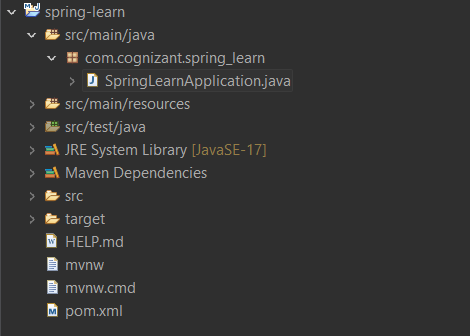
**COGNIZANT - DIGITALNURTURE4.0**

**DEEPSKILLING JAVA FSE**

**WEEK 01: SPRING REST USING SPRING BOOT 3**

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



**//SpringApplication.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) {

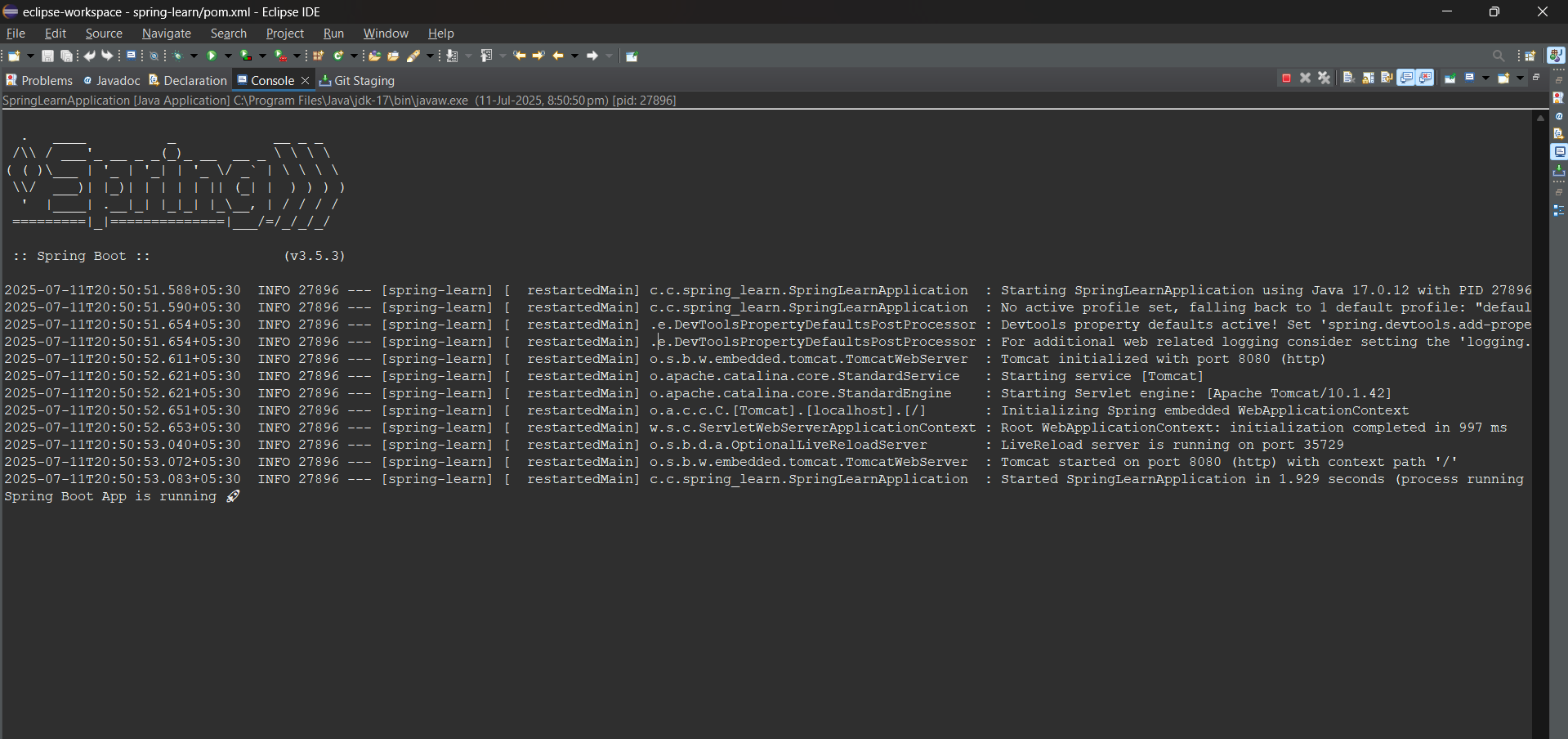
SpringApplication.run(SpringLearnApplication.class, args);

System.**out**.println("Spring Boot App is running 🚀");

}

}

**Output:**



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

**//SpringLearnApplicationn**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplicationn {

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

public static void main(String[] args) {

System.out.println("Main method running...");

LOGGER.debug("START of main()");

displayCountry();

LOGGER.debug("END of main()");

}

public static void displayCountry() {

try {

// Diagnostic print to check exact location

java.net.URL url = SpringLearnApplicationn.class.getClassLoader().getResource("country.xml");

System.out.println("country.xml URL = " + (url != null ? url.toString() : "NOT FOUND"));

// Load Spring context from XML

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

Country country = context.getBean("country", Country.class);

LOGGER.debug("Country : {}", country.toString());

context.close();

} catch (Exception e) {

System.out.println(" Error loading context or bean: " + e);

e.printStackTrace();

}

}

}

**//country.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="country" class="com.cognizant.spring\_learn.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

</beans>

**Output:**

Main method running...

DEBUG com.cognizant.spring\_learn.SpringLearnApplication - START of main()

DEBUG com.cognizant.spring\_learn.Country - Inside Country Constructor.

DEBUG com.cognizant.spring\_learn.Country - Setting country code to: IN

DEBUG com.cognizant.spring\_learn.Country - Setting country name to: India

DEBUG com.cognizant.spring\_learn.SpringLearnApplication - Country : Country{code='IN', name='India'}

DEBUG com.cognizant.spring\_learn.SpringLearnApplication - END of main()

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

**//SpringApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.Collections;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication app = new SpringApplication(SpringLearnApplication.class);

app.setDefaultProperties(Collections.singletonMap("server.port", "8083"));

app.run(args);

}

}

//spring-learn.properties

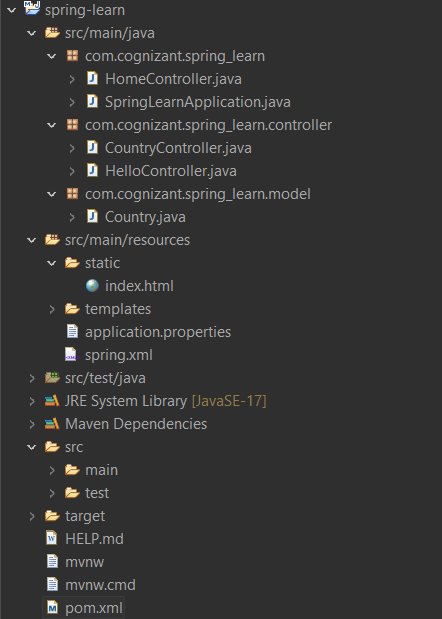
server.port=8083

spring.application.name=spring-learn

**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.beans.factory.BeanFactory;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

@RestController

public class CountryController {

private BeanFactory context;

@GetMapping("/country")

public Country getCountryIndia() {

context = new ClassPathXmlApplicationContext("spring.xml");

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

}

}

**//Country.java**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**//pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-learn</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<description>Spring Boot learning project</description>

<parent>

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

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

<version>3.2.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>20</java.version>

</properties>

<dependencies>

<!-- Add your app-specific dependencies here -->

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

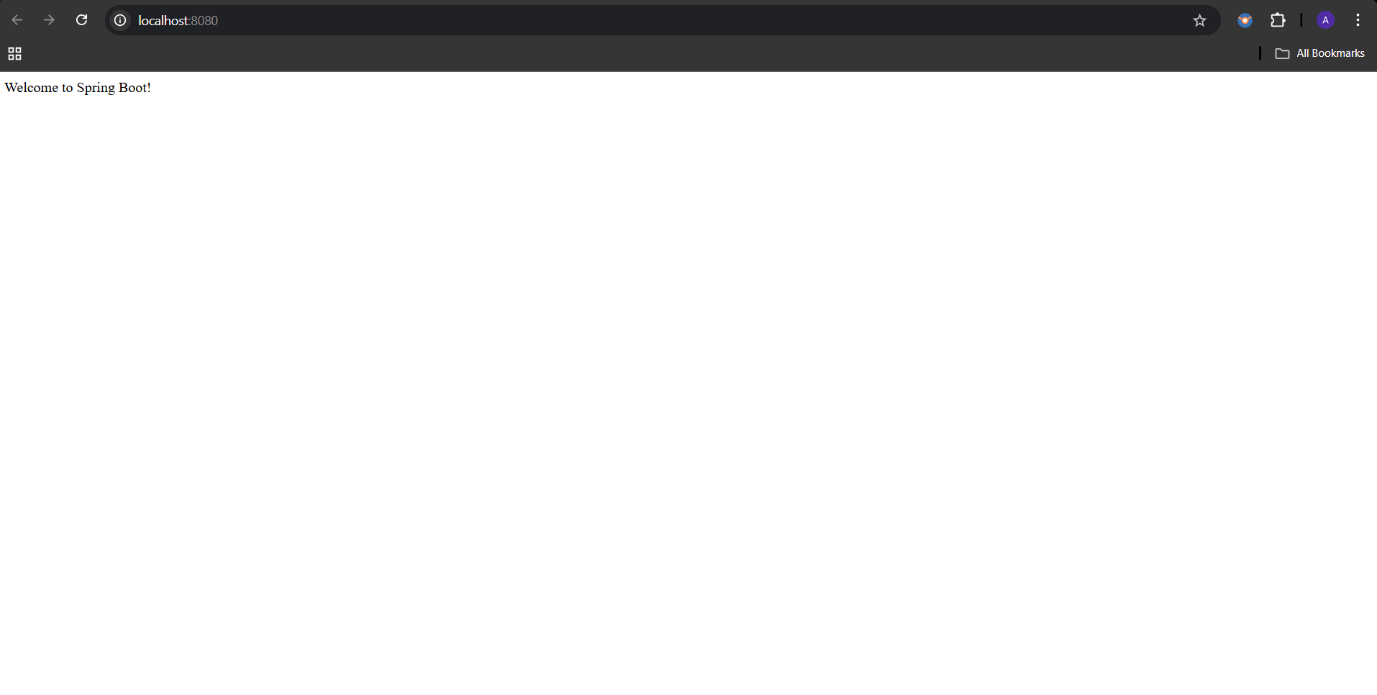
</plugin>

</plugins>

</build>

</project>

**Output:**

****

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

**//CountryService.java**

package com.cognizant.spring\_learn.service;

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

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

import jakarta.annotation.PostConstruct;

import org.springframework.core.io.ClassPathResource;

import org.springframework.oxm.jaxb.Jaxb2Marshaller;

import org.springframework.stereotype.Service;

import javax.xml.transform.stream.StreamSource;

import java.util.List;

*@Service*

public class CountryService {

private List<Country> countryList;

*@PostConstruct*

public void loadCountries() throws Exception {

Jaxb2Marshaller marshaller = new Jaxb2Marshaller();

marshaller.setClassesToBeBound(Countries.class);

marshaller.afterPropertiesSet(); //

// Load XML from classpath

ClassPathResource resource = new ClassPathResource("country.xml");

if (!resource.exists()) {

throw new RuntimeException("country.xml not found in classpath!");

}

// Use StreamSource with systemId

StreamSource source = new StreamSource(resource.getInputStream());

source.setSystemId(resource.getURL().toString());

Countries countries = (Countries) marshaller.unmarshal(source);

countryList = countries.getCountryList();

}

public Country getCountry(String code) {

return countryList.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElse(null);

}

}

**//Countries.java**

package com.cognizant.spring\_learn.model;

import java.util.List;

import jakarta.xml.bind.annotation.XmlRootElement;

import jakarta.xml.bind.annotation.XmlAccessorType;

import jakarta.xml.bind.annotation.XmlAccessType;

import jakarta.xml.bind.annotation.XmlElement;

*@XmlRootElement*(name = "countries")

*@XmlAccessorType*(*XmlAccessType*.***FIELD***)

public class Countries {

*@XmlElement*(name = "country")

private List<Country> countryList;

public List<Country> getCountryList() {

return countryList;

}

public void setCountryList(List<Country> countryList) {

this.countryList = countryList;

}

// Optional: Default constructor (JAXB requires it)

public Countries() {

}

// Optional: toString for debugging

*@Override*

public String toString() {

return "Countries{" +

"countryList=" + countryList +

'}';

}

}

**//Country.java**

package com.cognizant.spring\_learn.model;

import jakarta.xml.bind.annotation.XmlRootElement;

import jakarta.xml.bind.annotation.XmlAccessorType;

import jakarta.xml.bind.annotation.XmlAccessType;

import jakarta.xml.bind.annotation.XmlElement;

*@XmlRootElement*(name = "country")

*@XmlAccessorType*(*XmlAccessType*.***FIELD***)

public class Country {

*@XmlElement*(name = "code")

private String code;

*@XmlElement*(name = "name")

private String name;

// Default constructor required for JAXB

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

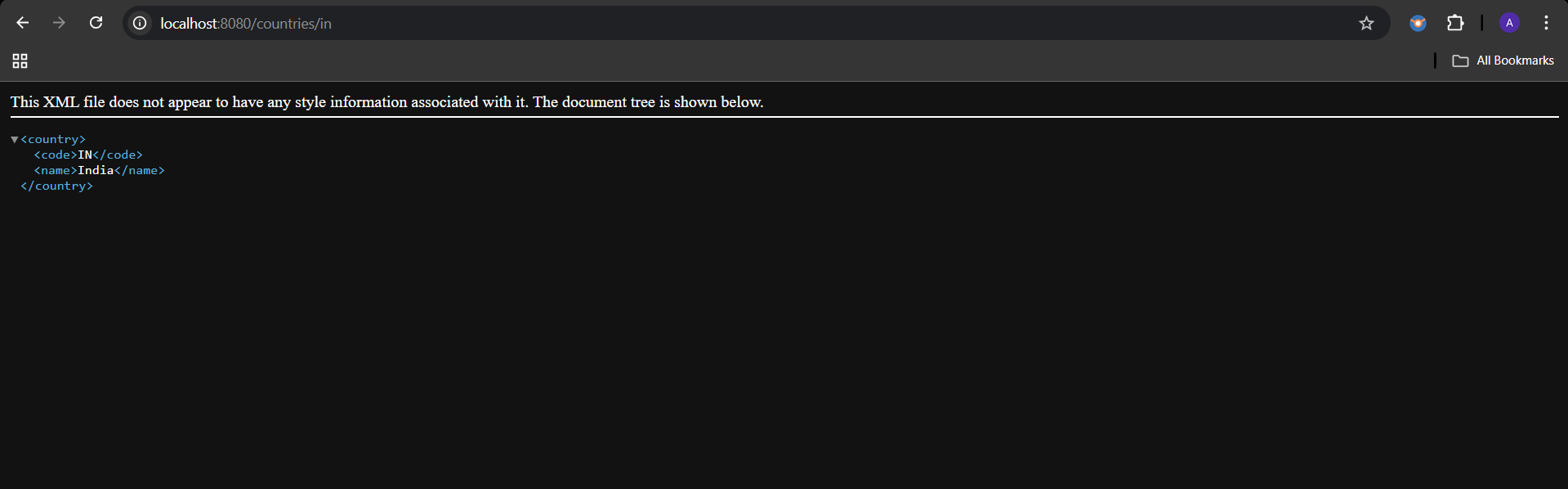
public void setName(String name) {

this.name = name;

}

}

**Output:**



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

**//AuthenticationController**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.util.JwtUtil;

import org.springframework.http.ResponseEntity;

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

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

return ResponseEntity.status(401).body("Missing or invalid Authorization header");

}

try {

// Decode base64 username:password

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

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

String credentials = new String(decoded);

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

if (values.length != 2) {

return ResponseEntity.status(401).body("Invalid Authorization format");

}

String username = values[0];

String password = values[1];

// Simple validation

if ("user".equals(username) && "pwd".equals(password)) {

String token = JwtUtil.generateToken(username);

// Proper JSON response

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

response.put("token", token);

return ResponseEntity.ok(response);

} else {

return ResponseEntity.status(401).body("Invalid credentials");

}

} catch (Exception e) {

return ResponseEntity.status(500).body("Error decoding credentials");

}

}

}

**//JwtUtil.java**

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class JwtUtil {

private static final String SECRET\_KEY = "secretkey123";

public static String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**//SecurityConfig.java**

package config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

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

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

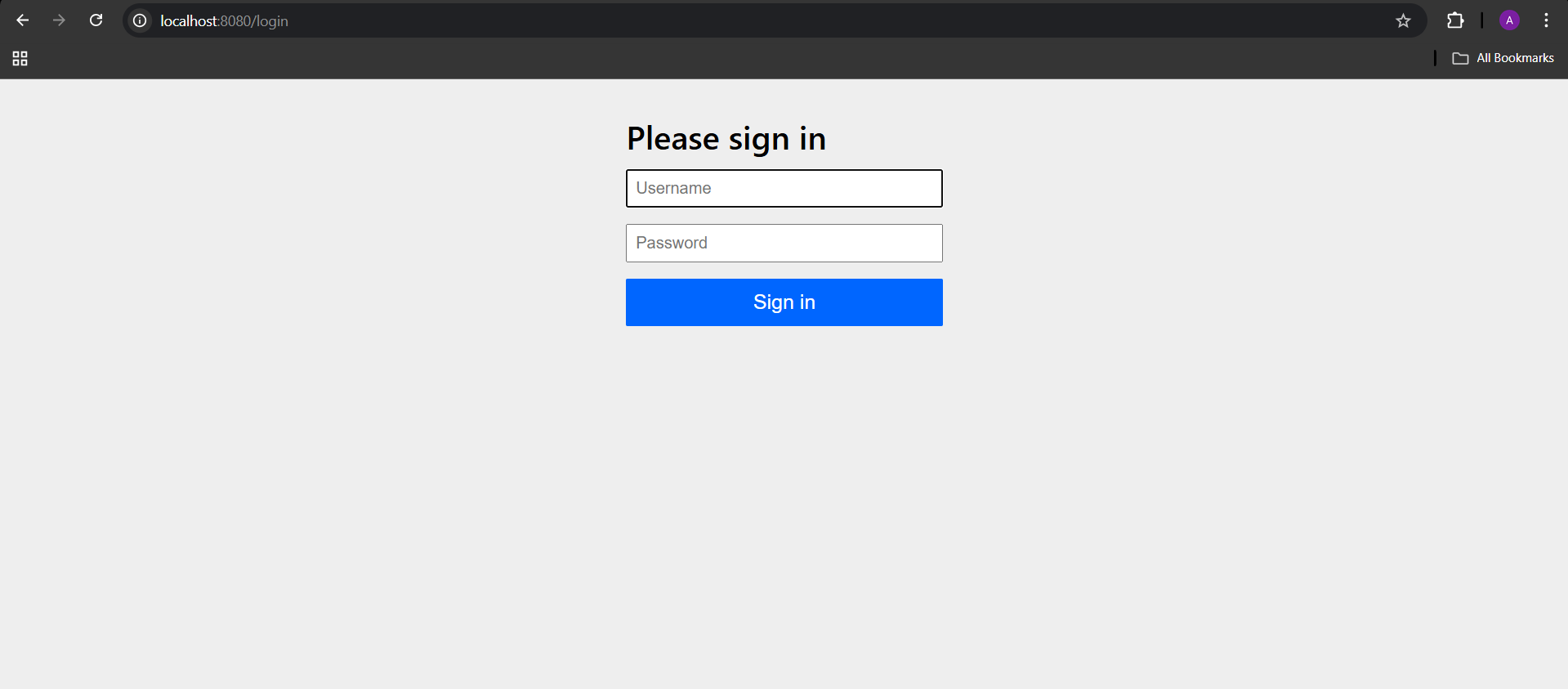
.httpBasic(Customizer.*withDefaults*()); // Enables Basic Auth

return http.build();

}

}

**Output:**



**NAME: HARINI G  
SUPERSET ID: 6385037**