**Library Management System Using Java and AWS**

**Objective**: Develop a Library Management System that facilitates CRUD operations on book and member data, incorporates role-based access control, and integrates seamlessly with AWS cloud services**.**

**Tech Stack:**

* Java 17
* Spring Boot 3
* Spring Data JPA
* Amazon RDS (for database)
* Amazon Cognito (for OAuth2 authentication)
* AWS Lambda
* Amazon S3 (for storage)
* Amazon SQS (for messaging)
* Amazon SNS (for notifications)
* Amazon API Gateway
* Amazon CloudWatch (for monitoring)
* Swagger (for API documentation)
* Postman (for API testing)
* MapStruct (for DTO and model mapping)

**Use Case**: A library aims to efficiently manage book inventories and member records with the following role-based access:

* Admins can add, update, and delete book and member records.
* Librarians can view and update book status and member interactions.
* Members can view book availability and their own borrowing history.

**Functional Requirements:**

* **Login System:** Implement OAuth2 authentication using Amazon Cognito.
* **Role-Based Access**: Define roles for Admin, Librarian, and Member.
* **CRUD Operations:**
  + **Add book/member**: Accessible by Admin only.
  + **View all books/members:** Accessible by Admin, Librarian, and Member (members can only view their own records).
  + **Update book/member:** Accessible by Admin and Librarian.
  + **Delete book/member:** Accessible by Admin only.
* **Data Storage**: Utilize Amazon RDS with Spring Data JPA for data management.

**Key Entities:**

* **User:** Contains ID, Username, Password, and Role (ADMIN, LIBRARIAN, MEMBER).
* **Book**: Contains ID, Title, Author, ISBN, Status (Available, Borrowed).
* **Member:** Contains ID, Name, Email, Phone, Borrowed Books.

**Cloud Services:**

1. **Authentication Service**
   * Manages user authentication and authorization using Amazon Cognito.
   * **APIs:**
     + **POST /auth/login**: Handles login and returns OAuth2 response.
2. **Book Management Service**
   * **Manages CRUD operations for book data.**
   * **APIs:**
     + **POST /books/:** Creates a new book record (Admin only).
     + **GET /books/:** Retrieves all book records (Admin, Librarian, Member).
     + **GET /books/{id}:** Retrieves a specific book record.
     + **PUT /books/{id}:** Updates a book record (Admin, Librarian).
     + **DELETE /books/{id}:** Deletes a book record (Admin only).
3. **Member Management Service**
   * **Manages CRUD operations for member data.**
   * **APIs:**
     + **POST /members/:** Creates a new member record (Admin only).
     + **GET /members/:** Retrieves all member records (Admin, Librarian).
     + **GET /members/{id}:** Retrieves a specific member record.
     + **PUT /members/{id}:** Updates a member record (Admin, Librarian).
     + **DELETE /members/{id}:** Deletes a member record (Admin only).
4. **Notification Service**
   * **Manages notifications for book due dates and member interactions.**
   * **AWS Lambda:** Triggered by events to send notifications.
   * **Amazon SQS**: Queue for managing notification messages.
   * **Amazon SNS**: Service for sending notifications.

**Implementation:**

* **Create Operation:** Admins can add new book and member records.
* **Read Operation**: Admins, Librarians, and Members can view all book records; Members can view their own records.
* **Update Operation**: Admins and Librarians can update book and member records.
* **Delete Operation:** Admins can delete book and member records.

**Intermediate Concepts Covered in Java and AWS:**

* OAuth2 Authentication with Amazon Cognito.
* Microservices Communication using Amazon SQS.
* Centralized Configuration with AWS Systems Manager Parameter Store.
* Exception Handling using Spring Boot Middleware and Filters.
* DTOs and Model Mapping with MapStruct.
* Environment-Based Configuration using Spring Boot Profiles.
* Real-Time Data Processing with AWS Lambda and Amazon SNS.