**Ex 6: Create authentication service that returns JWT**

**Add Maven Dependency for JWT**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

**Create JwtUtil Utility Class**

@Component

public class JwtUtil {

private String secretKey = "secret";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

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

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

}

**Create Authentication Controller**

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@RequestMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

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

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

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

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

String username = values[0];

String password = values[1];

if (username.equals("user") && password.equals("pwd")) {

String token = jwtUtil.generateToken(username);

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

} else {

throw new ResponseStatusException(HttpStatus.UNAUTHORIZED, "Invalid Credentials");

}

}

}  
**Security Configuration**

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests().anyRequest().permitAll();

}

}

**Output:**

