**Create authentication service that returns JWT**

**SecurityConfig.java**

package com.debangshu.jwt.demo.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.config.http.SessionCreationPolicy;

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

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/auth/login").permitAll() // allow login API

.anyRequest().authenticated() // secure all other endpoints

)

.sessionManagement(sess -> sess.sessionCreationPolicy(SessionCreationPolicy.STATELESS))

.formLogin(login -> login.disable()); // disables the default UI form

return http.build();

}

}

**AuthController.java**

package com.debangshu.jwt.demo.controller;

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

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

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

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

import com.debangshu.jwt.demo.util.JwtUtil;

@RestController

@RequestMapping("/auth")

public class AuthController {

@PostMapping("/login")

public String login(@RequestParam String username, @RequestParam String password) {

if ("admin".equals(username) && "password".equals(password)) {

return JwtUtil.generateToken(username);

} else {

throw new RuntimeException("Invalid credentials");

}

}

}

**JwtUtil.java**

package com.debangshu.jwt.demo.util;

import java.util.Date;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import javax.crypto.SecretKey;

public class JwtUtil {

// This generates a 256-bit key for HS256

private static final SecretKey SECRET\_KEY = Keys.secretKeyFor(SignatureAlgorithm.HS256);

public static String generateToken(String username){

return Jwts.builder()

.setSubject(username)

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

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

.signWith(SECRET\_KEY)

.compact();

}

}

**Output from Postman**

