**Exercise 1: Create authentication service that returns JWT**

Code:

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>jwt-auth-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>jwt-auth-service</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>

<!-- Spring Boot Starter Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- Lombok -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.32</version>

<scope>provided</scope>

</dependency>

<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>

<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>

User.java

**package** com.example.demo.model;

**public** **class** User {

**private** String username;

**private** String password;

// Constructors

**public** User() {}

**public** User(String username, String password) {

**this**.username = username;

**this**.password = password;

}

// Getters and Setters

**public** String getUsername() { **return** username; }

**public** **void** setUsername(String username) { **this**.username = username; }

**public** String getPassword() { **return** password; }

**public** **void** setPassword(String password) { **this**.password = password; }

}

JwtUtil.java

**package** com.example.demo.util;

**import** java.util.Date;

**import** org.springframework.stereotype.Component;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** io.jsonwebtoken.security.Keys;

**import** javax.crypto.SecretKey;

@Component

**public** **class** JwtUtil {

**private** **final** SecretKey secretKey = Keys.*secretKeyFor*(SignatureAlgorithm.***HS256***); // generates a secure 256-bit key

// Generate token for user

**public** String generateToken(String username) {

**return** Jwts.*builder*()

.setSubject(username)

.setIssuedAt(**new** Date(System.*currentTimeMillis*()))

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

.signWith(secretKey)

.compact();

}

// Validate the token (basic structure only)

**public** **boolean** validateToken(String token, String username) {

**final** String extractedUsername = extractUsername(token);

**return** (extractedUsername.equals(username) && !isTokenExpired(token));

}

// Extract username from token

**public** String extractUsername(String token) {

**return** Jwts.*parserBuilder*()

.setSigningKey(secretKey)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

}

// Check if the token is expired

**public** **boolean** isTokenExpired(String token) {

**final** Date expiration = Jwts.*parserBuilder*()

.setSigningKey(secretKey)

.build()

.parseClaimsJws(token)

.getBody()

.getExpiration();

**return** expiration.before(**new** Date());

}

}

AuthController.java

**package** com.example.demo.controller;

**import** com.example.demo.util.JwtUtil;

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

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

**import** java.util.Base64;

@RestController

**public** **class** AuthController {

@Autowired

**private** JwtUtil jwtUtil;

@GetMapping("/authenticate")

**public** String authenticate(@RequestHeader("Authorization") String authHeader) {

// Decode Basic Auth

String base64Credentials = authHeader.substring("Basic ".length());

String credentials = **new** String(Base64.*getDecoder*().decode(base64Credentials));

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

// Dummy check (you can use DB here)

**if** ("user".equals(username) && "pwd".equals(password)) {

String token = jwtUtil.generateToken(username);

**return** "{\"token\":\"" + token + "\"}";

} **else** {

**return** "Invalid credentials!";

}

}

}

SecurityConfig.java

**package** com.example.demo.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~~().disable()

.authorizeHttpRequests(authz -> authz

.anyRequest().permitAll()

);

**return** http.build();

}

}

JwtAuthServiceApplication.java

**package** com.example.demo;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** JwtAuthServiceApplication {

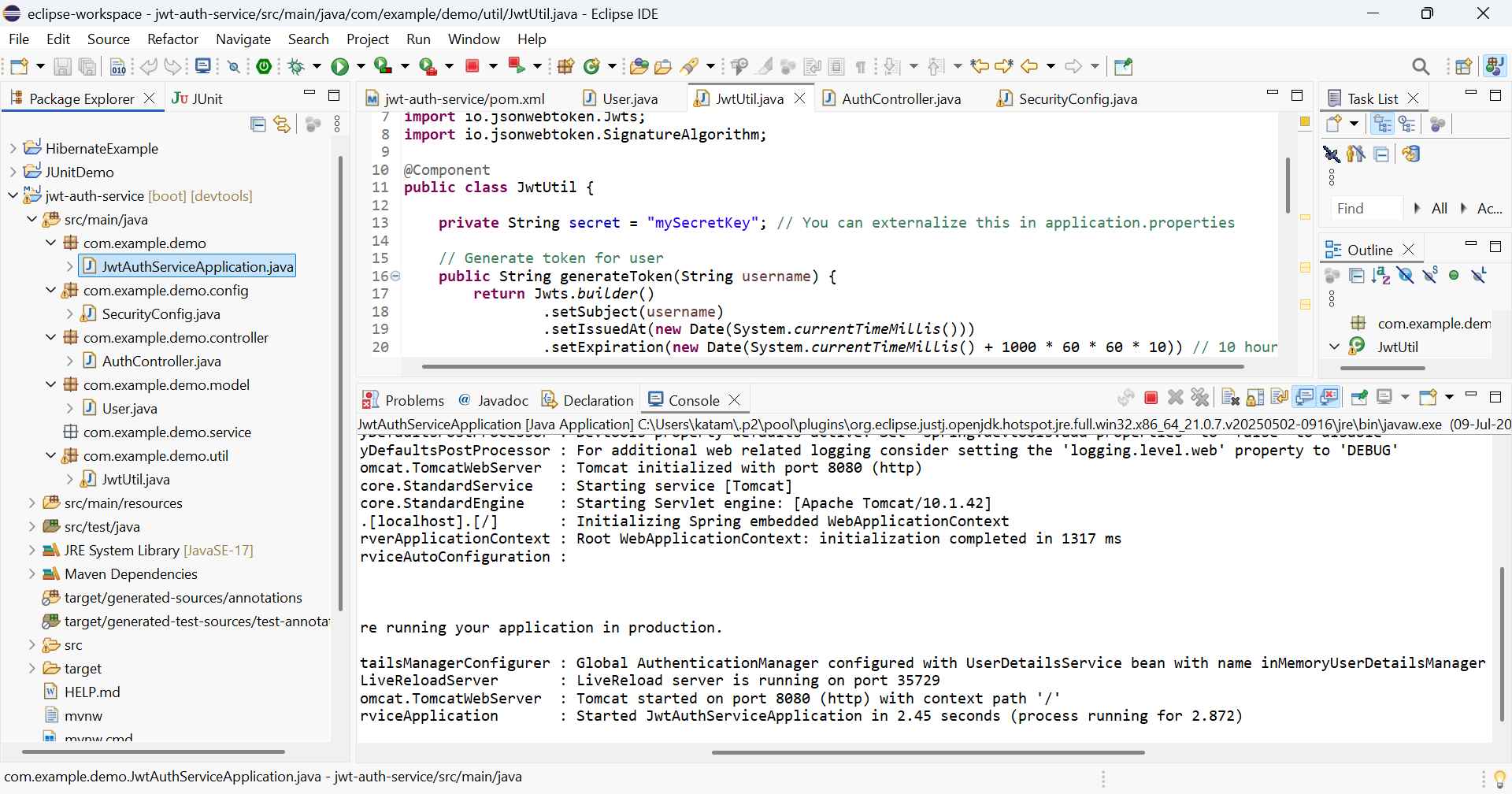
**public** **static** **void** main(String[] args) {

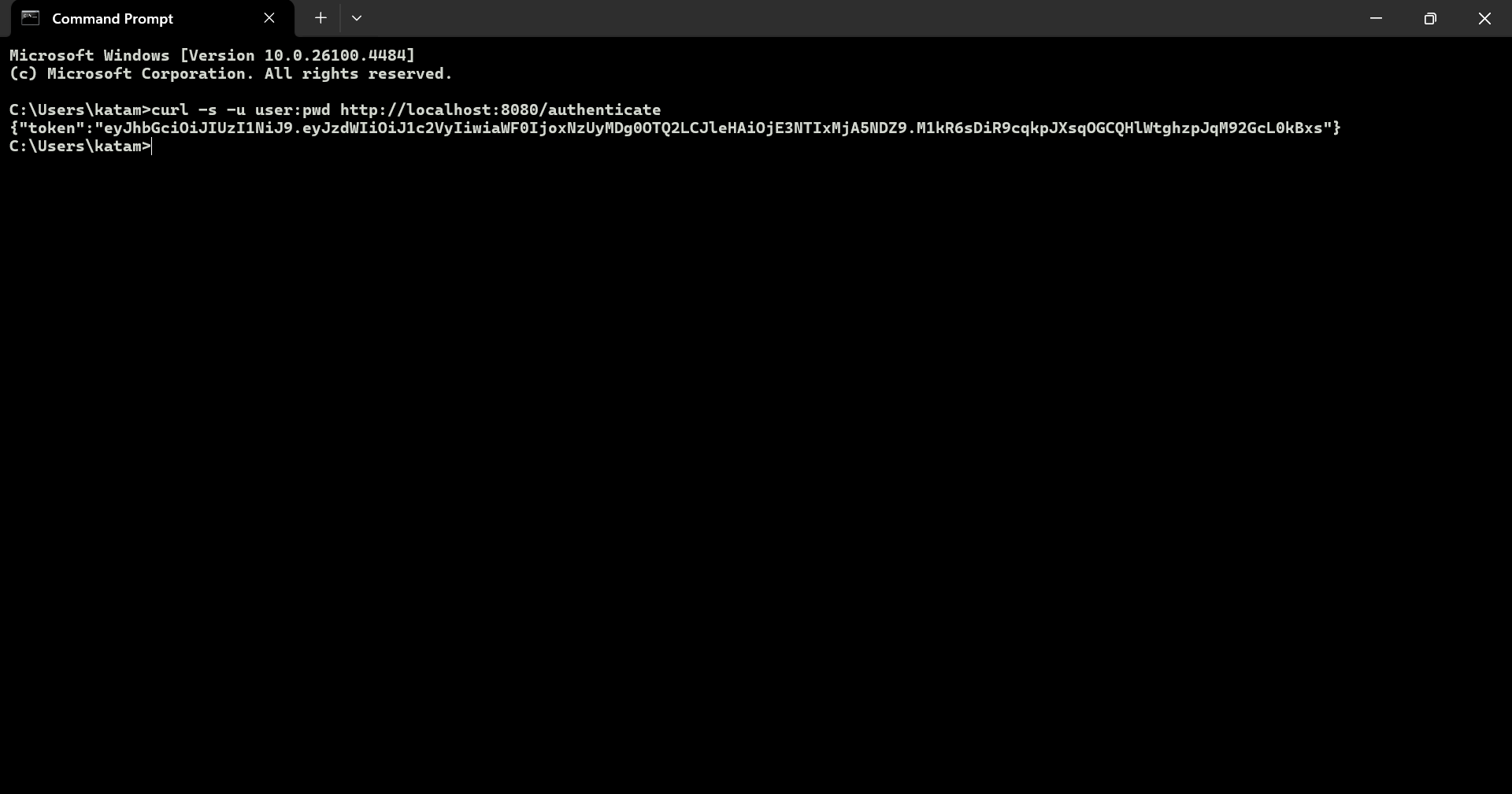
SpringApplication.*run*(JwtAuthServiceApplication.**class**, args);

}

}

Output:

****

****