**AuthenticationController.java**

package com.cognizant.spring\_learn.controller;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

@RestController

public class AuthenticationController {

private Logger logger = LoggerFactory.getLogger(AuthenticationController.class);

@GetMapping("/authenticate")

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

logger.info("START - /authenticate");

logger.debug("Authorization header: {}", authHeader);

String user = getUser(authHeader);

logger.debug("Decoded user: {}", user);

String token = generateJwt(user);

Map<String, String> map = new HashMap<>();

map.put("token", token);

logger.info("END - /authenticate");

return map;

}

private String getUser(String authHeader) {

logger.debug("Inside getUser()");

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

byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);

String decodedCredentials = new String(decodedBytes);

logger.debug("Decoded credentials: {}", decodedCredentials);

String user = decodedCredentials.split(":")[0];

return user;

}

private String generateJwt(String user) {

logger.debug("Inside generateJwt()");

JwtBuilder builder = Jwts.builder();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date((new Date()).getTime() + 1200000)); // 20 mins

builder.signWith(SignatureAlgorithm.HS256, "secretkey");

return builder.compact();

}

}

**SecurityConfig.java**

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.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

@Configuration

public class SecurityConfig {

// Define users in memory

@Bean

public UserDetailsService userDetailsService() {

var user = User.withUsername("user")

.password("pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

// Define security rules

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.requestMatchers("/countries").hasRole("USER")

.anyRequest().authenticated()

)

.httpBasic(Customizer.withDefaults()); // For curl -u to work

return http.build();

}

// Password encoder (not recommended for production)

@Bean

public static NoOpPasswordEncoder passwordEncoder() {

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

}

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication // Enables auto-configuration, component scan, etc.

public class SpringLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);

public static void main(String[] args) {

LOGGER.info("START");

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("END");

}

}