**5.JWT-HandsOn**

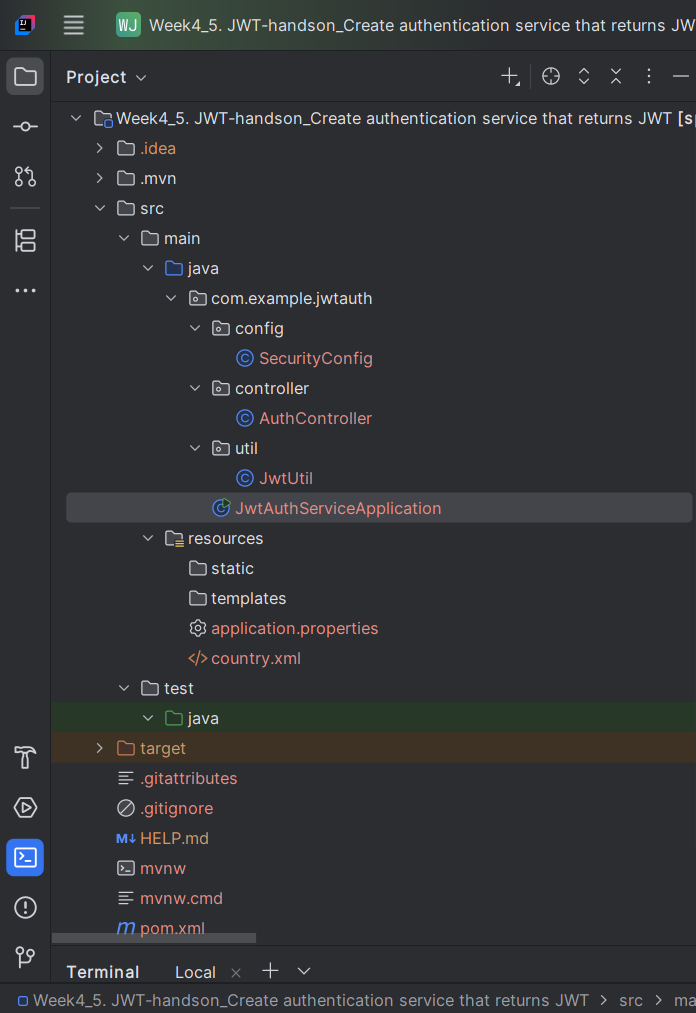
**Create authentication service that returns JWT   
  
As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.  
  
Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.  
  
Request**

**curl -s -u user:pwd http://localhost:8090/authenticate**

**Response**

**{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}**

**Folder Structure:**

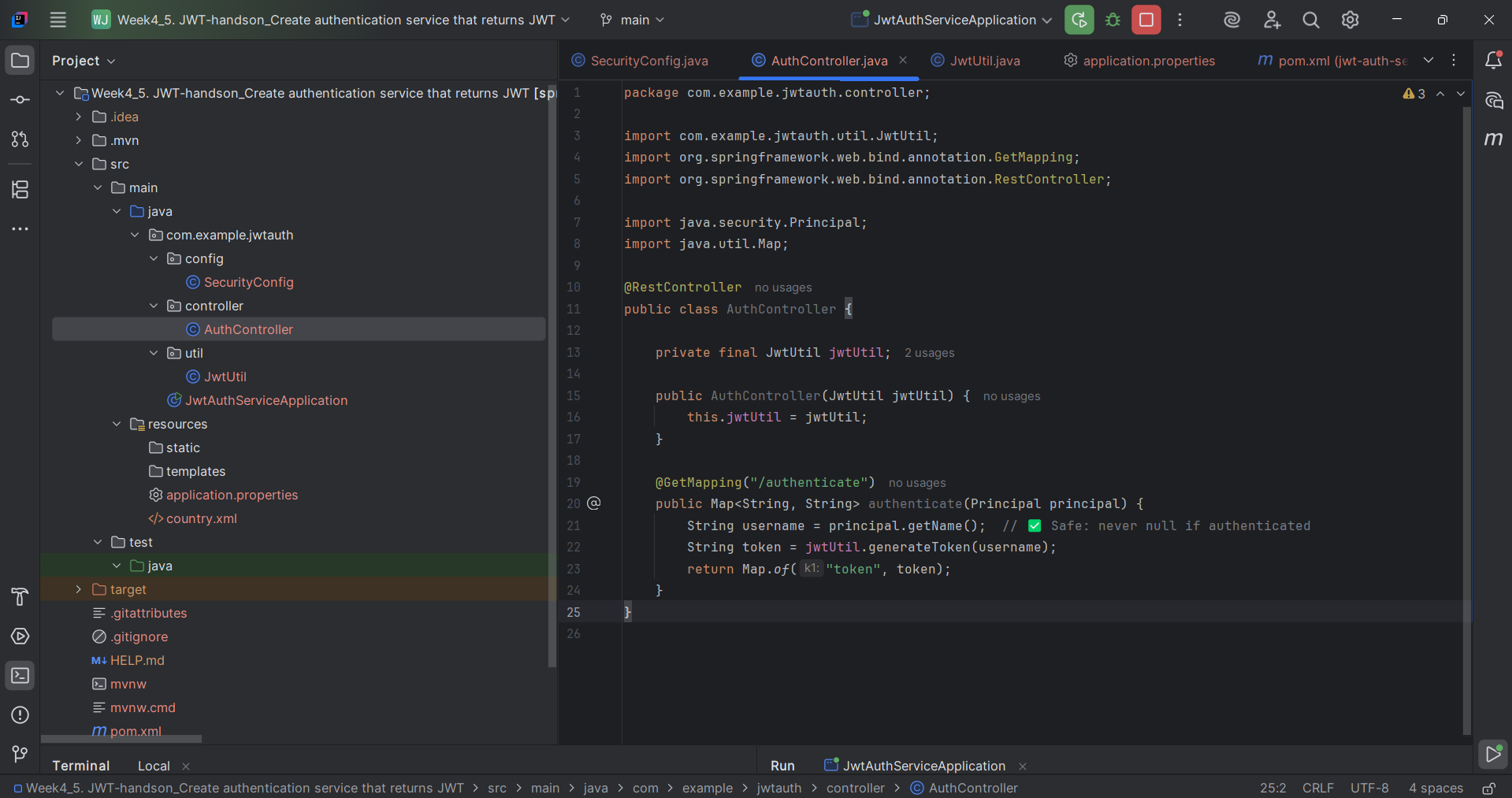
****

**SecurityConfig.java**

package com.example.jwtauth.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
  
import java.util.Collections;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 @Bean  
 public UserDetailsService userDetailsService() {  
 return new InMemoryUserDetailsManager(  
 User.*withUsername*("user")  
 .password("{noop}pwd")  
 .roles("USER")  
 .build()  
 );  
 }  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").authenticated()  
 .anyRequest().permitAll()  
 )  
 .httpBasic(httpBasic -> {})  
 .csrf(csrf -> csrf.disable());  
  
 return http.build();  
 }  
}

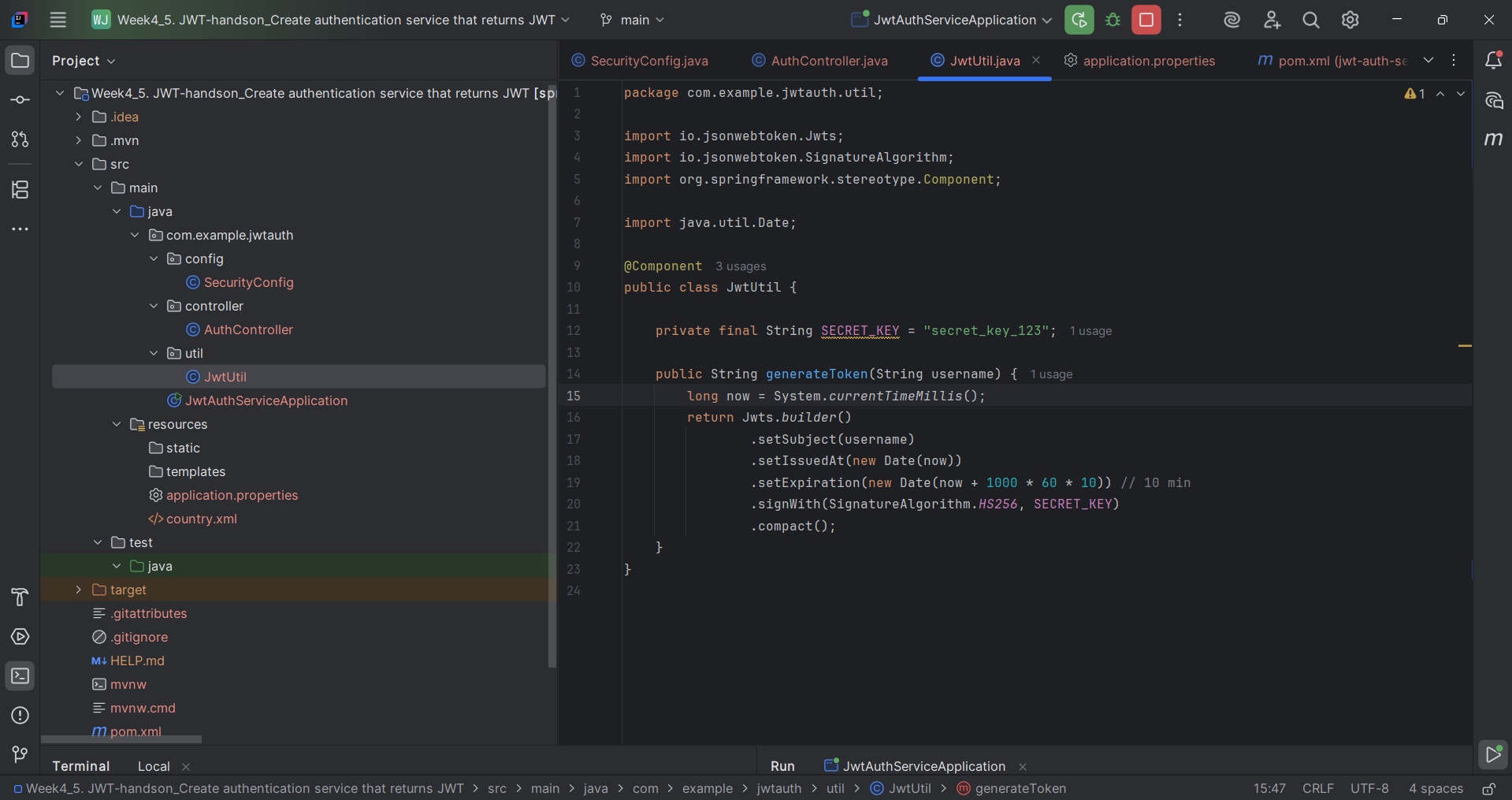
**AuthController.java**

package com.example.jwtauth.controller;  
  
import com.example.jwtauth.util.JwtUtil;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.security.Principal;  
import java.util.Map;  
  
@RestController  
public class AuthController {  
  
 private final JwtUtil jwtUtil;  
  
 public AuthController(JwtUtil jwtUtil) {  
 this.jwtUtil = jwtUtil;  
 }  
  
 @GetMapping("/authenticate")  
 public Map<String, String> authenticate(Principal principal) {  
 String username = principal.getName(); // ✅ Safe: never null if authenticated  
 String token = jwtUtil.generateToken(username);  
 return Map.*of*("token", token);  
 }  
}



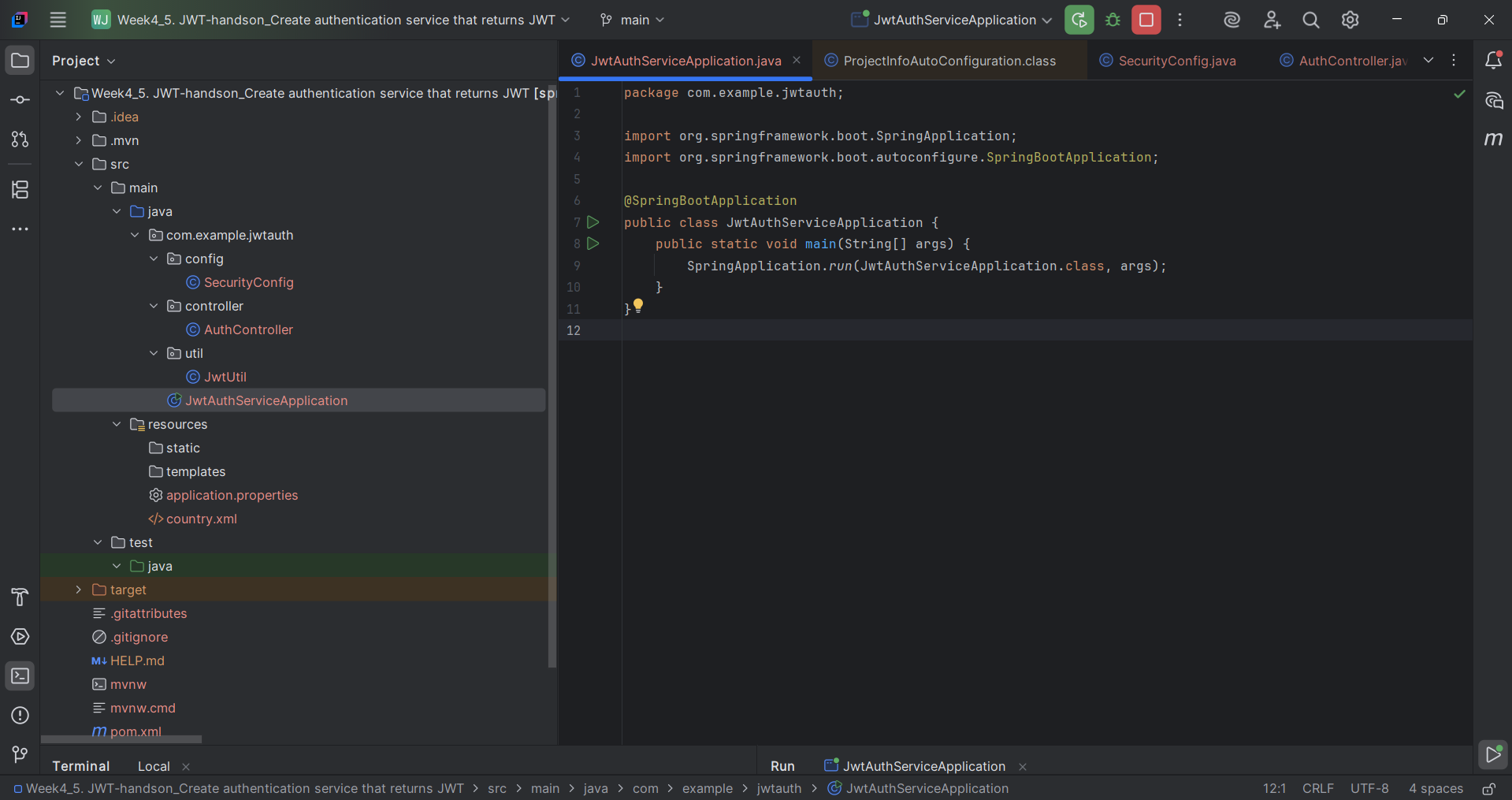
**JwtUtil.java**

package com.example.jwtauth.util;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import org.springframework.stereotype.Component;  
  
import java.util.Date;  
  
@Component  
public class JwtUtil {  
  
 private final String SECRET\_KEY = "secret\_key\_123";  
  
 public String generateToken(String username) {  
 long now = System.*currentTimeMillis*();  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date(now))  
 .setExpiration(new Date(now + 1000 \* 60 \* 10)) // 10 min  
 .signWith(SignatureAlgorithm.*HS256*, SECRET\_KEY)  
 .compact();  
 }  
}



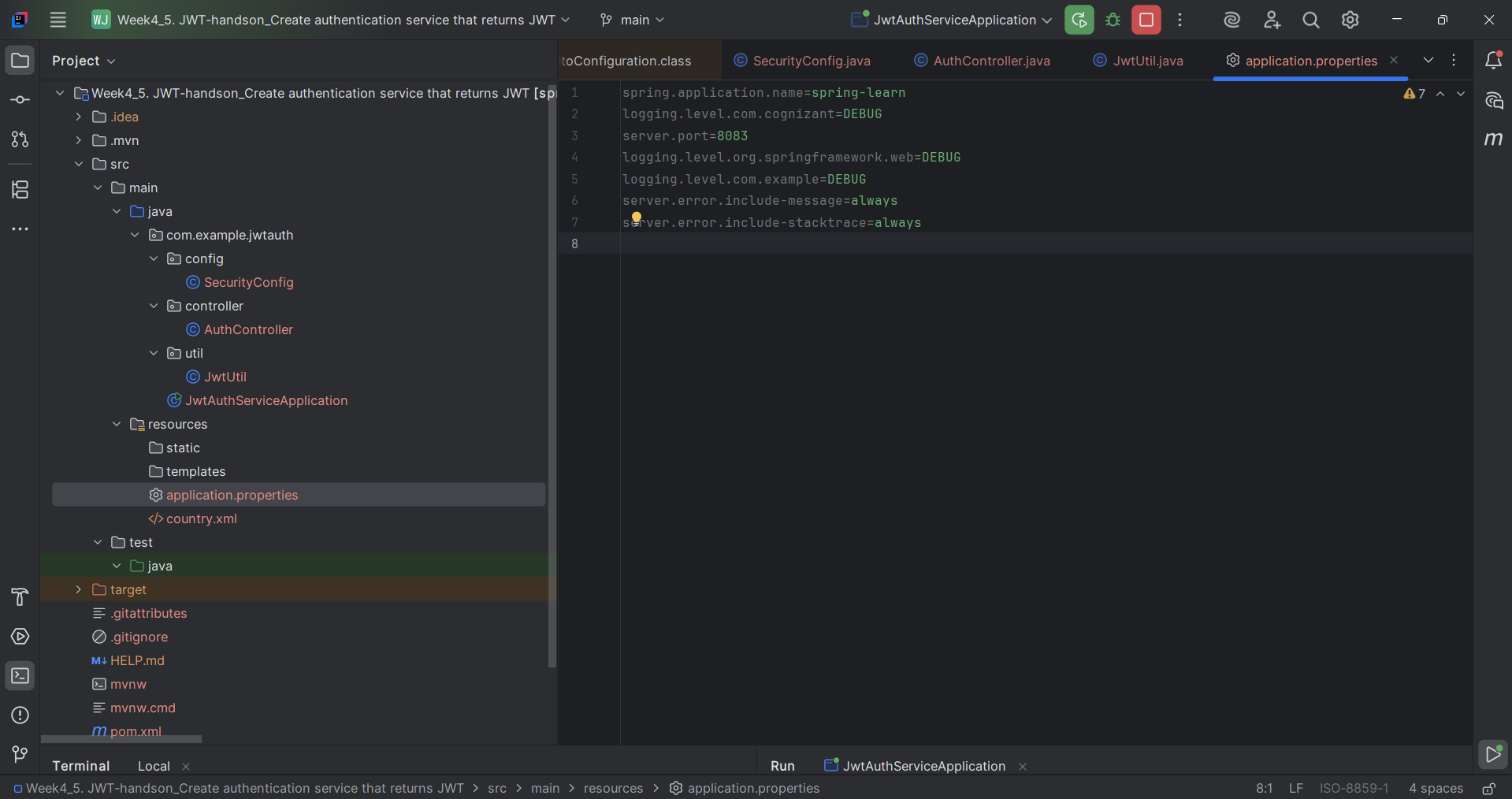
**JwtAuthServiceApplication.java**

package com.example.jwtauth;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class JwtAuthServiceApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(JwtAuthServiceApplication.class, args);  
 }  
}

****

**application.properties (src/main/resources)**

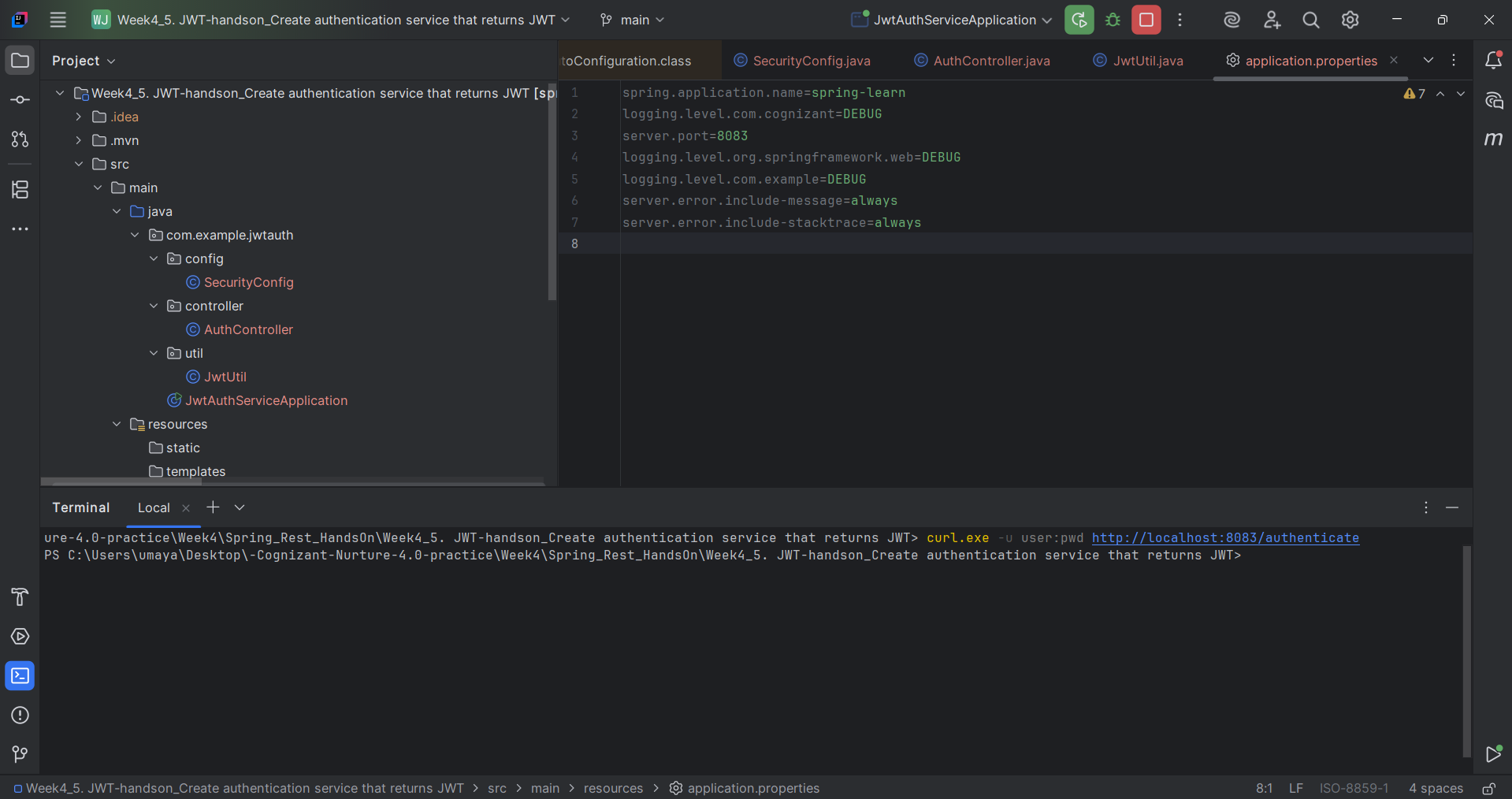
spring.application.name=spring-learn  
logging.level.com.cognizant=DEBUG  
server.port=8083  
logging.level.org.springframework.web=DEBUG  
logging.level.com.example=DEBUG  
server.error.include-message=always  
server.error.include-stacktrace=always

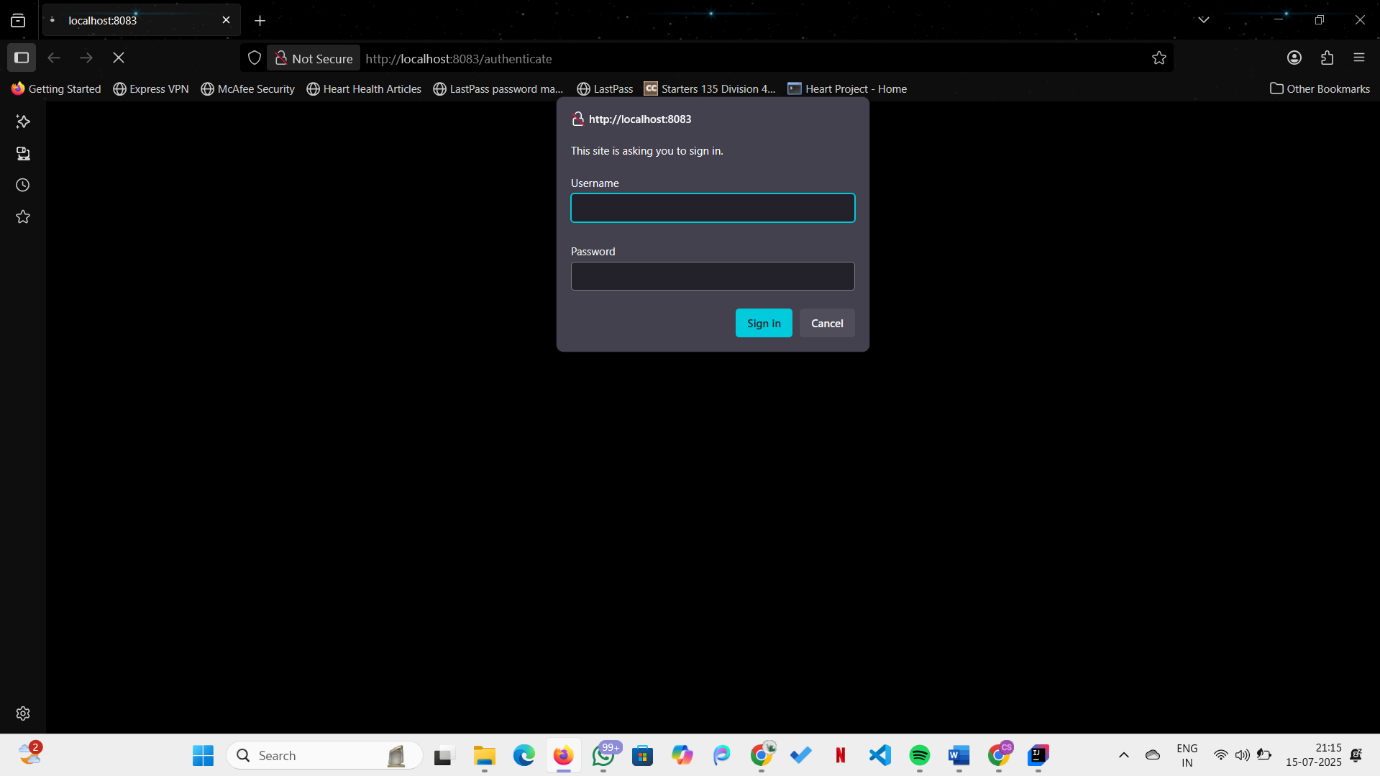


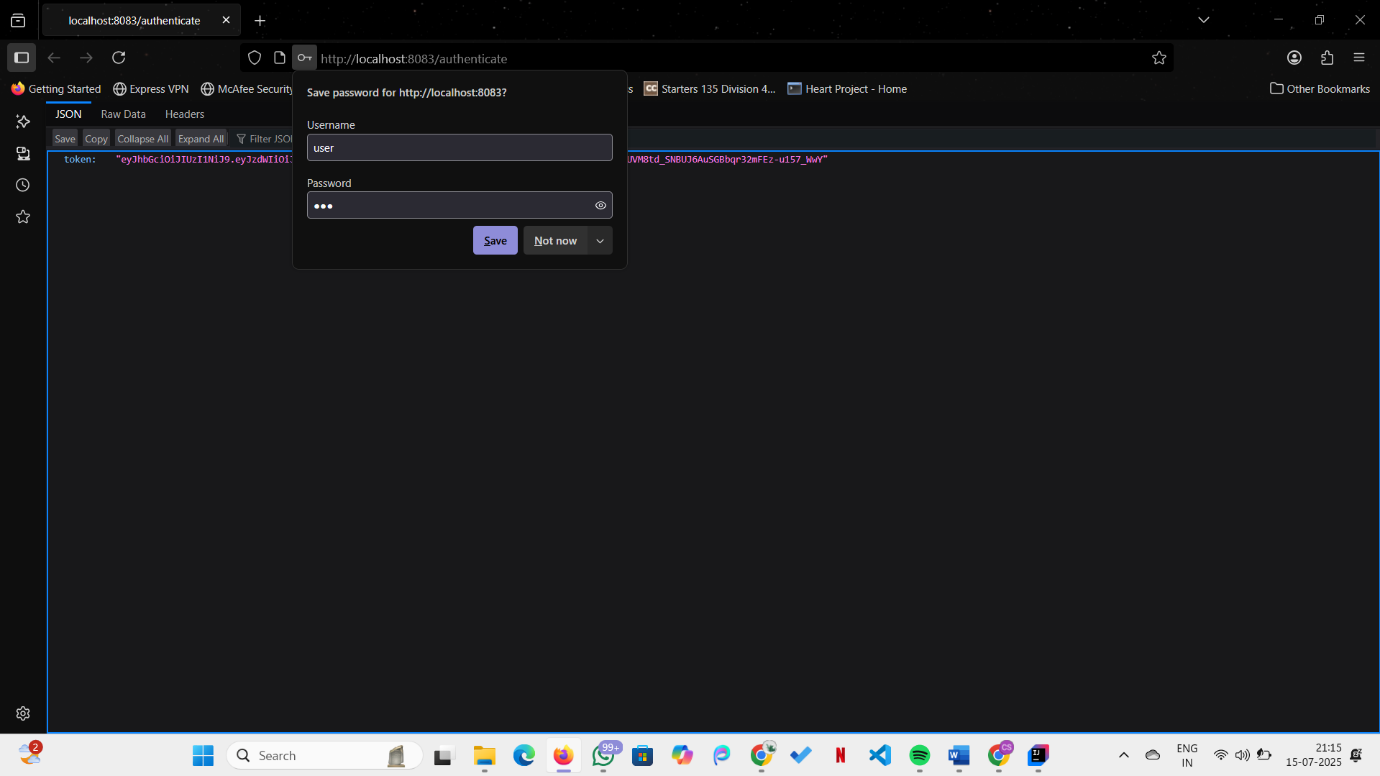
**pom.xml**

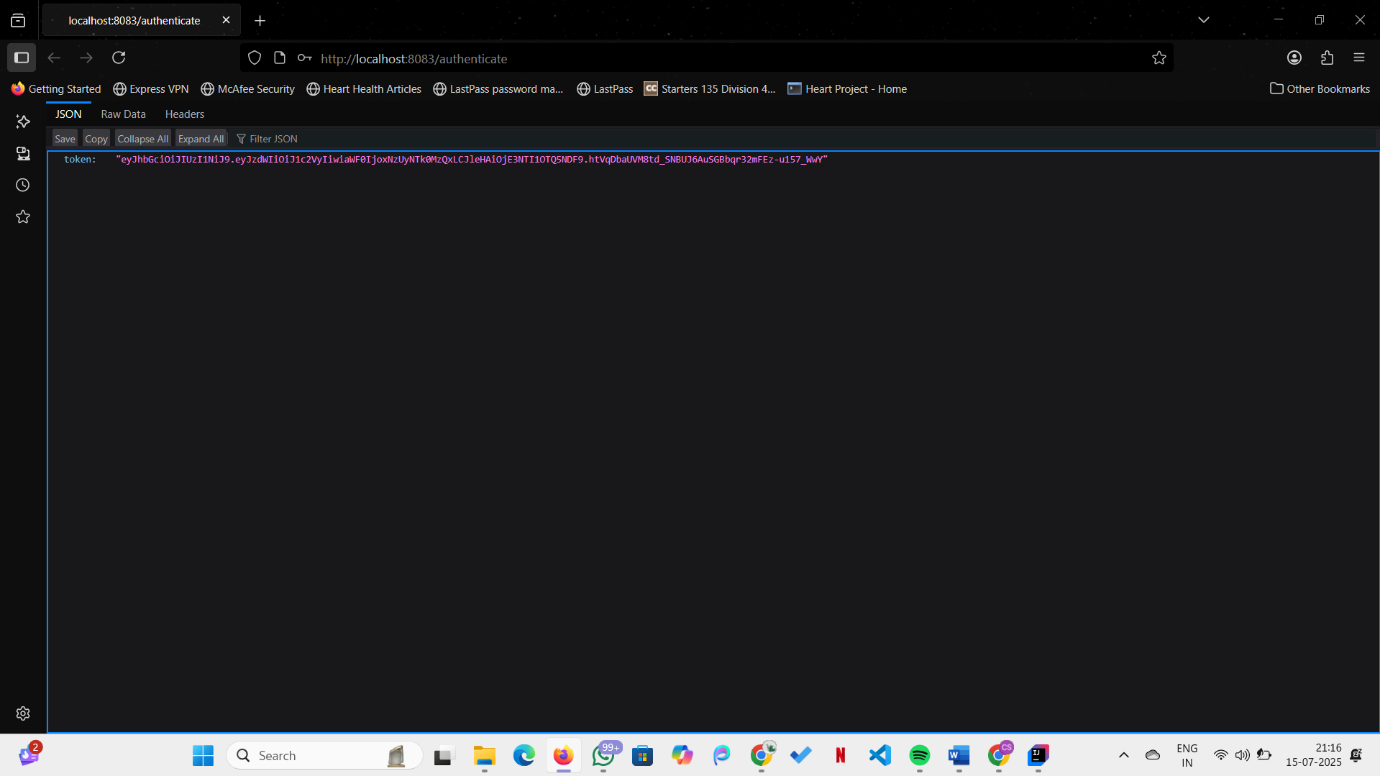
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>spring-learn</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Output:**

****

****

****

****