**WEEK-4**

**Create authentication service that returns JWT**

**JwtUtil.java**

package com.example.jwt\_auth;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.util.Date;

import javax.crypto.SecretKey;

public class JwtUtil {

private static final SecretKey ***key*** = Keys.*hmacShaKeyFor*("MySuperSecretKeyForJwtAuth1234567890".getBytes());

private static final long ***EXPIRATION\_TIME*** = 1000 \* 60 \* 60; // 1 hour

public static String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.*currentTimeMillis*() + ***EXPIRATION\_TIME***))

.signWith(***key***, *SignatureAlgorithm*.***HS256***)

.compact();

}

}

**SecurityConfig.java**

package com.example.jwt\_auth.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.web.SecurityFilterChain;

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

*@Configuration*

public class SecurityConfig {

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests(auth -> auth.anyRequest().permitAll());

return http.build();

}

}

**AuthController.java**

package com.example.jwt\_auth.controller;

import com.example.jwt\_auth.JwtUtil;

import org.springframework.http.ResponseEntity;

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

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

import java.util.Base64;

*@RestController*

public class AuthController {

*@GetMapping*("/authenticate")

public ResponseEntity<?> authenticate(jakarta.servlet.http.HttpServletRequest request) {

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

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

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

}

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

String credentials = new String(Base64.*getDecoder*().decode(base64Credentials));

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

if (parts.length != 2) {

return ResponseEntity.*status*(400).body("Invalid credentials format");

}

String username = parts[0];

String password = parts[1];

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

String token = JwtUtil.*generateToken*(username);

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

} else {

return ResponseEntity.*status*(401).body("Invalid credentials");

}

}

}

**TokenResponse.java**

package com.example.jwt\_auth.model;

public class TokenResponse {

private String token;

public TokenResponse() {}

public TokenResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

public void setToken(String token) {

this.token = token;

}

}

**Pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

OUTPUT



