Welcome to KAU live chat

- For Prof. Badruddin Alturki
- Course: CPIT-305
  - Students
    - Meshal -
    - Ahmed Abdulrhman -
    - Rayan -
    - Abdulrahman -

Introduction

Our project is a Java-based chat website that enables real-time communication, allowing Students, Lectures and Admins to send and receive messages instantly for a dynamic social experience.

The goal of this Java program is to create a simple chat room application specifically designed for students and lecturers. The application focuses on providing an easy-to-use interface, supporting multiple users, and ensuring instant messaging, making it accessible for both students and lecturers to engage effectively in conversations.

Programming Language: Java

IDEs used: Eclipse or IntelliJ IDEA

Libraries and Frameworks:

Examples: Spring, JavaFX

Need for the App

- In educational environments, effective communication between students and lecturers is crucial for enhancing learning experiences. This chat room application addresses the need for a dedicated platform where students can easily connect with their lecturers and peers. It fosters collaboration, allows for quick question-and-answer sessions, and provides a space for informal discussions outside of traditional classroom settings. With the increasing reliance on digital communication, having a real-time messaging tool specifically tailored for educational purposes can significantly improve engagement, support, and accessibility, ultimately contributing to a more interactive and supportive learning environment.

Project Structure

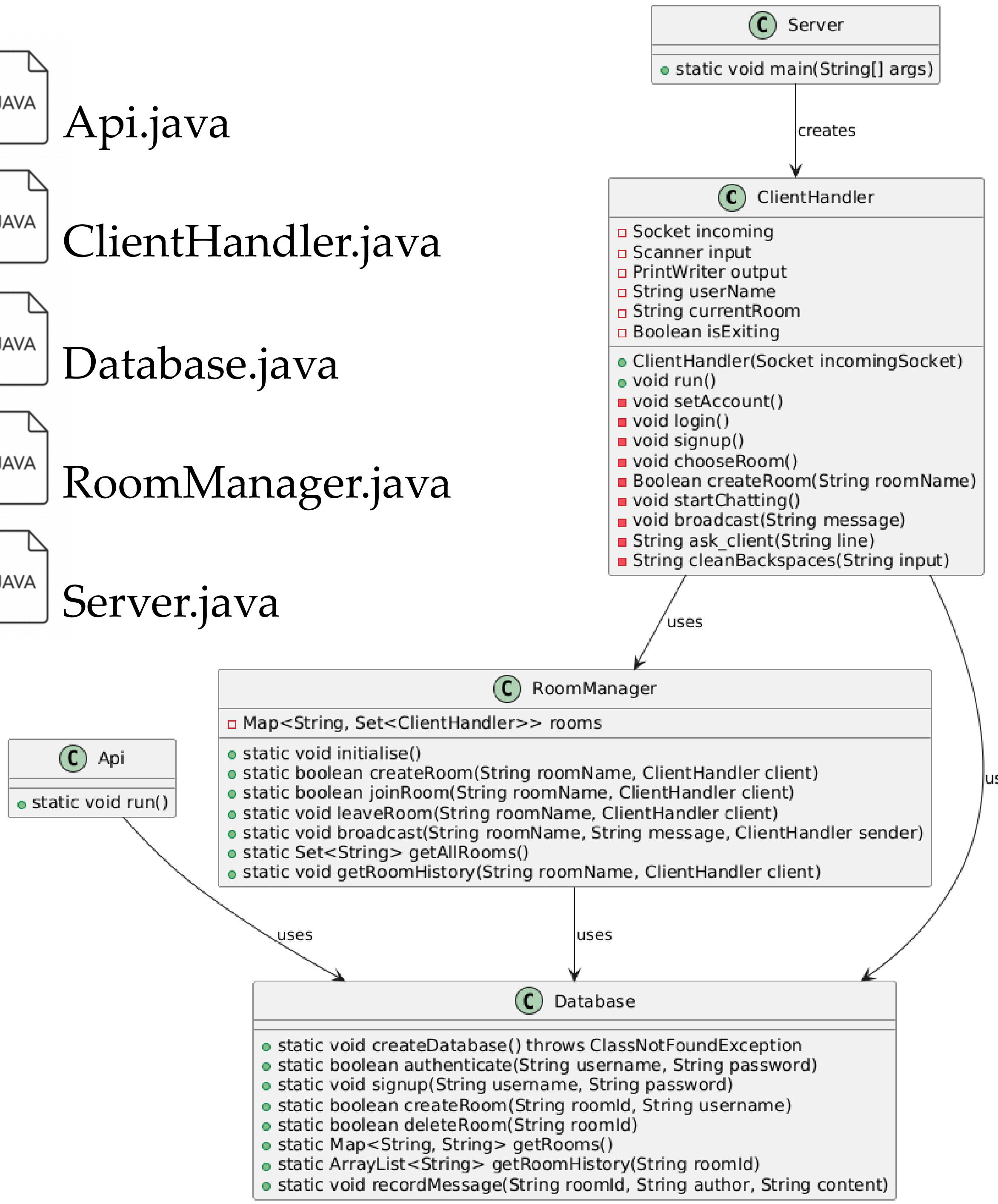
Api.java

ClientHandler.java

Database.java

RoomManager.java

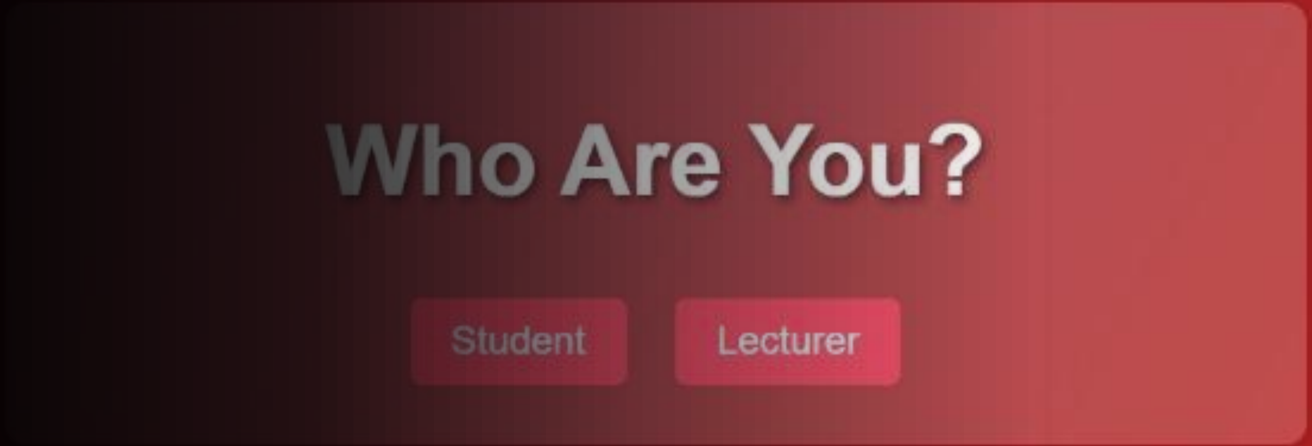
Server.java

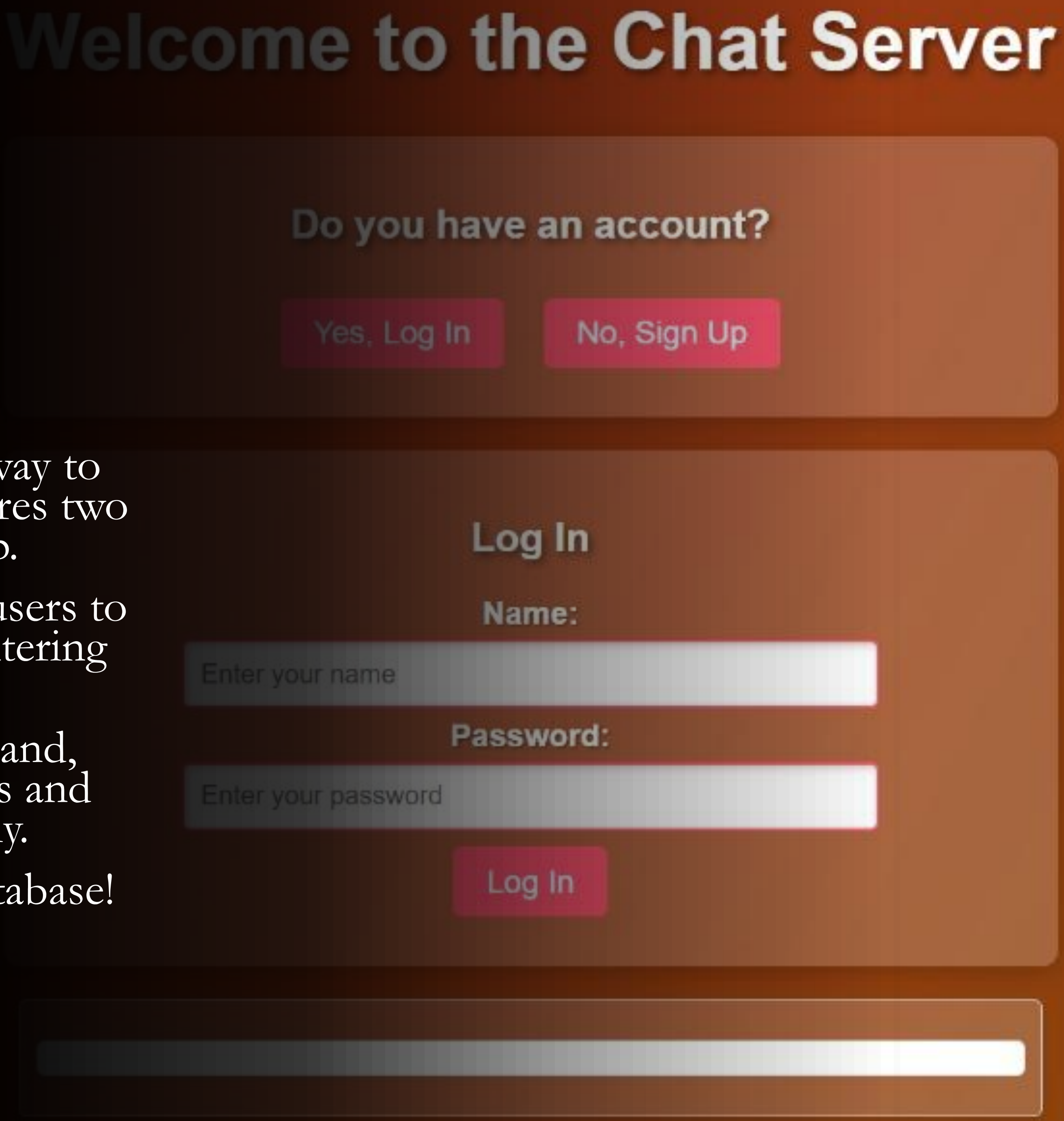

Server
static void main(String[] args)
creates
ClientHandler
Socket incoming
Scanner input
PrintWriter output
String userName
String currentRoom
Boolean isExiting
ClientHandler(Socket incomingSocket)
void run()
void setAccount()
void login()
void signup()
void chooseRoom()
Boolean createRoom(String roomName)
void startChatting()
void broadcast(String message)
String ask_client(String line)
String cleanBackspaces(String input)
uses
uses
RoomManager
Map<String, Set<ClientHandler>> rooms
static void initialise()
static boolean createRoom(String roomName, ClientHandler client)
static boolean joinRoom(String roomName, ClientHandler client)
static void leaveRoom(String roomName, ClientHandler client)
static void broadcast(String roomName, String message, ClientHandler sender)
static Set<String> getAllRooms()
static void getRoomHistory(String roomName, ClientHandler client)
Api
static void run()
uses
uses
Database
static void createDatabase() throws ClassNotFoundException
static boolean authenticate(String username, String password)
static void signup(String username, String password)
static boolean createRoom(String roomId, String username)
static boolean deleteRoom(String roomId)
static Map<String, String> getRooms()
static ArrayList<String> getRoomHistory(String roomId)
static void recordMessage(String roomId, String author, String content)

First..

- First is the most important question! Who are you?

Nothing else is needed here so after choosing you will be taken to the login signup page.



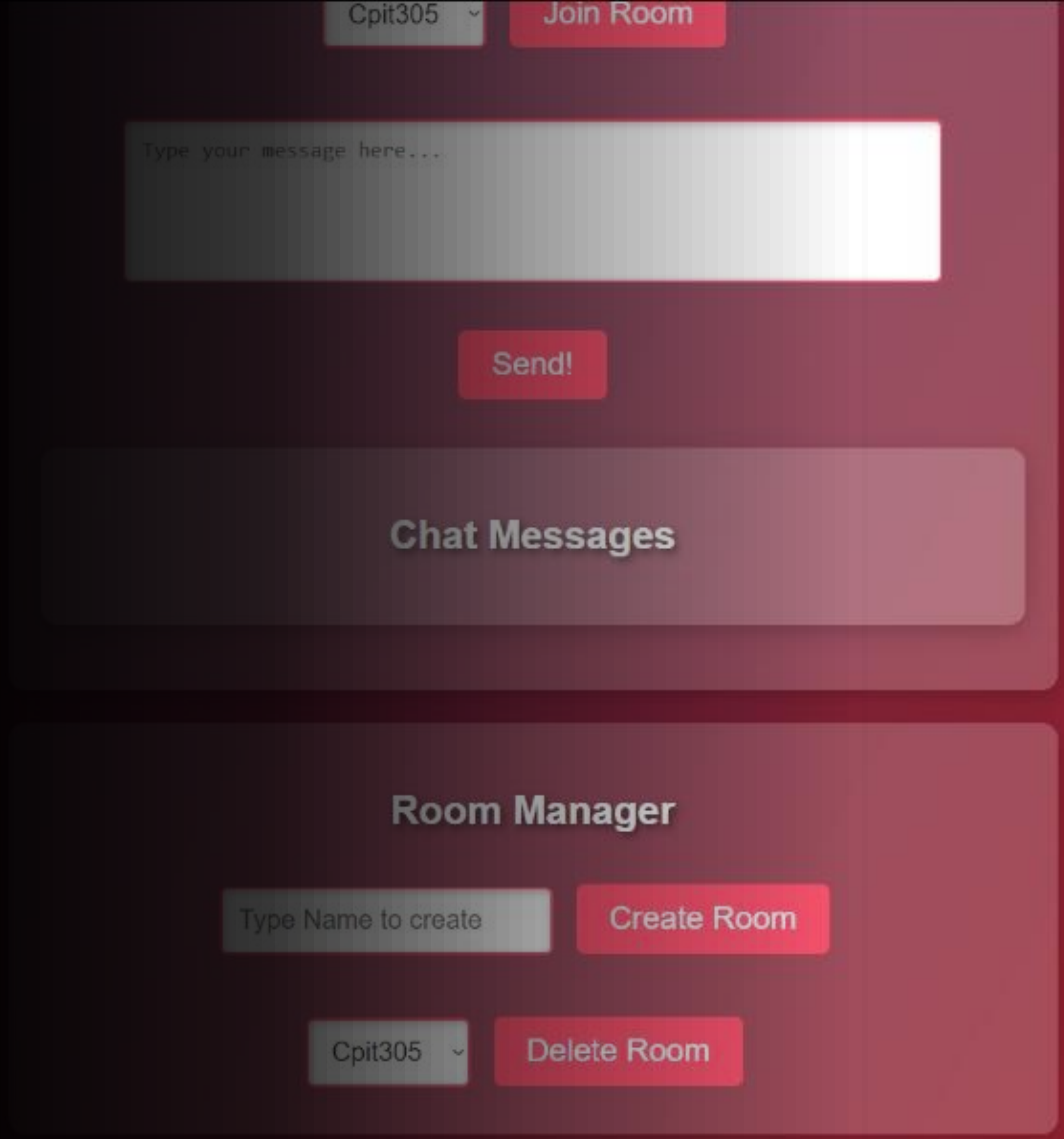


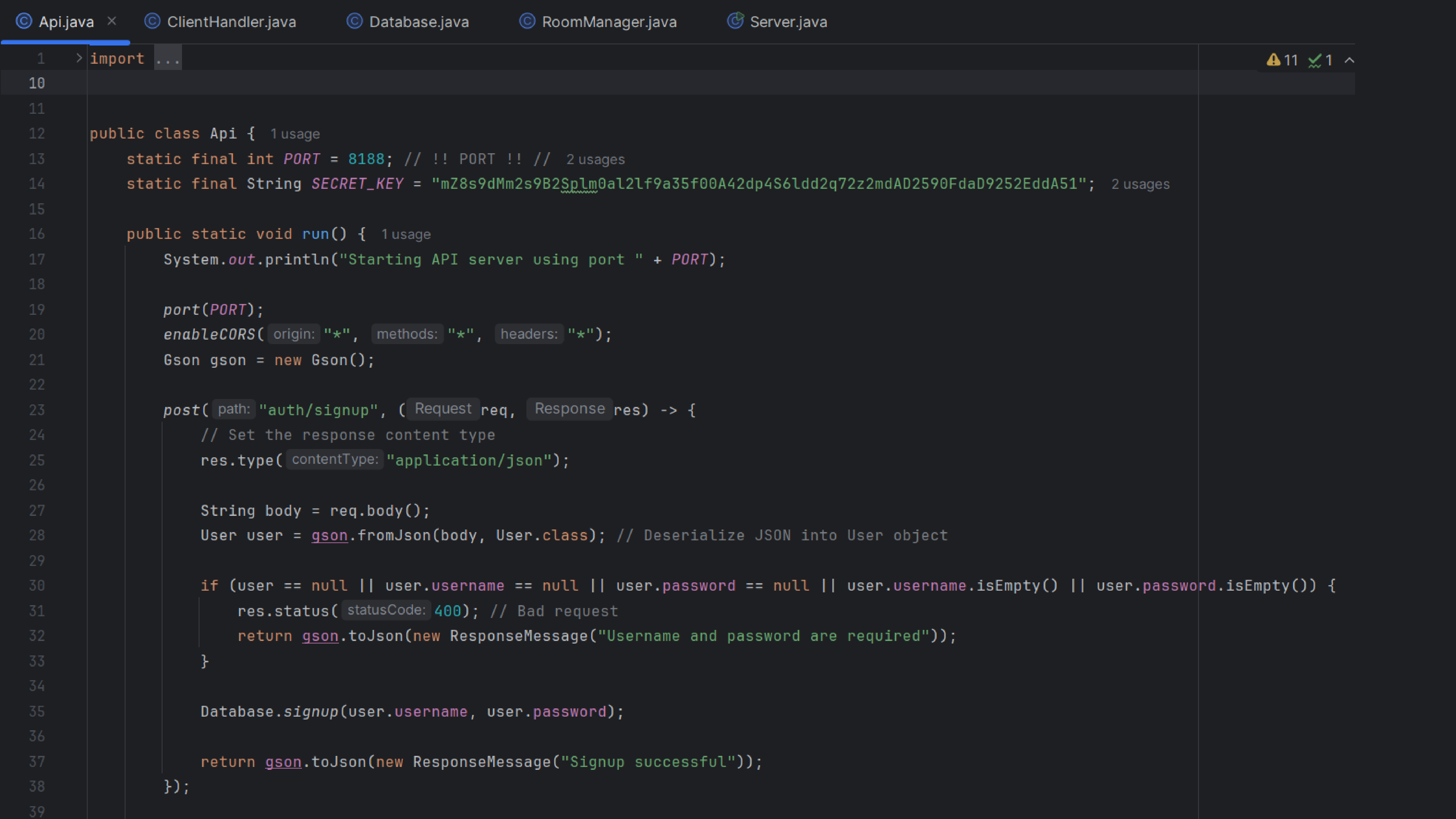
Second..

- The login/signup page is the gateway to the chat room application. It features two primary buttons: Login and Signup.
- The Login button allows existing users to access their accounts quickly by entering their credentials.
- The Signup button, on the other hand, enables new users, such as students and lecturers, to create an account easily.
- All credentials are saved in the Database!

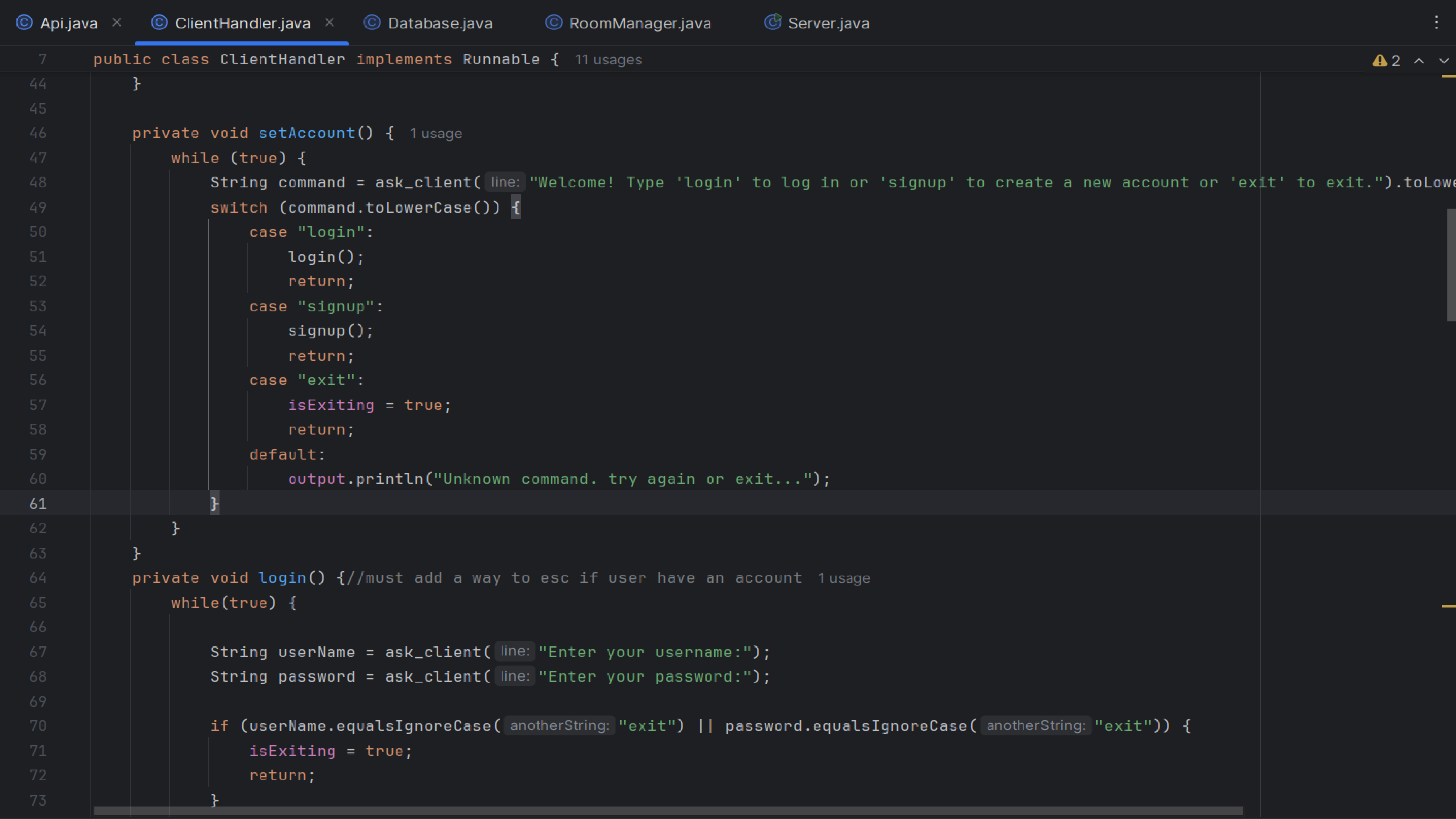
Finally

- Now we are in !
- You can now either search for a Room to join by typing the rooms name.
- Or you can Create of even Delete the Room you want.

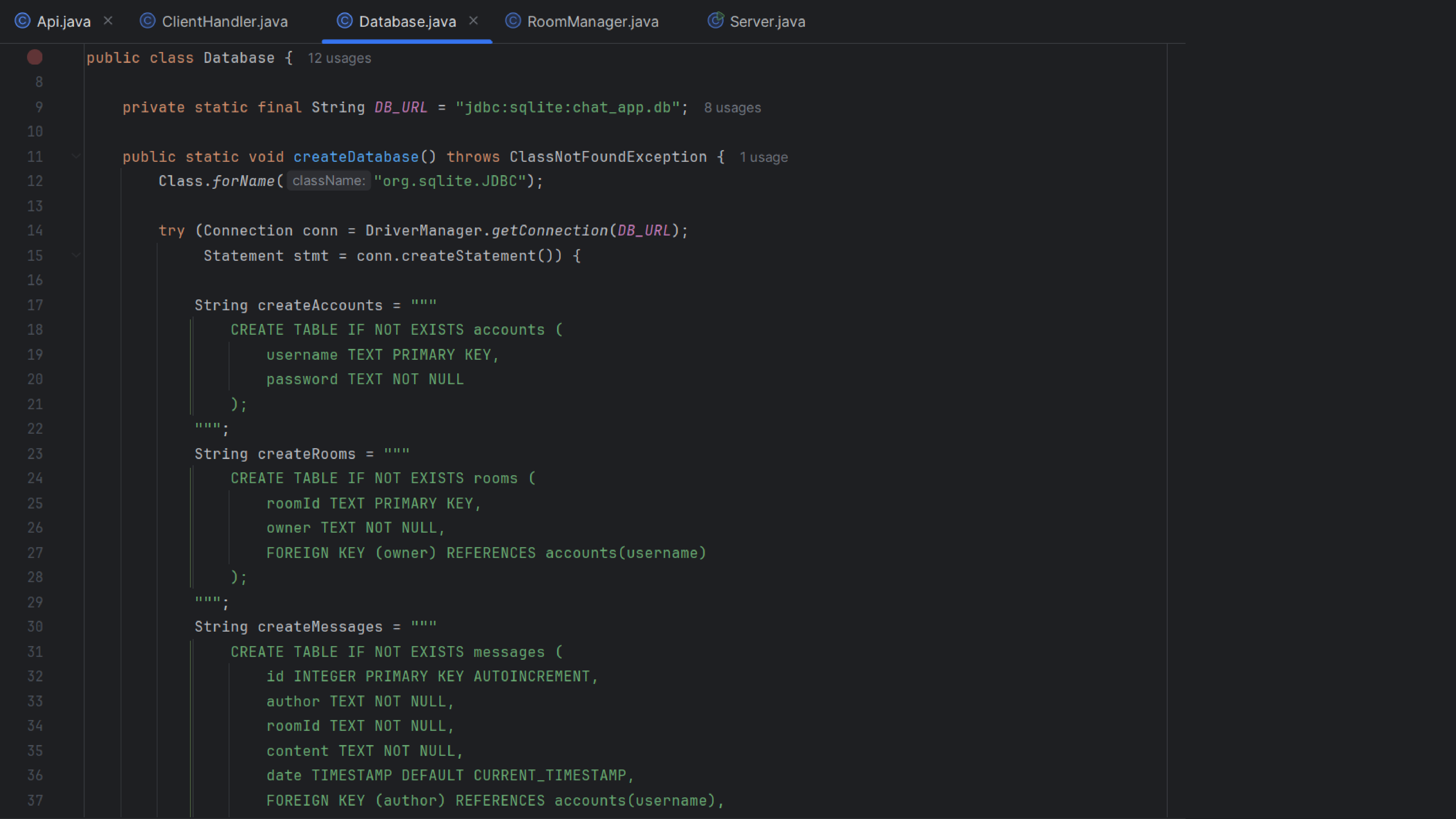




```java
import ...


public class Api {  1 usage
    static final int PORT = 8188; // !! PORT !! //  2 usages
    static final String SECRET_KEY = "mZ8s9dMm2s9B2Splm0al2lf9a35f00A42dp4S6ldd2q72z2mdAD2590FdaD9252EddA51";  2 usages

    public static void run() {  1 usage
        System.out.println("Starting API server using port " + PORT);

        port(PORT);
        enableCORS( origin: "*",  methods: "*",  headers: "*");
        Gson gson = new Gson();

        post( path: "auth/signup", ( Request req,  Response res) -> {
            // Set the response content type
            res.type( contentType: "application/json");

            String body = req.body();
            User user = gson.fromJson(body, User.class); // Deserialize JSON into User object

            if (user == null || user.username == null || user.password == null || user.username.isEmpty() || user.password.isEmpty()) {
                res.status( statusCode: 400); // Bad request
                return gson.toJson(new ResponseMessage("Username and password are required"));
            }

            Database.signup(user.username, user.password);

            return gson.toJson(new ResponseMessage("Signup successful"));
        });

```

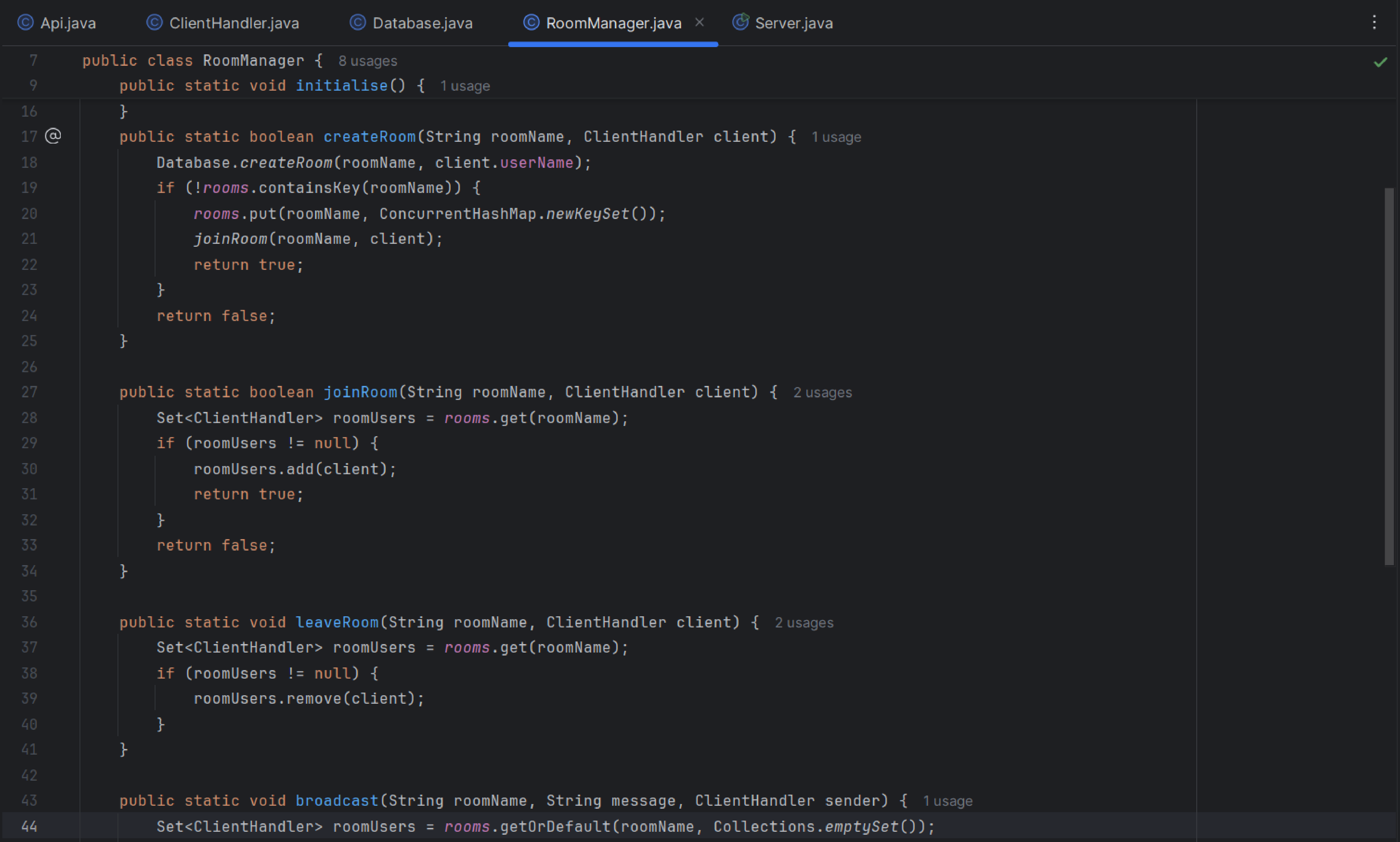


```java
public class ClientHandler implements Runnable {  11 usages
    }

    private void setAccount() {  1 usage
        while (true) {
            String command = ask_client( line: "Welcome! Type 'login' to log in or 'signup' to create a new account or 'exit' to exit.").toLowe
            switch (command.toLowerCase()) {
                case "login":
                    login();
                    return;
                case "signup":
                    signup();
                    return;
                case "exit":
                    isExiting = true;
                    return;
                default:
                    output.println("Unknown command. try again or exit...");
            }
        }
    }
    private void login() {//must add a way to esc if user have an account  1 usage
        while(true) {

            String userName = ask_client( line: "Enter your username:");
            String password = ask_client( line: "Enter your password:");

            if (userName.equalsIgnoreCase( anotherString: "exit") || password.equalsIgnoreCase( anotherString: "exit")) {
                isExiting = true;
                return;
            }
```

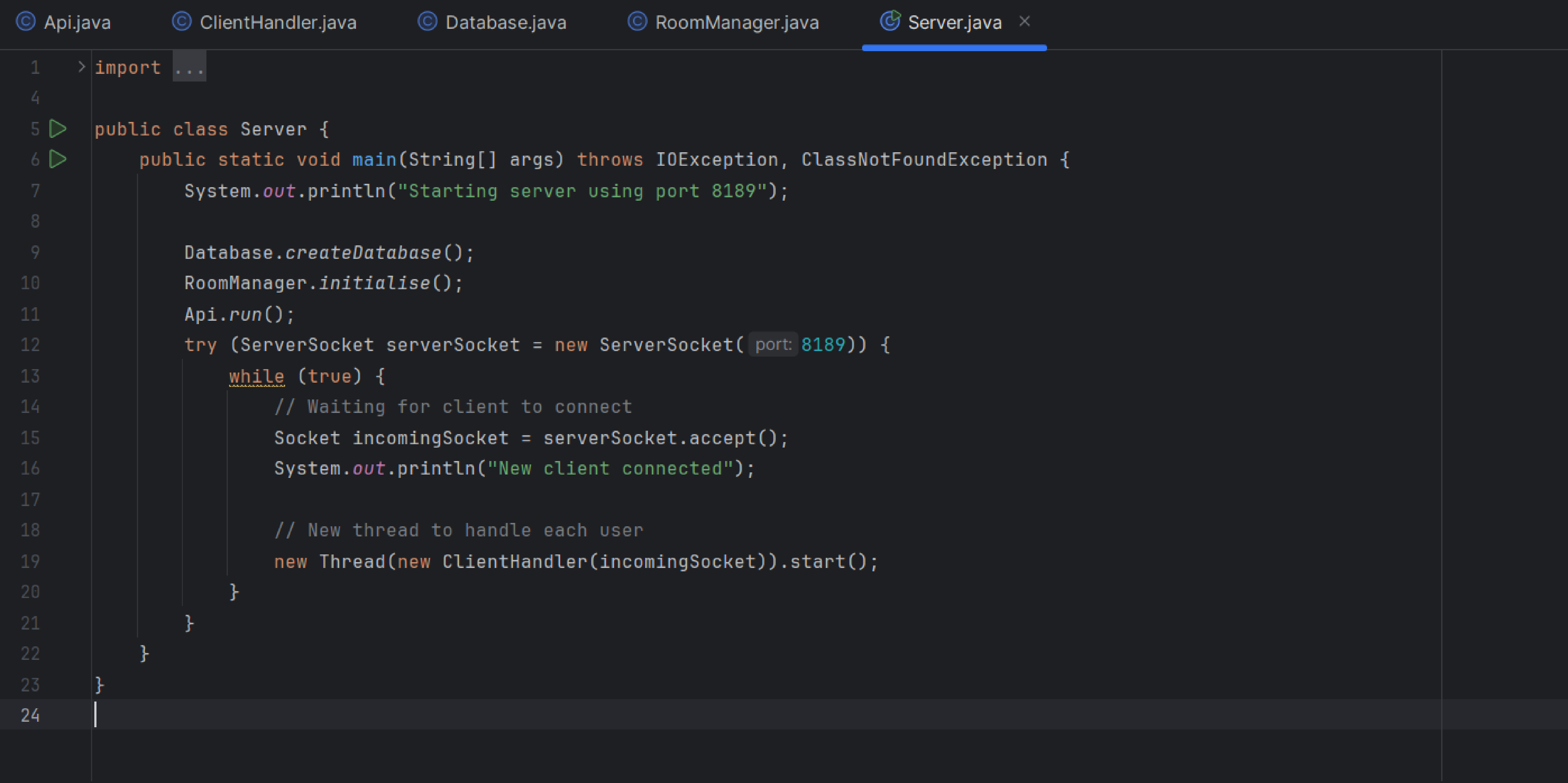


Api.java × ClientHandler.java Database.java × RoomManager.java Server.java

```
public class Database {  12 usages

    private static final String DB_URL = "jdbc:sqlite:chat_app.db";  8 usages

    public static void createDatabase() throws ClassNotFoundException {  1 usage
        Class.forName( className: "org.sqlite.JDBC");

        try (Connection conn = DriverManager.getConnection(DB_URL);
             Statement stmt = conn.createStatement()) {

            String createAccounts = """
                CREATE TABLE IF NOT EXISTS accounts (
                    username TEXT PRIMARY KEY,
                    password TEXT NOT NULL
                );
            """;
            String createRooms = """
                CREATE TABLE IF NOT EXISTS rooms (
                    roomId TEXT PRIMARY KEY,
                    owner TEXT NOT NULL,
                    FOREIGN KEY (owner) REFERENCES accounts(username)
                );
            """;
            String createMessages = """
                CREATE TABLE IF NOT EXISTS messages (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    author TEXT NOT NULL,
                    roomId TEXT NOT NULL,
                    content TEXT NOT NULL,
                    date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                    FOREIGN KEY (author) REFERENCES accounts(username),
```


Api.java ClientHandler.java Database.java RoomManager.java Server.java

```java
public class RoomManager {  8 usages
    public static void initialise() {  1 usage
    }
    public static boolean createRoom(String roomName, ClientHandler client) {  1 usage
        Database.createRoom(roomName, client.userName);
        if (!rooms.containsKey(roomName)) {
            rooms.put(roomName, ConcurrentHashMap.newKeySet());
            joinRoom(roomName, client);
            return true;
        }
        return false;
    }

    public static boolean joinRoom(String roomName, ClientHandler client) {  2 usages
        Set<ClientHandler> roomUsers = rooms.get(roomName);
        if (roomUsers != null) {
            roomUsers.add(client);
            return true;
        }
        return false;
    }

    public static void leaveRoom(String roomName, ClientHandler client) {  2 usages
        Set<ClientHandler> roomUsers = rooms.get(roomName);
        if (roomUsers != null) {
            roomUsers.remove(client);
        }
    }

    public static void broadcast(String roomName, String message, ClientHandler sender) {  1 usage
        Set<ClientHandler> roomUsers = rooms.getOrDefault(roomName, Collections.emptySet());
```


Api.java ClientHandler.java Database.java RoomManager.java Server.java

```java
import ...

public class Server {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        System.out.println("Starting server using port 8189");

        Database.createDatabase();
        RoomManager.initialise();
        Api.run();
        try (ServerSocket serverSocket = new ServerSocket( port: 8189)) {
            while (true) {
                // Waiting for client to connect
                Socket incomingSocket = serverSocket.accept();
                System.out.println("New client connected");

                // New thread to handle each user
                new Thread(new ClientHandler(incomingSocket)).start();
            }
        }
    }
}
```

Technologies Used

Java:	The primary programming language for building the application.
Socket Programming:	Utilized for real-time communication between the client and server. The Socket class is used for handling incoming connections.
Java AWT:	Used for handling input/output streams (Transferable and PrintWriter).
Database:	**SQLite**: A lightweight database used for storing user accounts, chat rooms, and messages. Managed through JDBC (Java Database Connectivity).
Multi-threading:	Each client connection is handled in a separate thread using the Runnable interface, allowing multiple clients to connect simultaneously.
JSON:	The application uses JSON for data exchange, particularly in the API endpoints, facilitated by the Gson library for serialization and deserialization.

Conclusion

- In the end, our chat application effectively demonstrates real-time communication using Java, showcasing key technologies such as socket programming and SQLite for data management. The integration of JWT enhances security through reliable user authentication, while the Spark framework simplifies API interactions.
- This project highlights essential programming concepts, including multi-threading and data handling, and offers a scalable solution for concurrent user engagement. Future improvements could focus on enhancing features and optimizing performance.
- Thank you for your attention! I'm happy to take any questions or feedback.