QUESTION 1:

Create a Spring Boot application with two entities: "Teacher" and "TeacherProfile." Each Teacher can have one TeacherProfile, and each TeacherProfile can be assigned to one Teacher. Implement a bidirectional one-to-one mapping between these entities using Spring Data JPA.

Functional Requirements:

Create folders named controller, model, repository, and service inside the WORKSPACE/springapp/src/main/java/com/example/springapp.

Inside the controller folder, create classes named TeacherController and TeacherProfileController.

Inside the model folder, create a class named Teacher with the following attributes:

- teacherId - int (auto-generated primary key)
- name - String
- subject - String
- email - String
- teacherProfile - TeacherProfile (@OneToOne, mappedBy = "teacher", @JsonManagedReference)

Create another class named TeacherProfile with the following attributes:

- profileId - int (auto-generated primary key)
- bio - String
- experience - int
- teacher - Teacher (@OneToOne, @JoinColumn, @JsonBackReference)

Implement getters, setters, and constructors (both parameterized and no-argument) for the Teacher and TeacherProfile entities.

Inside the repository folder, create interfaces named TeacherRepo and TeacherProfileRepo.

Inside the service folder, create interfaces named TeacherService and TeacherProfileService.

Also, create classes TeacherServiceImpl and TeacherProfileServiceImpl which implement TeacherService and TeacherProfileService, respectively.

Project Structure:

```
WORKSPACE/springapp/src/main/java/com/example/springapp
|
├── controller
```

```
│   ├── TeacherController.java
│   └── TeacherProfileController.java
│
├── model
│   ├── Teacher.java
│   └── TeacherProfile.java
│
├── repository
│   ├── TeacherRepo.java
│   └── TeacherProfileRepo.java
│
└── service
    ├── TeacherService.java
    ├── TeacherProfileService.java
    ├── TeacherServiceImpl.java
    └── TeacherProfileServiceImpl.java
```

API ENDPOINTS:

- **POST -** /api/teacher - Returns response status 201 with teacher object on successful creation or else 500. In case of a DuplicateTeacherException, specifically when the email already exists, it returns a status of 409 (CONFLICT) with an appropriate error message as "Teacher with email {email} already exists!".
- **POST -** /api/teacherprofile/teacher/{teacherId} - Returns response status 201 with a teacherProfile object on successfully mapping the teacherProfile to the teacherId or else 500.
- **GET -** /api/teacher/{teacherId} - Returns response status 200 with teacher object, which includes details of teacherProfile on successful retrieval or else 404.
- **GET -** /api/teacher - Returns response status 200 with List<Teacher> object, which includes details of teacherProfile on successful retrieval or else 404.
- **GET -** /api/teacherprofile/experience/{experience} - Returns a response status 200 with List<TeacherProfile> object on successful retrieval based on the specified experience or else 404.

- **DELETE -** /api/teacherprofile/{profileId} - Returns response status 200 with String "Teacher Profile {profileId} deleted successfully" on successful deletion or else "Teacher Profile not found with ID: {profileId}".

QUESTION 2:

Overview:

Create a Spring Boot application with two entities: **Manager** and **Department**. Each Manager can be associated with one Department, and each Department can have one Manager. Implement a unidirectional one-to-one mapping from Manager to Department using Spring Data JPA.

Functional Requirements:

Create folders named controller, model, repository, and service inside WORKSPACE/springapp/src/main/java/com/example/springapp.

Controller Folder: Create classes ManagerController and DepartmentController.

Model Folder: Create classes Manager and Department.

Manager Attributes:

- managerId - Long (auto-generated primary key)
- name - String
- email - String
- department - Department (@OneToOne, @JoinColumn, @JsonBackReference)

Department Attributes:

- departmentId - Long (auto-generated primary key)
- name - String
- location - String
- manager - Manager (@OneToOne, mappedBy = "department", @JsonManagedReference)

Implement getters, setters, and constructors (both parameterized and no-argument) for the Manager and Department entities.

Repository Folder: Create interfaces ManagerRepository and DepartmentRepository.

Service Folder: Create interfaces ManagerService and DepartmentService. Also, create classes ManagerServiceImpl and DepartmentServiceImpl which implement ManagerService and DepartmentService, respectively.

Project Structure:

css

Copy code

```
WORKSPACE/springapp/src/main/java/com/example/springapp
│
├── controller
│   ├── ManagerController.java
│   └── DepartmentController.java
│
├── model
│   ├── Manager.java
│   └── Department.java
│
├── repository
│   ├── ManagerRepository.java
│   └── DepartmentRepository.java
│
└── service
    ├── ManagerService.java
    ├── DepartmentService.java
    ├── ManagerServiceImpl.java
    └── DepartmentServiceImpl.java
```

API Endpoints:

- **POST -** /api/departments - Returns status 201 with the created Manager object on success. In case of a conflict, returns 409 (CONFLICT) with an appropriate error message.
- **POST -** /api/managers/departments/{departmentId} - Returns status 201 with the created Department object on success or 500 in case of an error.
- **GET -** /api/departments/{name} - Returns status 200 with the Department object matching the specified name, or status 404 if no match is found.

- **GET -** /api/departments/{id} - Returns status 200 with the Department object corresponding to the specified ID on success. If not found, returns 404.
- **GET -** /api/managers - Returns status 200 with a list of all Manager objects or 404 if no managers are found.
- **DELETE -** /api/departments/{id} - Returns status 200 with a confirmation message on successful deletion of the Department entity.

QUESTION 3:
Overview:

Create a Spring Boot application with two entities: **Employee** and **ProfileDetails**. Each Employee can have one ProfileDetails, and each ProfileDetails is associated with one Employee. Implement a unidirectional one-to-one mapping from ProfileDetails to Employee using Spring Data JPA.

Functional Requirements:

Create folders named controller, model, repository, and service inside WORKSPACE/springapp/src/main/java/com/example/springapp.

Inside the controller folder, create classes named EmployeeController and ProfileDetailsController.

Inside the model folder, create a class named Employee with the following attributes:

- employeeId - Long (auto-generated primary key)
- name - String
- email - String
- department - String
- profileDetails - ProfileDetails (@OneToOne, @JsonManagedReference)

Create another class named ProfileDetails with the following attributes:

- profileDetailsId - Long (auto-generated primary key)
- employee - Employee (@OneToOne, @JsonBackReference)
- address - String
- phoneNumber - String
- dateOfBirth - LocalDate

Implement getters, setters, and constructors (both parameterized and no-argument) for the Employee and ProfileDetails entities.

Inside the repository folder, create interfaces named EmployeeRepo and ProfileDetailsRepo.

Inside the service folder, create interfaces named EmployeeService and ProfileDetailsService. Also, create classes EmployeeServiceImpl and ProfileDetailsServiceImpl that implement EmployeeService and ProfileDetailsService, respectively.

Project Structure:

css

Copy code

```
WORKSPACE/springapp/src/main/java/com/example/springapp
│
├── controller
│   ├── EmployeeController.java
│   └── ProfileDetailsController.java
│
├── model
│   ├── Employee.java
│   └── ProfileDetails.java
│
├── repository
│   ├── EmployeeRepo.java
│   └── ProfileDetailsRepo.java
│
└── service
    ├── EmployeeService.java
    ├── ProfileDetailsService.java
    ├── EmployeeServiceImpl.java
    └── ProfileDetailsServiceImpl.java
```

API Endpoints:

- **POST -** /api/employees - Creates a new Employee object and returns response status 201 upon successful creation. In case of a conflict, returns status 409 (CONFLICT) with an error message.

- **POST -** /api/profile-details/employees/{employeeId} - Creates a new ProfileDetails object and returns response status 201 upon successful creation or 500 in case of an error.
- **GET -** /api/profile-details/{profileDetailsId} - Retrieves a ProfileDetails object by its profileDetailsId with response status 200 if found, otherwise 404 if not found.
- **DELETE -** /api/profile-details/{profileDetailsId} - Deletes a ProfileDetails by its ID and returns response status 200 with a success message if deleted or 404 if not found.
- **GET -** /api/employees - Returns status 200 with a List<Employee> object on success, or status 500 in case of an error.
- **GET -** /api/employees/by-email - Retrieves an Employee object by email with response status 200 if found or 404 if not found.

QUESTION 4:

Overview:

Create a Spring Boot application with two entities: **Library** and **LibraryDetails**. Each Library can have one LibraryDetails, and each LibraryDetails is associated with one Library. Implement a unidirectional one-to-one mapping from LibraryDetails to Library using Spring Data JPA.

Functional Requirements:

Create folders named controller, model, repository, and service inside WORKSPACE/springapp/src/main/java/com/example/springapp.

Inside the controller folder, create classes named LibraryController and LibraryDetailsController.

Inside the model folder, create a class named Library with the following attributes:

- libraryId - Long (auto-generated primary key)
- libraryName - String
- location - String
- contactInfo - String

Create another class named LibraryDetails with the following attributes:

- libraryDetailId - Long (auto-generated primary key)
- establishedYear - int
- numberOfBooks - int
- openingHours - String

- websiteUrl - String
- library - Library (@OneToOne, @JoinColumn)

Implement getters, setters, and constructors (both parameterized and no-argument) for the Library and LibraryDetails entities.

Inside the repository folder, create interfaces named LibraryRepo and LibraryDetailsRepo.

Inside the service folder, create interfaces named LibraryService and LibraryDetailsService. Also, create classes LibraryServiceImpl and LibraryDetailsServiceImpl which implement LibraryService and LibraryDetailsService, respectively.

Project Structure:

css

Copy code

```
WORKSPACE/springapp/src/main/java/com/example/springapp
│
├── controller
│   ├── LibraryController.java
│   └── LibraryDetailsController.java
│
├── model
│   ├── Library.java
│   └── LibraryDetails.java
│
├── repository
│   ├── LibraryRepo.java
│   └── LibraryDetailsRepo.java
│
└── service
    ├── LibraryService.java
    ├── LibraryDetailsService.java
    ├── LibraryServiceImpl.java
    └── LibraryDetailsServiceImpl.java
```

API Endpoints:

- **POST -** /api/libraries - Returns a 201 response status with the created Library object upon successful creation. If a Library with the same name already exists, throws a DuplicateLibraryException with the message "Library with name {libraryName} already exists!" or a 500 status in the event of an error.
- **POST -** /api/library-details/libraries/{libraryId} - Returns a 201 response status with the created LibraryDetails object if the mapping to the library is successful, or a 500 status in the event of an error.
- **PUT -** /api/libraries/{id} - Returns a 200 response status with the updated Library object upon a successful update. If the library is not found, it returns a 404 status.
- **GET -** /api/library-details/{id} - Returns a 200 response status with the LibraryDetails object corresponding to the specified ID upon successful retrieval. If not found, it returns a 404 status.
- **GET -** /api/library-details - Returns a 200 response status with a list of all LibraryDetails objects upon successful retrieval, or a 404 status if no library details are found.
- **GET -** /api/libraries/name/{libraryName} - Returns a 200 response status with the Library object that matches the specified name upon successful retrieval. If the library is not found, it returns a 404 status.