**1. Create authentication service that returns JWT**

**Code:**

**JwtAuthApplication.java:**package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**Application.properties:**

server.port=8090

spring.security.user.name=user

spring.security.user.password=pwd

**AuthController.java:**

package com.example.controller;

import com.example.util.JwtUtil;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import jakarta.servlet.http.HttpServletRequest;

import java.security.Principal;

import java.util.Map;

@RestController

public class AuthController {

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

Principal principal = request.getUserPrincipal();

if (principal == null) {

return ResponseEntity.status(401).build();

}

String username = principal.getName();

String token = JwtUtil.generateToken(username);

return ResponseEntity.ok(Map.of("token", token));

}

}

**JwtUtil.java:**

package com.example.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Date;

public class JwtUtil {

private static final Key SECRET\_KEY = Keys.hmacShaKeyFor("mySuperSecretKey12345678901234567890".getBytes());

public static String generateToken(String username) {

long now = System.currentTimeMillis();

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(now))

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

.signWith(SECRET\_KEY, SignatureAlgorithm.HS256)

.compact();

}

}

**SecurityConfig.java:**

package com.example.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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();

return http.build();

}

}

***Result:***

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.