 6 - database structure

## Graphical User Interface

The system Graphical user interfaces (GUIs) have several parts. Besides the GUIs of Sign In, Sign Up and Chatroom, we also designed plenty GUIs for game.

### Sign In

In the Fig 7, we have designed our home page which support users to sign in. If a user doesn’t have an account, we allow them to sign up for a new account. The user is able to sign in when the server is online which is indicated by a light. The green light signifies that the server is online, users can launch our system and start a connection. When the red light is visible the server is offline and users temporarily can’t use our system. Only after a user has inputted their username and password, will the Sign in button be active. On the right upper corner of dialog, we designed an exit icon for users to exit system at any time.

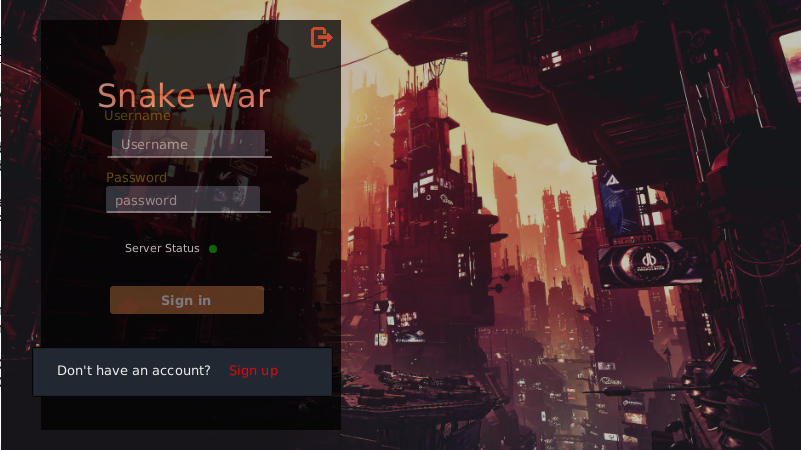


Fig 7 - Home Page

### Sign Up

The Sign-up page is shown on Fig 8, any symbol can be used to set username, and password. The password must be entered twice, in order to make sure that users input the password correctly. The system will automatically check if the username has been occupied. If the input name is a new username, a green “OK” will be displayed next to the username, while a red “NO” will show when a repeated username has been entered. We also import an agreement choice for our terms of service, though there is no detailed term in the hyperlink.

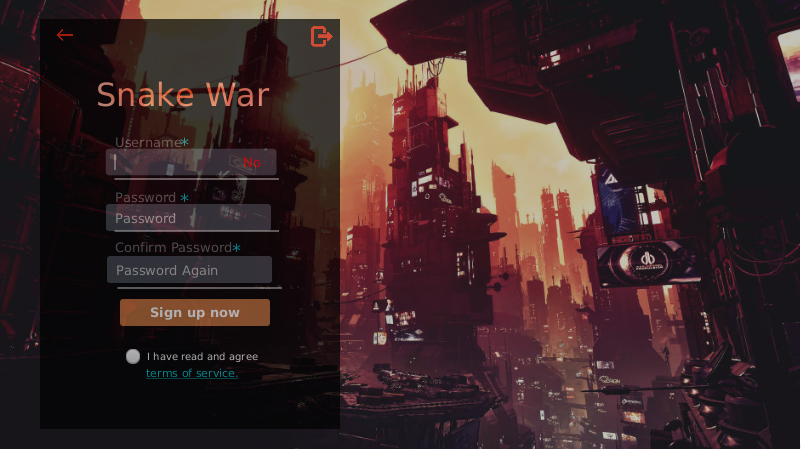


Fig 8 - Sign Up Page

### Chatroom

The chatroom allows multiple users to communicate messages and serves as a game lobby before or after game. (Fig 9)

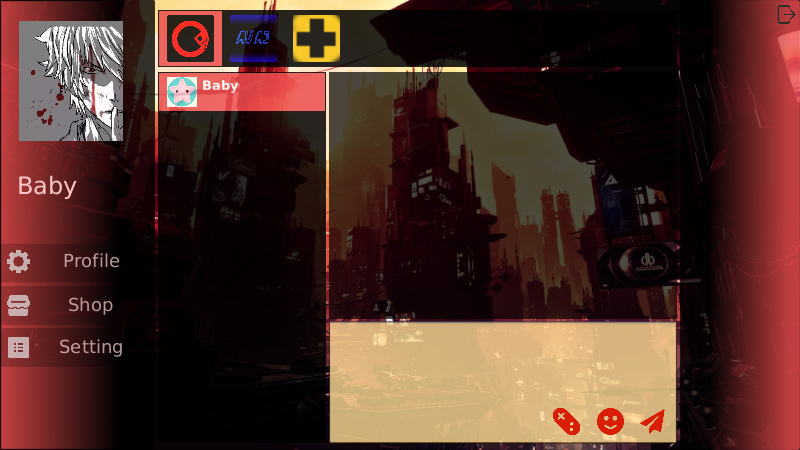


Fig 9 - Chat Room Page

At the left upper corner of GUI, the username is displayed to the user and the chat list is displayed on the left. Each user’s friend will be allocated an icon and their username will be displayed as well. Once a user sends a message to another user, his or her name and icon will be put on the top of chat list. Users can decide the person who he or she wants to talk with by clicking his or her icon, and the chat window will display on the right. There are two areas in the chat window, which are located at the upper and downer of that, respectively. The downer one is worked for users to edit their message, while the other one is used to save the message history. If the user wants to use emoji’s in their message, they can click the smiley face button. After the user has finished edition of the message, they can send it by clicking the paper plane button. They can also invite the friend to play a game together at any time when they select game in the list, which is located at the top of the window and then push the console button. At the left downer corner of the GUI, users can enter Profile, Shop or Setting Page by selecting corresponding button.

The Profile, Shop or Setting Page have not yet been implemented but plan to be included in the future. The profile page is used to manage personal information. Users can buy games and peripheral products in the shop page. The setting page is used to set personal preferences.

### Game GUI

#### Menu’s

The game has several menu’s that are used to handle each state the game could be in, the possible game states which are ready, running, paused and end of game.

#### Waiting Screen

Below in Fig 10 is the design for the waiting screen, users will be presented with this screen after the game invitation has been accepted. This menu shows the title of the game, the game rules and with the users' details. The user details are will are retrieved from the database and include their username, total number of games played and the number of games they have won.

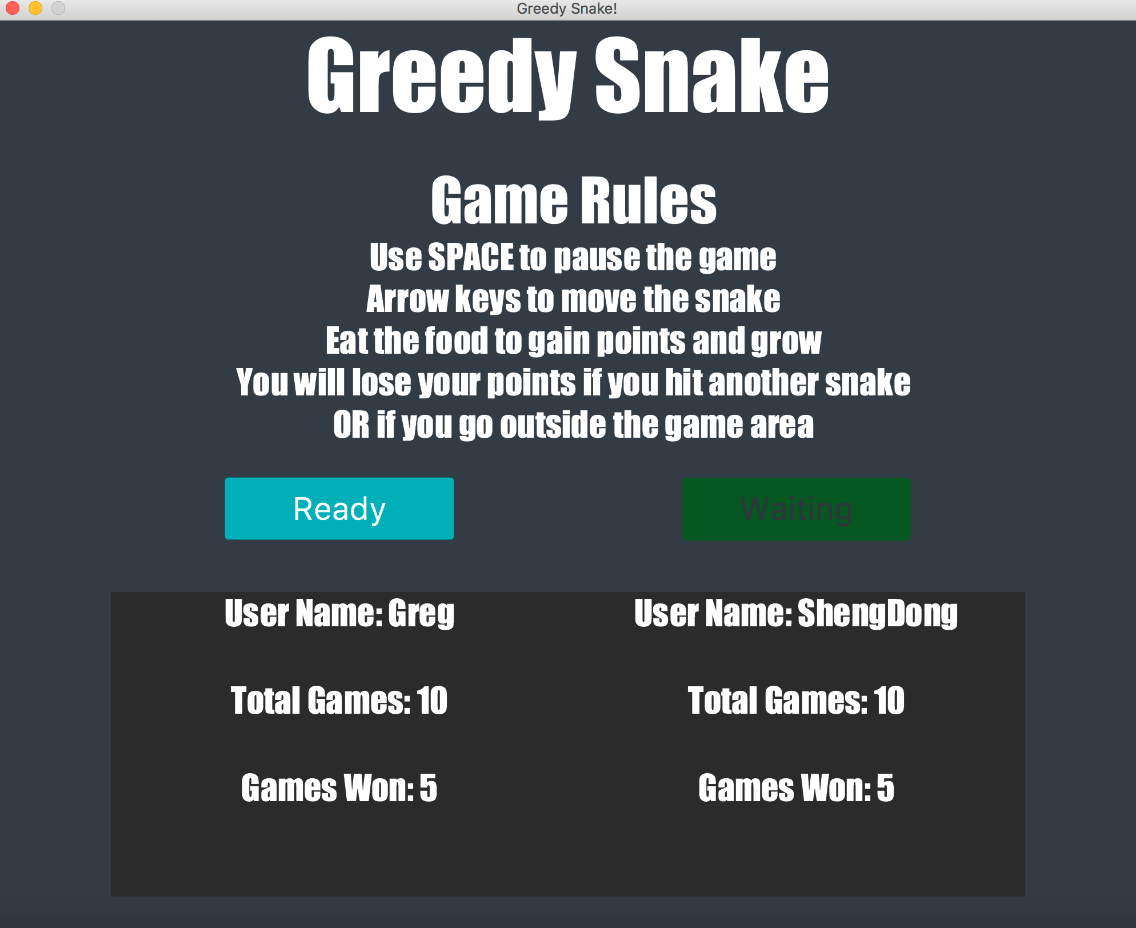


Fig 10 - Waiting screen

#### Pause Menu

In Fig 11 Is the pause menu, the user can enter this screen during the game by pressing the space key. Only the user who has paused the game is able to un-pause the game.



Fig 11 - Pause Screen

#### End of Game Menu

In Fig 12, Is the end game screen, this will show the result of the game and allow for the user to return to the Chatroom.

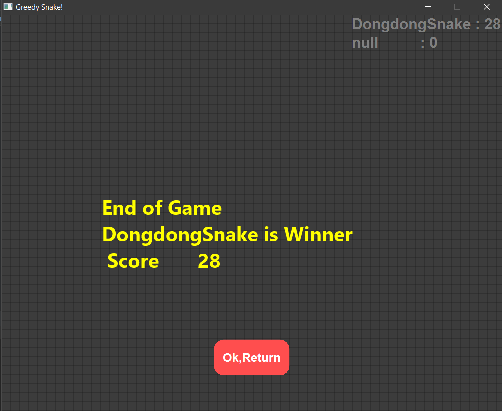


Fig 12 - End of Game Screen

# Test Plan

We attempted many tests on our application, and conclude them into a table. There are also several test cases listed below. The plan is designed to test the vital components of the system to ensure a user can create an account, login and control their snake within the game.

## Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Method of Testing | Expected | Actual |
| Create a user account | Click sign up button, enter username, enter password, click user agreement checkbox and then click sign up button | String username, String password (Encrypted) | String username, String password (Encrypted) |
| Correct Username and Password | Enter username and password and click Sign In button | success = yes  reply = " Click confirm to continue " | success = yes  reply = " Click confirm to continue " |
| Correct Username Incorrect Password | Enter username and wrong password | success = no  reply = "Incorrect username or password" | success = no  reply = "Incorrect username or password" |
| SQL Injection | In username field enter “1"; drop table user\_info;”  Enter anything into password field, Click Login button | success = no  reply = "Incorrect username or password" | success = no  reply = "Incorrect username or password" |
| Game Invite | Select other user, click game invite button | Pop box on other client, with game invitation | Pop box on other client, with game invitation |
| Move snake up | Up arrow key pressed once | direction = DIR\_UP | direction = DIR\_UP |
| Move snake right | Right arrow key pressed once | direction = DIR\_RIGHT | direction =DIR\_RIGHT |
| Move snake down | Down arrow key pressed once | direction =DIR\_DOWN | direction =DIR\_DOWN |
| Move snake left | Left arrow key pressed once | direction = DIR\_LEFT | direction = DIR\_LEFT |
| Move snake north east | Starting direction DIR\_UP then right key pressed once | direction = DIR\_RIGHTUP | direction = DIR\_RIGHTUP |
| Move snake north west | Starting direction DIR\_UP then left key pressed once | direction = DIR\_LEFTUP | direction = DIR\_LEFTUP |
| Move snake south east | Starting direction DIR\_DOWN then right key pressed once | direction = DIR\_RIGHTDOWN | direction = DIR\_RIGHTDOWN |
| Move snake south west | Starting direction DIR\_DOWN then left key pressed once | direction = DIR\_LEFTDOWN | direction = DIR\_LEFTDOWN |
| Snake collision with node | SnakeBody.size() = Constants. DEFAULT\_LENGTH  Score = 0 | SnakeBody.size() == 7  Score == 1 | SnakeBody.size() == 7  Score == 1 |
| Move snake in opposite direction | Starting direction DIR\_UP  Press down arrow key | Direction = DIR\_UP | Direction = DIR\_UP |

## Cases

### Sign up Successful

If there is a new user sign up, and the username has not been registered in the system. The sign-up process will succeed and there will be a pop-up message displayed. (Fig 13)

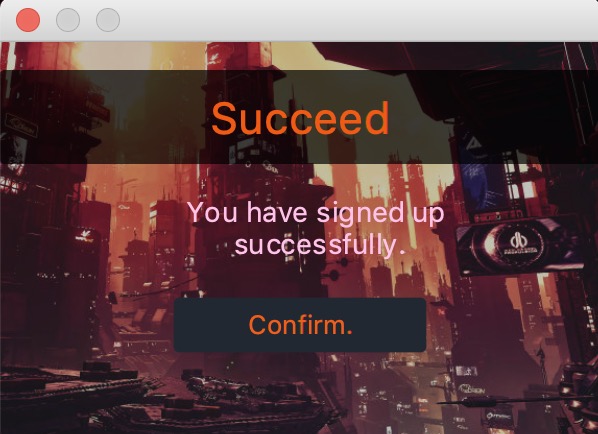


Fig 13 Sign Up Succeed

### Sign up Fail

If the user wants to register with a username that already exists in systems database, a pop-up message will be presented to remind the user it failed. (Fig 14)

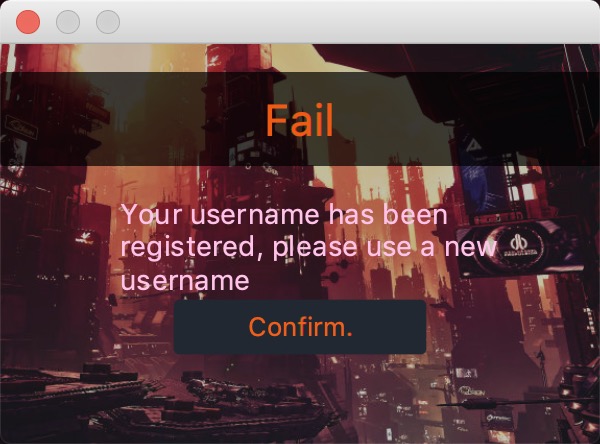


Fig 14 - Sign Up Failed

### Log In succeed

If the login is successful a pop-up message when users input correct username and password saved in system database, only then can they be authorized to enter our system. A pop-up message will display after the corresponding operations. (Fig 15)

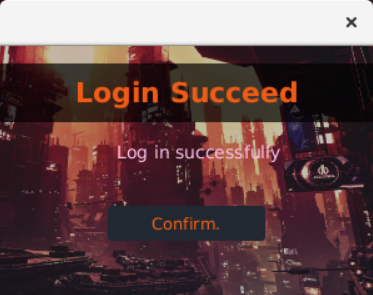


Fig 15 - Log In Successful

### Log In fail

If the user enters a username or password that is incorrect, they will not be allowed to enter the system. (Fig 16)

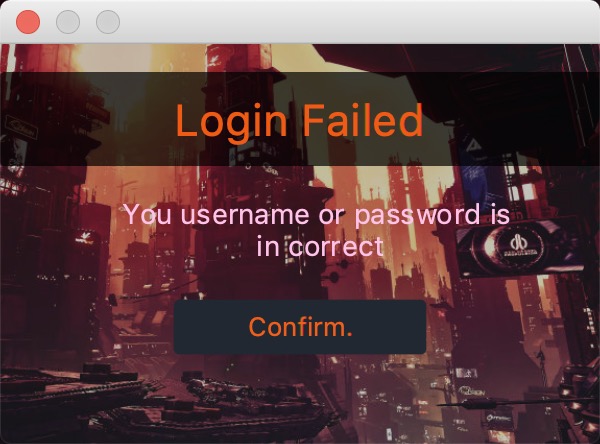


Fig 16 - Log In Failed

### Send Invitation

After a user sends a game invitation, the user will be sent a confirmation (Fig 17) and a pop-up message will be displayed on the receiver’s computer. (Fig 18)

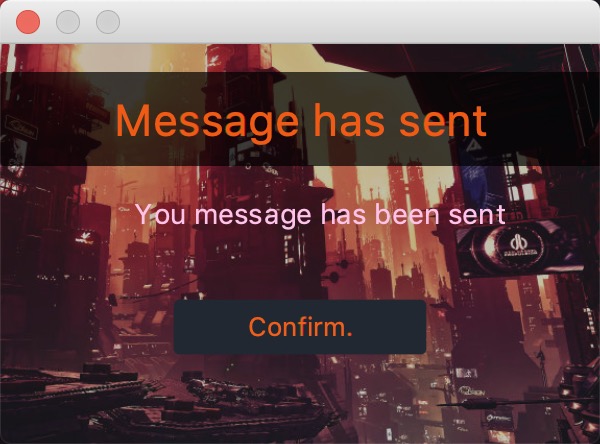


Fig 17 - Sent Confirmation

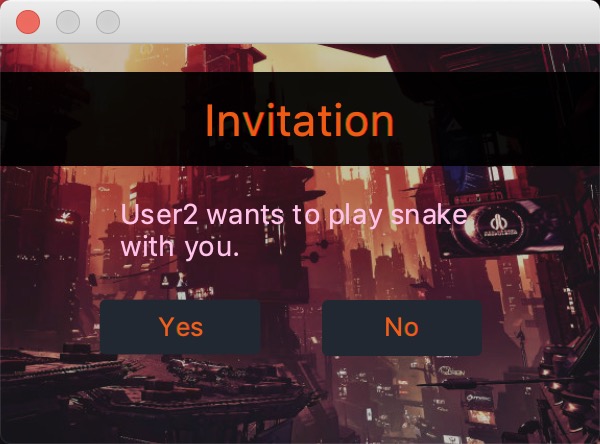


Fig 18 - Invitation

The receivers can decide whether to accept an invitation, the feedback information is sent to inviter if they disagree. (Fig 19)

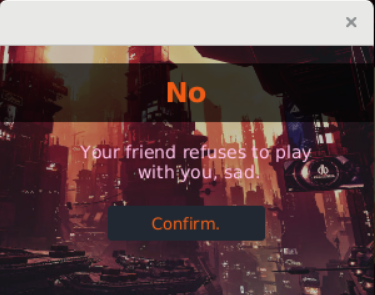


Fig 19 - Refuse invitation

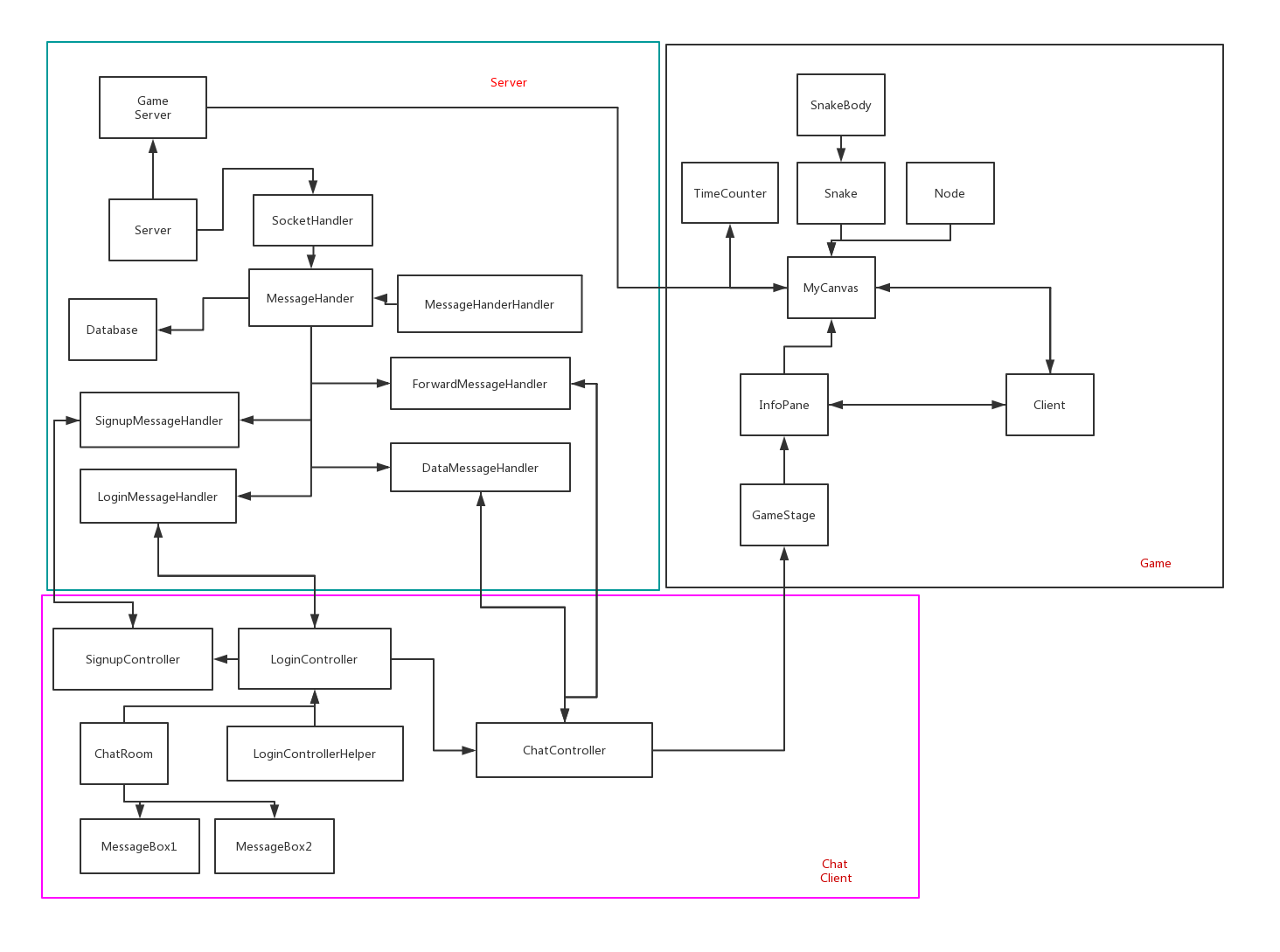
# Evaluation

The project successfully designed and implemented a system that uses a client server architecture to communicate over a network. The system allows for clients to send messages to other clients connected to the system and also send game invitations. The system also successfully stores user accounts and game information in a database. The system also successfully stores user accounts and game information in a database. The game is able to keep a record of users scores and determine a winner. The project did face some technical issues, these were overcome by sharing existing knowledge within the group.

In future, we plan to implement a chat bubble box into the chat window to make our UI more aesthetically pleasing. We would like to implement emoji’s and image sending. We also plan to allow users to reset their password and use custom avatars. We are also considering to allow more players to join a game and add AI to control multiple snakes to make our game more exciting and competitive. We plan to import multiple games into our system, and then allow users set up groups for communication. We also plan to make our system scalable to facilitate a large number of users to make our system available around the world. Thus, the Chatroom would become a game lobby and users could communicate about their game experience with friends. We will add a nickname function to support users to change their name at any time. We have locally recorded the game information; we plan to share the user game information during the game in future.

# Appendix

## A. Class Diagram



## B. Minutes

1st Meeting

Date: 24/02/19 at 14:00

Duration: 60 minutes

Members present:

* Greg Field
* Bo Sun
* Shengdong Yan
* Hao Feng
* Lixuan Dai

Work Agreed upon:

1. We agreed to make a 2D multiplayer game. The game would have a time limit, when the game finished the winner would be the user with the highest score.
2. We also agreed to use JavaFX for the GUI.
3. We agreed to meet at 14:00 every Wednesday.

2nd Meeting

Date: 27/02/19 at 15:00

Duration: 90 minutes

Members present:

* Greg Field
* Bo Sun
* Shengdong Yan
* Hao Feng
* Lixuan Dai

Work Agreed upon:

1. We discussed which game we are going to implement, and finally the game we decided to do is inspired from the game Great Ghoul Duel (<https://www.google.com/doodles/halloween-2018>). We would simplify it to a 1 vs 1 version.

1. We also allocated the work for each member: ShengDong, Feng Hao and Greg would do the game part, while Bo and Lixuan would do the server and client.

3rd Meeting

Date: 02/03/19 at 14:00

Duration: 2 hours

Members present:

* Greg Field
* Bo Sun
* Shengdong Yan
* Hao Feng
* Lixuan Dai

Work Agreed upon:

1. We agreed to change the game we would make, since sometimes the rule of the old game is a little confusing. We would implement the multiplayer snake game, which is a little similar with the original one.
2. We decided to use JSON to achieve message sending and receiving, and we decided the structure of server and client, and we also defined the message type and format for different phases.
3. We also defined the database structure and created database tables, and then finished the code that using JDBC to achieve query, insert and delete operation.

4th Meeting

Date: 06/03/19 at 14:00

Duration: 60 minutes

Members present:

* Greg Field
* Bo Sun
* Shengdong Yan
* Hao Feng
* Lixuan Dai

Work Agreed upon:

1. We summarized the work we had finished from last week. The server and client part were almost finished, now the data transmission between server and client was successful.
2. We agreed to implement the chatroom. After signing in successfully, the user will open the chatroom interface, which allows user to send message or game invitation to others.
3. We made an offline snake game, so the next work is to implement the online version.
4. We also agreed that we would start to make the GUI using JAVAFX.

5th Meeting

Date: 13/03/19 at 14:00

Duration: 60 minutes

Members present:

* Greg Field
* Bo Sun
* Shengdong Yan
* Hao Feng
* Lixuan Dai

Work Agreed upon:

1. We finished the first version of GUI, and agreed to continue making it better, and we wrote controller to combine the fxml file and client. We could use GUI to achieve login, sign up and a series of operation.
2. We also finished the fundamental functions of chatroom.
3. Now the online version of snake game is implemented, but it can only support one player. We found that we need to reconstruct some codes to achieve multiplayer online version.

6th Meeting

Date: 17/03/19 at 14:00

Duration: 60 minutes

Members present:

* Greg Field
* Bo Sun
* Shengdong Yan
* Hao Feng
* Lixuan Dai

Work Agreed upon:

1. Now we finished almost works.
2. We got a new version GUI with same style, including the sign up/in page, chatroom page and message box.
3. We finished the 1 vs 1 version online snake game, and we tried to use our own server and client to implement it.

7th Meeting

Date: 20/03/19 at 15:00

Duration: 8 hours

Members present:

* Greg Field
* Bo Sun
* Shengdong Yan
* Hao Feng
* Lixuan Dai

Work Agreed upon:

1. We meet together for debugging and report writing. The game is successfully implemented by our server and client, but now there are lots of bugs. We tried to fix them.
2. We agreed to expand it as a game platform for future use, and we reconstruct the code of game server, it would only forward message to clients, so that it is easier to access different games in the future.

8th Meeting

Date: 21/03/19 at 17:00

Duration: 7 hours

Members present:

* Greg Field
* Bo Sun
* Shengdong Yan
* Hao Feng
* Lixuan Dai

Work Agreed upon:

We meet together for debugging and report writing.

## C. Reference

[1] Ipfs.io. (2019). *PLATO (computer system)*. [online] Available at: https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/PLATO\_(computer\_system).html [Accessed 22 Mar. 2019].

## D. Statement of Contribution

Bo Sun bxs863 - 30 %

Gregory Field gxf894 - 17.5 %

Hao Feng hxf879 - 17.5 %

Lixuan Dai lxd901 - 17.5 %

Shengdong Yan sxy777 - 17.5 %

The contribution of **codes** has been listed below:

**Servers**

*Main Server*

Bo Sun

*Game Server*

Bo Sun

**Clients**

*Chat Room Client*

Lixuan Dai

*Game Client*

Greg Field

*Game design*

Shengdong Yan

**Database**

*Hao Feng*

**GUI**

*Lixuan Dai*

*Greg Field*

**Git maintain:** Bo

**JavaDoc:** Shendong, Bo

The contribution of **report** has been listed below:

**Consultant:**  Bo

**Modify:**  Greg

**Draft:**

Introduction: Greg

Project Planning Greg, Shendong

Brief Description Greg

System Design (Preface) Greg, Bo

Server (Preface) Hao, Bo

Main Server Hao

Game Server Hao

Client (Preface) Hao

Main Client Hao, Bo

Game Client Greg, Shendong, Hao, Bo

Database Hao

Graphical User Interface (Preface) Hao

Sign In Hao, Bo

Sign Up Hao

Chatroom Hao, Lixuan

Game Greg, Shendong

Testing (Table) Greg

Cases Hao, Lixuan

Evaluation Greg, Hao, Shendong, Lixuan

Appendix

A. Class Diagram Bo

B. Minutes Lixuan, Greg

C. Reference Greg

D. Statement of Contribution Hao