**Create authentication service that returns JWT**

**SecurityConfig.java**

package com.cognizant.spring\_learn.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;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf(AbstractHttpConfigurer::disable)  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 )  
 .httpBasic(Customizer.*withDefaults*());  
  
 return http.build();  
 }  
}

**AuthController.java:**

package com.cognizant.spring\_learn.controller;  
  
import com.cognizant.spring\_learn.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.nio.charset.StandardCharsets;  
import java.util.Base64;  
import java.util.Collections;  
  
@RestController  
public class AuthController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @PostMapping("/authenticate")  
 public ResponseEntity<?> authenticate(@RequestHeader(value = "Authorization", required = false) String authHeader) {  
 if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("Missing or invalid Authorization header");  
 }  
  
 // Decode credentials  
 String base64Credentials = authHeader.substring("Basic ".length());  
 byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);  
 String credentials = new String(credDecoded, StandardCharsets.*UTF\_8*);  
  
 // credentials = "username:password"  
 final String[] values = credentials.split(":", 2);  
 if (values.length != 2) {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("Invalid basic authentication token");  
 }  
  
 String username = values[0];  
 String password = values[1];  
  
 // Hardcoded user validation  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = jwtUtil.generateToken(username);  
 return ResponseEntity.*ok*(Collections.*singletonMap*("token", token));  
 } else {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("Invalid credentials");  
 }  
 }  
}

**JwtUtil.java:**

package com.cognizant.spring\_learn.util;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import io.jsonwebtoken.security.Keys;  
import org.springframework.stereotype.Component;  
  
import java.util.Date;  
import javax.crypto.SecretKey;  
  
@Component  
public class JwtUtil {  
  
 private static final String *SECRET* = "mysecretkey1234567890mysecretkey1234567890"; // 256-bit key (32+ characters)  
  
 public String generateToken(String username) {  
 long currentTimeMillis = System.*currentTimeMillis*();  
 Date now = new Date(currentTimeMillis);  
 Date expiryDate = new Date(currentTimeMillis + 10 \* 60 \* 1000); // 10 minutes  
  
 SecretKey key = Keys.*hmacShaKeyFor*(*SECRET*.getBytes());  
  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(now)  
 .setExpiration(expiryDate)  
 .signWith(key, SignatureAlgorithm.*HS256*)  
 .compact();  
 }  
}

**Output:**

**curl -u user:pwd -X POST** [**http://localhost:8090/authenticate**](http://localhost:8090/authenticate)

**{**

**"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNzI4OTE4MDAwLCJleHAiOjE3Mjg5MjE2MDB9.LYlE8ZCNL2\_xKsvBAnlITh1mUVRJgb9RGRkAa3g8znk"**

**}**