**WEEK 4 : JWT HANDSON**

**SUPERSET ID : 6386707**

**Create authentication service that returns JWT**

**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.example</groupId>  
 <artifactId>Create-authentication-service-that-returns-JWT</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Create-authentication-service-that-returns-JWT</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>  
 <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.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>io.jsonwebtoken</groupId>  
 <artifactId>jjwt</artifactId>  
 <version>0.9.1</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**config/**

**SecurityConfig:**

package com.example.Create.authentication.service.that.returns.JWT.config;  
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;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 );  
 return http.build();  
 }  
}

**controller/**

**AuthenticationController:**

package com.example.Create.authentication.service.that.returns.JWT.controller;  
  
  
import com.example.Create.authentication.service.that.returns.JWT.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import com.example.Create.authentication.service.that.returns.JWT.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Base64;  
  
@RestController  
public class AuthenticationController {  
  
 @Autowired  
 private JwtUtil jwtUtil;  
  
 @RequestMapping("/authenticate")  
 public ResponseEntity<?> authenticate(@RequestHeader(HttpHeaders.*AUTHORIZATION*) String authHeader) {  
 if (authHeader != null && authHeader.startsWith("Basic ")) {  
 String base64Credentials = authHeader.substring("Basic ".length()).trim();  
 byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);  
 String credentials = new String(credDecoded);  
 String[] userDetails = credentials.split(":", 2);  
 String username = userDetails[0];  
 String password = userDetails[1];  
  
 // Hardcoded check  
 if ("user".equals(username) && "pwd".equals(password)) {  
 String token = jwtUtil.generateToken(username);  
 return ResponseEntity.*ok*().body("{\"token\":\"" + token + "\"}");  
 }  
 }  
  
 return ResponseEntity.*status*(401).body("Unauthorized");  
 }  
}

**util/**

**JwtUtil:**

package com.example.Create.authentication.service.that.returns.JWT.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* = "secret";  
 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();  
 }  
}

**CreateAuthenticationServiceThatReturnsJwtApplication:**

package com.example.Create.authentication.service.that.returns.JWT;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class CreateAuthenticationServiceThatReturnsJwtApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CreateAuthenticationServiceThatReturnsJwtApplication.class, args);  
 }  
  
}

**Appication Properties:**

server.port=8090