Create authentication service that returns JWT

Code:

AuthenticationController.java

package com.cognizant.springlearnsecure.controller;  
  
import com.cognizant.springlearnsecure.security.JwtUtil;  
import jakarta.servlet.http.HttpServletRequest;  
import org.springframework.http.ResponseEntity;  
import org.springframework.util.StringUtils;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Base64;  
  
@RestController  
public class AuthenticationController {  
  
 @RequestMapping(value = "/authenticate", method = RequestMethod.*GET*)  
 public ResponseEntity<?> authenticate(HttpServletRequest request) {  
 String authHeader = request.getHeader("Authorization");  
  
 if (authHeader == null || !authHeader.startsWith("Basic ")) {  
 return ResponseEntity.*status*(401).body("Missing or invalid Authorization header");  
 }  
  
 String base64Credentials = authHeader.substring("Basic ".length()).trim();  
 byte[] decoded = Base64.*getDecoder*().decode(base64Credentials);  
 String credentials = new String(decoded);  
 String[] values = credentials.split(":", 2);  
  
 if (values.length != 2) {  
 return ResponseEntity.*status*(400).body("Invalid basic auth format");  
 }  
  
 String username = values[0];  
 String password = values[1];  
  
 // Hardcoded validation  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = JwtUtil.*generateToken*(username);  
 return ResponseEntity.*ok*().body("{\"token\":\"" + token + "\"}");  
 }  
  
 return ResponseEntity.*status*(401).body("Invalid credentials");  
 }  
}

Country.java

package com.cognizant.springlearnsecure.model;  
  
public class Country {  
 private String code;  
 private String name;  
  
 public Country() {  
 }  
  
 public Country(String code, String name) {  
 this.code = code;  
 this.name = name;  
 }  
  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country{" +  
 "code='" + code + '\'' +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

JWUtil.java

package com.cognizant.springlearnsecure.security;  
  
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* = 3600000; // 1 hour  
  
 public static String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date(System.*currentTimeMillis*()))  
 .setExpiration(new Date(System.*currentTimeMillis*() + *EXPIRATION\_TIME*))  
 .signWith(*key*)  
 .compact();  
 }  
}

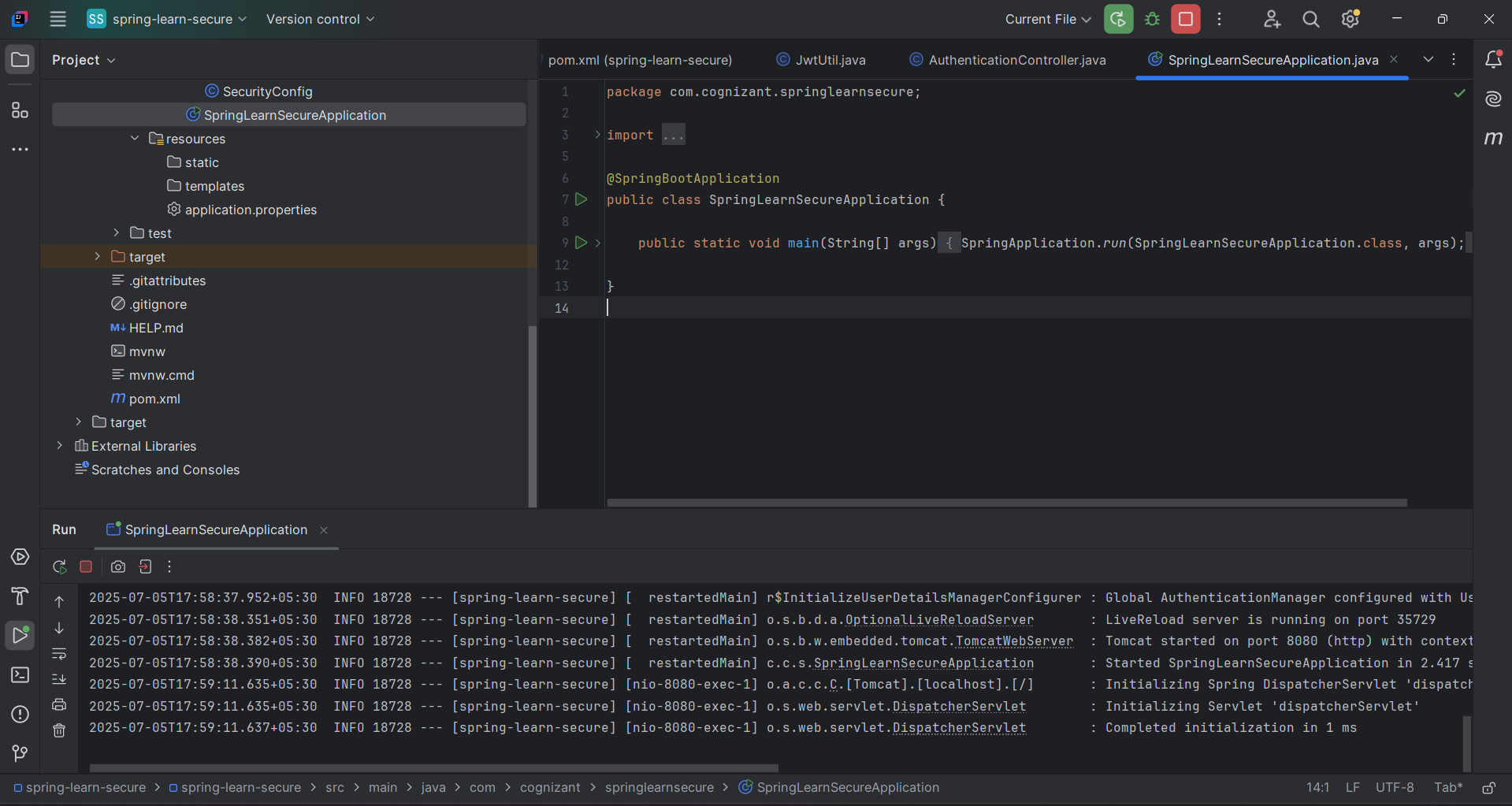
SecurityConfig.java

package com.cognizant.springlearnsecure.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.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.config.Customizer;  
  
@Configuration  
public class SecurityConfig {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(SecurityConfig.class);  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public UserDetailsService userDetailsService(PasswordEncoder encoder) {  
 InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();  
 manager.createUser(User.*withUsername*("user").password(encoder.encode("pwd")).roles("USER").build());  
 manager.createUser(User.*withUsername*("admin").password(encoder.encode("pwd")).roles("ADMIN").build());  
 return manager;  
 }  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http.csrf(AbstractHttpConfigurer::disable)  
 .authorizeHttpRequests(authz -> authz  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 )  
 .httpBasic(Customizer.*withDefaults*());  
  
 return http.build();  
 }  
}

pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-learn-secure</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>spring-learn-secure</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <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>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

Output:



A screenshot of a computer

AI-generated content may be incorrect.