**Assignment 05**

# Movie Ticket Management System - Spring Boot

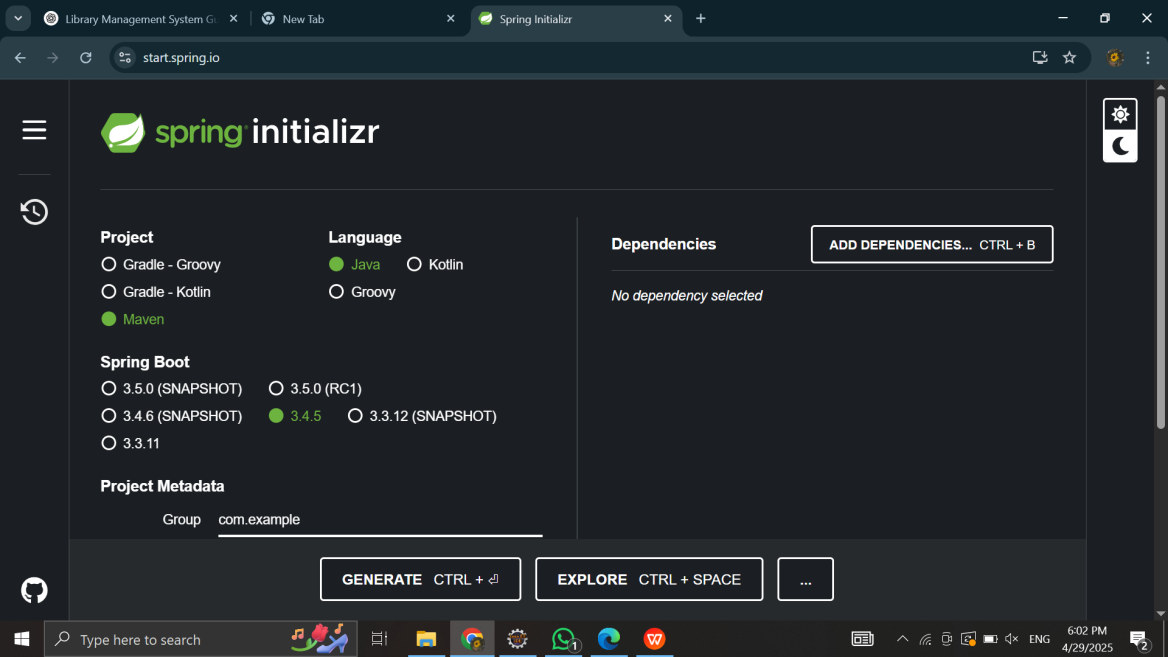
# Step 1: Create Project Using Spring Initializr

**Step 1: Create Project Using Spring Initializr**

Go to: <https://start.spring.io>  
Choose:

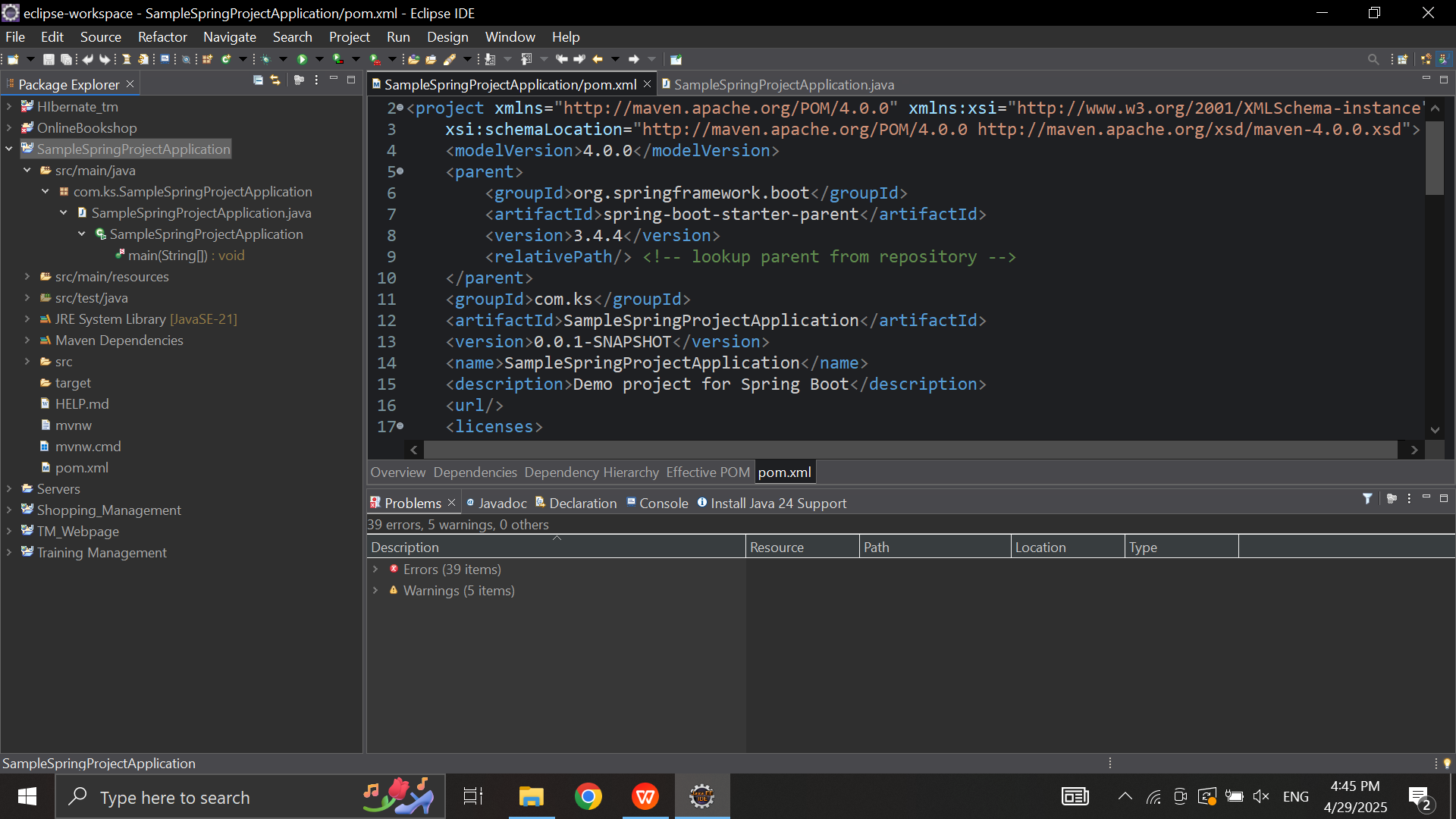
* **Project:** Maven
* **Language:** Java
* **Spring Boot Version:** Latest stable
* **Group:** com.example
* **Artifact:** movieticketsystem
* **Name:** MovieTicketManagementSystem
* **Packaging:** Jar
* **Java:** 17 or above
* **Dependencies:**
  + Spring Web
  + Spring Data JPA
  + MySQL Driver
  + Spring Boot DevTools (optional)

Click on **Generate** → Extract the ZIP file.

)  
Click on Generate → Extract the ZIP file.  


# Step 2: Import the Project

# Open IDE → File > Import > Maven > Existing Maven Projects → Select the extracted folder → Finish



# Step 3: Configure pom.xml dependencies

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

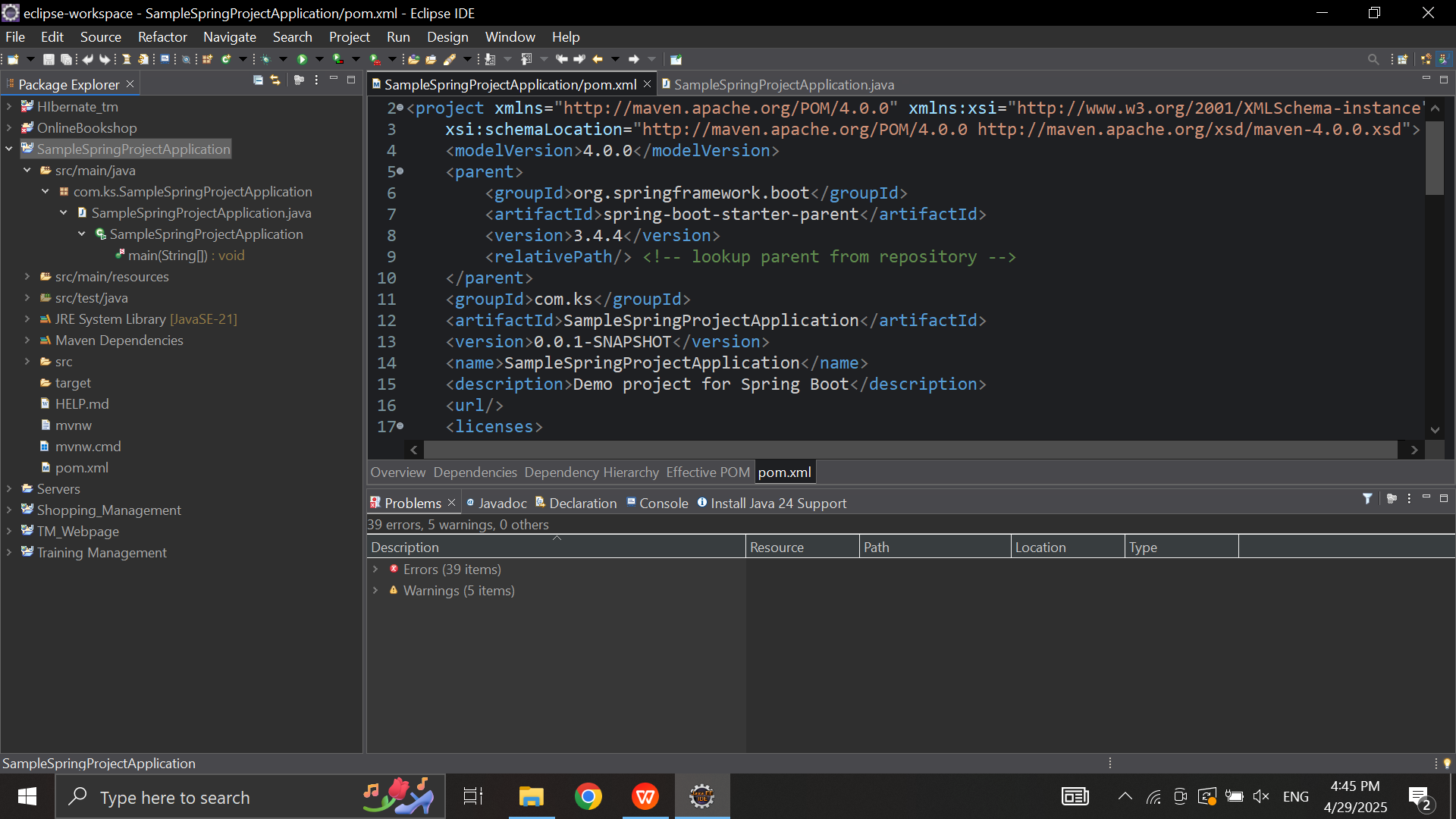
<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>



# Step 4: Create Model (Entity) Class — Train.java

package com.example.movieticketsystem.model;

import jakarta.persistence.\*;

@Entity

public class Ticket {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String customerName;

private String movieName;

private String seatNumber;

private String showTime;

private String theater;

// Getters & Setters

}  
}

# Step 5: Create Repository — TrainRepository.java

# package com.example.movieticketsystem.repository;

# import com.example.movieticketsystem.model.Ticket;

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

# import org.springframework.stereotype.Repository;

# @Repository

# public interface TicketRepository extends JpaRepository<Ticket, Long> {

# }Step 6: Create Service Layer — TrainService.java

# package com.example.movieticketsystem.service;

# import com.example.movieticketsystem.model.Ticket;

# import com.example.movieticketsystem.repository.TicketRepository;

# import org.springframework.beans.factory.annotation.Autowired;

# import org.springframework.stereotype.Service;

# import java.util.List;

# @Service

# public class TicketService {

# @Autowired

# private TicketRepository ticketRepo;

# public List<Ticket> getAllTickets() {

# return ticketRepo.findAll();

# }

# public Ticket addTicket(Ticket ticket) {

# return ticketRepo.save(ticket);

# }

# public void deleteTicket(Long id) {

# ticketRepo.deleteById(id);

# }

# }Step 7: Create Controller — TrainController.java

package com.example.movieticketsystem.controller;

import com.example.movieticketsystem.model.Ticket;

import com.example.movieticketsystem.service.TicketService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/tickets")

public class TicketController {

@Autowired

private TicketService ticketService;

@GetMapping

public List<Ticket> getTickets() {

return ticketService.getAllTickets();

}

@PostMapping

public Ticket addTicket(@RequestBody Ticket ticket) {

return ticketService.addTicket(ticket);

}

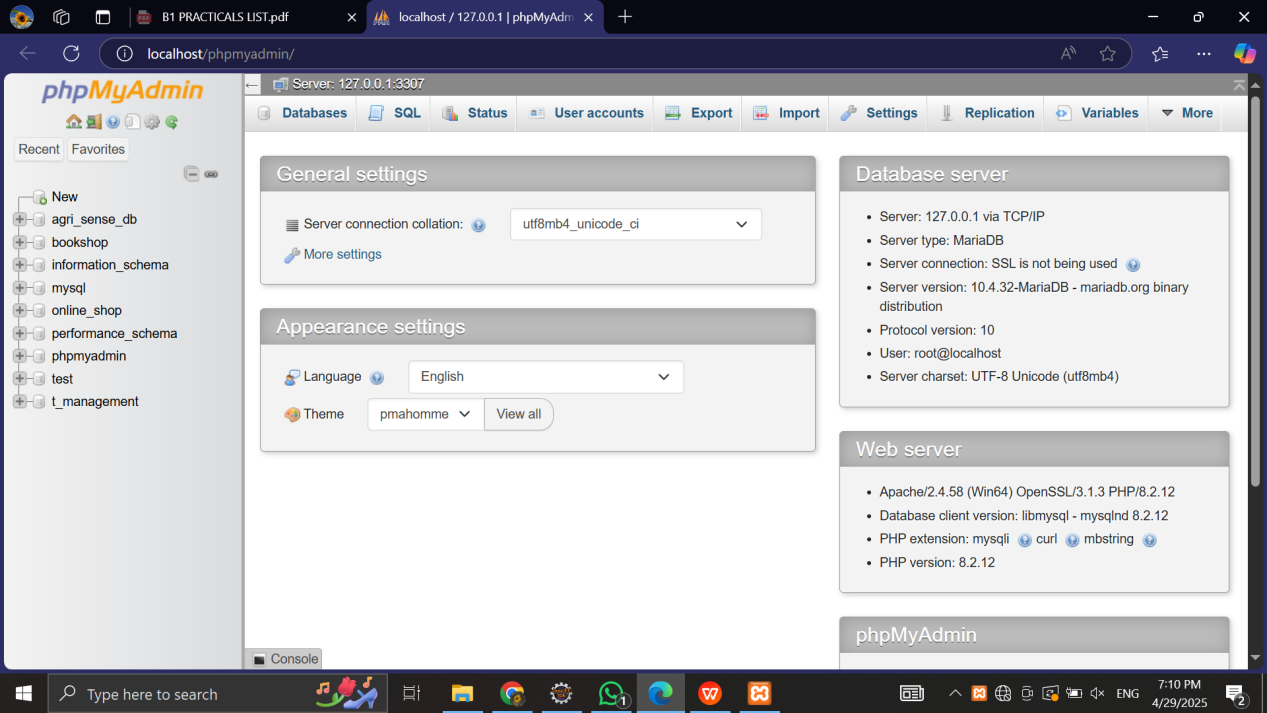
@DeleteMapping("/{id}")

public void deleteTicket(@PathVariable Long id) {

ticketService.deleteTicket(id);

}

}eleteTrain(@PathVariable Long id) {  
 trainService.deleteTrain(id);  
 }  
}



# Step 8: Configure MySQL Connection in application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/railwaydb  
spring.datasource.username=root  
spring.datasource.password=your\_password  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
  
Note: Create the database in MySQL before running the project:  
CREATE DATABASE railwaydb;

# Step 9: Run the Application

# Right-click on MovieTicketManagementSystemApplication.java → Run As → Java Application

# Test Cases

| **Test Case ID** | **Test Scenario** | **Input** | **Expected Output** |
| --- | --- | --- | --- |
| TC01 | Get all tickets | GET /tickets | List of all tickets |
| TC02 | Add new ticket | POST /tickets with JSON {customerName, movieName...} | Ticket with generated ID |
| TC03 | Delete ticket | DELETE /tickets/1 | Deletes ticket with ID 1 |
| TC04 | Empty ticket list | GET /tickets | Returns [] |
| TC05 | Add ticket with missing data | POST /tickets with missing fields | 400 Bad Request |