**Create authentication service that returns JWT**

**Add JJWT dependency manually in 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>

**AuthenticationController.java**

package com.example.jwt\_auth.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import java.util.Base64;

import java.util.Date;

import java.util.Map;

@RestController

public class AuthenticationController {

private final String secretKey = "mySecretKey123456"; // Should be stored securely

@GetMapping("/authenticate")

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

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

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

}

// Step 1: Decode username and password

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

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

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

String username = values[0];

String password = values[1];

// Step 2: Hardcoded user validation (can connect to DB instead)

if (!"user".equals(username) || !"pwd".equals(password)) {

throw new RuntimeException("Invalid credentials");

}

// Step 3: Generate JWT token

String token = Jwts.builder()

.setSubject(username)

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

.setExpiration(new Date(System.currentTimeMillis() + 60 \* 60 \* 1000)) // 1 hour

.signWith(SignatureAlgorithm.HS256, secretKey.getBytes())

.compact();

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

}

}

**SecurityConfig.java**

package com.example.jwt\_auth.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().disable() // Disable CSRF for simplicity

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic(); // Enable HTTP Basic Auth

return http.build();

}

}

**Test the Service**

Run your Spring Boot application, and test using:

bash

curl -s -u user:pwd <http://localhost:8090/authenticate>

**OUTPUT:**

A black screen with white text

AI-generated content may be incorrect.