# Secure Chat Room Implementation Reference Document

## Introduction

Welcome to the Secure Chat Room Implementation assignment for LeadMint. The objective of this assignment is to develop a secure and interactive chat room system using JavaScript, Node.js, MySQL, and Express. This document will guide you through the steps required to complete the assignment, ensuring security measures, user authentication, and real-time communication are effectively implemented. By the end of this assignment, you will have created a robust chat room system with comprehensive functionalities.

## Step-by-Step Instructions

### Step 1: User Registration and Authentication

- **Objective**: Implement user registration and authentication using JWT.

- Users need to provide the following details for registration:

- `userId`: Unique identifier.

- `deviceId`: Device identifier.

- `name`: Full name.

- `phone`: Contact number.

- `availCoins`: Available coins.

- **Steps**:

1. Set up your Node.js environment and initialize a new project using `npm init`.

2. Install necessary packages like `express`, `jsonwebtoken`, `bcrypt`, and `mysql`.

3. Create a registration API endpoint to handle user registration data.

4. Store passwords securely using bcrypt.

5. Implement JWT for user authentication and create a login endpoint.

### Step 2: Chat Room Creation

- **Objective**: Allow prime members to create chat rooms with a maximum capacity of 6 people.

- Only authenticated prime members should have access.

- **Steps**:

1. Create an endpoint (`POST /api/chatrooms`) for chat room creation.

2. Ensure only prime members can create chat rooms.

3. Implement logic to prevent participants from joining if the room is full.

### Step 3: Inviting Participants

- **Objective**: Allow chat room creators to invite other prime members using a secure token system.

- Non-prime members can join one room for free, then must pay 150 coins for additional rooms.

- **Steps**:

1. Implement a secure invitation mechanism using tokens.

2. Track the number of rooms non-prime members have joined and manage coin deductions accordingly.

### Step 4: Joining a Room as a Non-Prime Member

- **Objective**: Allow non-prime members to join a room, with specific conditions for coins and access.

- If the user has already joined one room for free, ensure they have 150 coins for additional rooms.

- **Steps**:

1. Create an endpoint (`POST /api/joinroom`) to handle room joining for non-prime members.

2. Verify prime membership status and manage free and paid room access.

### Step 5: Chat Functionality

- **Objective**: Enable real-time message sending and receiving within chat rooms.

- Implement WebSocket for this purpose.

- **Steps**:

1. Set up WebSocket communication in your Node.js server.

2. Create an endpoint (`POST /api/messages`) for sending messages.

3. Ensure messages are delivered in real-time using WebSocket.

### Step 6: Profile Viewing

- **Objective**: Allow users to view each other's profiles.

- Fetch user profiles via an API endpoint.

- **Steps**:

1. Create an endpoint (`GET /api/profile/:userId`) for profile retrieval.

2. Implement logic to fetch and display user details securely.

### Step 7: Friend Requests

- **Objective**: Enable users to send and accept friend requests.

- **Steps**:

1. Create an endpoint (`POST /api/friend-requests`) for sending friend requests.

2. Manage request status and notifications.

### Step 8: Database Management

- **Objective**: Store user details, chat room information, messages, and friend requests in MySQL.

- **Steps**:

1. Design a MySQL database schema for the required data.

2. Implement SQL queries and integrate with your Node.js server.

### Step 9: Security Measures

- **Objective**: Implement security features such as secure password storage and data protection.

- **Steps**:

1. Use bcrypt for password hashing.

2. Implement JWT for secure user sessions.

3. Perform thorough validation and error handling.

### Step 10: Error Handling and Validation

- **Objective**: Ensure robust error handling and input validation.

- **Steps**:

1. Implement middleware for input validation.

2. Provide clear error messages and status codes.

### Step 11: Version Control

- **Objective**: Use Git for version control throughout the development process.

- **Steps**:

1. Initialize a Git repository for your project.

2. Commit regularly with clear messages.

3. Push your changes to a remote GitHub repository.

## Best Practices

1. **Security First**: Always prioritize secure password storage, JWT management, and protected endpoints.

2. **Code Readability**: Write clean, commented code to ensure maintainability and understandability.

3. **Error Handling**: Implement comprehensive error handling and clear user feedback.

4. **Frequent Commits**: Use Git for regular commits, making it easier to track changes and revert if needed.

5. **Documentation**: Document your endpoints, expected request/response formats, and any additional features clearly.

## Submission Guidelines

1. **GitHub Repository**: Provide a link to your GitHub repository containing the complete codebase.

2. **README File**: Include a README file with instructions for setting up and running the application.

3. **SQL File**: Share the database schema in a separate `.sql` file.

4. **API Documentation**: Clearly document API endpoints, request/response formats, and any additional features.

### Submission Deadline

- **Due Date**: 6th June 2024, by 7 P.M.

- **Submit To**: Email submission to rishav@leadmint.io with a cc to shabaj@leadmint.io

## Frequently Asked Questions (FAQ)

**Q1: How do I ensure secure password storage?**

- Use bcrypt for hashing passwords before storing them in the database.

**Q2: How do I implement real-time chat functionality?**

- Use WebSocket for establishing real-time communication.

**Q3: What are the conditions for non-prime members joining chat rooms?**

- Non-prime members can join one chat room for free. For additional rooms, they must have 150 coins.

**Q4: How do I set up my development environment?**

- Install Node.js, set up an Express server, connect to MySQL, and initialize WebSocket for real-time capabilities.

**Q5: What should I include in the README file?**

- Provide setup instructions, running the application, API documentation, and any additional notes or features.

For any queries, please reach out to rishav@leadmint.io or shabaj@leadmint.io. Good luck with your assignment!