**Exercise 8: Employee Management System - Creating Projections**

Business Scenario:

Create projections to fetch specific data subsets from the employee and department entities.

**1. Introduction**

* **Purpose**: Describe the purpose of the Employee Management System, including managing employee and department data.
* **Scope**: Outline what the system covers, such as CRUD operations for employees and departments, auditing, pagination, sorting, and custom queries.

**2. System Architecture**

* **Overview**: Provide a high-level overview of the system architecture.
* **Components**:
  + **Controllers**: Handle HTTP requests and responses for employees and departments.
  + **Entities**: Define the Employee and Department entities and their attributes.
  + **Repositories**: Interfaces for data access and custom queries.
  + **Configuration**: Spring Boot and JPA configuration, including auditing setup.
  + **Projections**: Define projections for specific data subsets.

**Implementation Details**

* **Entities**:
  + **Employee**:
    - Fields: id, name, email, department, createdDate, lastModifiedDate.
    - Relationships: Many-to-one with Department.
  + **Department**:
    - Fields: id, name, createdDate, lastModifiedDate.
    - Relationships: One-to-many with Employee.
* **Repositories**:
  + **DepartmentRepository**: Provides CRUD operations for Department.
  + **EmployeeRepository**: Provides CRUD operations and custom queries for Employee.
* **Controllers**:
  + **DepartmentController**: Handles requests related to Department.
  + **EmployeeController**: Handles requests related to Employee.
* **Configuration**:
  + **JpaConfig**: Enables JPA auditing.

**6. Features**

* **Auditing**: Tracks creation and modification timestamps for entities.
* **Pagination and Sorting**: Allows for paginated and sorted data retrieval.
* **Custom Queries**: Uses JPQL and method queries to retrieve specific data.
* **Projections**: Fetches specific subsets of data from entities.

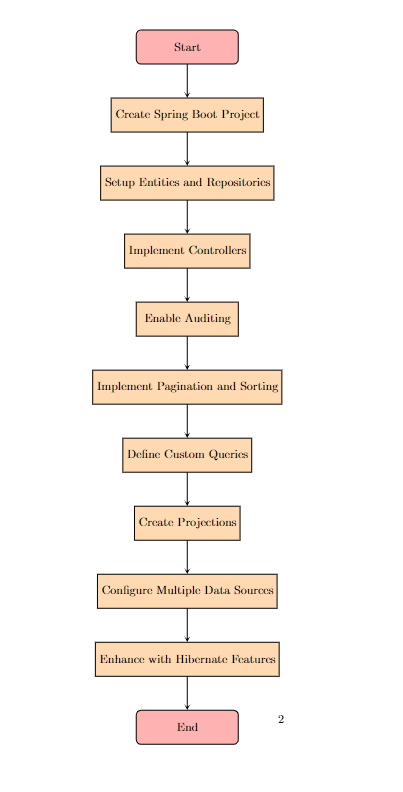
**7. Configuration**

* **application.properties**:
  + Spring Boot application settings.
  + Data source configuration for H2 database.
  + JPA settings, including Hibernate dialect.

**8. Dependencies**

* **Maven Dependencies**:
  + spring-boot-starter-data-jpa: For JPA and Hibernate integration.
  + spring-boot-starter-web: For creating web applications.
  + h2: In-memory database for testing.
  + lombok: To reduce boilerplate code.
  + spring-boot-starter-test: For testing purposes.

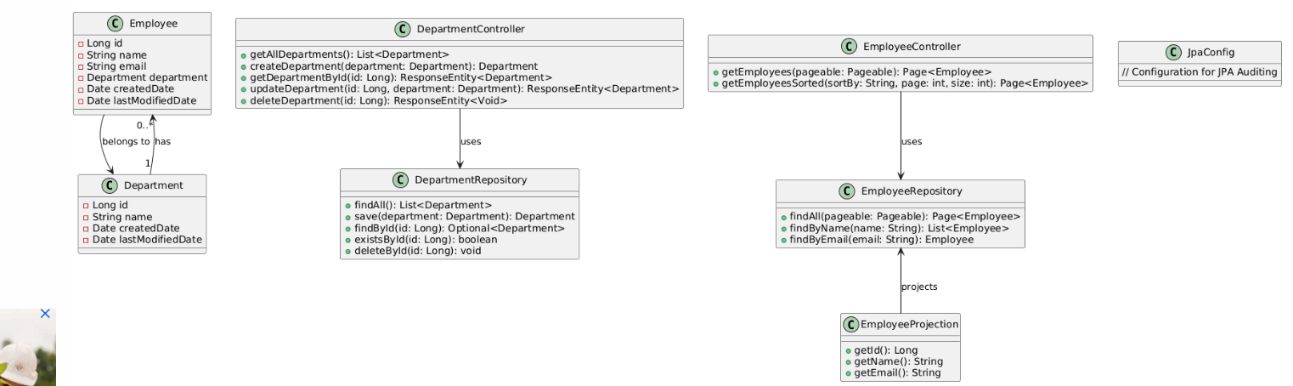
**FLOWCHART :**



**Explanation**

1. **Start**: Represents the beginning of the process.
2. **Create Spring Boot Project**: Setting up the initial Spring Boot project for the Employee Management System.
3. **Setup Entities and Repositories**: Defining the JPA entities (Employee and Department) and their corresponding repositories.
4. **Implement Controllers**: Creating RESTful controllers for handling HTTP requests related to employees and departments.
5. **Enable Auditing**: Adding auditing features to track creation and modification timestamps.
6. **Implement Pagination and Sorting**: Adding pagination and sorting capabilities to the employee search functionality.
7. **Define Custom Queries**: Implementing custom queries using JPQL and query methods.
8. **Create Projections**: Defining projections to fetch specific data subsets.
9. **Configure Multiple Data Sources**: Setting up and configuring multiple data sources if needed.
10. **Enhance with Hibernate Features**: Leveraging Hibernate-specific features for performance improvements.
11. **End**: Represents the completion of the flow.

**CLASS DIAGRAM :**



**Explanation**

* **Classes**:
  + Employee: Represents an employee entity with attributes such as id, name, email, and relationships to Department.
  + Department: Represents a department entity with attributes such as id, name, and auditing timestamps.
  + DepartmentRepository: Provides CRUD operations for Department entities.
  + EmployeeRepository: Provides CRUD operations and custom queries for Employee entities.
  + DepartmentController: Handles HTTP requests related to Department entities.
  + EmployeeController: Handles HTTP requests related to Employee entities.
  + JpaConfig: Configuration class for enabling JPA auditing.
  + EmployeeProjection: Defines a projection for selecting specific subsets of Employee data.
* **Relationships**:
  + Employee class has a many-to-one relationship with the Department class.
  + Department class has a one-to-many relationship with the Employee class.
  + EmployeeController and DepartmentController use their respective repositories.
  + EmployeeProjection is used to create projections for Employee data.