**Exercise 6: Create Authentication Service that Returns JWT**

1. **Add Dependencies (pom.xml)**

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Starter Security -->

<dependency>

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

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

</dependency>

<!-- JWT Library -->

<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>

</dependencies>

1. **SecurityConfig.java**

package com.example.demo.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated())

.httpBasic(); // Accept Basic Auth

return http.build();

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config)

throws Exception {

return config.getAuthenticationManager();

}

}

1. **JwtUtil.java**

package com.example.demo.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 key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

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)

.compact();

}

}

1. **AuthResponse.java**

package com.example.demo.model;

public class AuthResponse {

private String token;

public AuthResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

}

1. **AuthController.java**

package com.example.demo.controller;

import com.example.demo.model.AuthResponse;

import com.example.demo.util.JwtUtil;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.Authentication;

@RestController

public class AuthController {

@Autowired

private AuthenticationManager authManager;

@GetMapping("/authenticate")

public AuthResponse authenticate(@RequestHeader("Authorization") String authHeader) {

// Decode Basic Auth

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

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

String credentials = new String(credDecoded);

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

String username = values[0];

String password = values[1];

// Authenticate

Authentication authentication = authManager.authenticate(

new UsernamePasswordAuthenticationToken(username, password)

);

// Generate Token

String token = JwtUtil.generateToken(username);

return new AuthResponse(token);

}

}

1. **In-Memory Authentication (application.properties)**

spring.security.user.name=user

spring.security.user.password=pwd

**Output :**

