**Create authentication service that returns JWT**

**CODE:**

**JwtAuthApplication.java:**

package com.example.jwtauth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**SecurityConfig.java:**

package com.example.jwtauth.config;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.config.annotation.web.configuration.\*;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated();

}

}

**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 = "mysecretkey";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10)) // 10 min

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

AuthenticationController.java:

package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;

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

import org.springframework.http.\*;

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

import java.nio.charset.StandardCharsets;

import java.util.\*;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {

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

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

}

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

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

String credentials = new String(credDecoded, StandardCharsets.UTF\_8);

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

String username = values[0];

String password = values[1];

// Hardcoded credential check

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

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid credentials");

}

String token = jwtUtil.generateToken(username);

return ResponseEntity.ok(Collections.singletonMap("token", token));

}

}

application.properties:

server.port=8090

**pom.xml:**

<dependencies>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- Spring Security -->

<dependency>

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

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

</dependency>

<!-- JWT -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

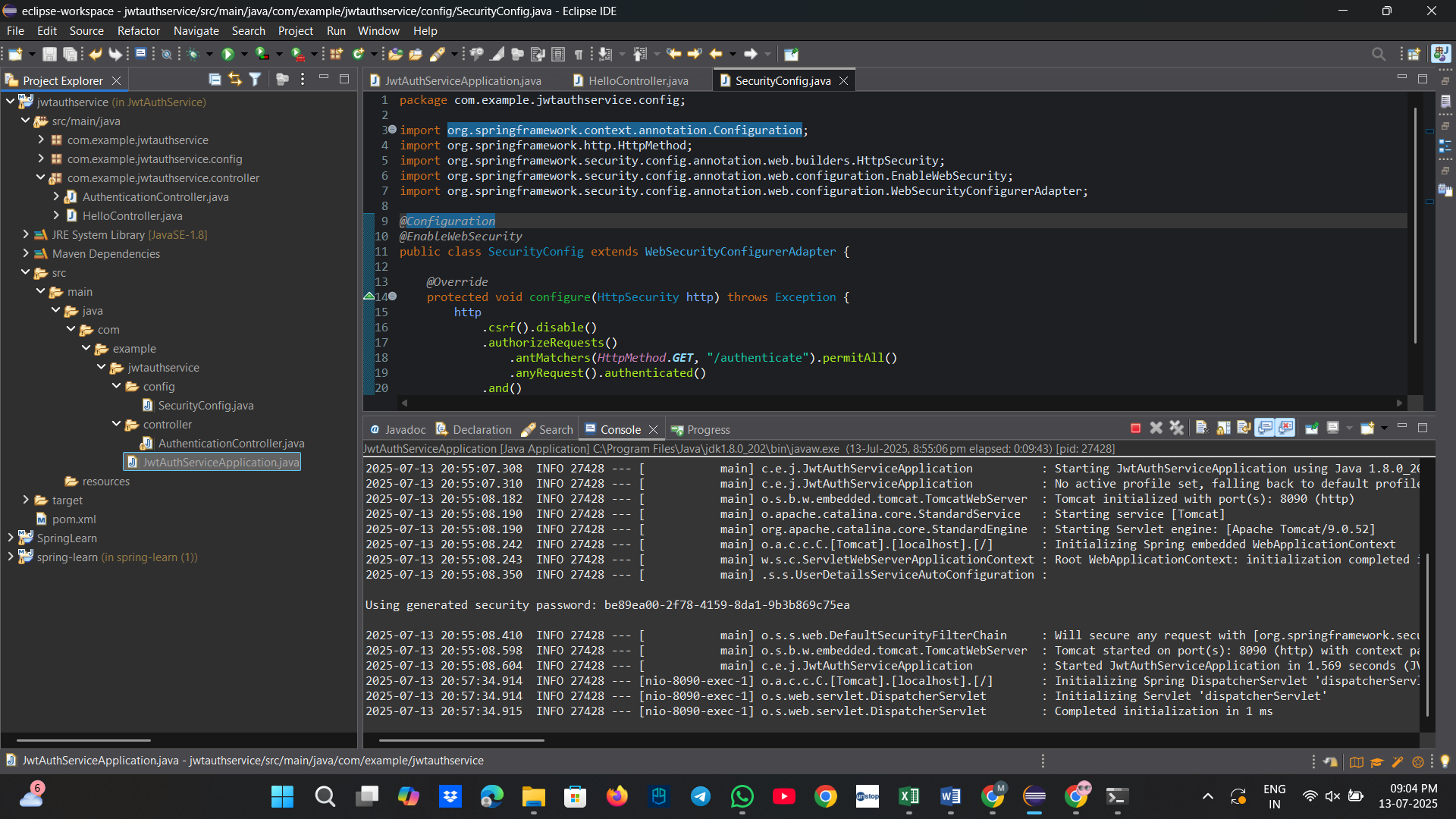
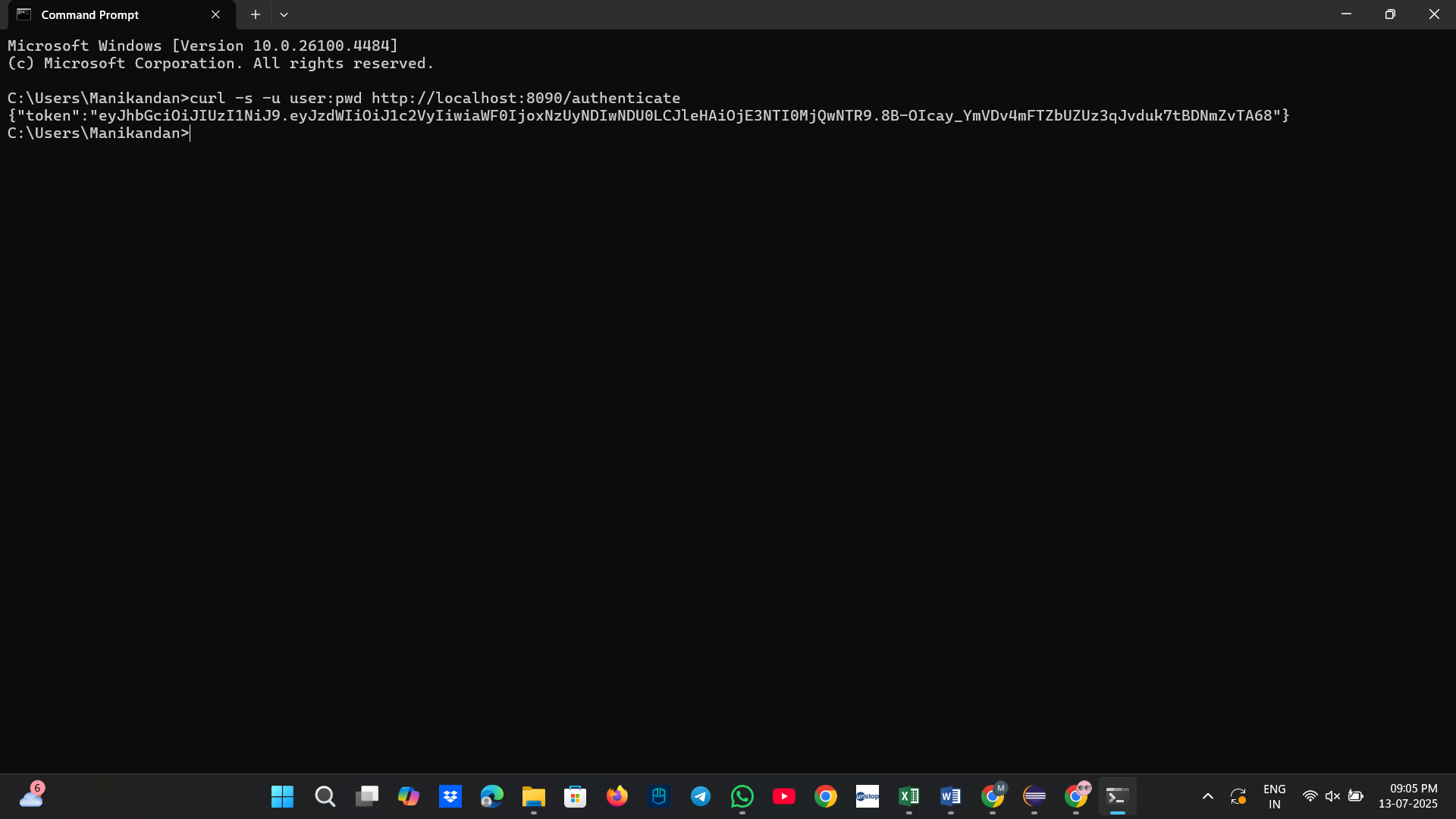
</dependency>

</dependencies>

**TEST:**

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

**OUTPUT:**

**** ****