**Scenario**

You have been asked to develop a basic Employee Management System using Spring Boot. This system will handle basic operations like adding, updating, retrieving, and deleting employees. Each employee will have an ID, name, designation, and department.

**Requirements:**

1. **Employee Class**:
   * ID (int)
   * Name (String)
   * Designation (String)
   * Department (String)
2. **Employee Service**:
   * **addEmployee**: Add a new employee.
   * **getEmployeeById**: Retrieve an employee by their ID.
   * **updateEmployee**: Update an existing employee's details.
   * **deleteEmployee**: Delete an employee by their ID.
   * **getAllEmployees**: Retrieve a list of all employees.
3. **Employee Controller**:
   * Provide endpoints for performing CRUD operations using HTTP methods:
     + POST /employees: Add a new employee.
     + GET /employees/{id}: Retrieve an employee by their ID.
     + PUT /employees/{id}: Update an employee by their ID.
     + DELETE /employees/{id}: Delete an employee by their ID.
     + GET /employees: Get a list of all employees.
4. **Employee Repository**:
   * Use an in-memory database like H2 for storing employee data.

**Constraints:**

* Use RESTful APIs for communication.
* Input validations are required (e.g., name and designation should not be null).
* Use an in-memory H2 database.

**Additional Tasks:**

1. Implement a **Search API** to find employees by name.
2. Create a **custom exception** for employee not found.
3. Use **Spring Security** to secure the APIs. Only authenticated users can perform CRUD operations.

**Project Structure:**

css

Copy code

src

│── main

│ └── java

│ └── com.example.employeemanagement

│ ├── controller

│ ├── service

│ ├── repository

│ ├── model

│ └── exception

│── resources

│ └── application.properties