**WEEK 4 (6397530 – Nayudu Praveen Kumar)**

**Create authentication service that returns JWT**

**CODE:**

**JwtAuthApplication.java:**

package com.example.jwtauth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

public static void main(String[] args) {

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**AuthController.java:**

package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.Base64;

import java.util.Collections;

@RestController

public class AuthController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {

try {

if (authHeader == null || !authHeader.startsWith("Basic ")) {

return ResponseEntity.status(HttpStatus.UNAUTHORIZED)

.body("Missing or invalid Authorization header");

}

// Decode and extract username:password

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

byte[] decodedBytes = Base64.getDecoder().decode(base64Credentials);

String decoded = new String(decodedBytes);

String[] parts = decoded.split(":", 2);

if (parts.length != 2) {

return ResponseEntity.status(HttpStatus.BAD\_REQUEST)

.body("Invalid Basic Auth format");

}

String username = parts[0];

String token = jwtUtil.generateToken(username);

return ResponseEntity.ok(Collections.singletonMap("token", token));

} catch (Exception e) {

e.printStackTrace(); // for console

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR)

.body("Error: " + e.getMessage());

}

}

}

**JwtUtil.java:**

package com.example.jwtauth.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import java.security.Key;

import java.util.Date;

import org.springframework.stereotype.Component;

@Component

public class JwtUtil {

private final Key secretKey = Keys.secretKeyFor(SignatureAlgorithm.HS256); // generate random key

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();

}

public String extractUsername(String token) {

return Jwts.parserBuilder()

.setSigningKey(secretKey)

.build()

.parseClaimsJws(token)

.getBody()

.getSubject();

}

}

**SecurityConfig.java:**

package com.example.jwtauth.security;

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.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@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() {

return NoOpPasswordEncoder.getInstance();

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.httpBasic();

return http.build();

}

}

**AuthResponse.java:**

package com.example.jwtauth.model;

public class AuthResponse {

private String token;

public AuthResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

}

**pom.xml:**

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>jwtauth</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<name>JWT Auth</name>

<description>Spring Boot JWT Authentication</description>

<parent>

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

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

<version>3.2.4</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>17</java.version>

<jjwt.version>0.11.5</jjwt.version>

</properties>

<dependencies>

<!-- Spring Boot 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>

<!-- JJWT API -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>${jjwt.version}</version>

</dependency>

<!-- JJWT Implementation -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>${jjwt.version}</version>

<scope>runtime</scope>

</dependency>

<!-- JJWT Jackson Serializer -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>${jjwt.version}</version>

<scope>runtime</scope>

</dependency>

<!-- Lombok (optional but recommended for cleaner code) -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<!-- For testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

</plugin>

</plugins>

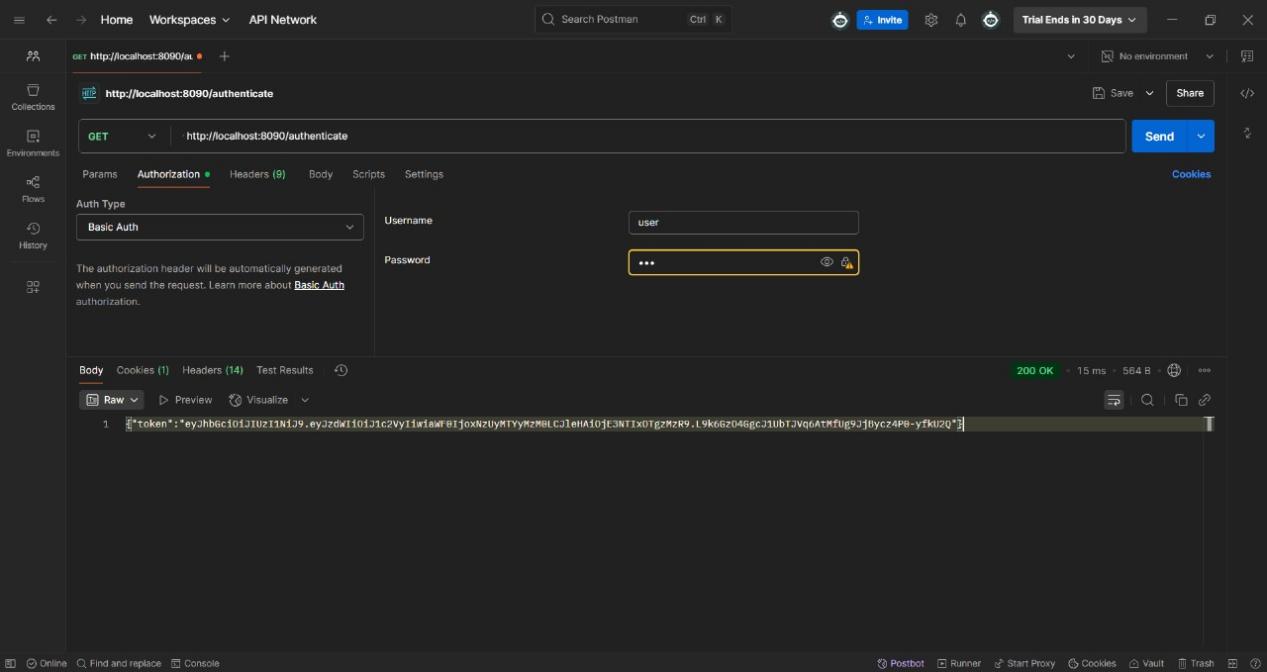
</build>

</project>

**application.properties:**

Server.port=8090

**OUTPUT:**

****

