**Project Documentation: Role-Based Access Control System (RBAC)**

**Introduction**

This project is a **Role-Based Access Control (RBAC) System** designed to manage user authentication, roles, and profiles securely and efficiently. The backend architecture follows a microservices approach to ensure modularity and scalability. It integrates robust security practices such as password encryption, token-based authentication, and role validation.

**Core Components:**

1. **User Authentication Service**:
   * Manages user registration and login.
   * Ensures secure password storage using **BCryptPasswordEncoder**.
   * Issues tokens signed with **HMAC SHA-256** for user sessions.
   * Exposes endpoints for user authentication.
2. **Role Management Service**:
   * Handles the creation, retrieval, updating, and deletion of roles.
   * Assigns roles to users dynamically.
   * Validates roles for access control through API communication with other services.
3. **User Profile Management Service**:
   * Manages CRUD operations for user profiles.
   * Updates user information like name, contact, and preferences.
   * Interacts with Authentication and Role Management services to ensure secure data handling.
4. **Role-Based Access Control (RBAC)**:
   * Implements a robust RBAC model to restrict access to resources based on user roles.
   * Ensures only authorized users can perform specific actions.
   * Integrates seamlessly with the Authentication and Role Management services.

**Technologies Used**

* **Java 17**: Language used for backend development.
* **Spring Boot 3.x**: Framework for rapid application development.
* **MySQL**: Database for persistent storage.
* **OAuth 2.0**: Standard protocol for secure authentication and authorization.
* **HMAC SHA-256**: Token signing for secure communication.
* **Lombok**: Simplifies Java code by reducing boilerplate.
* **Slf4j**: Logging for better monitoring and debugging.
* **ModelMapper**: Facilitates object mapping between DTOs and entities.
* **RESTful APIs**: Ensures proper communication between microservices.
* **RBAC Principles**: Implements secure role-based access control.

**Key Features**

1. **Authentication and Security**:
   * Secure user registration and login with encrypted passwords.
   * Token-based authentication for user sessions.
   * Role-based access control to manage permissions.
2. **Role Management**:
   * CRUD operations for roles.
   * Flexible role assignment to users.
   * API communication for role validation.
3. **RBAC (Role-Based Access Control)**:
   * Defines roles and associated permissions.
   * Restricts access to endpoints based on user roles.
   * Simplifies management of user privileges.
4. **User Profile Management**:
   * Create, update, and delete user profiles.
   * Retrieve profile details via secure APIs.
   * Interoperability with other services.
5. **Microservices Architecture**:
   * Separation of concerns through modular services.
   * Independent deployment and scaling of services.

**How to Run the Project**

**Pre-Requisites**

* Java 17 installed.
* Maven installed for building the project.
* MySQL database running locally or on a server.

**Setup Instructions**

1. Unzip all three folders by clicking extract all.
2. Now open them in your IDE as follows right chick

**import – Existing maven project – now select one by one folder.**

1. Import all the required libraries.
2. **Database Setup**:
3. Create a database named rbac\_db in MySQL.
4. Update the application.properties or application.yml files in each service with the following:
   * spring.datasource.url=jdbc:mysql://localhost:3306/?user=root
   * spring.datasource.username=local
   * spring.datasource.password=Ameer@123
5. **Build the Project**:
   * Navigate to each service folder and build:
   * mvn clean install
6. **Run the Services**:
   * Start each service from its main class or use:
   * **mvn spring-boot:run**
7. **Test the APIs**:
   * Use tools like **Postman** or **cURL** to interact with the services.

**Highlighted Skills**

* **Backend Development**: Expertise in Java, Spring Boot, and RESTful API design.
* **Database Management**: Schema design and efficient querying using MySQL.
* **Security**: Implemented OAuth 2.0, password encryption, HMAC SHA-256, and token-based authentication.
* **RBAC Implementation**: Designed and enforced secure role-based access control policies.
* **Microservices Architecture**: Modular service design for scalability and maintainability.
* **Code Optimization**: Used Lombok for reducing boilerplate code and ModelMapper for object transformations.
* **Logging and Monitoring**: Integrated Slf4j for structured logging.

**API Endpoints:** Below are some of the key API endpoints for each service:

**User Authentication Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /auth/register | Register a new user |
| POST | /auth/login | Login and get JWT token |

**Role Management Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /roles | Create a new role |
| GET | /roles/{id} | Get role by ID |
| PUT | /roles/{id} | Update role by ID |
| DELETE | /roles/{id} | Delete role by ID |

**User Profile Management Service**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /profiles | Create a new user profile |
| GET | /profiles/{id} | Get user profile by ID |
| PUT | /profiles/{id} | Update user profile by ID |
| DELETE | /profiles/{id} | Delete user profile by ID |