**WEEK - 4**

**Spring REST using Spring Boot 3**

**Exercise 1: Create a Spring Web Project using Maven**

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

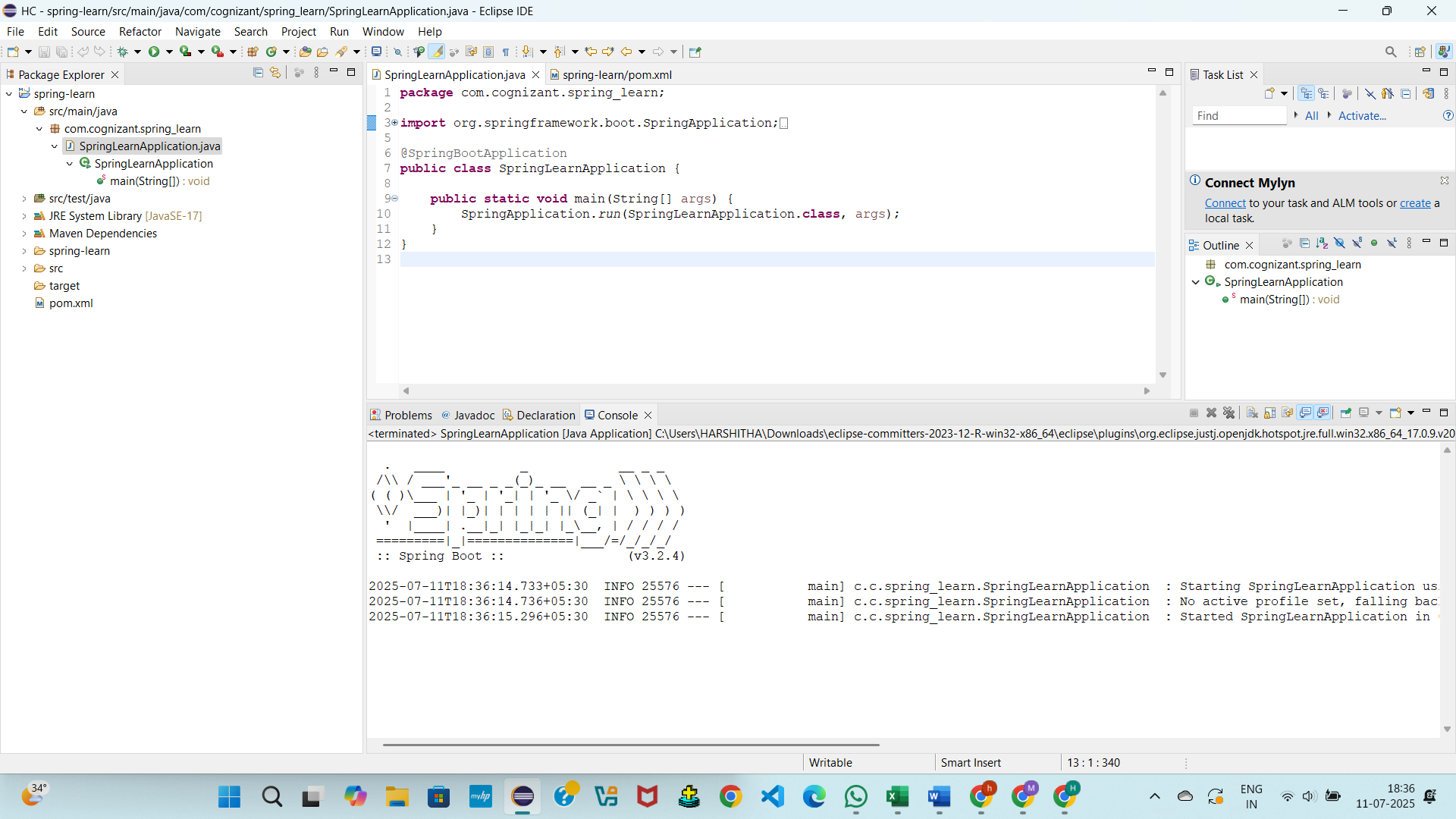
public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**OUTPUT:**

****

**Exercise 2: Spring Core – Load Country from Spring Configuration XML**

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) throws Exception {

SpringApplication.run(SpringLearnApplication.class, args);

displayDate();

}

public static void displayDate() throws Exception {

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

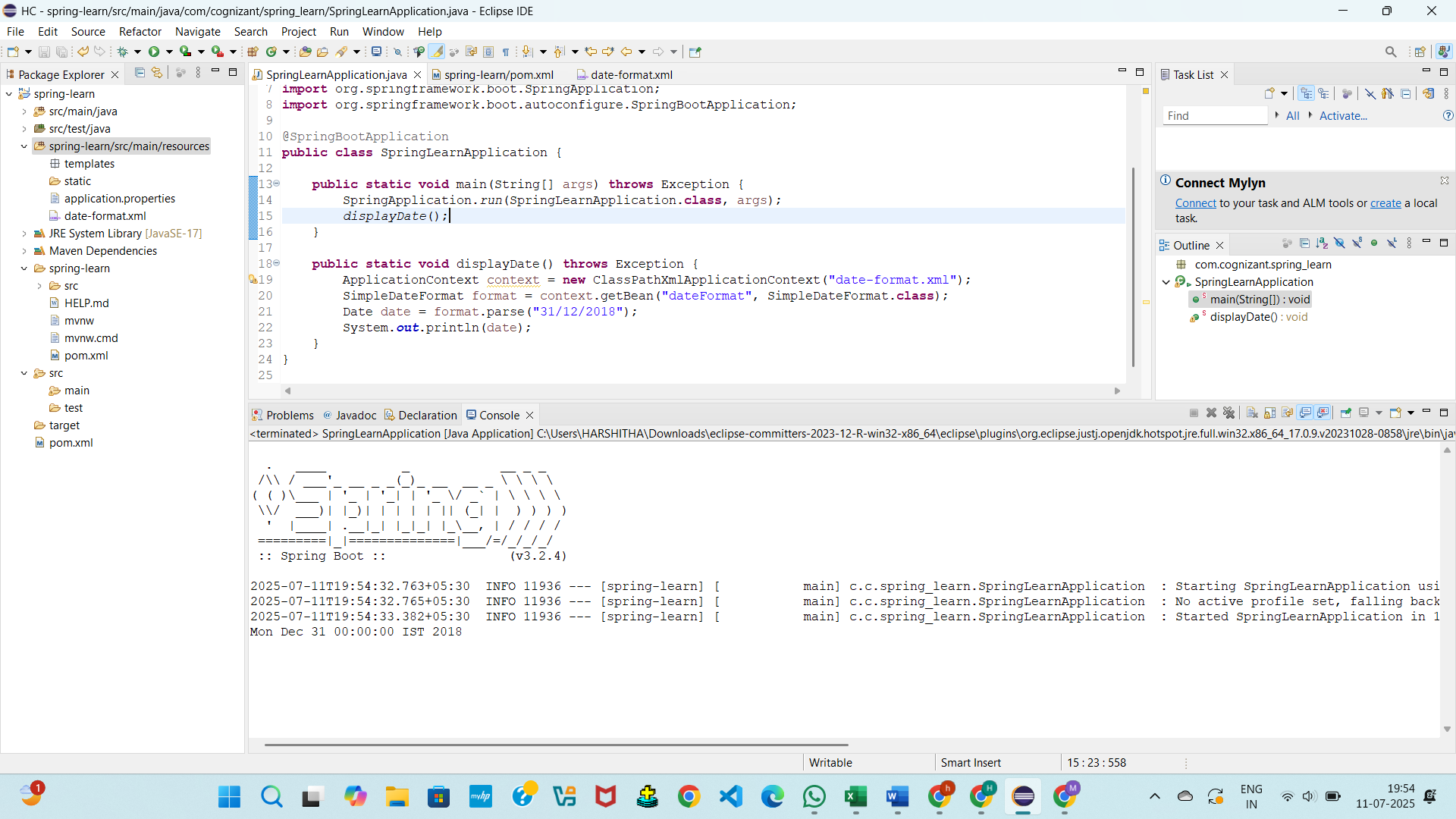
Date date = format.parse("31/12/2018");

System.out.println(date);

}

}

**OUTPUT:**

****

**Exercise 3: Hello World RESTful Web Service**

**HelloController.java**package com.cognizant.spring\_learn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {

private static final Logger LOGGER = LoggerFactory.getLogger(HelloController.class);

@GetMapping("/hello")

public String sayHello() {

LOGGER.info("START sayHello()");

String response = "Hello World!!";

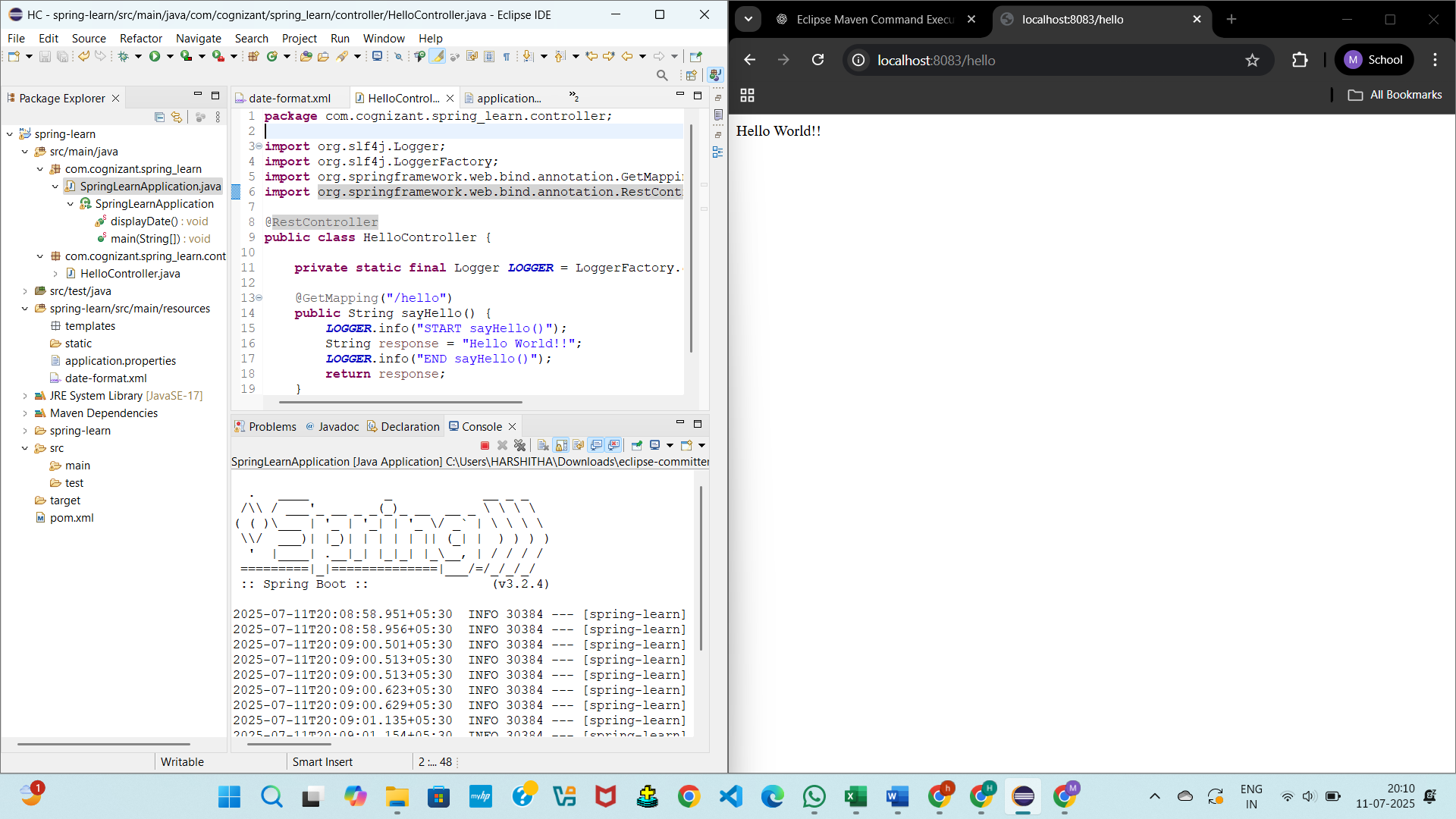
LOGGER.info("END sayHello()");

return response;

}

}

**OUTPUT:**

****

**Exercise 4: REST - Country Web Service**

**CountryController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class CountryController {

@RequestMapping("/country")

public Country getCountryIndia() {

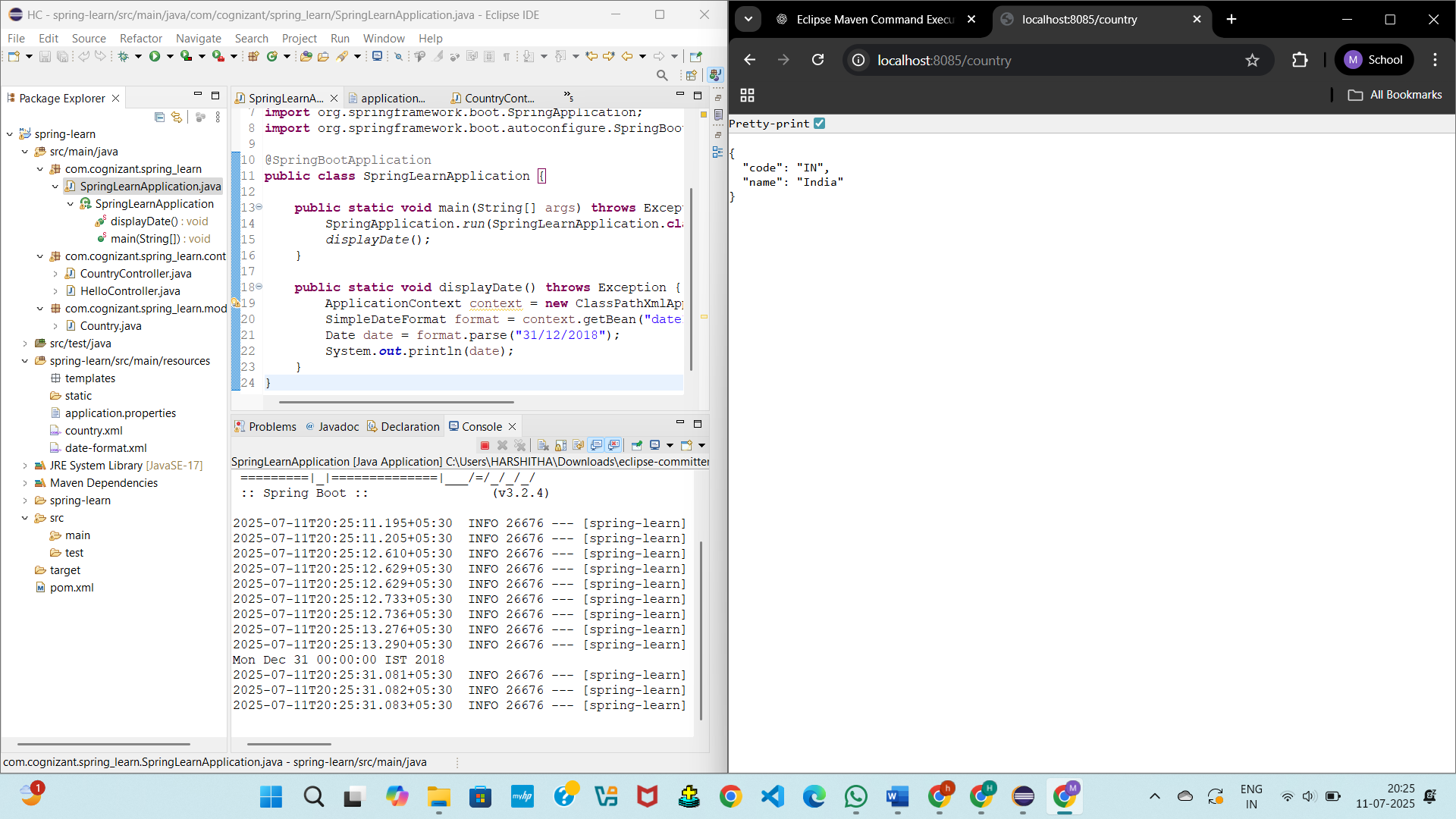
ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

return (Country) context.getBean("in");

}

}

**OUTPUT:**

****

**Exercise 5: REST - Get country based on country code**

**CountryService.java**

package com.cognizant.spring\_learn.service;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import com.cognizant.spring\_learn.model.Country;

@Service

public class CountryService {

public Country getCountry(String code) {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

List<Country> countryList = (List<Country>) context.getBean("countryList");

return countryList.stream().filter(country -> country.getCode().equalsIgnoreCase(code))

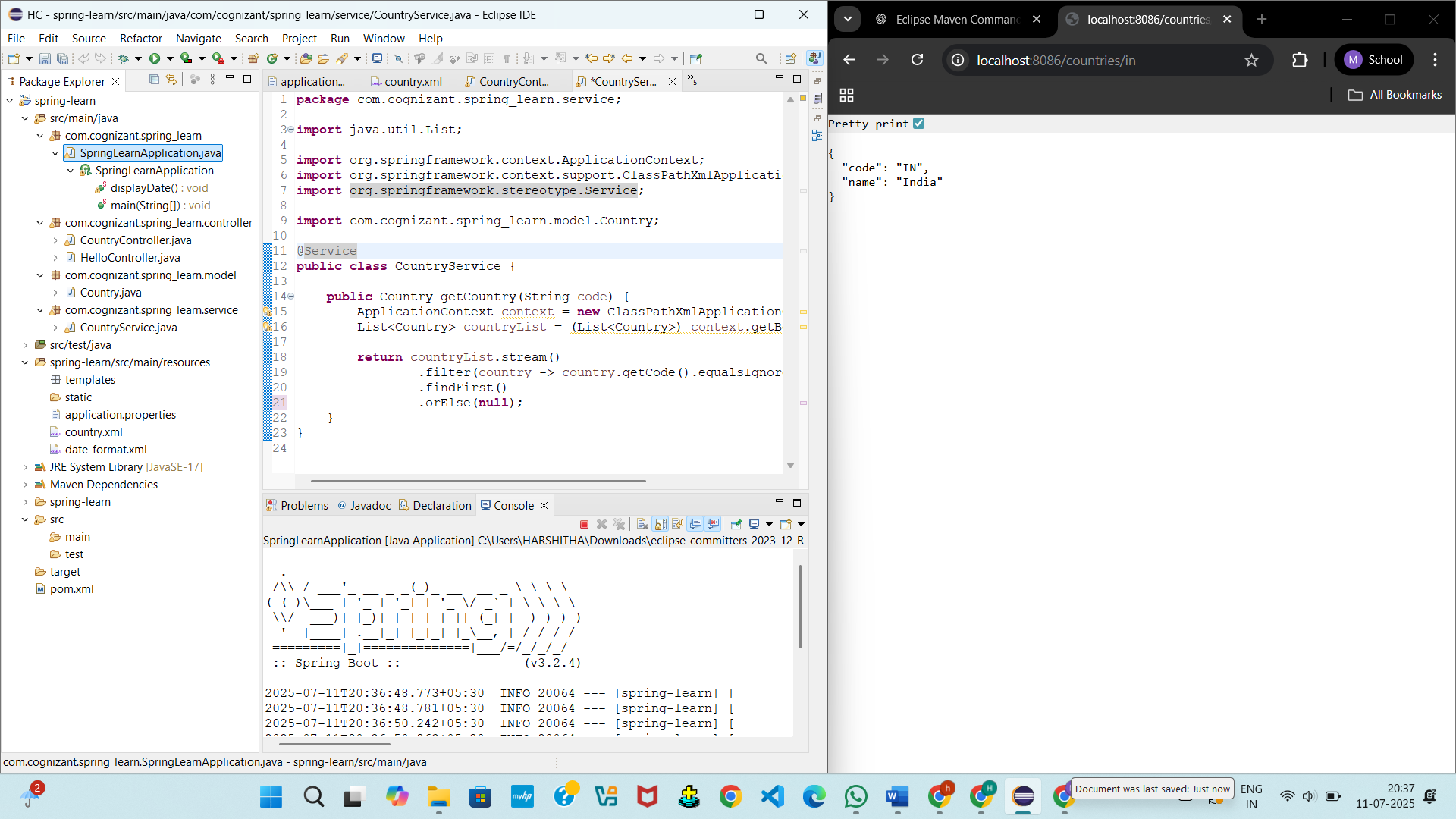
.findFirst()

.orElse(null);

}

}

**OUTPUT:**

****

**Exercise 6: Create authentication service that returns JWT**

**AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import jakarta.servlet.http.HttpServletRequest;

@RestController

public class AuthenticationController {

private static final String SECRET\_KEY = "mySecretKey123"; // should be stronger & stored safely

@RequestMapping(value = "/authenticate", method = RequestMethod.GET)

public Map<String, String> authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

if (authHeader == null || !authHeader.startsWith("Basic ")) {

throw new RuntimeException("Missing or invalid Authorization header");

}

String base64Credentials = authHeader.substring("Basic ".length());

byte[] credDecoded = Base64.getDecoder().decode(base64Credentials);

String credentials = new String(credDecoded);

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

if (!username.equals("user") || !password.equals("pwd")) {

throw new RuntimeException("Invalid username or password");

}

String token = Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 60 \* 60 \* 1000)) // 1 hour

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

Map<String, String> response = new HashMap<>();

response.put("token", token);

return response;

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic(httpBasic -> {})

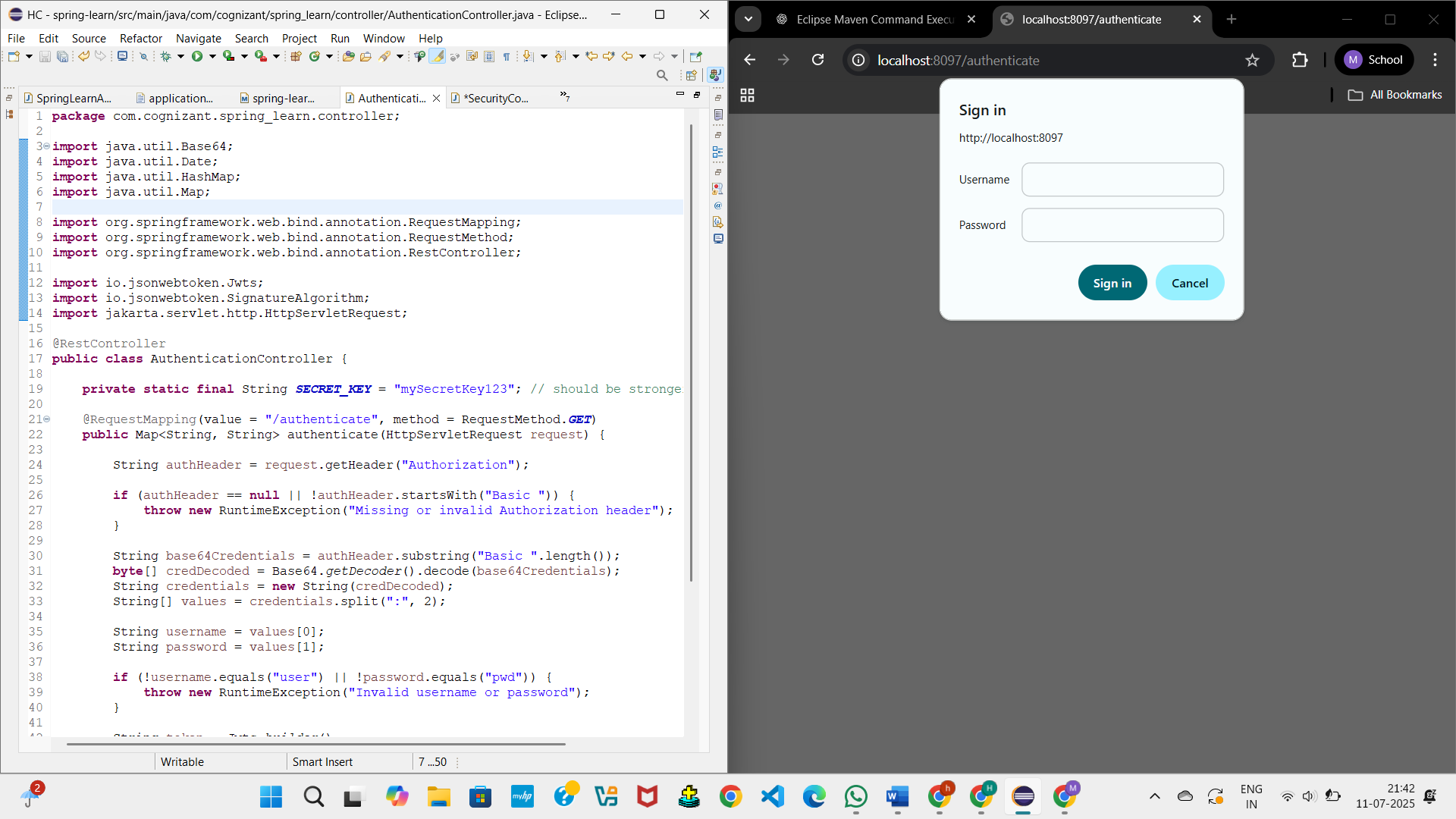
.csrf(csrf -> csrf.disable());

return http.build();

}

}

**OUTPUT:**

****