



Case Study: Library Management System



Objective:

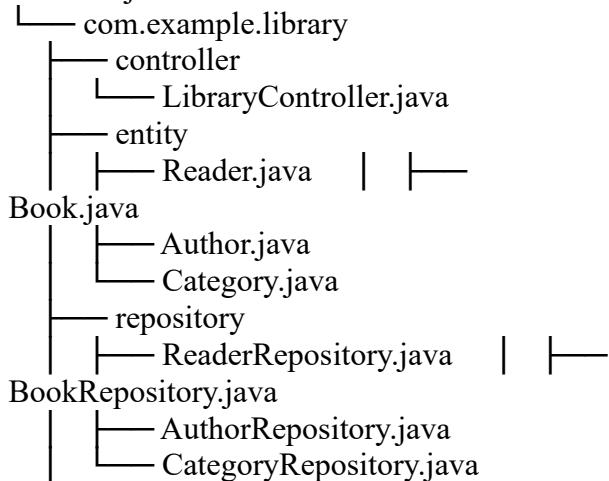
Design a Library Management System where:

- Readers can borrow books
- Books belong to categories • Authors can write multiple books



Folder Structure

src/main/java



SOLUTION:

Step 1: Create MySQL Database

1. Open MySQL Workbench
2. Execute the following SQL:

CREATE DATABASE library_db;

Step 2: Generate Spring Boot Project

- Open Spring Tool Suite (STS)
- Go to: File → New → Spring Starter Project
- Fill in:
 - Name: library-management
 - Group: com.example
 - Artifact: library
 - Type: Maven
 - Java Version: 17
- Click Next, then choose dependencies:
 - Spring Web
 - Spring Data JPA
 - MySQL Driver
 - Lombok
- Click Finish

Step 3: Configure application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/library_db
spring.datasource.username=root
spring.datasource.password=Spoorthib2003$
spring.jpa.hibernate.ddl-auto=update
```

```
spring.jpa.show-sql=true
```

Step 4: Create Entity Classes

```
//Reader.java
com.example.library.entity
import java.util.List;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNullArgsConstructor;
import lombok.RequiredArgsConstructor;
import lombok.Value;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Reader {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String email;

    @OneToMany(mappedBy = "reader")
    private List<Book> books;
}

//Author.java
import java.util.List;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNullArgsConstructor;
import lombok.RequiredArgsConstructor;
import lombok.Value;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    @OneToMany(mappedBy = "author")
    private List<Book> books;
}

// Category.java
```

```
import java.util.List;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    @OneToMany(mappedBy = "category")
    private List<Book> books;
}

//Book.java
import java.util.List;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;
    private LocalDate publishDate;

    @ManyToOne
    private Reader reader;

    @ManyToOne
    private Author author;

    @ManyToOne
    private Category category;
}
```

Step 5: Create Repository Interfaces

com.example.library.repository

//appointmentRepo

```
package com.example.hospital.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.hospital.entity.Appointment;

public interface AppointmentRepo extends JpaRepository<Appointment, Long> {

}

//DoctorRepo

package com.example.hospital.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.hospital.entity.Doctor;

public interface DoctorRepo extends JpaRepository<Doctor, Long> {

}

//MedicalRecordRepo

package com.example.hospital.repository;

import com.example.hospital.entity.MedicalRecord;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface MedicalRecordRepo extends JpaRepository<MedicalRecord, Long> {

    // Custom finder method based on Patient ID
    List<MedicalRecord> findByPatientId(Long patientId);
}

//PatientRepo

package com.example.hospital.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.hospital.entity.Patient;

public interface PatientRepo extends JpaRepository<Patient, Long> {

}
```

Step 6: Create Controller

com.example.library.controller

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.hospital.entity.Appointment;
import com.example.hospital.entity.Doctor;
import com.example.hospital.entity.MedicalRecord;
import com.example.hospital.entity.Patient;
import com.example.hospital.repository.AppointmentRepo;
import com.example.hospital.repository.DoctorRepo;
import com.example.hospital.repository.MedicalRecordRepo;
import com.example.hospital.repository.PatientRepo;

import lombok.RequiredArgsConstructor;

@RestController
@RequestMapping("/api")
@RequiredArgsConstructor
public class LibraryController {

    private final ReaderRepository readerRepo;
    private final BookRepository bookRepo;
    private final AuthorRepository authorRepo;
    private final CategoryRepository categoryRepo;

    @PostMapping("/readers")
    public Reader addReader(@RequestBody Reader reader) {
        return readerRepo.save(reader);
    }

    @PostMapping("/authors")
    public Author addAuthor(@RequestBody Author author) {
        return authorRepo.save(author);
    }

    @PostMapping("/categories")
    public Category addCategory(@RequestBody Category category) {
        return categoryRepo.save(category);
    }

    @PostMapping("/books")
    public Book addBook(@RequestBody Book book) {
```

```

        return bookRepo.save(book);
    }

    @GetMapping("/books")
    public List<Book> getBooks() {
        return bookRepo.findAll();
    }
}

```

Step 7: Run the Application

1. Right-click on LibraryManagementApplication.java
2. Choose Run As → Spring Boot App
3. Check console — it should say Tomcat started on port(s): 8080

Step 8: Test in Postman

POST http://localhost:8080/api/categories
Content-Type: application/json

```
{
  "name": "Fiction"
}
```

POST http://localhost:8080/api/authors

```
{
  "name": "George Orwell"
}
```

POST http://localhost:8080/api/readers

```
{
  "name": "Alice",
  "email": "alice@gmail.com"
}
```

POST http://localhost:8080/api/books

```
{
  "title": "1984",
  "publishDate": "1949-06-08",
  "reader": { "id": 1 },
  "category": { "id": 1 },
  "author": { "id": 1 }
}
```



Case Study Title: Hospital Management System using Spring Boot and Spring Data JPA



1. Overview

The Hospital Management System helps manage patients, doctors, appointments, and medical records. It allows hospital staff to:

- Add/update patient and doctor records
- Schedule appointments
- Track medical history



3. Entity Relationship Diagram (ERD)

Patient (1) ----- (M) Appointment (M) ----- (1) Doctor



4. JPA Entity Class Summary

SOLUTION :

Step 1: Create Database in MySQL Workbench

- Open MySQL Workbench
- Run this SQL:

```
CREATE DATABASE hospitaldb;
```

Step 2: Create Spring Boot Project

- File > New > Spring Starter Project
- Fill in:
 - Name: hospital-management
 - Group: com.example
 - Artifact: hospital
 - Package: com.example.hospital
- Click Next, then add dependencies:
 - Spring Web
 - Spring Data JPA
 - MySQL Driver
 - Lombok
- Finish → Project will be created.
-

Step 3: Configure application.properties

Open src/main/resources/application.properties and add properties

spring.datasource.url=jdbc:mysql://localhost:3306/hospitaldb

spring.datasource.username=root

spring.datasource.password=Spoorthib2003\$

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format_sql=true

Step 4: Create Entity Classes

In com.example.hospital.entity, create:

```
//Patient.java
```

```
import java.util.List;
```

```
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Patient {

    @Id @GeneratedValue
    private Long id;
    private String name;
    private int age;
    private String gender;
    private String address;

    @OneToMany(mappedBy = "patient", cascade = CascadeType.ALL)
    private List<Appointment> appointments;

    @OneToMany(mappedBy = "patient", cascade = CascadeType.ALL)
    private List<MedicalRecord> records;
}

//Doctor.java
import java.util.List;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Doctor {

    @Id @GeneratedValue
    private Long id;
    private String name;
    private String specialization;
    private String email;
    private String phone;

    @OneToMany(mappedBy = "doctor", cascade = CascadeType.ALL)
    private List<Appointment> appointments;
}
```

```
//Appointment.java
import java.util.List;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDate;
import java.time.LocalTime;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Appointment {
    @Id @GeneratedValue
    private Long id;
    private LocalDate date;
    private LocalTime time;
    private String notes;

    @ManyToOne
    private Patient patient;

    @ManyToOne
    private Doctor doctor;
}
```

```
//MedicalRecord.java
import java.util.List;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MedicalRecord {
    @Id @GeneratedValue
```

```

private Long id;
private String diagnosis;
private String treatment;
private LocalDate date;

@ManyToOne
private Patient patient;
}

```

Step 5: Create Repository Interfaces

In com.example.hospital.repository, create:

```

public interface PatientRepository extends JpaRepository<Patient, Long> {}
public interface DoctorRepository extends JpaRepository<Doctor, Long> {}
public interface AppointmentRepository extends JpaRepository<Appointment, Long> {}
public interface MedicalRecordRepository extends JpaRepository<MedicalRecord, Long> {}

```

Step 6: Create Controller Class

In com.example.hospital.controller, create:

```

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.hospital.entity.Appointment;
import com.example.hospital.entity.Doctor;
import com.example.hospital.entity.MedicalRecord;
import com.example.hospital.entity.Patient;
import com.example.hospital.repository.AppointmentRepo;
import com.example.hospital.repository.DoctorRepo;
import com.example.hospital.repository.MedicalRecordRepo;
import com.example.hospital.repository.PatientRepo;

import lombok.RequiredArgsConstructor;

@RestController
@RequestMapping("/api")
@RequiredArgsConstructor
public class HospitalController {

    private final PatientRepository patientRepo;
    private final DoctorRepository doctorRepo;
    private final AppointmentRepository appointmentRepo;
    private final MedicalRecordRepository medicalRecordRepo;
}

```

```

@PostMapping("/patients")
public Patient addPatient(@RequestBody Patient patient) {
    return patientRepo.save(patient);
}

@GetMapping("/patients")
public List<Patient> getAllPatients() {
    return patientRepo.findAll();
}

@PostMapping("/doctors")
public Doctor addDoctor(@RequestBody Doctor doctor) {
    return doctorRepo.save(doctor);
}

@PostMapping("/appointments")
public Appointment bookAppointment(@RequestBody Appointment appointment) {
    return appointmentRepo.save(appointment);
}

@GetMapping("/appointments")
public List<Appointment> getAppointments() {
    return appointmentRepo.findAll();
}

@PostMapping("/medical-records")
public MedicalRecord addRecord(@RequestBody MedicalRecord record) {
    return medicalRecordRepo.save(record);
}

@GetMapping("/patients/{id}/records")
public List<MedicalRecord> getPatientRecords(@PathVariable Long id) {
    Patient patient = patientRepo.findById(id).orElseThrow();
    return patient.getRecords();
}

```

Step 7: Run the Application

- Right-click project → Run As → Spring Boot App
- App should start on <http://localhost:8080>

Step 8: Test APIs in Postman

POST <http://localhost:8080/api/patients>

json

CopyEdit

```
{
  "name": "John Doe",
  "age": 35,
```

```
"gender": "Male",
"address": "123 Main Street"
}
```

Add Doctor

POST http://localhost:8080/api/doctors

json

CopyEdit

```
{
  "name": "Dr. Smith",
  "specialization": "Cardiologist",
  "email": "drsmith@example.com",
  "phone": "9876543210"
}
```

Book Appointment

POST http://localhost:8080/api/appointments

json

CopyEdit

```
{
  "date": "2025-08-03",
  "time": "10:00:00",
  "notes": "Follow-up",
  "patient": { "id": 1 },
  "doctor": { "id": 1 }
}
```

Add Medical Record

POST http://localhost:8080/api/medical-records

json

CopyEdit

```
{
  "diagnosis": "Hypertension",
  "treatment": "Medication",
  "date": "2025-08-03",
  "patient": { "id": 1 }
}
```

View Patient Records

GET http://localhost:8080/api/patients/1/records