Assignment: JWT Authentication Service in Spring Boot

# ✅ Objective:

Create an authentication service in Spring Boot that accepts user credentials and returns a JWT (JSON Web Token). This service reads the credentials passed via HTTP Basic Auth, authenticates them, and generates a token.

# ✅ Sample Request (cURL Command)

Request:  
curl -s -u user:pwd http://localhost:8090/authenticate  
  
Response:  
{"token":"<JWT\_TOKEN\_STRING>"}

# ✅ Implementation Steps

• Create Authentication Controller and configure it in SecurityConfig

• Read Authorization header and decode the username and password

• Generate JWT token based on the authenticated user

# ✅ Step 1: Create Authentication Controller

@RestController  
public class AuthenticationController {  
  
 @GetMapping("/authenticate")  
 public ResponseEntity<Map<String, String>> authenticate(@RequestHeader("Authorization") String authHeader) {  
 // Decode Basic Auth credentials  
 String base64Credentials = authHeader.substring("Basic ".length()).trim();  
 byte[] decodedBytes = Base64.getDecoder().decode(base64Credentials);  
 String credentials = new String(decodedBytes, StandardCharsets.UTF\_8);  
 final String[] values = credentials.split(":", 2);  
 String username = values[0];  
 String password = values[1];  
  
 // Validate credentials (example: hardcoded for demo)  
 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).build();  
 }  
 }  
}

# ✅ Step 2: JWT Utility Class

public class JwtUtil {  
 private static final String SECRET\_KEY = "secretkey";  
  
 public static String generateToken(String username) {  
 return Jwts.builder()  
 .setSubject(username)  
 .setIssuedAt(new Date(System.currentTimeMillis()))  
 .setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10)) // 10 mins  
 .signWith(SignatureAlgorithm.HS256, SECRET\_KEY)  
 .compact();  
 }  
}

# ✅ Step 3: Security 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();  
 }  
}

# ✅ SME Explanation

## 📌 a) What does the controller do?

• It extracts the Authorization header containing Basic Auth credentials.  
• Decodes the username and password from Base64.  
• Validates the credentials and generates a JWT if they are correct.  
• Returns the token in a JSON response.

## 📌 b) How is JWT generated?

• The token is created using the io.jsonwebtoken library (JJWT).  
• A secret key and claims like subject, issuedAt, and expiration are added.  
• It is signed using HMAC-SHA256 (HS256) algorithm.

## 📌 c) How is SecurityConfig set up?

• The authenticate endpoint is excluded from authentication.  
• Other endpoints remain protected unless configured otherwise.  
• CSRF protection is disabled for simplicity.

# ✅ Conclusion

This implementation sets up a basic authentication service using Spring Boot that generates JWT tokens. It uses HTTP Basic Auth to accept credentials and JJWT library to create tokens. This is the first step in setting up secure communication between client and server using JWT.