**WEEK\_4- JWT HANDSON**

**Create authentication service that returns JWT**

Package com.cognizant.spring\_learn.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import jakarta.servlet.http.HttpServletRequest;

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

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

import java.util.Base64;

import java.util.Date;

@RestController

public class AuthenticationController {

private static final String SECRET\_KEY = "secretkey"; // Use env var in real projects

private static final long EXPIRATION\_TIME = 10 \* 60 \* 1000; // 10 min

@GetMapping("/authenticate")

public TokenResponse authenticate(HttpServletRequest request) {

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

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

throw new RuntimeException("Missing or invalid Authorization header");

}

// Decode Base64 username:password

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

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

String credentials = new String(credDecoded);

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

String username = values[0];

String password = values[1];

// For demo: accept if username=user and password=pwd

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

String token = Jwts.builder()

.setSubject(username)

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

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

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

.compact();

return new TokenResponse(token);

} else {

throw new RuntimeException("Invalid credentials");

}

}

// Simple DTO

public static class TokenResponse {

private String token;

public TokenResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

}

}

Security config

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;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults());

return http.build();

}

}

JwtAuth1Application:

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class JwtAuth1Application {

public static void main(String[] args) {

SpringApplication.*run*(JwtAuth1Application.class, args);

}

}

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.