**JWT Handson**

**Securing RESTful Web Services with Spring Security**

**Add Spring Security dependency in pom.xml**

<dependency>

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

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

</dependency>

**SecurityConfig.java**

package com.cognizant.spring\_hello\_world.security;

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;

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

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests()

.anyRequest().authenticated()

.and()

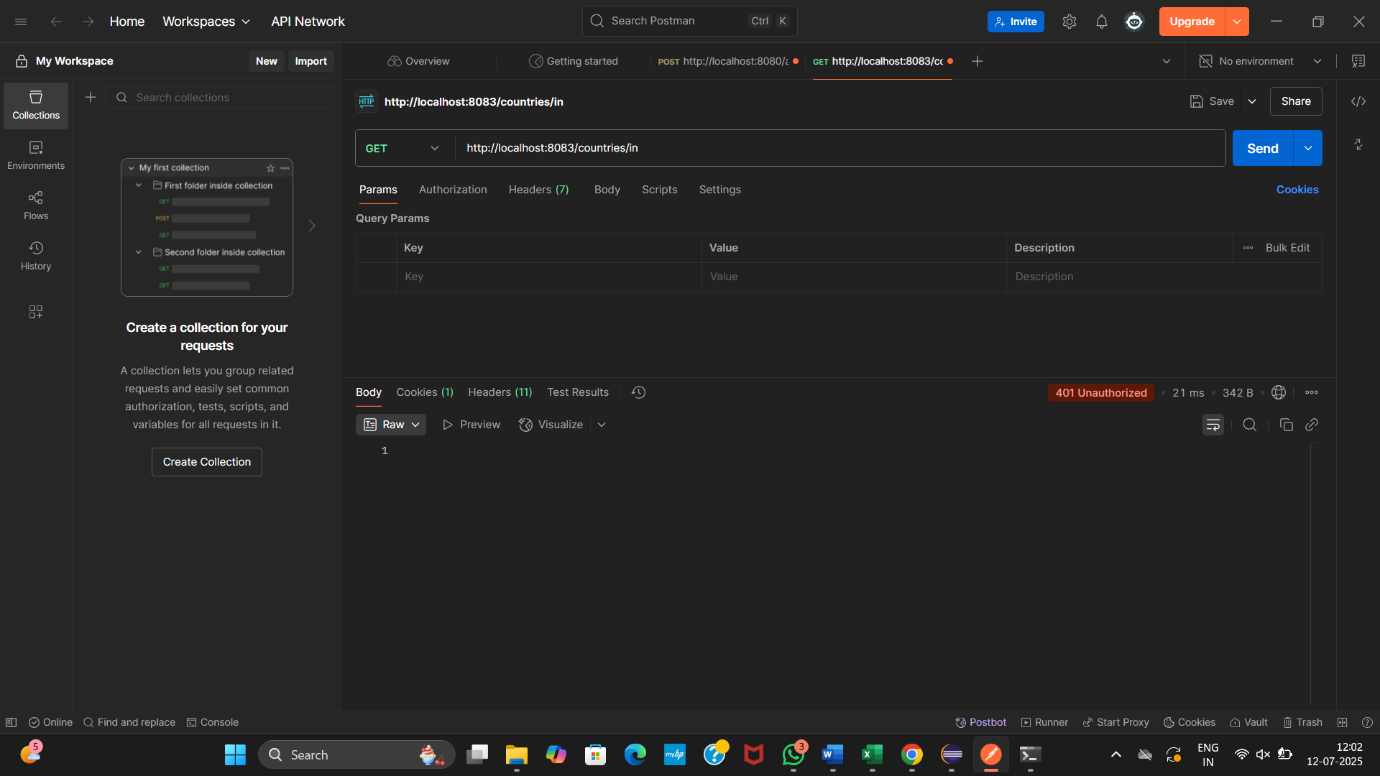
.httpBasic();

return http.build();

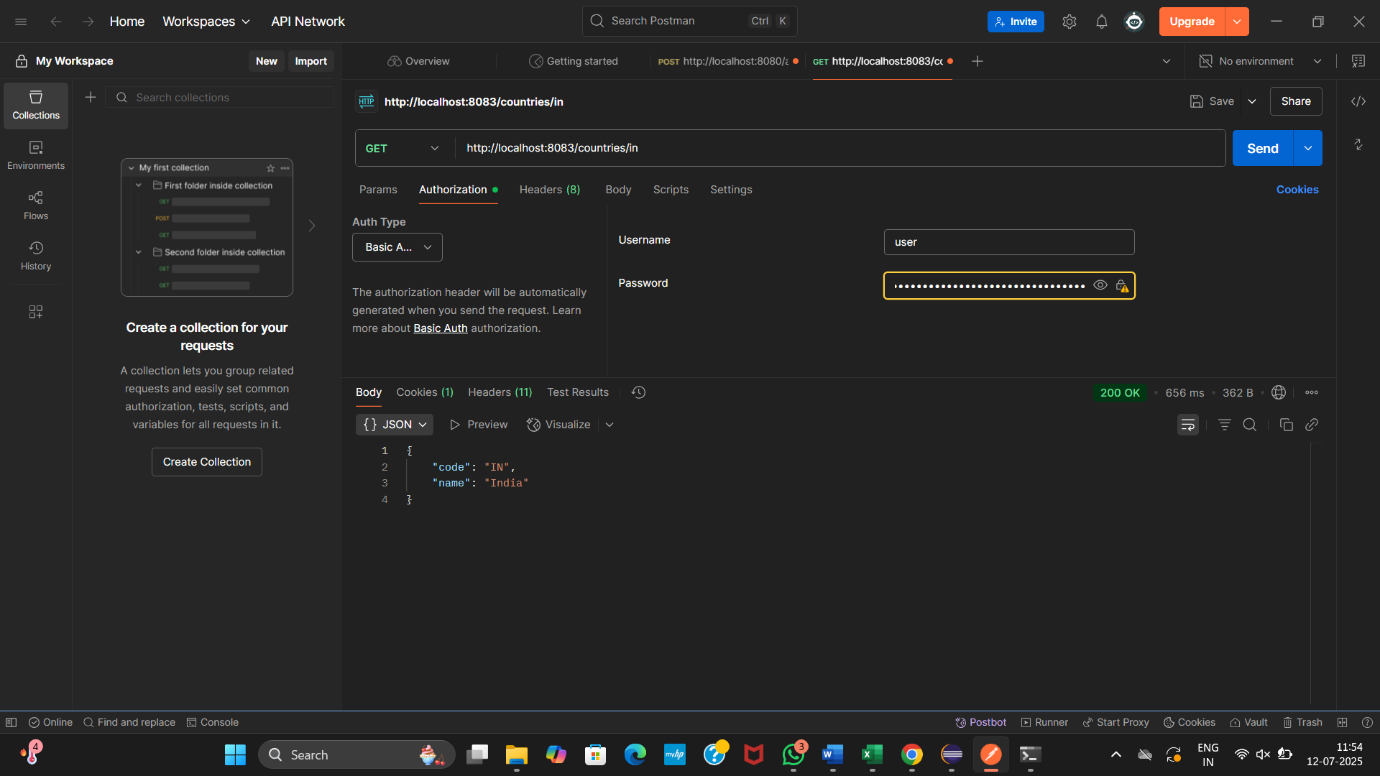
}

}

**Postman**

****

**After giving username and password(taken from console log):**



**Creating users and roles in Spring Security**

**SecurityConfig.java**

package com.cognizant.spring\_hello\_world.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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.core.userdetails.User;

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

import org.springframework.security.web.SecurityFilterChain;

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

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

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

// Define in-memory users with roles

*@Bean*

public UserDetailsService userDetailsService(PasswordEncoder passwordEncoder) {

***LOGGER***.info("Start userDetailsService");

InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();

manager.createUser(

User.*withUsername*("admin")

.password(passwordEncoder.encode("pwd"))

.roles("ADMIN")

.build()

);

manager.createUser(

User.*withUsername*("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build()

);

return manager;

}

// Define password encoding strategy

*@Bean*

public PasswordEncoder passwordEncoder() {

***LOGGER***.info("Start passwordEncoder");

return new BCryptPasswordEncoder();

}

// Define access rules using HttpSecurity

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

***LOGGER***.info("Start filterChain");

http.csrf().disable()

.authorizeHttpRequests()

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

.anyRequest().authenticated()

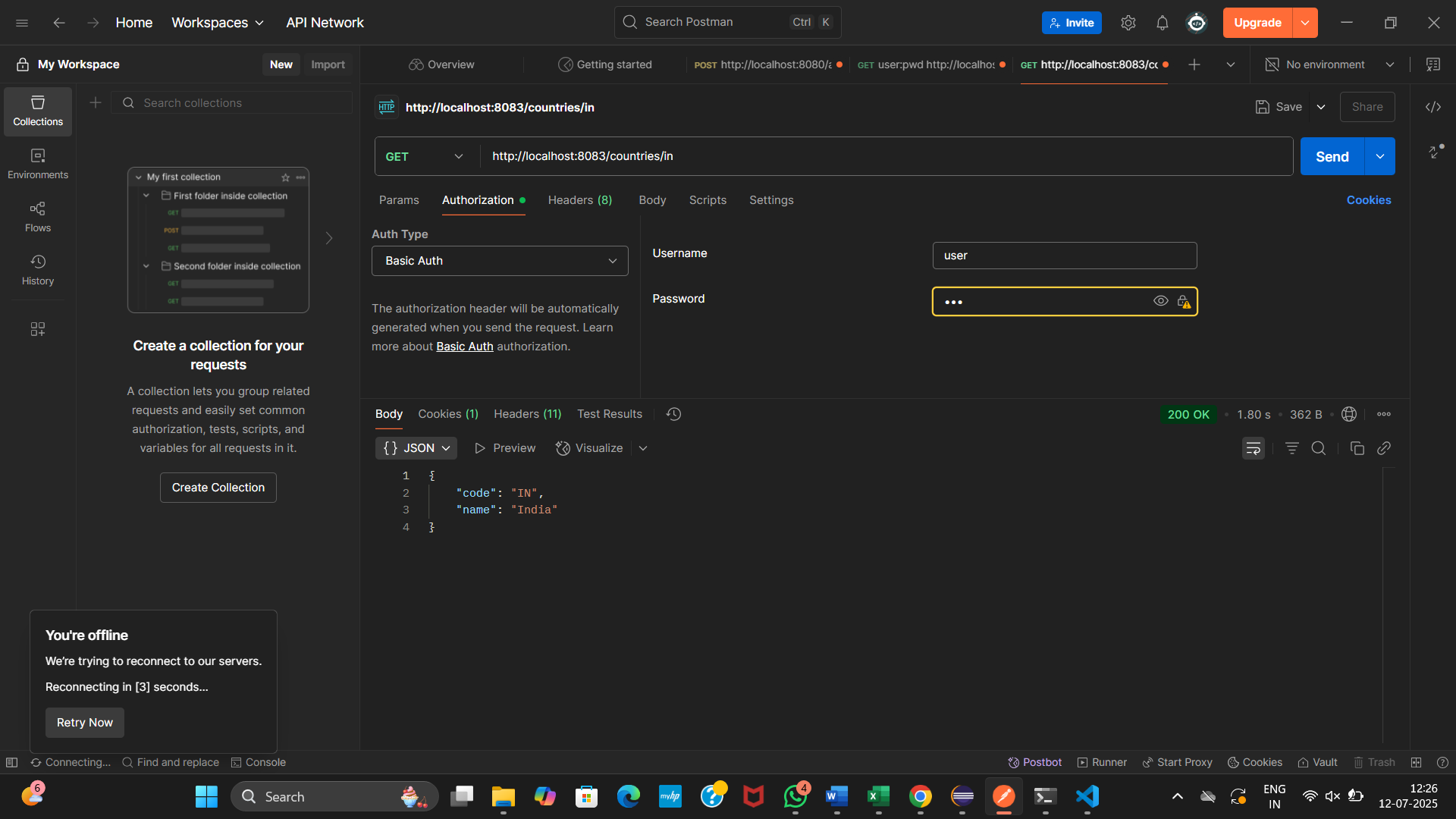
.and()

.httpBasic(); // Enable basic auth

return http.build();

}

}



**Create authentication service that returns JWT**

**Add dependencies in pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.1</version>

</dependency>

**JwtUtil.java**

package com.cognizant.spring\_hello\_world.security;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

*@Component*

public class JwtUtil {

private static final String ***SECRET\_KEY*** = "secret";

public String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(*SignatureAlgorithm*.***HS256***, ***SECRET\_KEY***)

.compact();

}

}

**AuthenticationController.java**

package com.cognizant.spring\_hello\_world.controller;

import com.cognizant.spring\_hello\_world.security.JwtUtil;

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

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

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

import java.util.HashMap;

import java.util.Map;

*@RestController*

public class AuthenticationController {

*@Autowired*

private JwtUtil jwtUtil;

*@GetMapping*("/authenticate")

public Map<String, String> authenticate() {

Authentication authentication = SecurityContextHolder.*getContext*().getAuthentication();

String username = authentication.getName();

String token = jwtUtil.generateToken(username);

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

response.put("token", token);

return response;

}

}

**SecurityConfig.java**

package com.cognizant.spring\_hello\_world.security;

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.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

*@Configuration*

*@EnableWebSecurity*

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(Customizer.*withDefaults*());

return http.build();

}

*@Bean*

public UserDetailsService userDetailsService(PasswordEncoder encoder) {

UserDetails user = User.*builder*()

.username("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

UserDetails admin = User.*builder*()

.username("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

*@Bean*

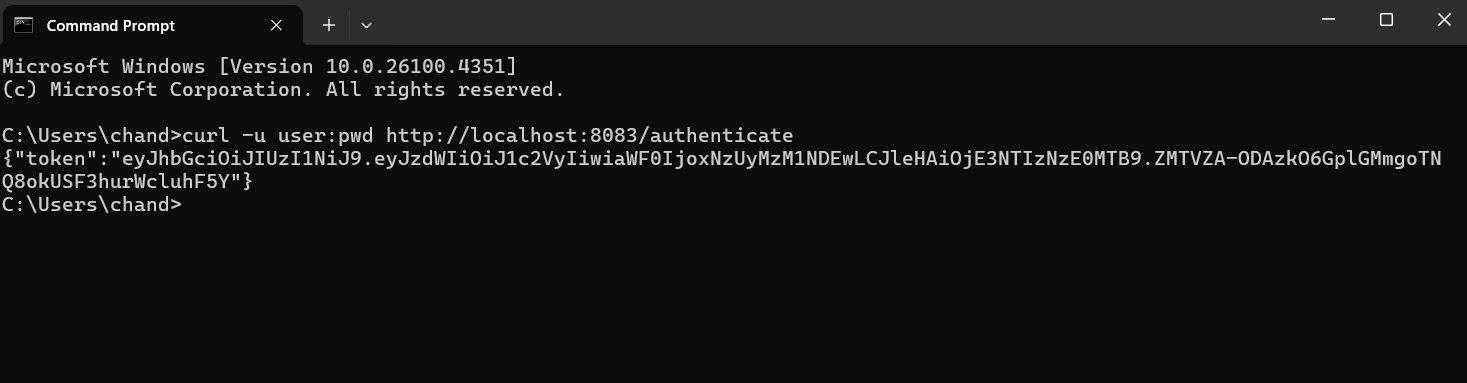
public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

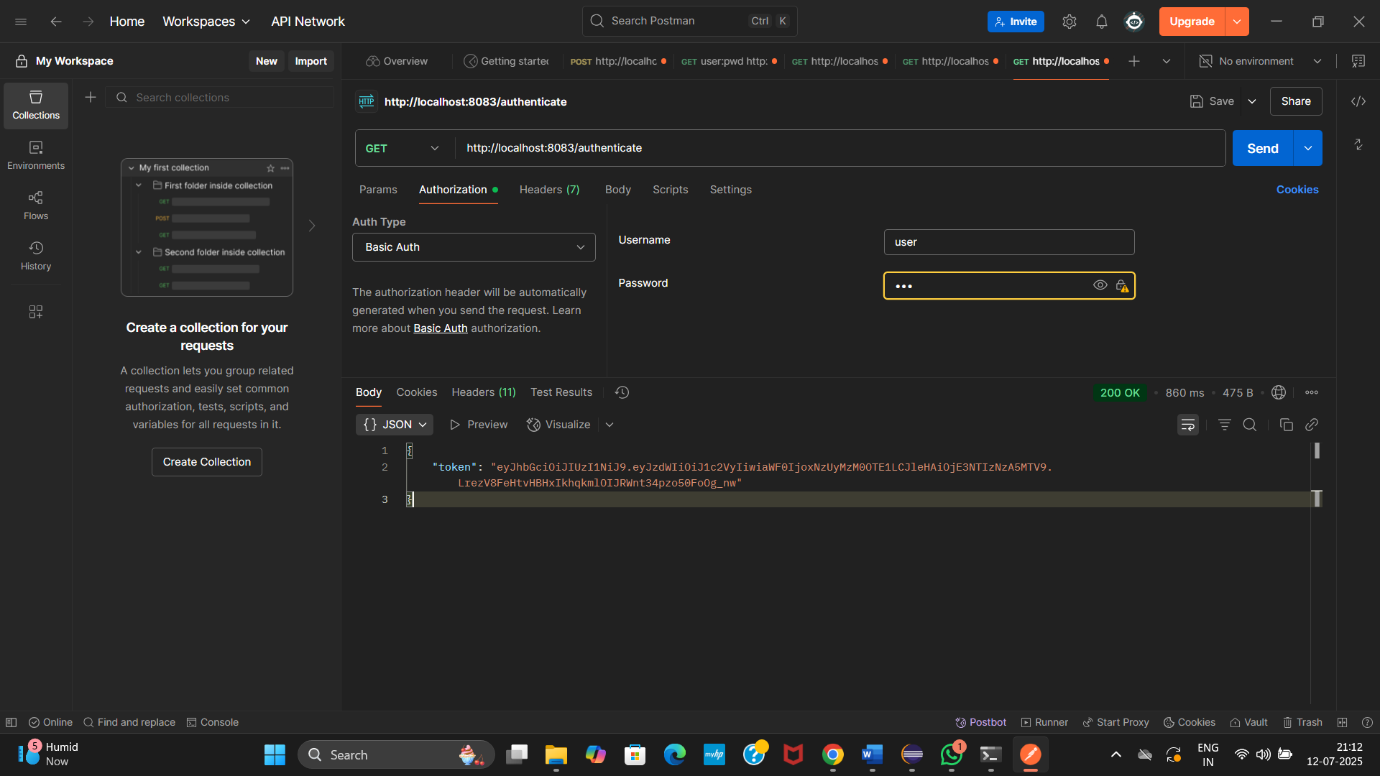
}

}

**curl:**

****

**Postman:**

****

**Read Authorization header and decode the username and password**

**AuthenticationController.java**

package com.cognizant.spring\_hello\_world.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

*@RestController*

public class AuthenticationController {

private static final 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 username = getUser(authHeader);

***LOGGER***.debug("Decoded username: {}", username);

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

response.put("token", "");

***LOGGER***.info("END - authenticate()");

return response;

}

private String getUser(String authHeader) {

***LOGGER***.debug("Start getUser()");

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

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

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

String decodedString = new String(decodedBytes);

***LOGGER***.debug("Decoded credentials: {}", decodedString);

String username = decodedString.split(":")[0];

***LOGGER***.debug("Extracted username: {}", username);

return username;

} else {

***LOGGER***.warn("Authorization header is missing or not in expected format.");

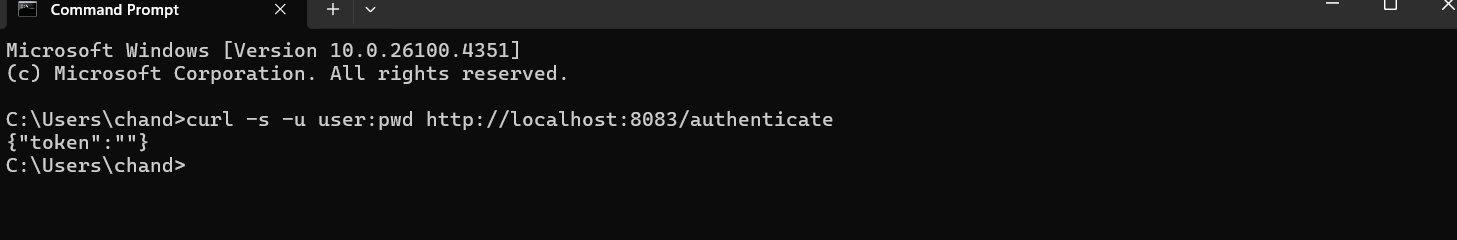
return null;

}

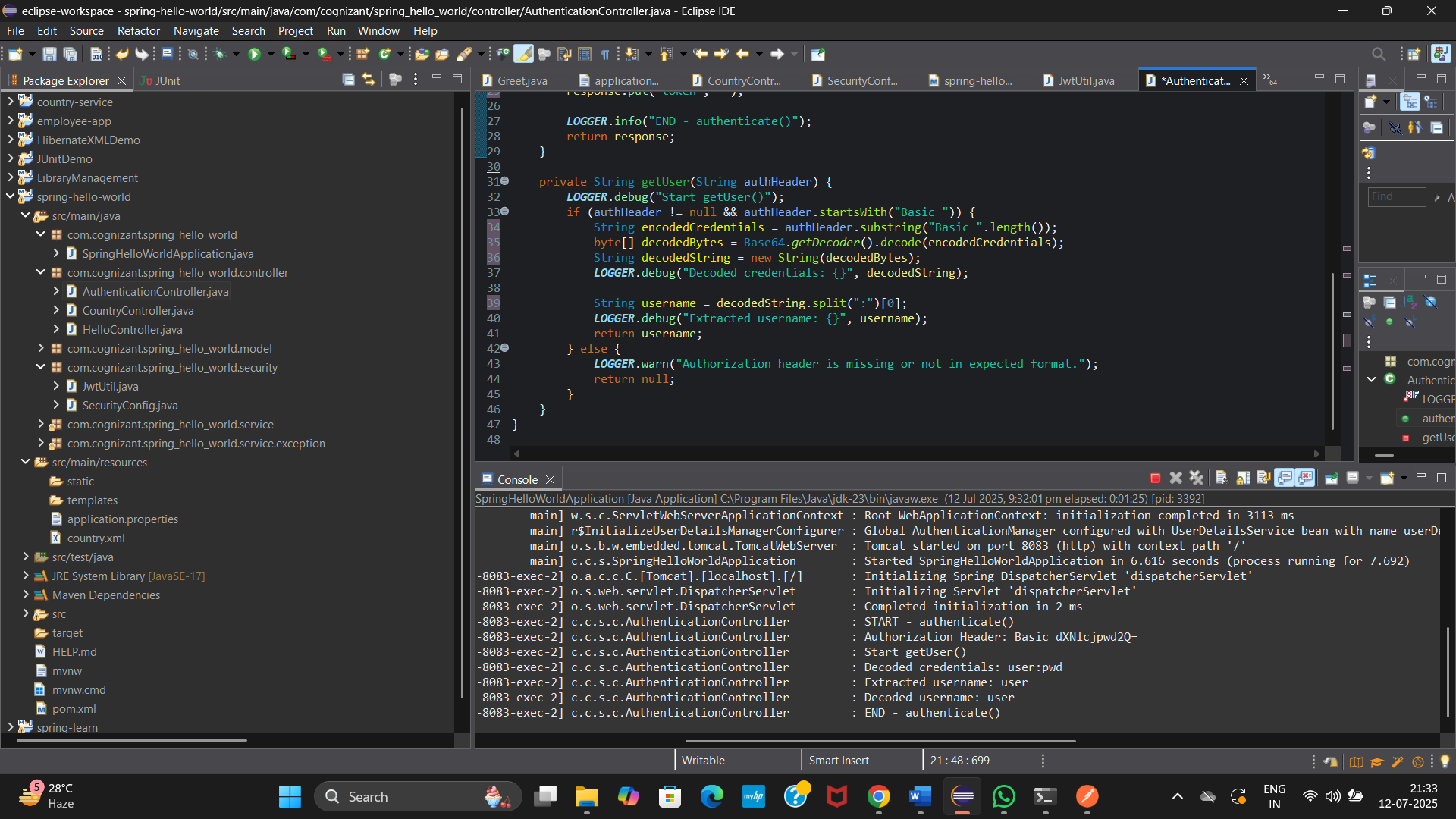
}

}

**curl:**

****

**Console logs:**

****

**Generate token based on the user**

**Pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

**AuthenticationController.java**

package com.cognizant.spring\_hello\_world.controller;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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;

*@RestController*

public class AuthenticationController {

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

*@GetMapping*("/authenticate")

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

***LOGGER***.info("START - authenticate()");

String user = getUser(authHeader);

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("Authorization Header: {}", authHeader);

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

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

String decodedString = new String(decodedBytes);

***LOGGER***.debug("Decoded credentials: {}", decodedString);

String username = decodedString.split(":")[0];

***LOGGER***.debug("Extracted username: {}", username);

return username;

}

private String generateJwt(String user) {

JwtBuilder builder = Jwts.*builder*();

builder.setSubject(user);

builder.setIssuedAt(new Date());

builder.setExpiration(new Date(System.*currentTimeMillis*() + 20 \* 60 \* 1000)); // 20 minutes

builder.signWith(*SignatureAlgorithm*.***HS256***, "secretkey"); // In practice, use environment variable

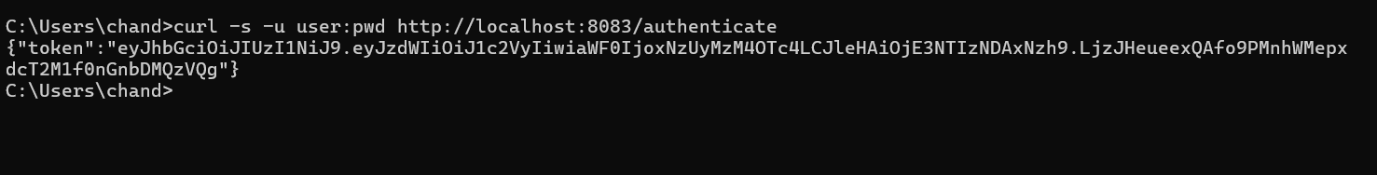
String token = builder.compact();

***LOGGER***.debug("Generated JWT: {}", token);

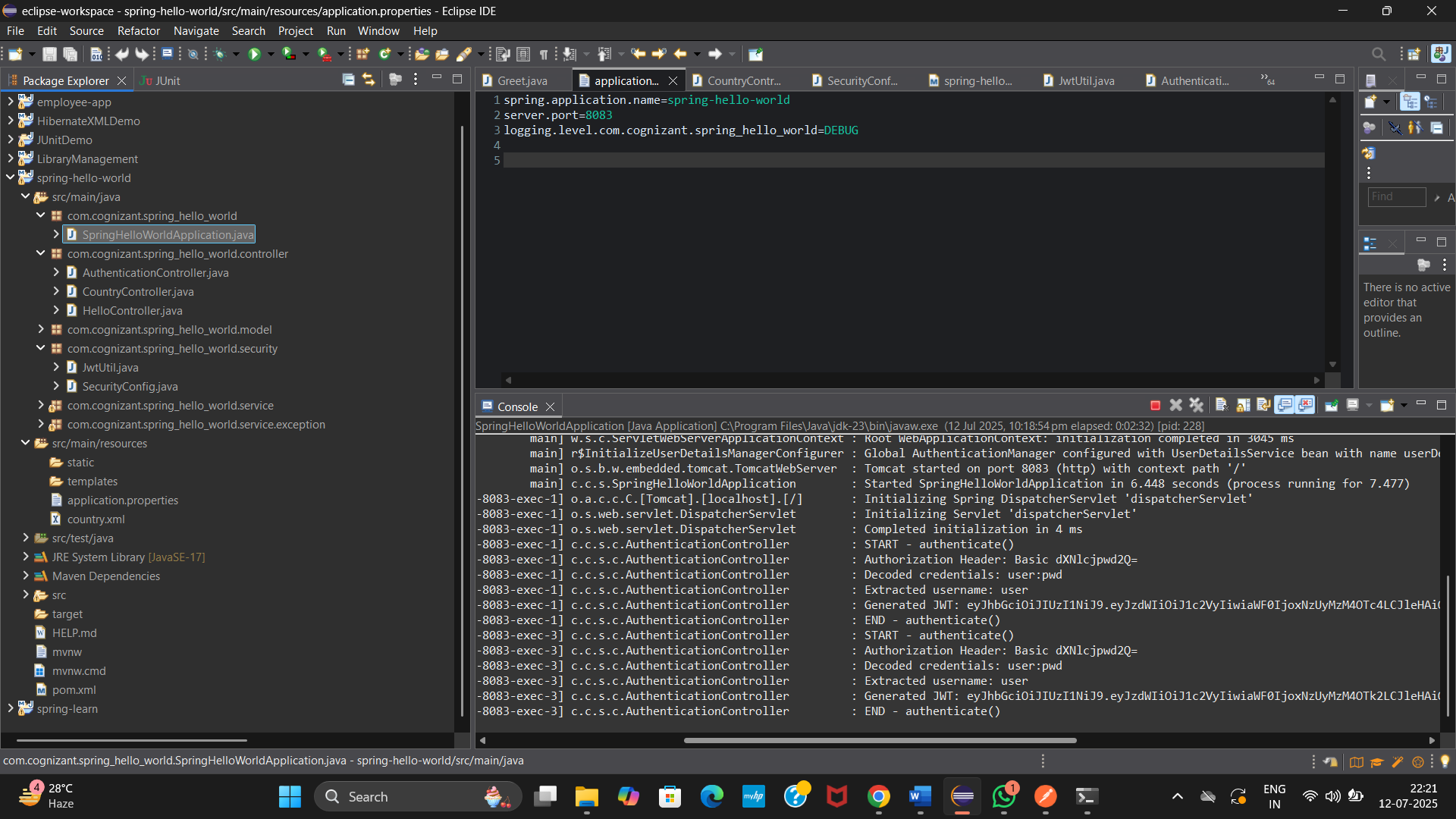
return token;

}

}



**Console logs:**

****

**Authorize based on JWT**

**JwtAuthorizationFilter.java**

package com.cognizant.spring\_hello\_world.security;

import java.io.IOException;

import java.util.ArrayList;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

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

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

***LOGGER***.info("JwtAuthorizationFilter Constructor - Start");

***LOGGER***.debug("AuthenticationManager: {}", authenticationManager);

}

*@Override*

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,

FilterChain chain) throws IOException, ServletException {

***LOGGER***.info("Start - doFilterInternal()");

String header = request.getHeader("Authorization");

***LOGGER***.debug("Authorization Header: {}", header);

if (header == null || !header.startsWith("Bearer ")) {

chain.doFilter(request, response);

return;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(request);

if (authentication != null) {

SecurityContextHolder.*getContext*().setAuthentication(authentication);

}

chain.doFilter(request, response);

***LOGGER***.info("End - doFilterInternal()");

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

String token = request.getHeader("Authorization");

if (token != null) {

try {

Jws<Claims> jws = Jwts.*parser*()

.setSigningKey("secretkey")

.parseClaimsJws(token.replace("Bearer ", ""));

String user = jws.getBody().getSubject();

***LOGGER***.debug("JWT User: {}", user);

if (user != null) {

return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());

}

} catch (JwtException e) {

***LOGGER***.error("Invalid JWT token", e);

return null;

}

}

return null;

}

}

**SecurityConfig.java**

package com.cognizant.spring\_hello\_world.security;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.ProviderManager;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

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

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

import java.util.Arrays;

*@Configuration*

*@EnableWebSecurity*

public class SecurityConfig {

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

*@Bean*

public SecurityFilterChain filterChain(HttpSecurity http, AuthenticationManager authenticationManager) throws Exception {

***LOGGER***.info("Setting up filter chain");

http

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

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.addFilter(new JwtAuthorizationFilter(authenticationManager))

.httpBasic(); // Retain for token request

return http.build();

}

*@Bean*

public AuthenticationManager authenticationManager(UserDetailsService userDetailsService) {

var provider = new DaoAuthenticationProvider();

provider.setUserDetailsService(userDetailsService);

provider.setPasswordEncoder(passwordEncoder());

return new ProviderManager(provider);

}

*@Bean*

public UserDetailsService userDetailsService() {

var userDetailsService = new InMemoryUserDetailsManager();

userDetailsService.createUser(User.*withUsername*("user").password(passwordEncoder().encode("pwd")).roles("USER").build());

userDetailsService.createUser(User.*withUsername*("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN").build());

return userDetailsService;

}

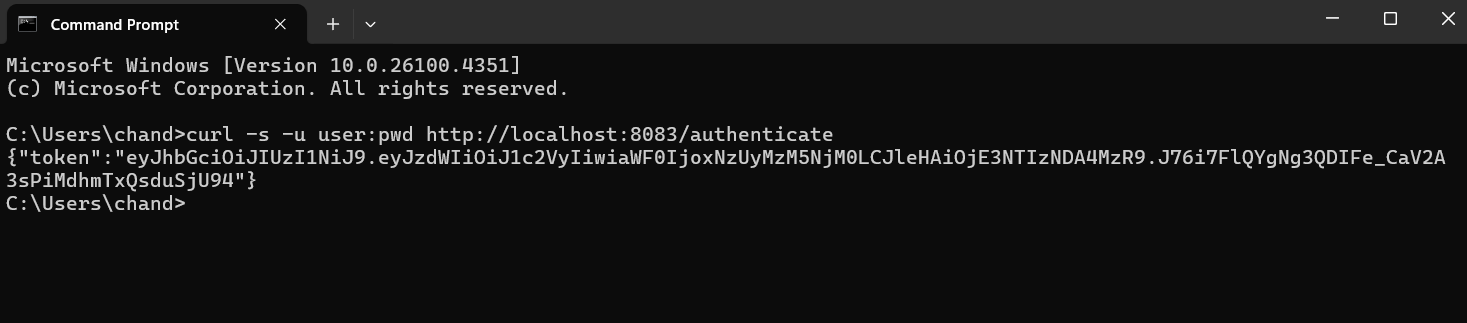
*@Bean*

public PasswordEncoder passwordEncoder() {

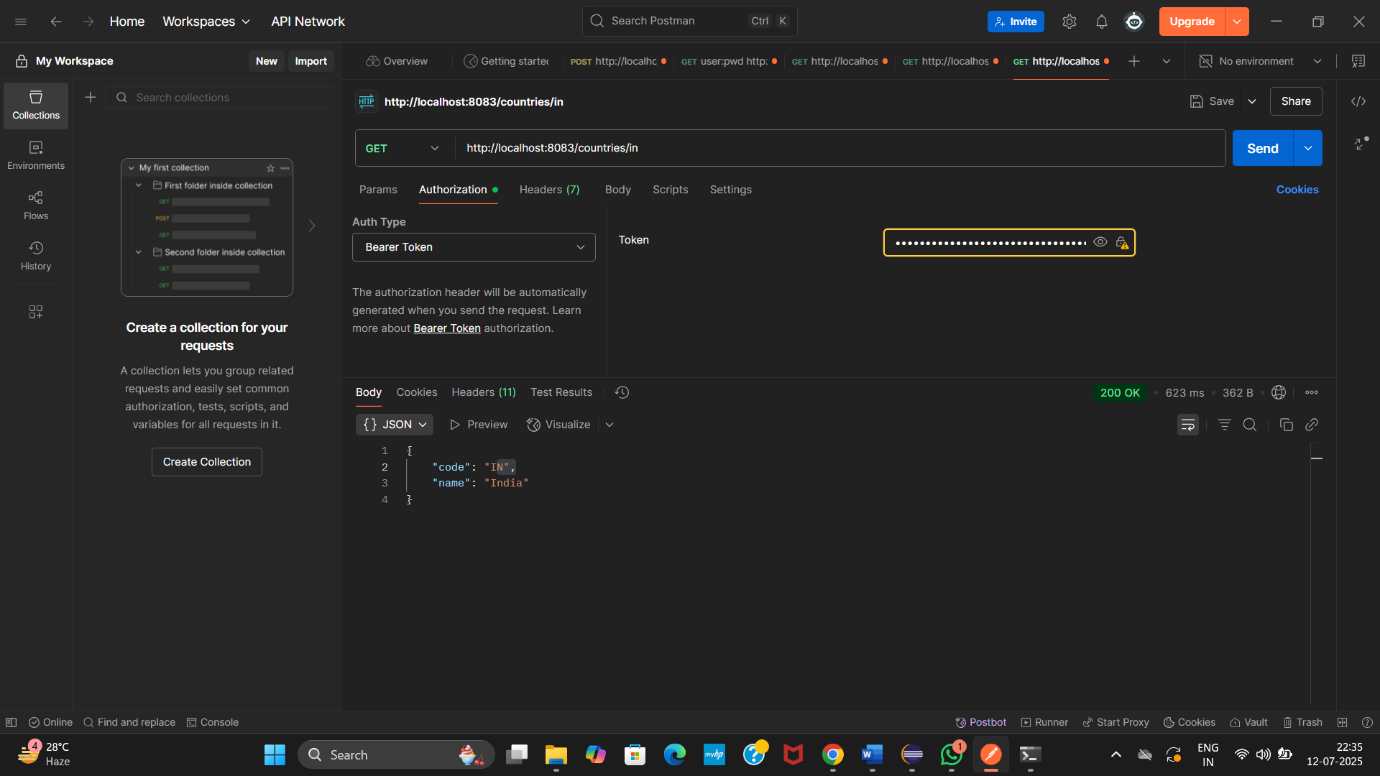
return new BCryptPasswordEncoder();

}

}



**make a GET request to** [**http://localhost:8083/countries/in**](http://localhost:8083/countries/in) **using generated token**

****