Name: Vikesh Kumar Bairwa,

Mob.9024220603,

email:vikeshbairwa890@gmail.com

Project Components:

1. Main Service:

This will be the core service responsible for handling user authentication and authorization using JWT.

Responsibilities.

- **User Authentication and Authorization:** Implement user login and registration functionality.
    - **Login Endpoint:** Handle user login requests. Verify provided credentials and generate a JWT token for successful login.
    - **Registration Endpoint:** Handle user registration requests. Store user data in the database.
    - **Protected Endpoints:** Implement protected routes that require a valid JWT token for access. These endpoints should only be accessible to authenticated users.
- **Candidate Management:** Manage candidate data.
    - **Add Candidate Endpoint:** Implement an endpoint to add new candidate data to the database. This endpoint will also store the user ID of the user who added the candidate.
    - **Retrieve Candidate Endpoint:** Implement an endpoint to retrieve candidate data. The endpoint should retrieve candidates added by the current user.

2.Public API Microservice:

This microservice provides a public API key that can be used to access main service routes without requiring a login.

Responsibilities

- **Public API Key:** Generate and manage public API keys.
- **Public Endpoints:** Implement public API endpoints that do not require user authentication.
    - **User Profile Endpoint:** Provide an endpoint to retrieve the profile information of the user who is using the public API key. This endpoint should return data in JSON format.

- **Candidate Retrieval Endpoint:** Implement an endpoint to retrieve all candidates related to the user who is using the public API key.

Database:

The user database should contain the following fields:

- `id`: Unique user identifier.
- `first_name`: User's first name.
- `last_Name`: User's last name.
- `email`: User's email address.
- `password_hash`: Hashed password of the user.

2. JWT Authentication Endpoints

- **POST** `/api/register`: Register a new user.
- **POST** `/api/login`: Authenticate an existing user.
- **POST** `/api/protected`: A protected endpoint that requires a valid JWT token for access.

3. Additional Endpoints

- **POST** `/api/candidate`: Add a new candidate to the database.
- **GET** `/api/candidate`: Retrieve candidates for the current user.

4. Public API Endpoints

- **POST** `/api/public/profile`: Retrieve user profile information based on the API key.
- **GET** `/api/public/candidate`: Retrieve candidates associated with the user using the API key.

Connection between Main Service and Public API Microservice

The main service should be able to communicate with the public API microservice to access its functionalities.

Documentation:

Provide a comprehensive documentation guide on how to set up and run both services. The documentation should include:

- **Installation Instructions:** Include all necessary commands to initialize the projects and install required dependencies.
- **Setup Guide:** Provide clear instructions on how to set up both services, including configuration steps.
- **API Documentation:** Document the endpoints of both the main service and the public API, including request parameters, response format, and expected responses.
- **Example Code** Provide examples of how to use the API endpoints.

Submission

- **Code Submission:** Submit the code for both the main service and the public API microservice. Include clear comments and documentation within the code.
- **Documentation Submission:** Submit a well-structured documentation guide in a separate document format (e.g., PDF). Include step-by-step instructions for setting up, running, and connecting both services. List all necessary commands and configurations.

IMPORTANT FOR SUBMISSION:

The zip file must contain the following:

- Code
- Documentation
- Database dump

The zip file must contain Details **Project_Link**:

Evaluation Criteria

Your assignment will be evaluated based on the following criteria:

- **Functionality:** Does the authentication system work as intended? Are the public API endpoints accessible with the public API?
- **Code Quality:** Is the code well-structured, modular, and follows best practices? Are there comments where necessary?
- **Documentation:** Is the documentation clear and comprehensive? Can someone new to the project easily set up and understand the system?
- **Microservices Architecture:** How well are the main service and public API microservice designed to work together as separate entities?
- **Security:** Is the authentication system secure, and are best practices followed for communication between microservices?