

1. Library Management System

Problem Statement:

Develop a REST API for managing a library's collection of books. Each book should have the following attributes: id, title, author, isbn, publishedDate.

Requirements: CRUD Operations:

Create a new book record.

Retrieve a list of all books.

Retrieve details of a specific book by its ID.

Update an existing book's details.

Delete a book record by its ID.

Fetch books by a specific author.

Fetch books published after a given year.

Layers:

Entity: Book

Controller: /api/books

Service: Handle business logic for book operations.

Repository: Use JPA for database operations.

2. Student Management System

Problem Statement:

Create a REST API to manage student records. Each student should have the following attributes: id, name, email, course, enrollmentDate.

Requirements: CRUD Operations:

Add a new student record.

Retrieve all student records.

Retrieve a specific student by ID.

Update a student's information.

Delete a student record.

Fetch students enrolled in a specific course.

Search students by name using a partial match.

Layers:

Entity: Student

Controller: /api/students

Service: Handle business logic for student operations.

Repository: Use JPA for database interactions.

3. Employee Management System

Problem Statement:

Develop a REST API to manage employee details. Each employee should have the following attributes: id, name, designation, department, salary, and joiningDate.

Requirements: CRUD Operations:

Add a new employee record.

Retrieve all employees.

Retrieve a specific employee by ID.

Update an existing employee's details.

Delete an employee record by ID.

Fetch employees by department.

Retrieve employees with a salary greater than a specified amount.

Layers:

Entity: Employee

Controller: /api/employees

Service: Handle business logic for employee operations.

Repository: Use JPA for database queries.

4. Order Management System

Problem Statement:

Create a REST API to manage customer orders. Each order should have the following attributes: id, customerName, product, quantity, orderDate, and status.

Requirements:CRUD Operations:

Add a new order.

Retrieve all orders.

Retrieve a specific order by ID.

Update an existing order's details.

Delete an order record by ID.

Update the status of an order (e.g., Pending, Shipped, Delivered).

Retrieve orders placed by a specific customer.

Layers:

Entity: Order

Controller: /api/orders

Service: Handle order-related logic.

Repository: Use JPA for database interactions.

5. Product Inventory System

Problem Statement:

Design a REST API to manage a product inventory. Each product should have the following attributes: id, name, category, price, stock, and addedDate.

Requirements: CRUD Operations:

Add a new product.

Retrieve all products.

Retrieve a specific product by ID.

Update product details.

Delete a product by ID.

Fetch products by category.

Retrieve products that are out of stock (stock quantity = 0).

Layers:

Entity: Product

Controller: /api/products

Service: Handle inventory-related logic.

Repository: Use JPA for database operations.

Guidelines:

- Students should design the database schema in MySQL and configure application.properties or application.yml for database connectivity.
- Ensure students use appropriate HTTP methods (GET, POST, PUT, DELETE) for their REST APIs.
- Ensure Security - authentication and authorization
- Optionally, ask students to include Swagger for API documentation to enhance their learning