Technical Test - Backend

Applicant are required to answer all question with write the functions by **any programming language and Postman (To Do the API Testing)**

Major: Develop messaging system

Question 1

Design a messaging system database using the ERD diagram based on requirement in **Question 2**.

Question 2

2.1
You are tasked with developing a secure, role-based messaging system API. This API will handle user registrations, login processes, email verification, and provide differentiated access levels based on user roles (Super Admin, Admin, Normal User). The system should ensure secure transmission and storage of credentials using hashing and JWT for authentication. Additionally, integrate email verification using an email service like Mailtrap for newly registered users.

Requirements:

User Registration and Email Verification:

- Implement a RegisterUser endpoint that accepts username, password, and email. Passwords must be hashed before storing in the database.
- Upon registration, the system should generate a unique verification link and send it to the user's email using **Mailtrap**.
- Include an VerifyEmail endpoint where users can confirm their email address by clicking the verification link.

Login Function:

- Develop a Login endpoint that accepts username and password. Use the hashed password for verification.
- After successful login, generate a JWT token that includes the user's ID, role, and a custom claim indicating whether the user's email has been verified.
- Users should not be able to login if their email is not verified.

Role-based Access Control:

- Implement middleware or filters to validate JWT tokens and enforce role-based access control for each endpoint.
- Normal users can only send and receive messages.
- Admins can send and read all messages.
- Super Admins have full CRUD capabilities, including the ability to create and delete users.

Bonus (Optional):

- Implement rate limiting on the Login and RegisterUser endpoints to protect against brute force attacks.
- Use refresh tokens to allow users to remain authenticated without needing to frequently re-enter credentials.

Questions:

System Architecture:

- Describe the architecture of your API, focusing on security, scalability, and maintainability. How do you ensure the system is secure against common vulnerabilities?

Code Implementation:

- Provide detailed code examples for the RegisterUser, Login, and VerifyEmail endpoints, including any middleware or services used for email sending, JWT handling, and role-based access control.

Security Considerations:

- Discuss the hashing algorithm chosen for password storage and the rationale behind it.
- Explain how the JWT token and role-based access control mechanism work together to secure the API.

Database Schema:

- Outline the database schema, including tables for users, roles, and permissions. Explain how relationships between these entities are managed.

Performance and Scalability:

- Address how your design supports scalability, especially in terms of handling a large number of concurrent connections and ensuring fast response times.

Error Handling and Logging:

- Describe your strategy for error handling and logging, including how you would provide meaningful error messages to the client while logging detailed information for debugging purposes.

2.2

Create APIs (Restful API) **based on 2.1** on:

You are tasked with designing a simple messaging system for an application. The system will allow users to send and receive messages in real-time. To ensure security and identify the users, JWT (JSON Web Tokens) will be used. The messages will be stored in a PostgreSQL database for persistence.

Requirements:

SendMessage Function:

- Develop a function named SendMessage that allows a user to send a message.
- The function must accept at least two parameters: the message content and a JWT token.
- The JWT token should be used to verify the identity of the user sending the message. Assume the token contains a user ID and is already valid.
- Upon successful verification, the message, along with the sender's user ID and timestamp, should be stored in a PostgreSQL database.

ReceiveMessage Function:

- Implement a function named ReceiveMessage that receiving of messages.
- When a new message is received in the system, it should be broadcasted to all connected clients in real-time.

Questions:

Design Considerations:

- Explain your design choices for the messaging system, particularly how you plan to implement JWT authentication.

Code Implementation:

- Provide pseudo-code snippets for both SendMessage and ReceiveMessage functions. Highlight how you handle JWT token verification.

Database Schema:

- Describe the PostgreSQL database schema you would use to store messages. Include the tables, columns, and any indexes you believe are necessary for efficient operations.

Security and Performance:

- Discuss the security measures you would implement to protect the messaging system, focusing on the use of JWT tokens.
- Additionally, outline any considerations or techniques you would employ to ensure the system performs well, especially under high load.

Submit Requirement

1. Please write down how long (hours and minutes) it took you to complete the assessment
2. Record your output as a video to clearly demonstrate all functions by clicking on all the necessary tabs, buttons, and so on.
3. Zip your source code, video record, and submit it to the person in charge

All The Best