**Hands on**

**Create authentication service that returns JWT**

Solution-

1. Created SecuirtyConfig file in Security package

SecurityConfig.java-

package com.cognizant.jwtauth.security;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.inMemoryAuthentication()  
 .withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")  
 .and()  
 .withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.csrf().disable()  
 .authorizeRequests()  
 .antMatchers("/authenticate").authenticated() // require login here  
 .antMatchers("/countries").hasRole("USER") // keep your existing rule  
 .anyRequest().permitAll() // (optional) everything else open  
 .and()  
 .httpBasic(); // triggers the Basic filter  
 }  
  
}

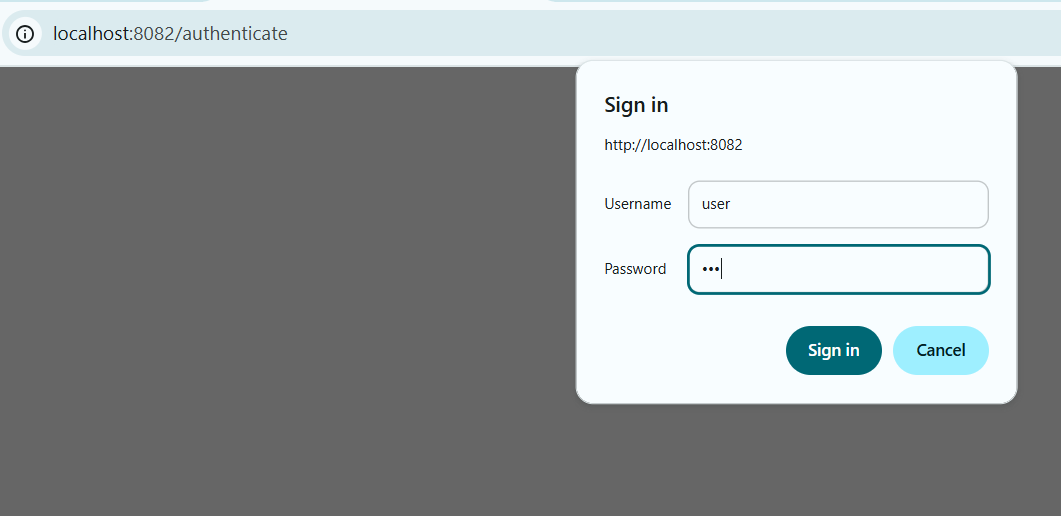
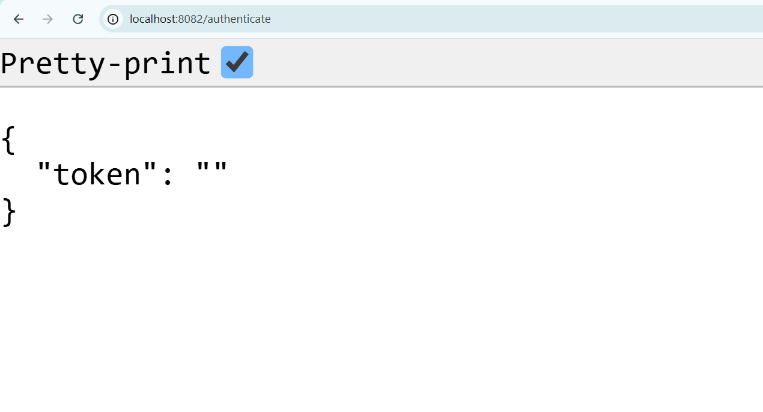
2. Created Controller class: com.cognizant.jwtauth.controller.AuthenticationController.java

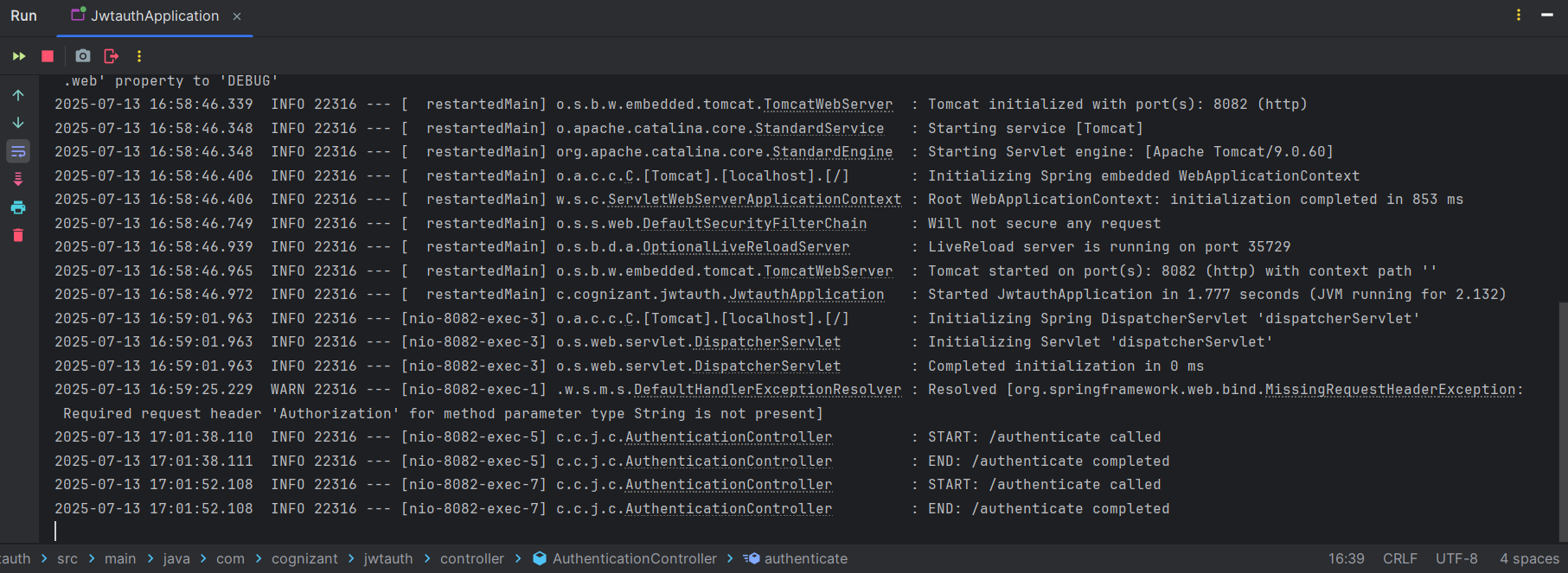
AuthenticationController.java-

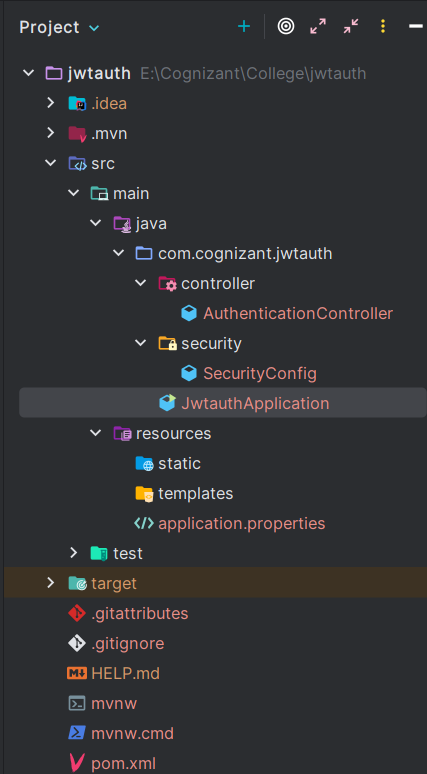
package com.cognizant.jwtauth.controller;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.HashMap;  
import java.util.Map;  
  
@RestController  
public class AuthenticationController {  
  
 private static final Logger *logger* = LoggerFactory.*getLogger*(AuthenticationController.class);  
  
 @GetMapping("/authenticate")  
 public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {  
 *logger*.info("START: /authenticate called");  
  
 *logger*.debug("Authorization Header: {}", authHeader);  
  
 Map<String, String> map = new HashMap<>();  
 map.put("token", "");  
  
 *logger*.info("END: /authenticate completed");  
 return map;  
 }  
}

3. Run the application

Output-

On port

Logs

4. Project Structure