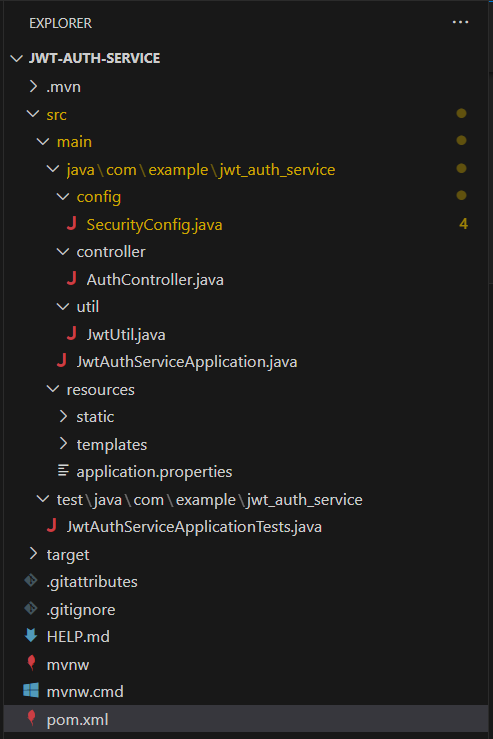
WEEK-4

**5. JWT-handson**

**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>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>

        <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>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>

    </dependencies>

    <build>

        <plugins>

            <plugin>

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

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**SecurityConfig.java:**

package com.example.jwt\_auth\_service.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.ProviderManager;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

    @Bean

    public UserDetailsService userDetailsService() {

        return new InMemoryUserDetailsManager(

            User.withUsername("user")

                .password("{noop}pwd") // {noop} means no encoding

                .roles("USER")

                .build()

        );

    }

    @Bean

    public AuthenticationManager authenticationManager(UserDetailsService userDetailsService) {

        DaoAuthenticationProvider provider = new DaoAuthenticationProvider();

        provider.setUserDetailsService(userDetailsService);

        return new ProviderManager(provider);

    }

    @Bean

    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

        http.csrf().disable()

            .authorizeHttpRequests(auth -> auth

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

                .anyRequest().authenticated()

            );

        return http.build();

    }

}

**AuthController.java:**

package com.example.jwt\_auth\_service.controller;

import com.example.jwt\_auth\_service.util.JwtUtil;

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

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

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

import java.util.Base64;

import jakarta.servlet.http.HttpServletRequest;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthController {

    @Autowired

    private AuthenticationManager authenticationManager;

    @GetMapping("/authenticate")

    public Map<String, String> authenticate(HttpServletRequest request) {

        String header = request.getHeader("Authorization");

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

            throw new RuntimeException("Missing or invalid Authorization header.");

        }

        String[] credentials = new String(Base64.getDecoder().decode(header.substring(6))).split(":");

        String username = credentials[0];

        String password = credentials[1];

        Authentication authentication = authenticationManager.authenticate(

                new UsernamePasswordAuthenticationToken(username, password));

        if (authentication.isAuthenticated()) {

            String token = JwtUtil.generateToken(username);

            Map<String, String> response = new HashMap<>();

            response.put("token", token);

            return response;

        } else {

            throw new RuntimeException("Invalid credentials");

        }

    }

}

**JwtUtil.java:**

package com.example.jwt\_auth\_service.util;

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 = 1000 \* 60 \* 10; // 10 minutes

    public static String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

                .setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

                .signWith(key)

                .compact();

    }

}

**JwtAuthServiceApplication.java:**

package com.example.jwt\_auth\_service;

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);

    }

}

**application.properties:**

spring.application.name=jwt-auth-service

server.port=8090

**JwtAuthServiceApplicationTests.java:**

package com.example.jwt\_auth\_service;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

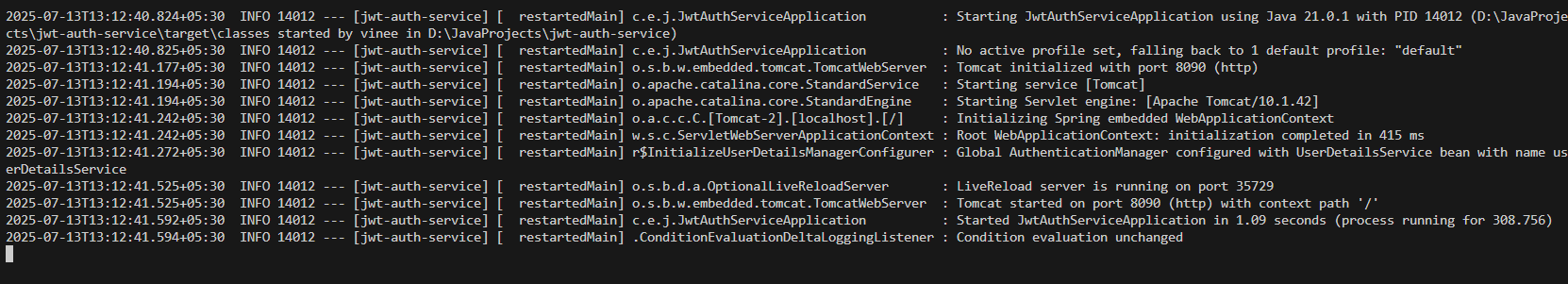
@SpringBootTest

class JwtAuthServiceApplicationTests {

    @Test

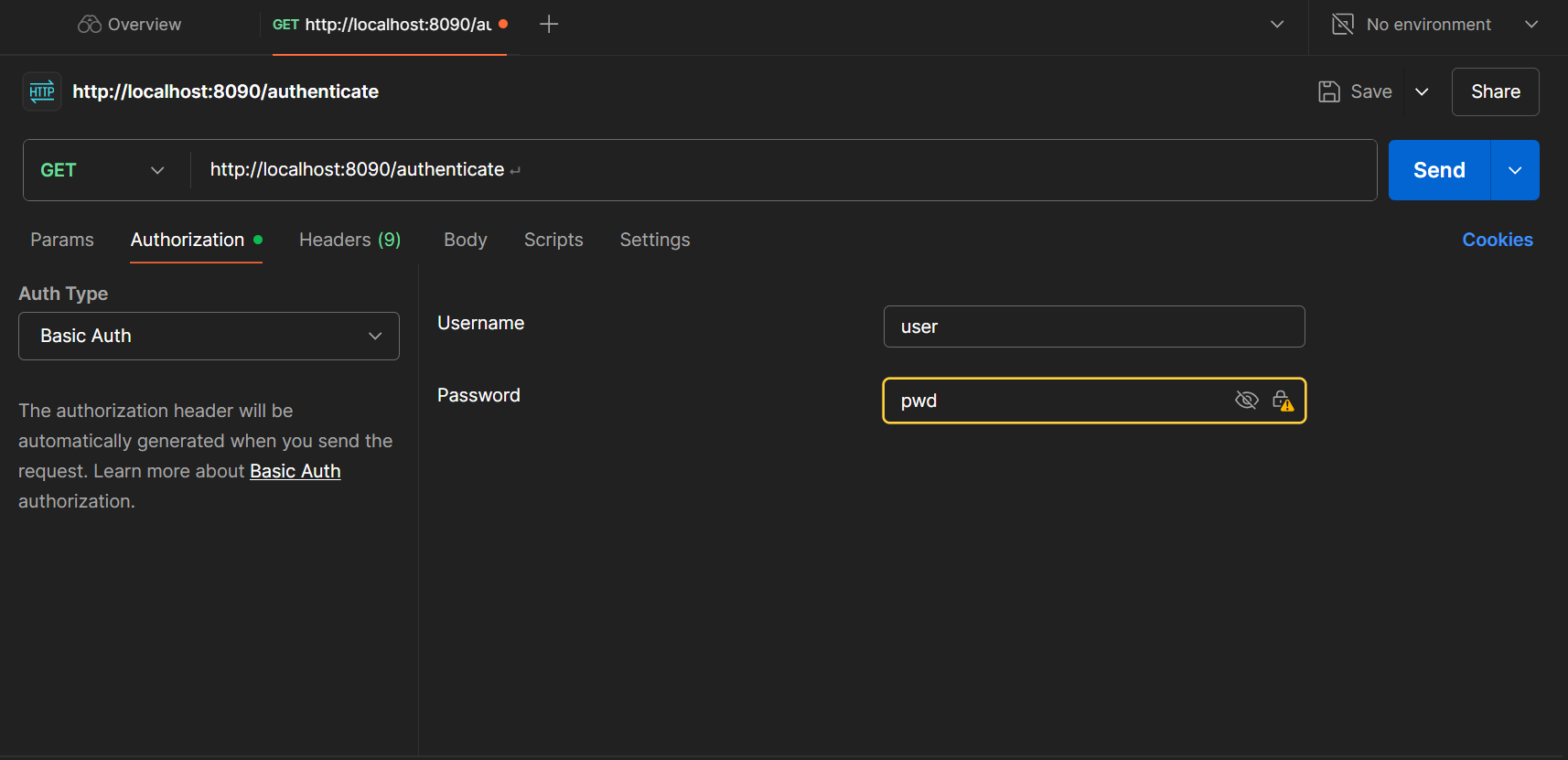
    void contextLoads() {

    }

}

(Tomcat started on port 8090 can be seen in the above terminal)

Tested through Postman:



Token generated successfully:  
