**Create authentication service that returns JWT**

Using the same spring-learn application, we add the security configurations and the authentication controller classes to generate the jwt token.

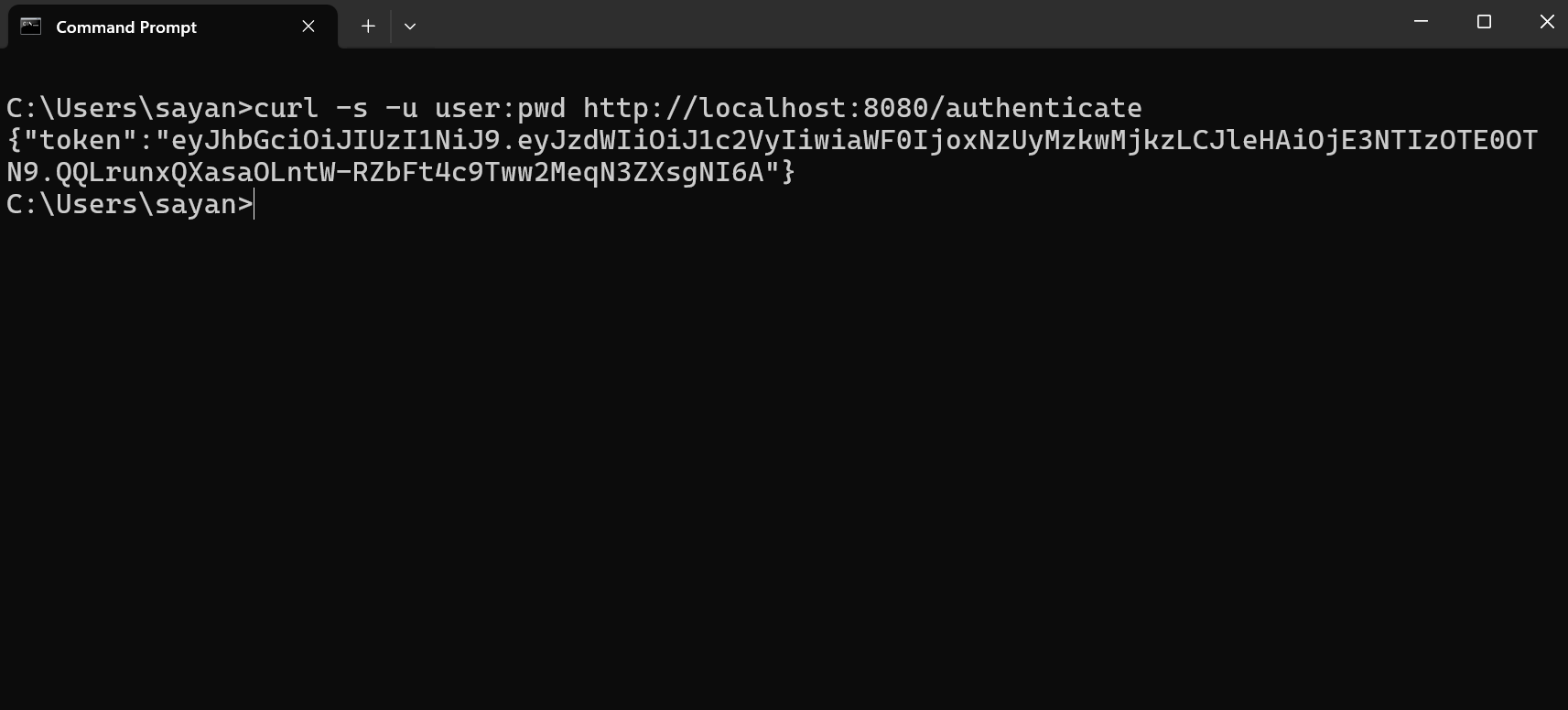
**SecurityConfig.java:**

*package* com.cognizant.springlearn.security;  
  
  
*import* org.slf4j.Logger;  
*import* org.slf4j.LoggerFactory;  
*import* org.springframework.context.annotation.*Bean*;  
*import* org.springframework.context.annotation.*Configuration*;  
*import* org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
*import* org.springframework.security.config.annotation.web.builders.HttpSecurity;  
*import* org.springframework.security.config.annotation.web.configuration.*EnableWebSecurity*;  
*import* org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
*import* org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
*import* org.springframework.security.crypto.password.PasswordEncoder;  
  
*@Configuration  
@EnableWebSecurity  
public class* SecurityConfig *extends* WebSecurityConfigurerAdapter {  
  
 *private static final* Logger LOGGER = LoggerFactory.getLogger(SecurityConfig.*class*);  
  
  
 *protected void* configure(AuthenticationManagerBuilder auth) *throws* Exception {  
 auth.inMemoryAuthentication()  
 .withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")  
 .and()  
 .withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");  
 }  
  
 *@Bean  
 public* PasswordEncoder passwordEncoder() {  
 LOGGER.info("Start");  
 *return new* BCryptPasswordEncoder();  
 }  
  
  
 *protected void* configure(HttpSecurity httpSecurity) *throws* Exception {  
 httpSecurity.csrf().disable().httpBasic().and()  
 .authorizeRequests().antMatchers("/countries").hasRole("USER");  
 }  
  
  
}

**AuthenticationController.java:**

*package* com.cognizant.springlearn.controller;  
  
*import* io.jsonwebtoken.JwtBuilder;  
*import* io.jsonwebtoken.Jwts;  
*import* io.jsonwebtoken.SignatureAlgorithm;  
*import* io.jsonwebtoken.security.Keys;  
*import* org.springframework.web.bind.annotation.*GetMapping*;  
*import* org.springframework.web.bind.annotation.*RequestHeader*;  
*import* org.springframework.web.bind.annotation.*RestController*;  
  
*import* java.util.Base64;  
*import* java.util.Date;  
*import* java.util.HashMap;  
*import* java.util.Map;  
  
*@RestController  
public class* AuthenticationController {  
  
 *@GetMapping*("/authenticate")  
 *public* Map<String, String> authenticate(*@RequestHeader*("Authorization") String authHeader) {  
 String user = getUser(authHeader);  
 String token = generateJwt(user);  
  
 Map<String, String> map = *new* HashMap<>();  
 map.put("token", token);  
 *return* map;  
 }  
  
 *private* String getUser(String authHeader) {  
 String encoded = authHeader.replace("Basic ", "");  
 *byte*[] decodedBytes = Base64.getDecoder().decode(encoded);  
 String decoded = *new* String(decodedBytes);  
 *return* decoded.split(":")[0]; *// get username* }  
  
 *private* String generateJwt(String user) {  
 JwtBuilder builder = Jwts.builder();  
 builder.setSubject(user);  
 builder.setIssuedAt(*new* Date());  
 builder.setExpiration(*new* Date(System.currentTimeMillis() + 1200000)); *// 20 mins* builder.signWith(Keys.hmacShaKeyFor("secretkeysecretkeysecretkey123456".getBytes()), SignatureAlgorithm.HS256);  
 *return* builder.compact();  
 }  
}

**OUTPUT**

****