ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

PBL4 : DỰ ÁN HỆ ĐIỀU HÀNH VÀ MẠNG MÁY TÍNH

Đề tài

XÂY DỰNG TRÒ CHƠI TÌM SỐ ONLINE THÔNG QUA MÔ HÌNH CLIENT – SERVER

GIẢNG VIÊN HƯỚNG DẪN: ThS. Trần Hồ Thuỷ Tiên.

SINH VIÊN THỰC HIỆN:

Tên sinh viên : Lê Hải Khoa	**LỚP: 22T_KHDL**	**NHÓM: 22N15B**
Tên sinh viên : Hồ Nguyễn Thế Vinh	**LỚP: 22T_KHDL**	**NHÓM: 22N15B**
Tên sinh viên : Ngô Xuân Vinh	**LỚP: 22T_KHDL**	**NHÓM: 22N15B**

Đà Nẵng, 05/01/2025

MỤC LỤC

DANH SÁCH HÌNH VẼ *4*
DANH SÁCH CÁC TỪ VIẾT TẮT *5*
MỞ ĐẦU *6*
CHƯƠNG 1. CƠ SỞ LÍ THUYẾT *7*
1.1. Thế nào là mạng máy tính ? 7
1.2. Các thành phần chính của mạng 7
1.3. Phân loại mạng máy tính 7
1.4. Mô hình MVC (Model – View – Controller) 7
1.5. Xử lý đồng bộ (synchronized) 9
CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG *10*
2.1. Chức năng hệ thống 10
2.2. Cơ sở dữ liệu 10
2.3. Biểu đồ tuần tự (Sequence Diagram) 11
2.4. Cách trò chơi hoạt động (Game Operation) 13
2.5. Xây dựng hệ thống: 14
CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ *24*
3.1. Cửa sổ đăng nhập 24
3.2. Cửa sổ đăng kí 26
3.3. Giao diện trang chủ 27
3.4. Giao diện sau khi nhấn nút PLAY 28
3.5. Giao diện đợi 28
3.6. Giao diện chơi trò chơi 29
3.7. Giao diện khi hai người đang chơi: 29
3.8. Giao diện sau khi chọn nút JOIN ROOM 30
3.9. Giao diện thông tin cá nhân 31
3.10. Giao diện hỗ trợ cách chơi và luật chơi 31
3.11. Giao diện khi muốn LOGOUT 32
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN *33*
TÀI LIỆU THAM KHẢO *34*

DANH SÁCH HÌNH VẼ

Hình 1. Mối quan hệ giữa các thành phần trong mô hình MVC 7
Hình 2. Sơ đồ hoạt động thông thường của một mô hình MVC 8
Hình 3. Sơ đồ tuần tự của một mô hình MVC thông thường 8
Hình 4. Mô hình Client-Server 9
Hình 5. Thư viện Java Swing 9
Hình 6. Cơ sở dữ liệu 11
Hình 7. Đăng nhập 11
Hình 8. Đăng kí 12
Hình 9. Tìm trận và các trạng thái tìm trận 12
Hình 10. Sơ lược một ván trò chơi tìm số 13
Hình 11. Các thư mục lớp View 14
Hình 12. Các thư mục lớp Model, Controller cũng như phần Application 17
Hình 13. Class ServerMain 17
Hình 14. Class ClientMain 18
Hình 15. Class ClientController 20
Hình 16. Class ServerController 22
Hình 17. Class RoomController 23
Hình 18. Cấu trúc file lớp Model 24
Hình 19. Class DatabaseManager 24
Hình 20. Class History 25
Hình 21. Class Player 26
Hình 22. Class Room 27
Hình 23. Màn hình đăng nhập 28
Hình 24. Đăng nhập thành công 28
Hình 25. Đăng nhập thất bại 29
Hình 26. Cửa sổ đăng kí 30
Hình 27. Đăng kí thành công 30
Hình 28. Đăng kí thất bại 31
Hình 29. Cửa sổ giao diện 32
Hình 30. Vào phòng chơi 33
Hình 31. Đợi trong lúc tìm trận 34
Hình 32. Giao diện một ván đấu 35
Hình 33. Giao diện của 2 client chơi với nhau 36
Hình 34. Nhấn nút JOIN ROOM 36
Hình 35. Thông tin người chơi 37
Hình 36. Giao diện luật chơi và thông tin nhà phát hành 37
Hình 37. Giao diện LOGOUT 38

DANH SÁCH CÁC TỪ VIẾT TẮT

Từ viết tắt	Giải nghĩa
LAN	Local Area Network
MAN	Metropolitan Area Network
WAN	Wide Area Network
P2P	Peer-to-Peer
MVC	Model – View – Controller

MỞ ĐẦU

Thời đại Internet kết nối vạn vật yêu cầu các sản phẩm phát triển cần kết nối hệ thống mạng máy tính để các thiết bị có thể dùng từ xa mà không ảnh hưởng đến mã nguồn của sản phẩm.

Trước bối cảnh đấy, việc hiểu mạng và lập trình mạng xem như nền tảng với ngành mạng máy tính. Cho đến thời điểm hiện tại đã có rất nhiều mô hình kết nối mạng máy tính khác nhau. Một trong số những mô hình phổ biến nhất hiện tại là mô hình mạng máy tính Client – Server (hay mô hình khách – chủ).

Việc hiểu được mô hình Client – Server hoạt động ra sao theo một cách tổng quát để đưa vào **"PBL4 : Dự án mạng máy tính và hệ điều hành"** là một điều khá mơ hồ. Thay vào đó chúng tôi dùng những tình huống cụ thể có thể áp dụng mô hình trên. Một trong số đó là trò chơi khá phổ biến trong giới học sinh – Tìm số.

Vốn dĩ được biết đến với luật chơi đơn giản : khoanh các số theo thứ tự từ 1 cho đến hết số được ghi sẵn trên một mặt phẳng như giấy, bảng,… luật chơi đơn giản nhưng rèn luyện được khả năng quan sát, phản xạ cũng như khả năng tranh chấp giữa các người chơi. Dễ chơi là vậy, dễ hiểu là vậy, nhưng để đưa trò chơi này lên máy tính lại là câu chuyện khác. Việc phải xây dựng trò chơi tìm số và kết nối ít nhất là hai máy tính với nhau để chơi (trừ khi bạn có các tay cầm chơi game để có thể cùng nhau chơi trên 1 máy duy nhất) không chỉ dừng ở vấn đề giao diện, thao tác xử lí,… mà còn liên quan tới việc truyền tín hiệu từ máy này sang máy khác, xử lí tín hiệu cũng như đồng bộ chúng lẫn nhau sao cho độ trễ thấp nhất.

Trong quá trình thực hiện, đã không ít lần chương trình chạy không đúng ý của nhóm chúng em. Do đó, chúng em rất mong nhận được các ý kiến đóng góp của thầy cô để có thể đề tài của chúng em ngày càng hoàn thiện hơn.

Ngoài ra, chúng em cũng xin cảm ơn đến ThS. Trần Hồ Thủy Tiên đã có nhưng tư vấn, hỗ trợ chúng em trong quá trình thực hiện đồ án, đồng thời giải đáp những thắc mắc và giúp chúng em nhìn ra những điểm sai, chưa tốt cần cải thiện để chúng em có thể hoàn thành đồ án này.

Một lần nữa, xin cảm ơn quý thầy cô đã tận tình theo sát và hướng dẫn chúng em trong suốt quá trình thực hiện đồ án này.

CHƯƠNG 1. CƠ SỞ LÍ THUYẾT

1.1. Thế nào là mạng máy tính ?

Mạng máy tính là một hệ thống bao gồm hai hoặc nhiều máy tính hoặc thiết bị kết nối với nhau thông qua các phương tiện truyền thông (như dây cáp, sóng vô tuyến, hoặc tín hiệu quang học) để chia sẻ tài nguyên, dữ liệu, và thông tin.

1.2. Các thành phần chính của mạng

- Thiết bị đầu cuối : Máy tính, điện thoại, máy chủ
- Thiết bị mạng (Network Devices)
- Môi trường truyền dẫn (Transmission Medium)
- Giao thức mạng (Protocol)

1.3. Phân loại mạng máy tính

Theo phạm vi địa lí :

- LAN (Local Area Network) : Mạng cục bộ
- MAN (Metropolitan Area Network) : Mạng đô thị, thường dùng trong phạm vi một thành phố
- WAN (Wide Area Network) : Mạng diện rộng, kết nối từ khoảng cách lớn
- Internet : Mạng toàn cầu, kết nối các mạng nhỏ hơn

Theo kiến trúc, mô hình kết nối :

- Peer-to-Peer (P2P, mô hình ngang hàng) :
- Client – Server (mô hình khách – chủ) :

1.4. Mô hình MVC (Model – View – Controller)

Là một kiến trúc phần mềm phổ biến để tổ chức và xây dựng ứng dụng. Mô hình MVC phân chia ứng dụng thành 3 lớp chính :

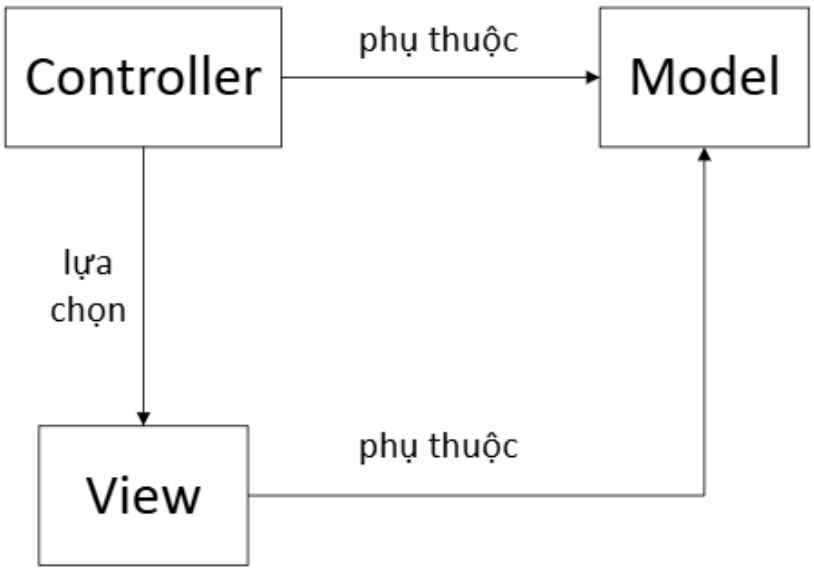
- Model :
  - Quản lí, lưu trữ, truy vấn dữ liệu.
  - Chịu trách nhiệm giao tiếp với cơ sở dữ liệu.
- View :
  - Giao diện của ứng dụng.

- Chịu trách nhiệm biểu diễn dữ liệu của ứng dụng thành các dạng nhìn thấy được.
- Không chứa logic xử lí dữ liệu.

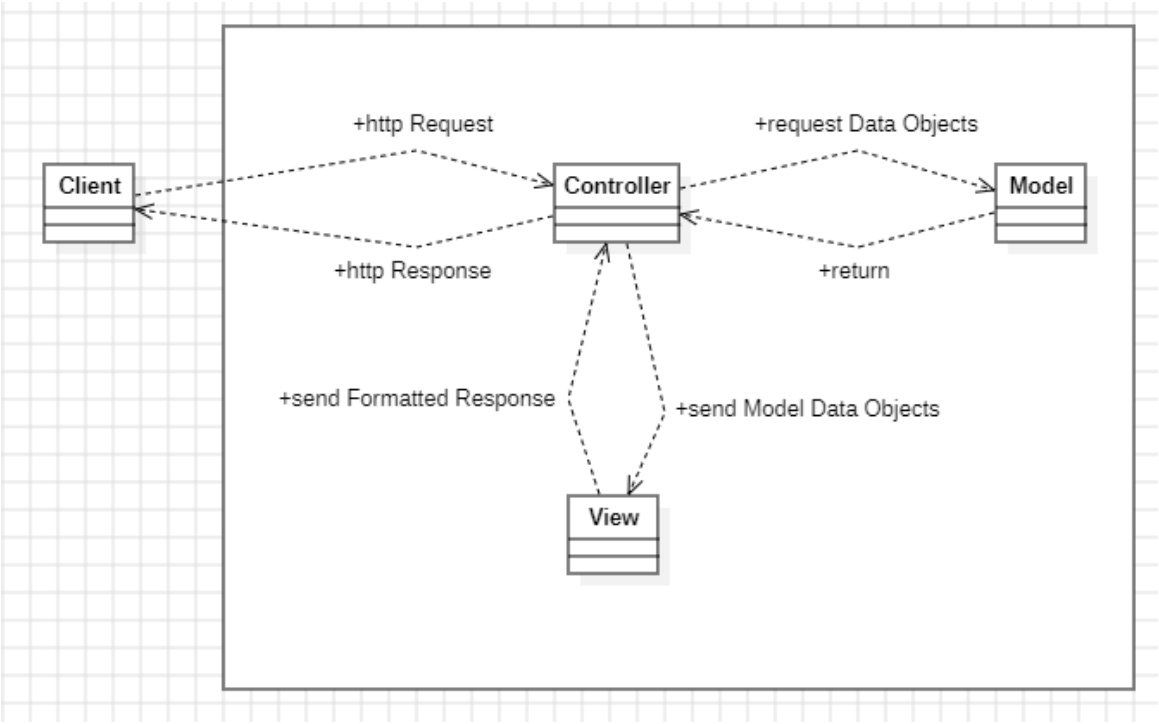
- Controller :
  - Quản lí và điều phối luồng hoạt động của ứng dụng.
  - Nhận request (yêu cầu) từ phía Client.
  - Điều phối các Model và View để có output thích hợp và trả kết quả về cho người dùng.

Mối quan hệ giữa Model, View và Controller :

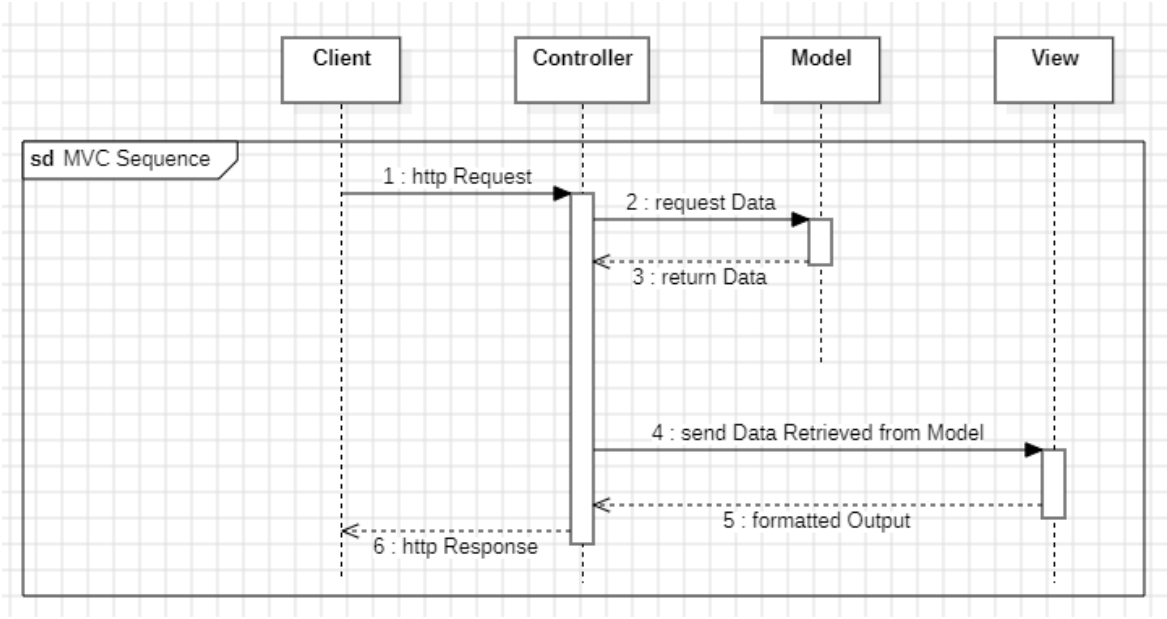
- Mối quan hệ giữa Model và View **: View <u>phụ thuộc</u> Model,** vì khi Model thay đổi về các tổ chức interface thì View phải thay đổi theo.
- Mối quan hệ giữa Model và Controller : Controller được thiết kế để kết nối với Model, điều khiển, truy xuất Model. Ngoài ra, khi Model có sự thay đổi thì bắt buộc Controller cũng phải có sự thay đổi theo, trong khi điều ngược lại không hoàn toàn xảy ra. Cho nên chúng ta kết luận : **Controller <u>phụ thuộc</u> vào Model**
- Mối quan hệ giữa View và Controller : **Controller là <u>đối tượng</u> lựa chọn View.**



Hình 1. Mối quan hệ giữa các thành phần trong mô hình MVC



Hình 2. Sơ đồ hoạt động thông thường của một mô hình MVC



Hình 3. Sơ đồ tuần tự của một mô hình MVC thông thường

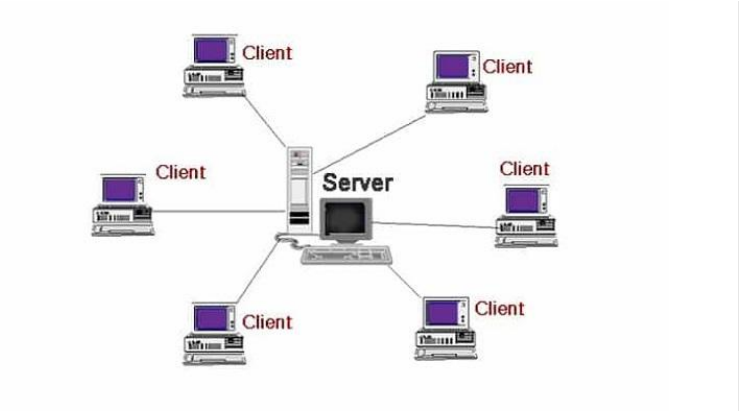
1.5. Xử lý đồng bộ (synchronized)

Đồng bộ hoá toàn bộ phương thức

Xử lý đồng bộ (Synchronization) trong Java được sử dụng để đảm bảo rằng các luồng (threads) không can thiệp lẫn nhau khi truy cập và thao tác trên các tài nguyên chia sẻ (shared resources). Java cung cấp nhiều cơ chế để thực hiện xử lý đồng bộ hóa

1.6. Mô hình Client-Server

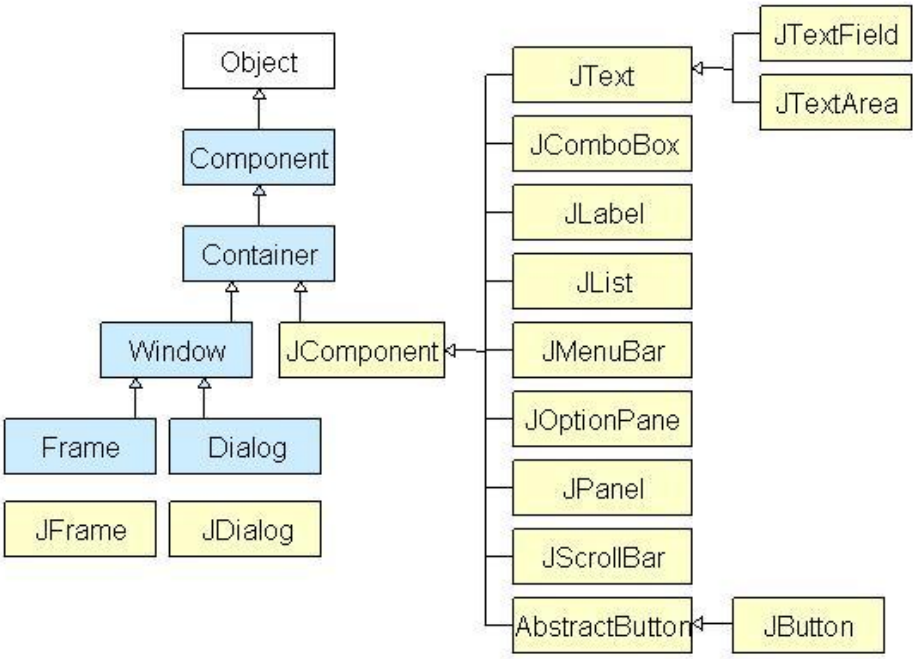
Client server là mô hình mạng máy tính gồm có 2 thành phần chính đó là máy khách (client) và máy chủ (server). Server chính là nơi giúp lưu trữ tài nguyên cũng như cài đặt các chương trình dịch vụ theo đúng như yêu cầu của client. Ngược lại, Client bao gồm máy tính cũng như các loại thiết bị điện tử nói chung sẽ tiến hành gửi yêu cầu đến server.



Hình 4. Mô hình Client-Server

1.7. Thư viện Java Swing

Thư viện Java Swing được xây dựng dựa trên Bộ công cụ tiện ích con trừu tượng Java (**AWT**), một bộ công cụ GUI phụ thuộc vào nền tảng cũ hơn. Bạn có thể sử dụng các thành phần lập trình GUI đơn giản của Java như nút, hộp văn bản, v.v., từ thư viện và không phải tạo các thành phần từ đầu.



Hình 5. Thư viện Java Swing

CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

2.1. Chức năng hệ thống

2.1.1. Server

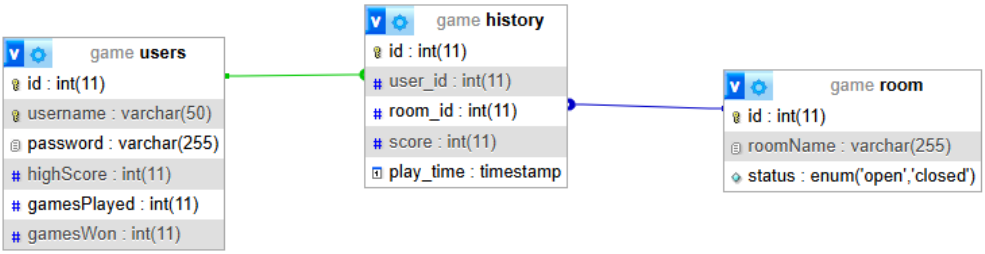
- Quản lí thông tin người chơi
- Điều phối người chơi vào các phòng chơi thích hợp
- Điều khiển các trận đấu

2.1.2. Client

- Đăng nhập
- Đăng kí
- Chơi trò chơi
- Xem các thông tin cá nhân
- Xem thông tin sản phẩm (trò chơi)

2.2. Cơ sở dữ liệu

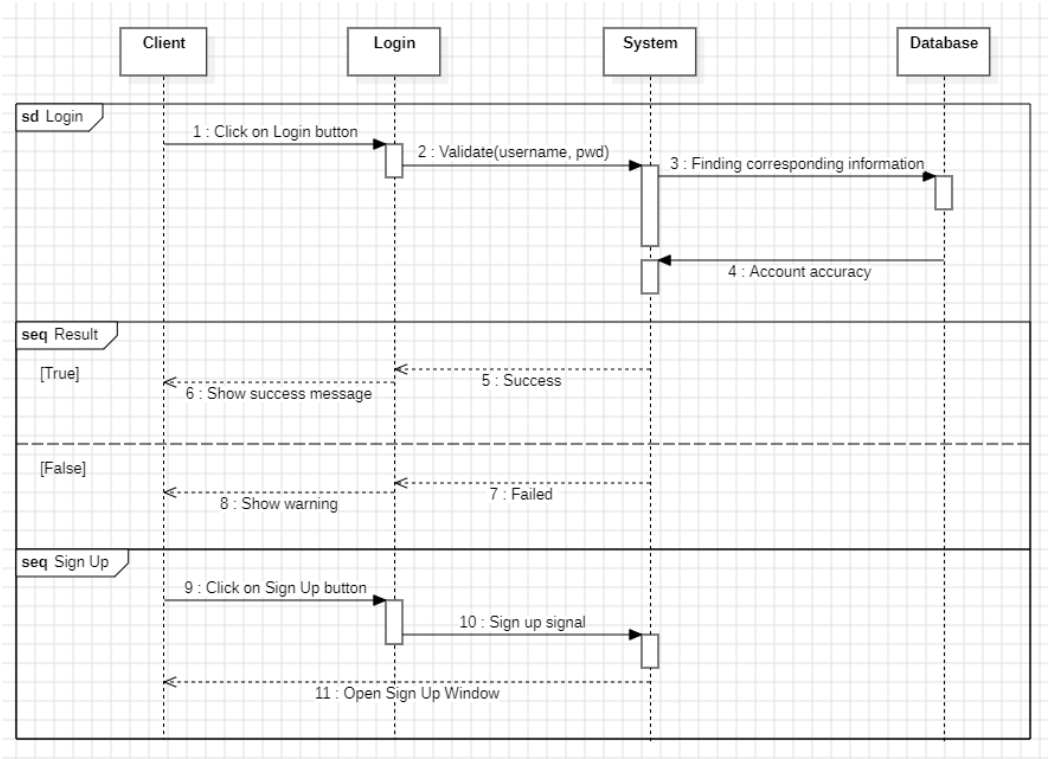
- Users (Players) : Bảng này gồm 5 column
  - Username : Tên đăng nhập của người dùng (người chơi)
  - Password : Mật khẩu người dùng (người chơi)
  - High Score : Điểm số cao nhất người chơi đạt được trong một trận
  - Games Played : Số trận đã chơi
  - Games Won : Số trận thắng
- Room : Bảng này gồm 2 column
  - Room Name : Thông thường hiển thị ID phòng để phân biệt các phòng chơi với nhau
  - Status : Trạng thái các phòng chơi (đang đợi, đang chơi,…)
- History : Bảng này gồm 4 column
  - User ID : Hiển thị đối thủ của trận
  - Room ID : ID phòng (trận đấu) đã chơi.
  - Score : Số điểm đạt được
  - Play Time : Thời gian trận đấu



Hình 6. Cơ sở dữ liệu

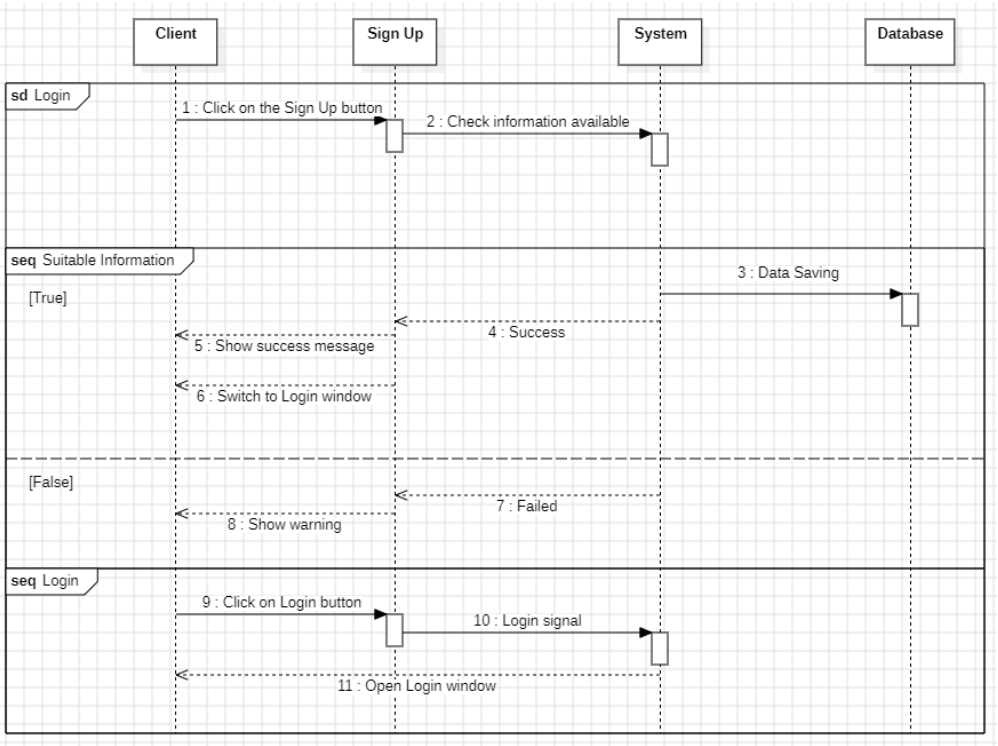
2.3. Biểu đồ tuần tự (Sequence Diagram)

2.3.1. Đăng nhập (Login)



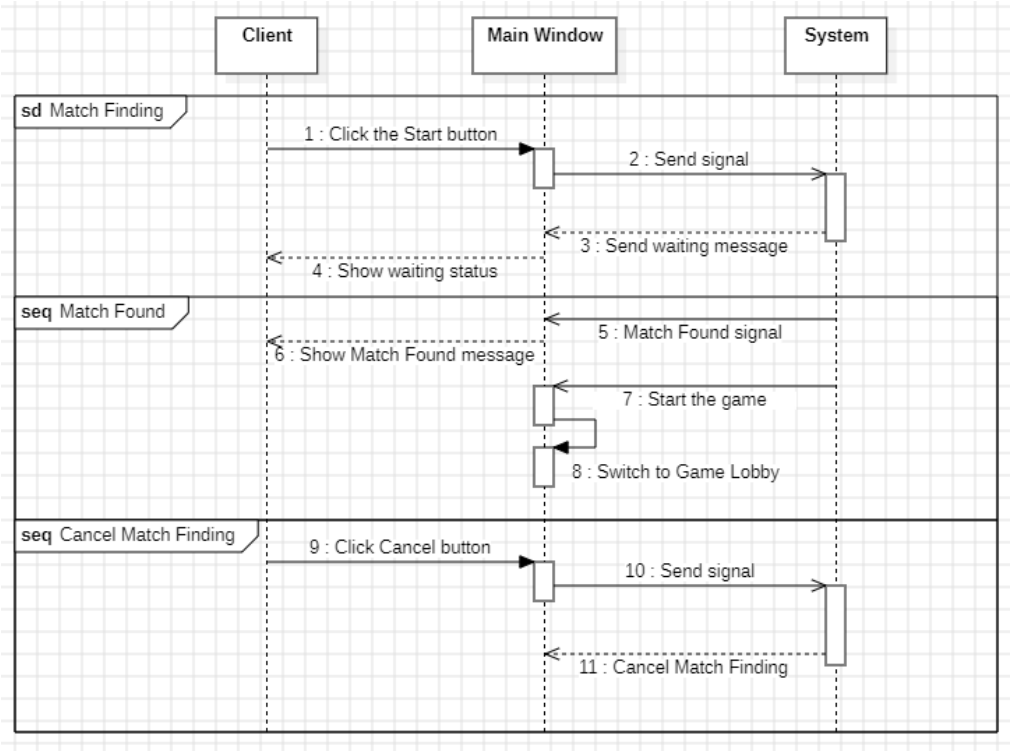
Hình 7. Đăng nhập

2.3.2. Đăng kí (Sign up)



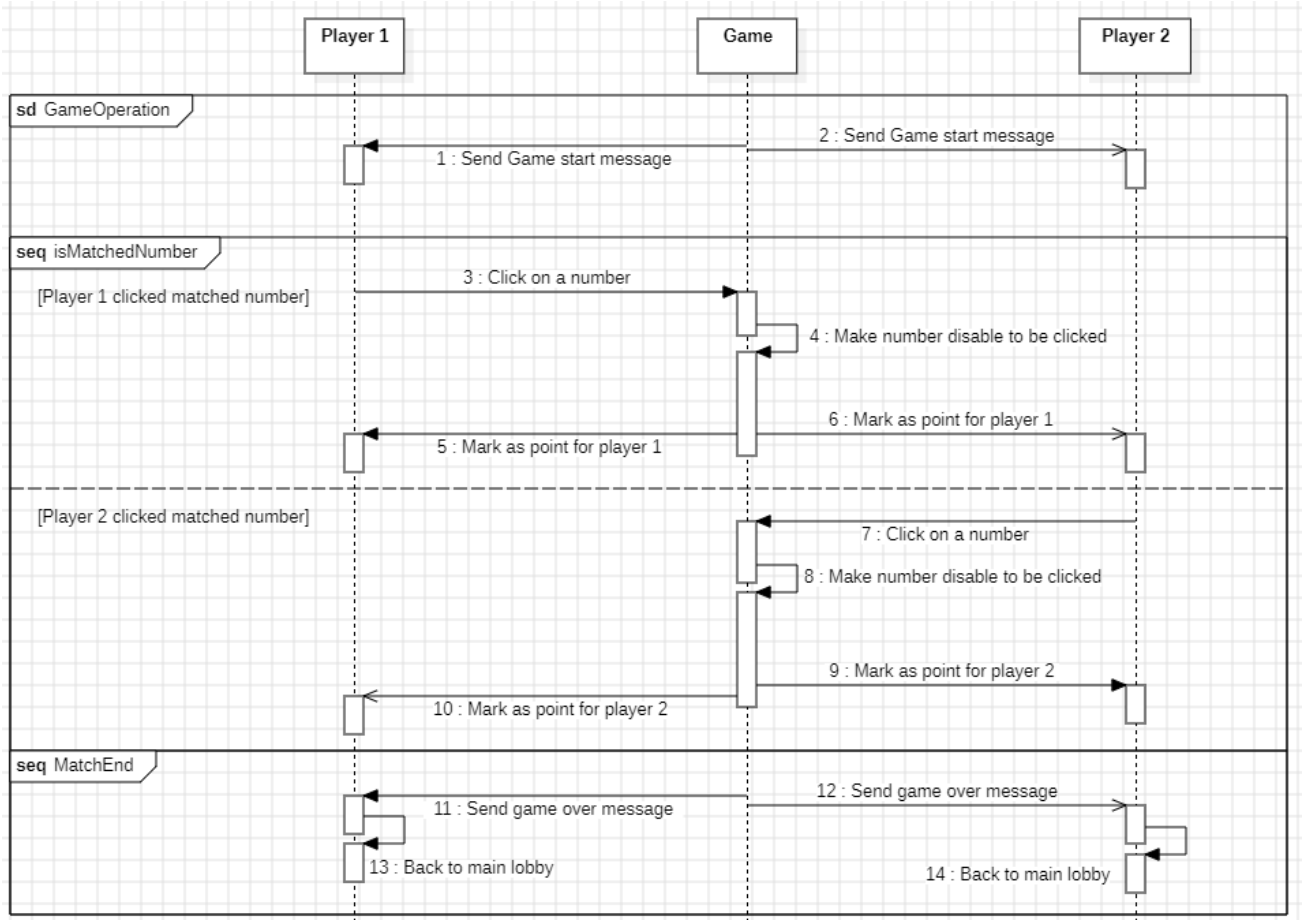
Hình 8. Đăng kí

2.3.3. Tìm trận và các trạng thái tìm trận



Hình 9. Tìm trận và các trạng thái tìm trận

2.4. Cách trò chơi hoạt động (Game Operation)



Hình 10. Sơ lược một ván trò chơi tìm số

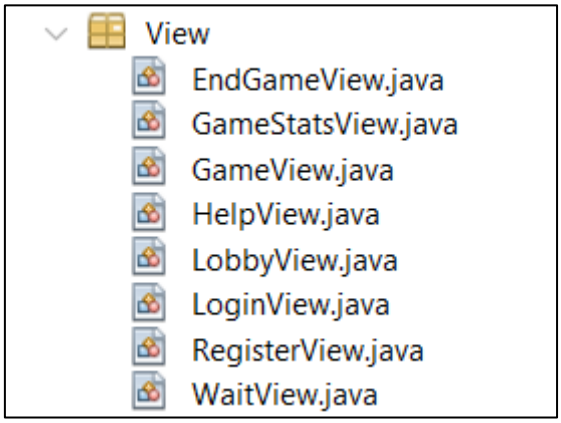
Trò chơi sẽ hiển thị một bảng số giống nhau cho cả 2 người chơi. Nhiệm vụ mỗi người là tìm các con số tăng dần cho đến hết bảng. Người chơi nào có số điểm nhiều hơn thì thắng trận.

Khi có một người chơi nhấn (click) vào một số bất kì, hệ thống sẽ kiểm tra xem số đó có hợp lệ hay không. Nếu đúng là số hợp lệ, số đó sẽ được đánh dấu là có điểm dành cho người chơi đó, đồng thời vô hiệu hóa (disable) số vừa nhần.

2.5. Xây dựng hệ thống:

2.5.1. Giao diện của hệ thống (Java – swing):

2.5.1.1. Cấu trúc thư mục của dự án:



Hình 11. Các thư mục lớp View

- View: Thư mục chứa các giao diện của chương trình
  - LoginView: Chứa các thành phần tạo nên giao diện đăng nhập hệ thống. LoginView sử dụng BorderLayout trong thư viện swing để tạo khung cho giao diện

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

public class LoginView extends JFrame {
    private ClientController clientController;

    // Khung của giao diện Login
    public LoginView(ClientController clientController) {...10 lines }

    // Tạo giao diện cho Login
    private void createUI() {...95 lines }

    // Thêm hiệu ứng hover cho nút
    private void addHoverEffect(JButton button, Color hoverColor) {...12 lines }
}
```

```
// Tạo giao diện cho Login
private void createUI() {
    setLayout(new BorderLayout());
```

  - RegisterView: Chứa các thành phần tạo nên giao diện đăng ký tài khoản.

- LobbyView: Chứa các thành phần tạo nên giao diện trang chủ thực hiện các chức năng hiển thị.
  LobbyView không sử dụng layout trong thư viện Swing mà dùng phương thức setBounds để xác định vị trí một cách thủ công.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

public class LobbyView extends JFrame {
    private final ClientController clientController;
    private final String playerName;

    // Khung của giao diện Lobby
    public LobbyView(ClientController clientController, String playerName) {...11 lines }

    // Tạo giao diện cho Lobby
    private void createUI() {...158 lines }

    // Hàm định dạng cho các nút chính (PLAY, JOIN ROOM, GAME STATS)
    private void styleMainButton(JButton button) {...18 lines }

    // Hàm định dạng cho các nút bên góc (HELP, LOGOUT)
    private void styleSideButton(JButton button) {...18 lines }
}
```

```
// Tạo giao diện cho Lobby
private void createUI() {
    setLayout(null); // Dùng layout null để định vị chính xác
    getContentPane().setBackground(new Color(240, 248, 255)); // Màu nền nhẹ nhàng
```

- WaitView: Chứa các thành phần tạo nên giao diện đợi chờ khi chuẩn bị bắt đầu trò chơi.
  WaiView sử dụng các layout của thư viện swing:
  - BorderLayout cho JFrame và mainPanel.
  - GridBagLayout cho centerPanel.
  - FlowLayout cho footerPanel.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

public class WaitView extends JFrame {
    private final ClientController clientController;
    private final String playerName;

    public WaitView(ClientController clientController, String playerName) {...6 lines }

    private void listenToServer() {...42 lines }

    private void initializeUI() {...60 lines }

    private void styleMainButton(JButton button) {...20 lines }
}
```

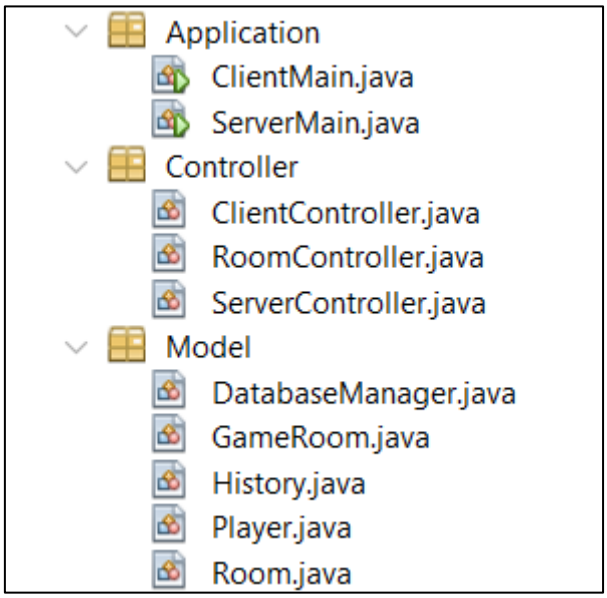
- GameView: Chứa các thành phần tạo nên giao diện chơi trò chơi.
  GameView sử dụng Layout của thư viện swing:
  - BorderLayout cho JFrame và mainPanel
  - GridLayout cho boardPanel
  - BoxLayout cho infoPanel
  - FlowLayout cho buttonPanel

```
import javax.swing.*;
import java.awt.*;
import java.io.PrintWriter;

public class GameView extends JFrame {
    private ClientController clientController;
    private int[] board; // Bàn cờ nhận từ server
    private JButton[] buttons; // Các nút đại diện cho số trên bàn cờ
    private JTextArea gameLog;
    private String playerName;

    private JLabel player1ScoreLabel;
    private JLabel player2ScoreLabel;
    private PrintWriter out;
    private JLabel timeLabel;
    private Timer timer;
    private int elapsedTime;

    public GameView(ClientController clientController, String playerName) {...13 lines }

    private void createUI() {...102 lines }

    private void listenToServer() {...81 lines }

    public void initializeBoard(String boardData) {...15 lines }

    private void updateBoard(String moveData) {...23 lines }

    private void updateScores(String scoreData) {...10 lines }

    private void appendToLog(String message) {...5 lines }

    private void startTimer() {...10 lines }

    private void handleInvalidMove(String message) {...6 lines }
}
```

- HelpView: Chứa các thành phần tạo nên giao diện hỗ trợ thông tin về trò chơi dành cho người chơi.

2.5.2. Model – Controller:

2.5.2.1. Cấu trúc thư mục của dự án:



Hình 12. Các thư mục lớp Model, Controller cũng như phần Application

2.5.2.1.1. Application:

- Application: Thư mục chứa các lớp thực hiện kết nối client – server (TCP/IP).
  - Server tạo cổng để các nhiều client có thể kết nối đến

```java
public class ServerMain {
    private static final int PORT = 12345; // Cổng để client kết nối

    public static void main(String[] args) {
        DatabaseManager databaseManager = new DatabaseManager();  // Khởi tạo cơ sở dữ liệu người chơi
        ServerController serverController = new ServerController(databaseManager);

        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Server is running and waiting for clients to connect...");

            while (true) {
                Socket clientSocket = serverSocket.accept();  // Chấp nhận kết nối từ client
                System.out.println("Client connected: " + clientSocket.getInetAddress());

                // Xử lý client trong một luồng riêng biệt
                new Thread(() -> serverController.handleClient(clientSocket)).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Hình 13. Class ServerMain

  - Client thực hiện kết nối thông qua địa chỉ IP

```java
public class ClientMain {

    private static final String SERVER_ADDRESS = "localhost"; // Địa chỉ server
    private static final int SERVER_PORT = 12345; // Cổng kết nối

    public static void main(String[] args) {
        try {
            // Kết nối đến server
            Socket socket = new Socket(SERVER_ADDRESS, SERVER_PORT);
            System.out.println("Connected to server.");

            // Khởi tạo ClientController để xử lý giao tiếp
            ClientController clientController = new ClientController(socket);

            // Hiển thị màn hình đăng nhập
            LoginView loginView = new LoginView(clientController);
            loginView.setVisible(true);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Hình 14. Class ClientMain

- Mô hình Client-Server sử dụng Socket ở chế độ có kết nối (TCP)

  Các bước thực hiện kết nối:

  - B1: Mở một socket nối kết đến server đã biết địa chỉ IP và số hiệu cổng
  - B2: Lấy Inputstream và Outputstream gán với Socket
  - B3: Trao đổi dữu liệu với Server nhờ vào các InputStream và OutputStream
  - B4: Đóng Socket trước khi kết thúc chương trình

2.5.2.1.2. Controller:

- Thư mục chứa các controller nhận các request từ giao diện và thực hiện các chức năng.
- ClientController:

```
public class ClientController {
    private Socket socket;
    private BufferedReader in;
    private PrintWriter out;

    public ClientController(Socket socket) throws IOException {...15 lines }

    // Gửi thông tin đăng nhập
    public void sendLogin(String username, String password) {...3 lines }

    public void sendCheckRoom(String username, String idRoom, String nameRoom) {...3 lines }

    // Gửi yêu cầu phòng
    public void sendRoom(String username,String roomName) {...3 lines }

    public void sendLeaveRoom(String username, int roomId) {...3 lines }

    public void sendMove(String playerName, int number) {...3 lines }

    public void sendStartGame(String username, int roomId) {...3 lines }

    public void sendExitGame(String username) {...3 lines }

    // Đăng xuất
    public void sendLogout(String username) {...3 lines }

    public void sendRegister(String username, String password) {...3 lines }

    // Gửi tin nhắn
    private void sendMessage(String message) {...3 lines }

    // Lắng nghe phản hồi từ server
    public String listen() {...10 lines }

    // Đóng kết nối
    public void closeConnection() throws IOException {...5 lines }
}
```

```
public ClientController(Socket socket) throws IOException {
    try {
        this.socket = socket;
        this.in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        this.out = new PrintWriter(socket.getOutputStream(), true);

        if (socket != null && socket.isConnected()) {
            System.out.println("Connection established with server.");
        } else {
            System.out.println("Failed to connect to server.");
        }
    } catch (IOException e) {
        System.err.println("Error connecting to server: " + e.getMessage());
    }
}
```

```java
// Gửi tin nhắn
private void sendMessage(String message) {
    out.println(message);
}

// Lắng nghe phản hồi từ server
public String listen() {
    try {
        String response = in.readLine();
        System.out.println("Received from server: " + response);
        return response;
    } catch (IOException e) {
        System.out.println("Error while listening from server: " + e.getMessage());
        return null;
    }
}
```

```java
// Đóng kết nối
public void closeConnection() throws IOException {
    if (socket != null && !socket.isClosed()) {
        socket.close();
    }
}
```

Hình 15. Class ClientController

ClientController nhận socket để thực hiện các phương thức:
+ sendMessage: Gửi thông điệp nhận được từ giao diện đến server,
+ listen: Nhận thông điệp từ server và hiển thị giao diện
+ closeConnection: đóng kết nối socket khi dừng chương trình

- ServerController:

```java
public void handleClient(Socket clientSocket) {
    try {
        BufferedReader reader = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
        PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);

        String request;
        while ((request = reader.readLine()) != null) {
            processRequest(request, writer);
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            clientSocket.close();
        } catch (IOException e) {
            System.err.println("Error closing client socket: " + e.getMessage());
        }
    }
}
```

```java
private void processRequest(String request, PrintWriter writer){
    System.out.println("Received request: " + request);
    if (request.startsWith("CHECK_USERNAME")) {
        handleCheckUsername(request, writer);
    } else if (request.startsWith("REGISTER")) {
        handleRegister(request, writer);
    } else if (request.startsWith("LOGIN")) {
        handleLogin(request, writer);
    } else if (request.startsWith("LOGOUT")) {
        handleLogout(request, writer);
    } else if (request.startsWith("GET_ROOMS")) {
        handleGetRooms(writer);
    } else if (request.startsWith("JOIN_ROOM")) {
        handleJoinRoom(request, writer);
    } else if (request.startsWith("CHECK_ROOM")) {
        handleCheckRoom(request, writer);
    } else if (request.startsWith("START_GAME")) {
        handleStartGame(request, writer);
    } else if (request.startsWith("LEAVE_ROOM")) {
        handleLeaveRoom(request, writer);
    } else if (request.startsWith("SAVE_HISTORY")) {
        handleSaveHistory(request, writer);
    } else if (request.startsWith("GET_HISTORY")) {
        handleGetHistory(request, writer);
    } else if (request.startsWith("MOVE")){
        handleMove(request, writer);
    } else if (request.startsWith("EXIT_GAME")){
        handleExitGame(request, writer);
    } else {
        writer.println("UNKNOWN_COMMAND");
        System.out.println("Unknown command received: " + request);
    }
}
```

```
// ---------- Các phương thức xử lý người dùng ----------
private void handleCheckUsername(String request, PrintWriter writer) {...14 lines }

private void handleRegister(String request, PrintWriter writer) {...21 lines }

private void handleLogin(String request, PrintWriter writer) {...24 lines }

private void handleLogout(String request, PrintWriter writer) {...16 lines }

// ---------- Các phương thức xử lý phòng (Room) ----------
private void handleGetRooms(PrintWriter writer) {...5 lines }

private void handleJoinRoom(String request, PrintWriter writer) {...30 lines }

private void handleCheckRoom(String request, PrintWriter writer){...33 lines }

private void handleStartGame(String request, PrintWriter writer) {...35 lines }

private void handleLeaveRoom(String request, PrintWriter writer) {...33 lines }

// ---------- Các phương thức xử lý lịch sử chơi (History) ----------
private void handleSaveHistory(String request, PrintWriter writer) {...25 lines }

private void handleGetHistory(String request, PrintWriter writer) {...24 lines }

private void handleMove(String request, PrintWriter writer) {...45 lines }

private void handleExitGame(String request, PrintWriter writer) {...36 lines }
```

Hình 16. Class ServerController

ServerController nhận socket để thực hiện các chức năng:
+ handleClient: nhận socket từ ServerMain, khi dừng hoạt động sẽ tự động đóng kết nối socket.
+ processRequest: nhận thông điệp được lấy thông qua socket sau đó thực hiện các handle kèm theo writer để có thể gửi thông điệp lại cho client.
+ Các phương thức xử lý dành cho người dùng
+ Các phương thức xử lý phòng
+ Các phương thức xử lý lịch sử chơi

- RoomController:

```java
public class RoomController{
    private List<GameRoom> rooms;

    public RoomController() {...3 lines }

    public synchronized GameRoom getOrCreateRoom() {...13 lines }

    public synchronized GameRoom getOrCreateRoom(int idRoom, String nameRoom) {...10 lines }

    public synchronized List<GameRoom> getAllRooms() {...3 lines }

    public synchronized void logRoomsState() {...8 lines }

    public synchronized GameRoom findRoomByPlayer(String username) {...10 lines }

    public synchronized void removeEmptyRooms() {...3 lines }

    public synchronized void removeRoom(GameRoom room) {...3 lines }
}
```

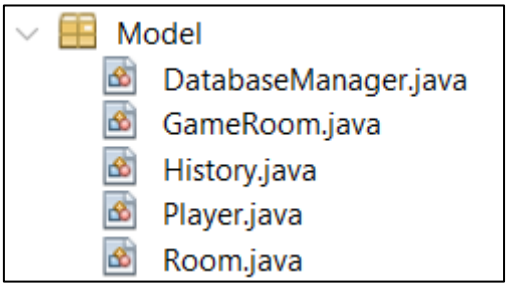
```java
public synchronized GameRoom getOrCreateRoom() {
    for (GameRoom room : rooms) {
        if (!room.isActive() && !room.isFull()) {
            System.out.println("Assigning player to existing Room ID " + room.getRoomId());
            return room;
        }
    }
    // Nếu không có phòng, tạo phòng mới
    GameRoom newRoom = new GameRoom();
    rooms.add(newRoom);
    System.out.println("Created new Room ID " + newRoom.getRoomId());
    return newRoom;
}

public synchronized GameRoom getOrCreateRoom(int idRoom, String nameRoom){
    for (GameRoom room : rooms) {
        if (room.getRoomId() == idRoom && room.getNameRoom().equals(nameRoom)) {
            return room;
        }
    }
    GameRoom newRoom = new GameRoom(idRoom, nameRoom);
    rooms.add(newRoom);
    return newRoom;
}
```

Hình 17. Class RoomController

RoomController: thực hiện các yêu cầu tạo phòng, tìm phòng từ ServerController
+ sử dụng synchronized để thực hiện đồng bộ các tính chất của phòng như điểm
+ sử dụng overloading (nạp chồng): để thực hiện tạo phòng cho người chơi bất kỳ và người chơi có tên phòng và mã phòng
+ các chức năng còn lại để dọn dẹp và xoá phòng đã chơi.

2.5.2.1.3. Model

Cấu trúc file của model



Hình 18. Cấu trúc file lớp Model

- DatabaseManager:

```java
public class DatabaseManager {
    private static final String DB_URL = "jdbc:mysql://localhost:4306/game";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "";

    // Kết nối đến cơ sở dữ liệu
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
    }
```

```java
// ---------- Phương thức cho bảng Player ----------
// Thêm người chơi mới vào cơ sở dữ liệu
public boolean addPlayer(Player player) {...15 lines }

public boolean updatePlayerStats(String playerName, int playerScore, boolean isWinner) {...15 lines }

public Player getUserByUsername(String username) {...15 lines }

public List<Player> getAllPlayers() {...13 lines }

public boolean usernameExists(String username) {...13 lines }

public boolean registerUser(String username, String password) {...12 lines }
```

```java
// ---------- Phương thức cho bảng Room ----------
public boolean addRoom(int idRoom, String nameRoom) {...13 lines }

public boolean updateRoom(Room room) {...12 lines }

public Room getRoomStatus(int idRoom, String nameRoom) {...18 lines }

public List<Room> getAllRooms() {...16 lines }

public boolean deleteRoom(int id) {...10 lines }
```

```java
// ---------- Phương thức cho bảng History ----------
public boolean addHistory(History history) {...13 lines }

public List<History> getHistoryByUserId(int userId) {...20 lines }

public List<History> getHistoryByRoomId(int roomId) {...20 lines }

public boolean deleteHistoryById(int id) {...10 lines }
```

Hình 19. Class DatabaseManager

DatabaseManager: Chứa các phương thức sau:

Cơ sở dữ liệu MySQL: MySQL được triển khai trên XAMPP, cấu hình kết nối cơ sở dữ liệu trong ứng dụng XAMPP thông qua localhost/phpMyAdmin.
+ getConnection: kết nối với cơ sở dữ liệu để các phương thức còn lại truy vấn đến cơ sở dữ liệu
+ truy vấn đến bảng user
+ truy vấn đến bảng room
+ truy vấn đến bảng history

Room, Player, History:
+ Các lớp này để đại diện cho dữ liệu của các bảng trong cơ sở dữ liệu
+ Chứa các phương thức getter, setter để truy cập và cập nhật giá trị của thuộc tính

```
public class History {
    private int id;
    private int userId;  // Liên kết tới Player (bảng users)
    private int roomId;  // Liên kết tới Room
    private int score;
    private String result;
    private Timestamp playTime;

    // Constructor
    public History(int id, int userId, int roomId, int score, Timestamp playTime) {...7 lines }

    public History(int userId, int roomId, int score, Timestamp playTime) {...6 lines }

    // Getters và Setters
    public int getId() {...3 lines }

    public void setId(int id) {...3 lines }

    public int getUserId() {...3 lines }

    public void setUserId(int userId) {...3 lines }

    public int getRoomId() {...3 lines }

    public void setRoomId(int roomId) {...3 lines }

    public int getScore() {...3 lines }

    public void setScore(int score) {...3 lines }

    public Timestamp getPlayTime() {...3 lines }

    public void setPlayTime(Timestamp playTime) {...3 lines }

    @Override
    public String toString() {...9 lines }
}
```

Hình 20. Class History

```java
public class Player {
    private int id;
    private String username;
    private String password;
    private int highScore;
    private int gamesPlayed;
    private int gamesWon;

    // Danh sách các số người chơi đã chọn
    private Set<Integer> selectedNumbers;

    // Constructor
    public Player(int id, String username, String password, int highScore, int gamesPlayed, int gamesWon) {...9 lines }

    // Constructor không có ID (Dùng khi thêm người chơi mới)
    public Player(String username, String password) {...8 lines }
```

```java
// Getter and Setter
public int getId() {...3 lines }

public void setId(int id) {...3 lines }

public String getUsername() {...3 lines }

public void setUsername(String username) {...3 lines }

public String getPassword() {...3 lines }

public void setPassword(String password) {...3 lines }

public int getHighScore() {...3 lines }

public void setHighScore(int highScore) {...3 lines }

public int getGamesPlayed() {...3 lines }

public void setGamesPlayed(int gamesPlayed) {...3 lines }

public int getGamesWon() {...3 lines }

public void setGamesWon(int gamesWon) {...3 lines }

// Phương thức chọn số
public void selectNumber(int number) {...3 lines }

// Phương thức lấy số lượng số đã chọn
public int getSelectedNumbersCount() {...3 lines }

// Phương thức reset lại các số đã chọn khi bắt đầu trò chơi mới
public void resetSelectedNumbers() {...3 lines }

@Override
public String toString() {...9 lines }
```

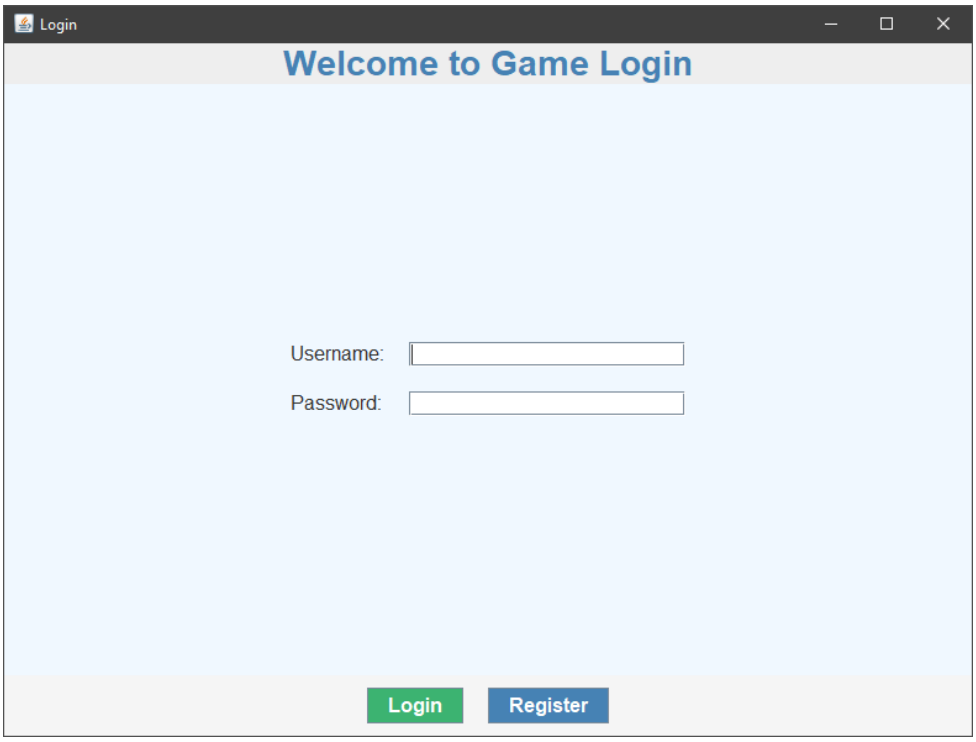
Hình 21. Class Player

```java
public class Room {
    private int idRoom;
    private String nameRoom;
    private String statusRoom; // "open" hoặc "close"
    private int playerRoom;

    // Constructor
    public Room(int idRoom, String nameRoom, String statusRoom) {...5 lines }

    // Getters và Setters
    public int getIdRoom() {...3 lines }

    public void setIdRoom(int idRoom) {...3 lines }

    public String getNameRoom() {...3 lines }

    public void setNameRoom(String nameRoom) {...3 lines }

    public String getStatusRoom() {...3 lines }

    public void setStatusRoom(String statusRoom) {...3 lines }

    @Override
    public String toString() {...7 lines }
}
```

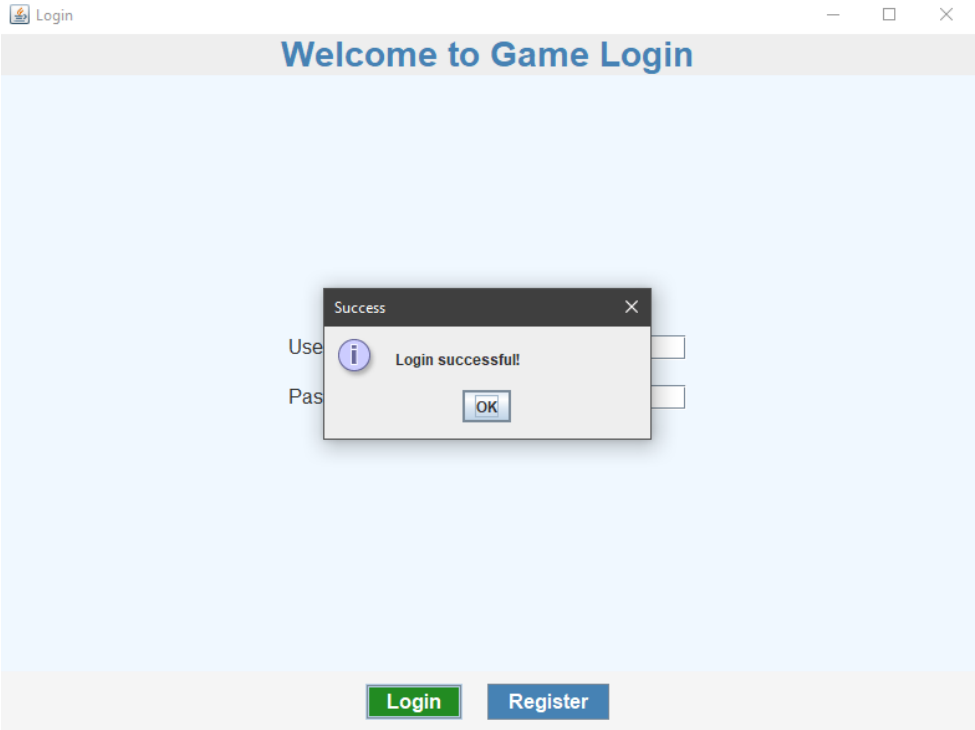
Hình 22. Class Room

CHƯƠNG 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

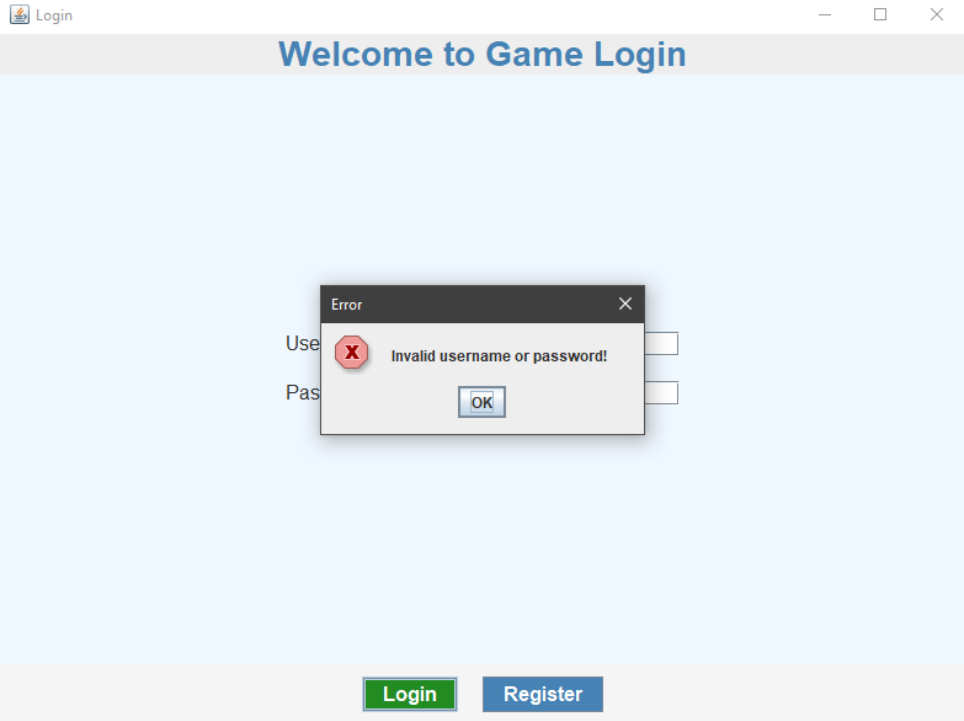
3.1. Cửa sổ đăng nhập



Hình 23. Màn hình đăng nhập



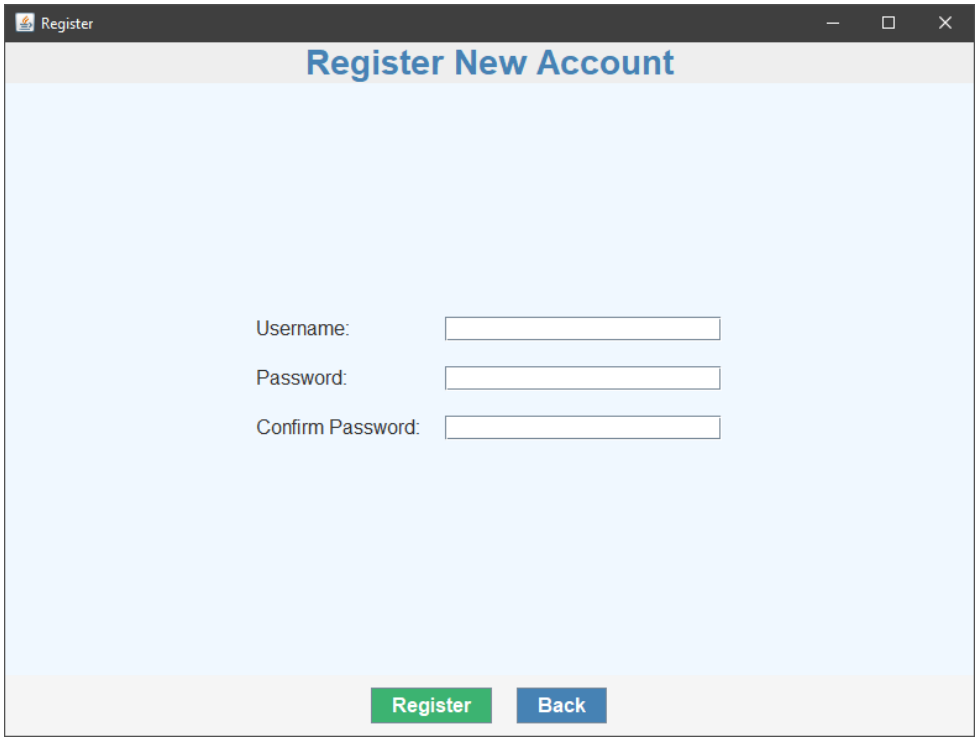
Hình 24. Đăng nhập thành công



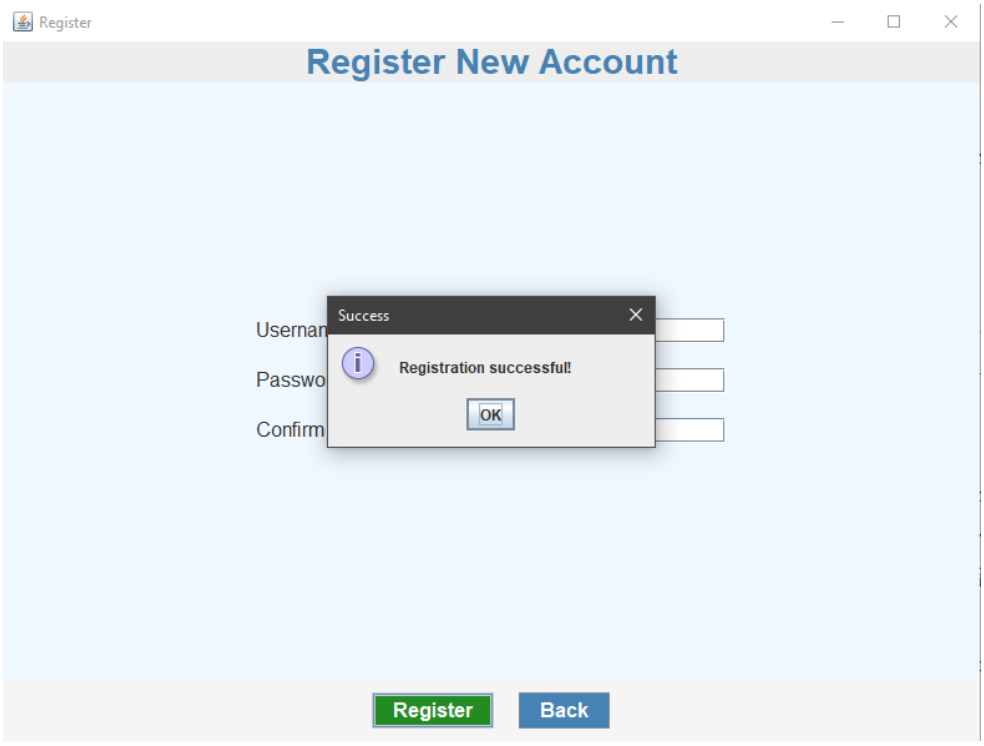
Hình 25. Đăng nhập thất bại

Giao diện	Trang đăng nhập		
Mô tả	Giúp người dùng mau chóng đăng nhập vào chương trình.		
Truy cập	Chạy chương trình		
Các hành động trong giao diện			
Hành động	**Mô tả**	**Thành công**	**Lỗi**
Đăng nhập.	Khi người dùng nhấn nút "Login", hệ thống sẽ kiểm tra Tài khoản với mật khẩu tương ứng có tồn tại trong hệ thống hay không, nếu có thì cho đăng nhập vào hệ thống, nếu không thì sẽ để đăng nhập lại.	Vào màn hình chính của chương trình và hiện thông báo.	Hiện thông báo.

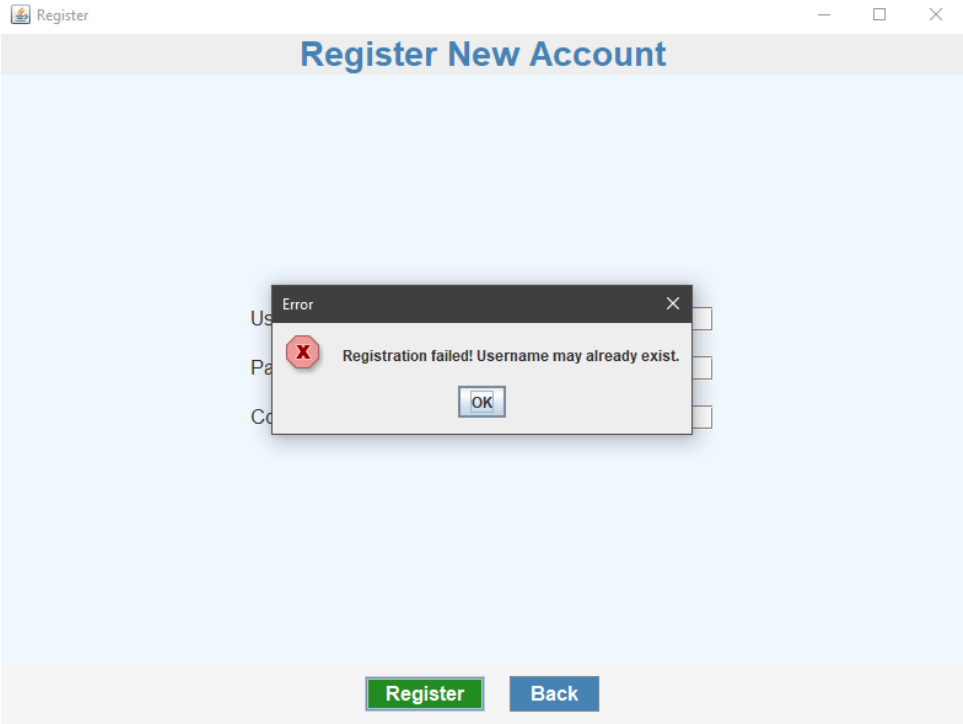
3.2. Cửa sổ đăng kí



Hình 26. Cửa sổ đăng kí



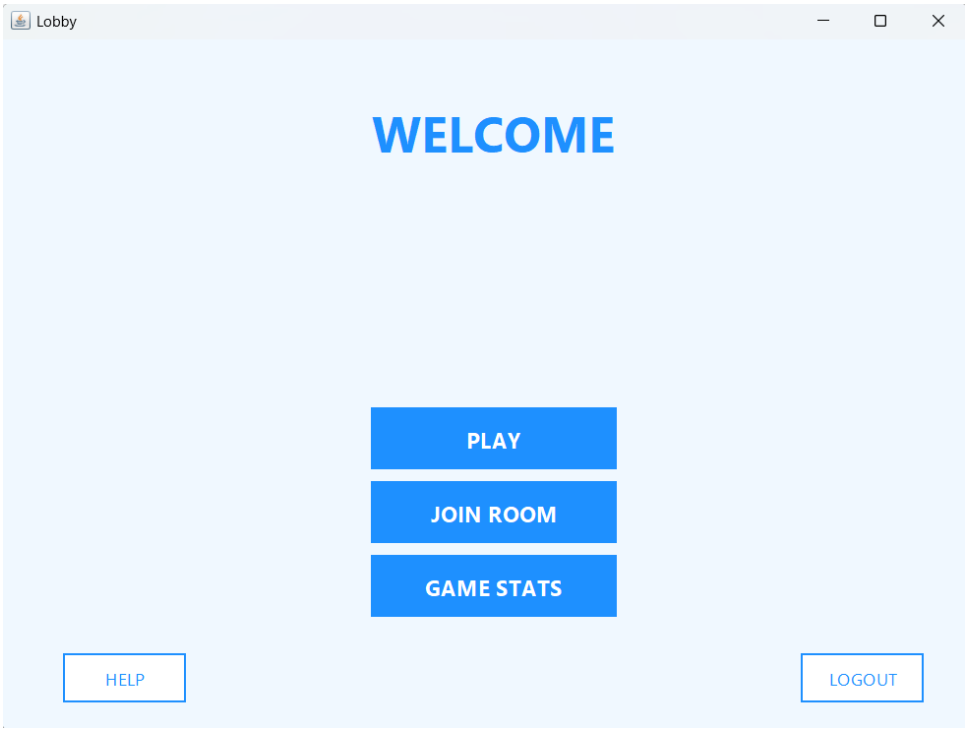
Hình 27. Đăng kí thành công



Hình 28. Đăng kí thất bại

Giao diện	Trang đăng ký		
Mô tả	Giúp người dùng mau chóng đăng ký tài khoản.		
Truy cập	Nhấn "Register" tại trang đăng nhập		
Các hành động trong giao diện			
Hành động	**Mô tả**	**Thành công**	**Lỗi**
Đăng nhập.	Khi người dùng nhấn nút "Register", hệ thống sẽ hiển thị trang đăng ký Tài khoản với ba dòng, "Username", "Password", "Confirm Password" để người dùng có thể nhập và đăng ký tài khoản mới.	Thông báo đăng ký thành công, quay lại trang đăng nhập để đăng ký.	Hiện thông báo.

3.3. Giao diện trang chủ

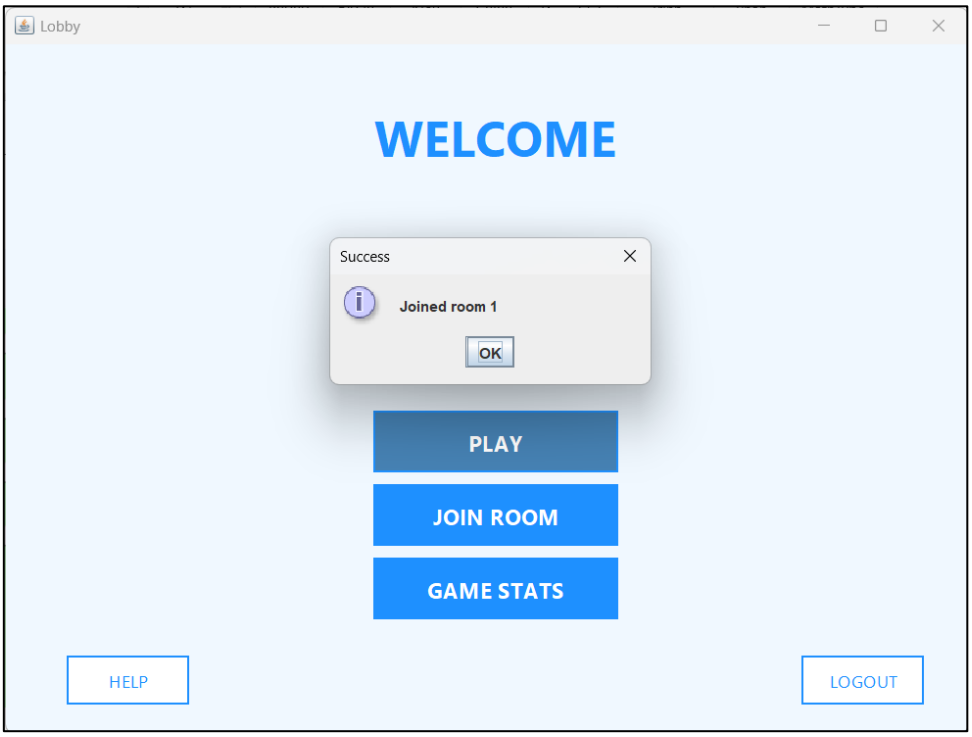


Hình 29. Cửa sổ giao diện

Giao diện	Trang chủ		
Mô tả	Giúp người dùng có cái nhìn tổng quan và thấy được các chức năng trong chương trình		
Truy cập	Sau khi đăng nhập thành công		
Các hành động trong giao diện			
Hành động	**Mô tả**	**Thành công**	**Lỗi**
Nhấn nút "PLAY"	Khi người dùng nhấn vào "PLAY", người dùng có thể vào phòng chơi bất kỳ và đợi cho đến khi đủ người, nếu chưa đủ người dùng sẽ thấy giao diện chờ đợi, còn nếu đủ người dùng sẽ vào giao diện trò chơi	Vào giao diện trò chơi	
Nhấn nút "JOIN ROOM"	Khi người dùng nhấn nút "JOIN ROOM", người dùng phải nhập idRoom và nameRoom để hệ thống xác nhận. Nếu đã có thì vào	Vào giao diện trò chơi	

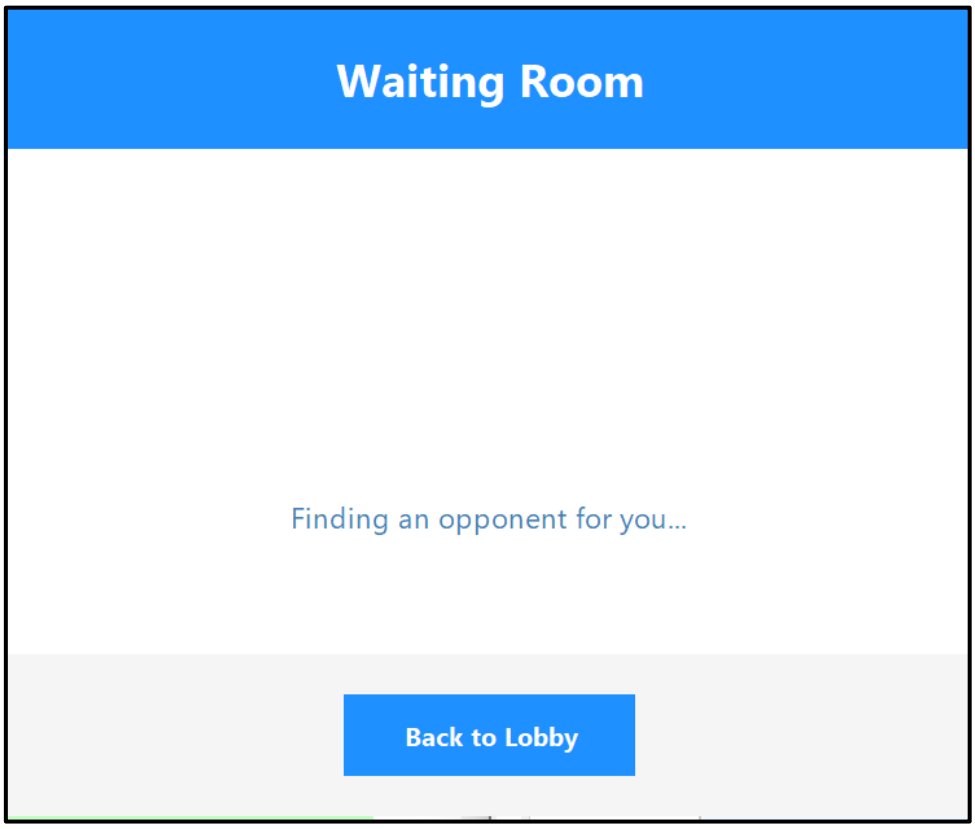
	phòng và chờ cho đến khi đủ người. Nếu không thì người nhận sẽ được tạo phòng mới và chờ đợi. Nếu phòng đã đủ thì sẽ nhận thông báo phòng đầy		
Nhấn nút "GAME STATS"	Khi người dùng nhấn nút "GAME STATS", hiển thị thông tin cá nhân và lịch sử chơi của người chơi.	Vào giao diện StatusView	
Nhấn nút "HELP"	Khi người dùng nhấn nút "HELP", hiển thị thông tin hỗ trợ về luật chơi và cách chơi.	Vào giao diện HelpView	
Nhấn nút "LOGOUT"	Khi người dùng nhấn nút "LOGOUT" hiển thị trang muốn đăng xuất hay không?	Trở về giao diện đăng nhập	

3.4. Giao diện sau khi nhấn nút PLAY



Hình 30. Vào phòng chơi

3.5. Giao diện phòng đợi



Hình 31. Đợi trong lúc tìm trận

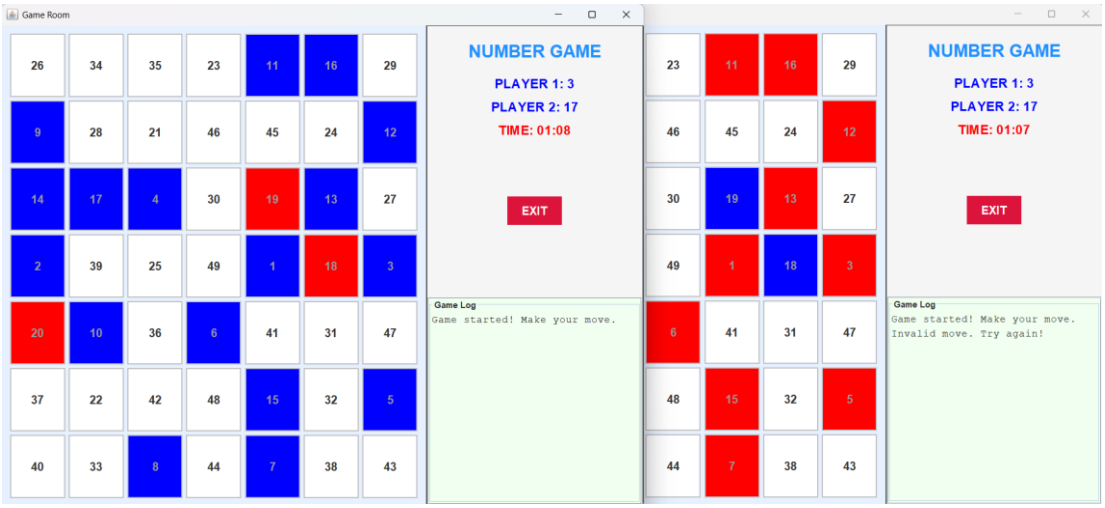
Giao diện	Trang WAITVIEW		
Mô tả	Giúp người dùng đợi cho đến khi đủ người chơi		
Truy cập	Nhấn "PLAY hoặc JOIN ROOM" tại trang chủ		
Các hành động trong giao diện			
Hành động	**Mô tả**	**Thành công**	**Lỗi**
Nhấn nút "PLAY", "JOIN ROOM"	Khi người dùng nhấn nút "PLAY", hệ thống sẽ hiển thị trang đợi cho đến khi đủ người.	Dẫn đến giao diện đợi "WAITING ROOM"	
Nhấn nút "Back to Lobby"	Khi người dùng nhấn nút "Back to Lobby", hệ thống sẽ huỷ phòng và dừng chờ đợi.	Dẫn đến giao diện trang chủ	

3.6. Giao diện chơi trò chơi

Hình 32. Giao diện một ván đấu

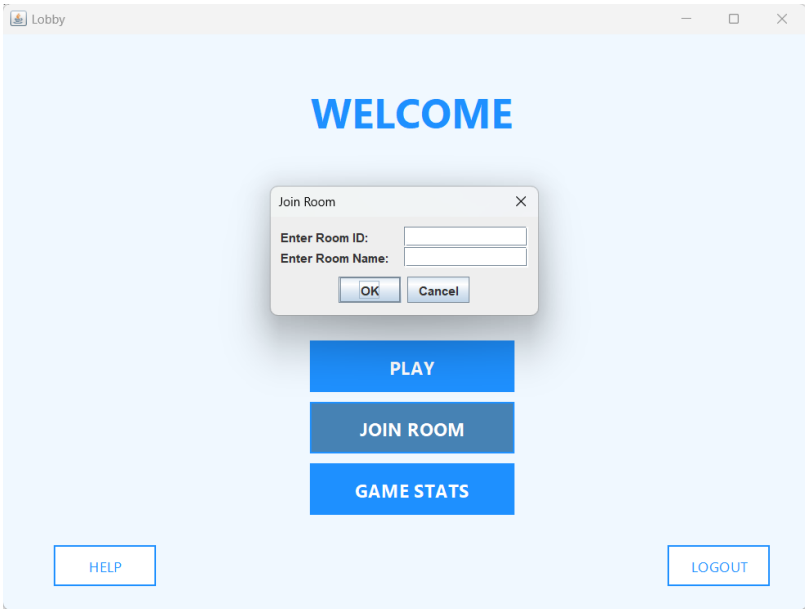
Giao diện	Trang GAMEVIEW		
Mô tả	Giúp người dùng vào trang chơi trò chơi		
Truy cập	Nhấn "PLAY hoặc JOIN ROOM" tại trang chủ		
Các hành động trong giao diện			
Hành động	**Mô tả**	**Thành công**	**Lỗi**
Nhấn các số	Khi người dùng chọn các nút chứa số thì sẽ khiến cho nút đổi màu nếu chọn đúng số	Đổi màu nút	
Nhấn nút "EXIT"	Khi người dùng nhấn nút "EXIT" sẽ hiển thị câu hỏi bạn có muốn thoát không.	Dẫn đến giao diện trang chủ	

3.7. Giao diện khi hai người đang chơi:



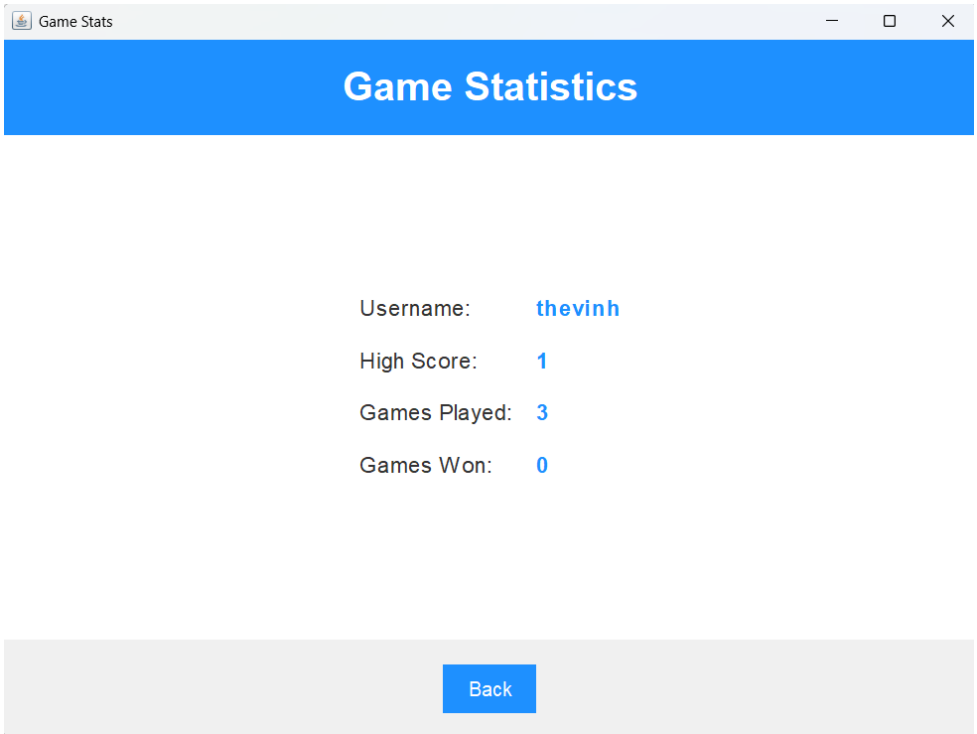
Hình 33. Giao diện của 2 client chơi với nhau

3.8. Giao diện sau khi chọn nút JOIN ROOM



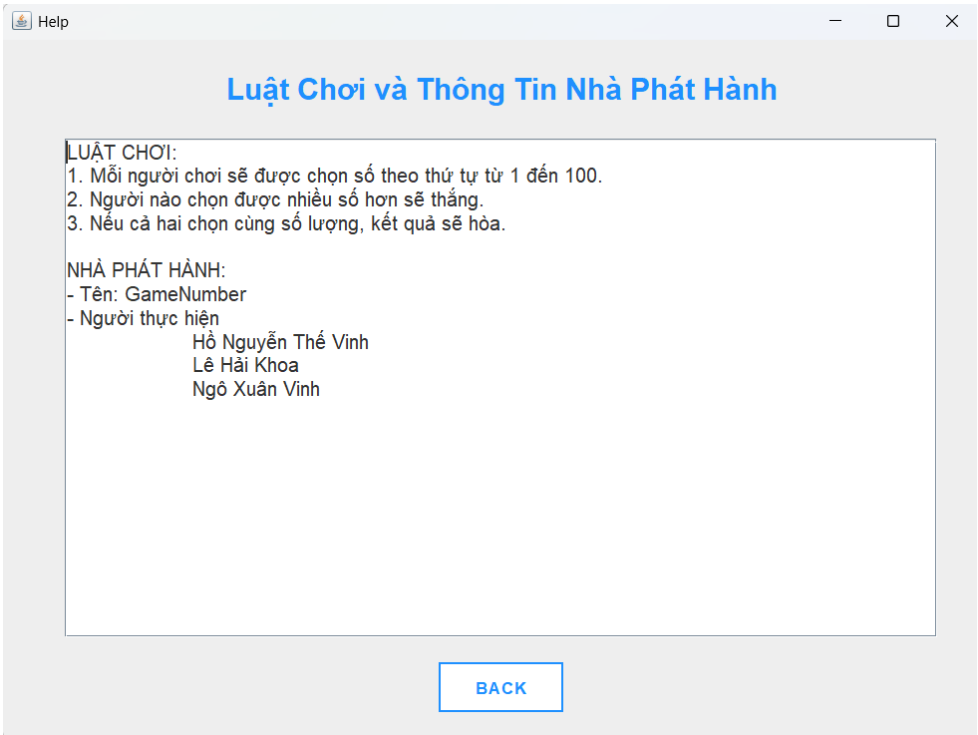
Hình 34. Nhấn nút JOIN ROOM

3.9. Giao diện thông tin cá nhân



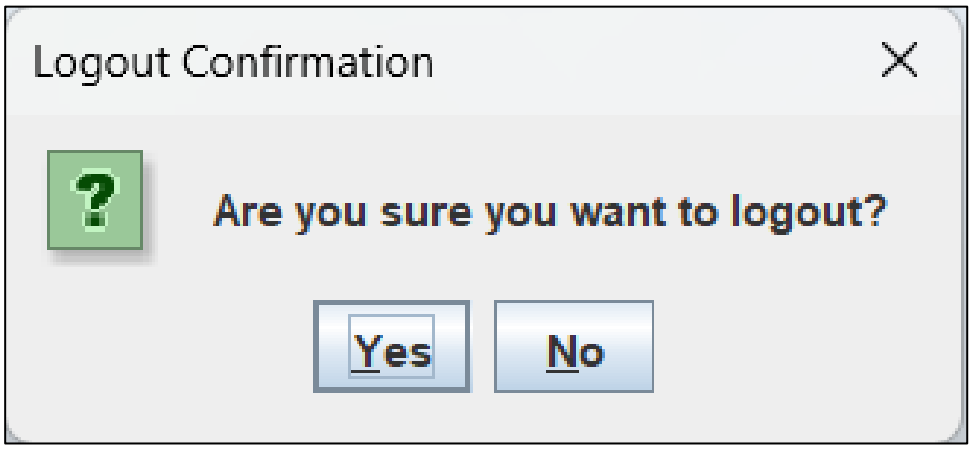
Hình 35. Thông tin người chơi

3.10. Giao diện hỗ trợ cách chơi và luật chơi



Hình 36. Giao diện luật chơi và thông tin nhà phát hành

3.11. Giao diện khi muốn LOGOUT



Hình 37. Giao diện LOGOUT

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận

Trò chơi tìm số hoàn thành được các chức năng như :

- Client :
  - Chức năng đăng nhập
  - Chức năng đăng kí
  - Chức năng chơi trò chơi
  - Chức năng xem các thông tin cá nhân
  - Chức năng xem thông tin luật và cách chơi
- Server :
  - Quản lí thông tin người chơi
  - Điều phối người chơi vào các phòng chơi thích hợp
  - Điều khiển các trận đấu

Hướng phát triển

Cập nhật thêm các chức năng, tính năng khác :

- Bảng xếp hạng : Hiển thị vị trí từng người chơi trong một khu vực (cụm máy chủ) hoặc toàn cầu
- Tính năng tùy chỉnh : có thể tùy chỉnh màu sắc hoặc chủ đề giao diện, bảng số,…mang trải nghiệm cá nhân hóa

---

TÀI LIỆU THAM KHẢO

[1] Tài liệu: https://gpcoder.com/3679-xay-dung-ung-dung-client-server-voi-socket-trong-java/

[2] Nguồn tham khảo: https://github.com/opengeogroep/filesetsync.git