**Create authentication service that returns JWT**

**SpringLearnApplication.java :-**

package com.cognizant.spring\_learn;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.ImportResource;  
  
@SpringBootApplication  
@ComponentScan(basePackages = "com.cognizant.spring\_learn")  
public class SpringLearnApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringLearnApplication.class, args);  
 }  
}

**SecurityConfig.java :-**

package com.cognizant.spring\_learn.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
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.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
//import org.springframework.security.web.util.matcher.AntPathRequestMatcher;  
import org.springframework.security.config.Customizer;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public UserDetailsService userDetailsService() {  
 UserDetails user = User  
 .*withUsername*("user")  
 .password("pwd")  
 .roles("USER")  
 .build();  
 return new InMemoryUserDetailsManager(user);  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 // Use NoOp for simplicity in dev/demo only  
 return NoOpPasswordEncoder.*getInstance*();  
 }  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate") // Preferred  
 .permitAll()  
 .anyRequest().authenticated()  
 )  
 .httpBasic(Customizer.*withDefaults*()); // ✅ Updated to avoid deprecation  
  
 return http.build();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration config)  
 throws Exception {  
 return config.getAuthenticationManager();  
 }  
}

**AuthenticationController.java :-**

package com.cognizant.spring\_learn.controller;  
  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import io.jsonwebtoken.security.Keys;  
  
import jakarta.servlet.http.HttpServletRequest;  
import javax.crypto.SecretKey;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.nio.charset.StandardCharsets;  
import java.util.Base64;  
import java.util.Collections;  
import java.util.Date;  
  
@RestController  
public class AuthenticationController {  
  
 private final SecretKey key;  
  
 public AuthenticationController() {  
 // A secure 256-bit key encoded in Base64 (generated once)  
 String secret = "MuJv8KNCf3zi+eZpQ/hhM0aZKskQy7WbwvIM1ZhK6YI=";  
 byte[] decodedKey = Base64.*getDecoder*().decode(secret);  
 this.key = Keys.*hmacShaKeyFor*(decodedKey);  
 }  
  
 @GetMapping("/authenticate")  
 public ResponseEntity<?> authenticate(HttpServletRequest request) {  
 String header = request.getHeader("Authorization");  
  
 if (header == null || !header.startsWith("Basic ")) {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("Missing or invalid Authorization header");  
 }  
  
 // Decode Base64 and extract username:password  
 String base64Credentials = header.substring("Basic ".length());  
 byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);  
 String credentials = new String(credDecoded, StandardCharsets.*UTF\_8*);  
 String[] values = credentials.split(":", 2);  
  
 String username = values[0];  
 String password = values[1];  
  
 // Dummy authentication logic (replace with real user validation)  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour  
 .signWith(key, SignatureAlgorithm.*HS256*)  
 .compact();  
  
 return ResponseEntity.*ok*(Collections.*singletonMap*("token", token));  
 }  
  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("Invalid credentials");  
 }  
}

**JwtUtil.java :-**

package com.cognizant.spring\_learn.util;  
  
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* = "secret123";  
 private static final long *EXPIRATION\_TIME* = 1000 \* 60 \* 10; // 10 minutes  
  
 public String generateToken(String username) {  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(new Date(System.*currentTimeMillis*()))  
 .setExpiration(new Date(System.*currentTimeMillis*() + *EXPIRATION\_TIME*))  
 .signWith(SignatureAlgorithm.*HS256*, *SECRET\_KEY*)  
 .compact();  
 }  
}

**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</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>spring\_learn</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>21</java.version>  
 </properties>  
 <dependencies>  
 <!-- Spring Web -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <!-- Spring Security -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
  
 <!-- JWT -->  
  
 <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> <!-- for JSON serialization -->  
 <version>0.11.5</version>  
 <scope>runtime</scope>  
 </dependency>  
  
 <!-- pom.xml -->  
  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Output :-**



