**Create authentication service that returns JWT**

As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.

**pom.xml :**

<dependencies>

<!-- Spring Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Security -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<!-- JWT Library -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

**JwtUtil.java :**

package com.cognizant.springlearn.util;

import java.util.Date;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

public class JwtUtil {

private static final String SECRET\_KEY = "your-secret-key";

public static String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

}

}

**AuthenticationController.java :**

package com.cognizant.springlearn.controller;

import java.util.Base64;

import javax.servlet.http.HttpServletRequest;

import com.cognizant.springlearn.util.JwtUtil;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class AuthenticationController {

private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

LOGGER.info("START /authenticate");

String authHeader = request.getHeader("Authorization");

if (authHeader == null || !authHeader.startsWith("Basic ")) {

return ResponseEntity.status(401).body("Missing or invalid Authorization header");

}

// Decode Base64

String[] credentials = new String(Base64.getDecoder().decode(authHeader.substring(6))).split(":");

String username = credentials[0];

String password = credentials[1];

LOGGER.debug("Received credentials: {} / {}", username, password);

// Simple validation (normally check against DB or service)

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

String token = JwtUtil.generateToken(username);

LOGGER.info("END /authenticate");

return ResponseEntity.ok().body("{\"token\":\"" + token + "\"}");

} else {

return ResponseEntity.status(403).body("Invalid credentials");

}

}

}

**SecurityConfig.java :**

package com.cognizant.springlearn.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.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(authz -> authz

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

}