**Create authentication service that returns JWT**

**pom.xml :**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

**AuthResponse.java :**

package com.example.jwt\_authentication.model;  
  
public class AuthResponse {  
 private String token;  
 public AuthResponse(String token) {  
 this.token = token;  
 }  
 public String getToken() {  
 return token;  
 }  
}

**JwtUtil.java :**

package com.example.jwt\_authentication.util;  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import io.jsonwebtoken.security.Keys;  
import org.springframework.stereotype.Component;  
  
@Component  
public class JwtUtil {  
 private static final String *SECRET\_KEY* = "thisIsASecretKeyWithEnoughLength12345";  
  
 public String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .signWith(Keys.*hmacShaKeyFor*(*SECRET\_KEY*.getBytes()), SignatureAlgorithm.*HS256*)  
 .compact();  
 }  
}

**AuthController.java :**

package com.example.jwt\_authentication.controller;  
  
import com.example.jwt\_authentication.model.AuthResponse;  
import com.example.jwt\_authentication.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import java.util.Base64;  
  
@RestController  
public class AuthController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @RequestMapping(value = "/authenticate", method = RequestMethod.*GET*)  
 public ResponseEntity<AuthResponse> authenticate(@RequestHeader(HttpHeaders.*AUTHORIZATION*) String authHeader) {  
  
 String[] credentials = extractCredentials(authHeader);  
 String username = credentials[0];  
 String password = credentials[1];  
  
 // Hardcoded validation for simplicity  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = jwtUtil.generateToken(username);  
 return ResponseEntity.*ok*(new AuthResponse(token));  
 } else {  
 return ResponseEntity.*status*(401).build();  
 }  
 }  
  
 private String[] extractCredentials(String header) {  
 String base64Credentials = header.substring("Basic ".length());  
 String decoded = new String(Base64.*getDecoder*().decode(base64Credentials));  
 return decoded.split(":", 2);  
 }  
}

**SecurityConfig.java :**

package com.example.jwt\_authentication.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 filterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 );  
 return http.build();  
 }  
}

**JwtAuthenticationApplication.java :**

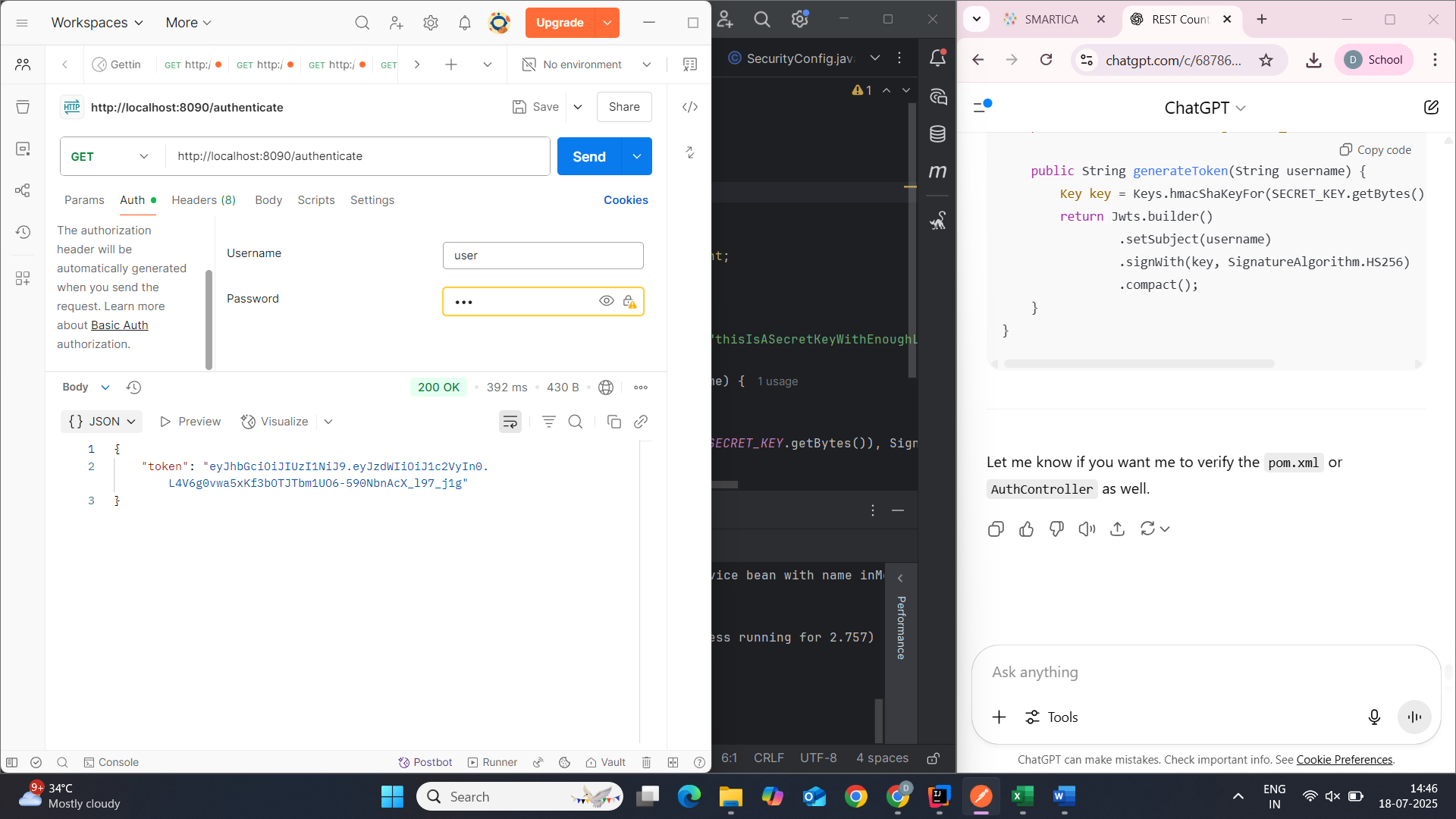
package com.example.jwt\_authentication;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class JwtAuthenticationApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(JwtAuthenticationApplication.class,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

**application.properties :**

server.port=8090

spring.application.name=jwt-authentication

**OUTPUT :**

****