**Create authentication service that returns JWT**

**SecurityConfig.java**

package com.example.jwtauth.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()

            .anyRequest().permitAll();

        return http.build();

    }

}

**AuthenticationController.java**

package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;

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

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestHeader;

import org.springframework.web.bind.annotation.RestController;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.Map;

@RestController

public class AuthenticationController {

    @Autowired

    private JwtUtil jwtUtil;

    @GetMapping("/authenticate")

    public ResponseEntity<?> authenticate(@RequestHeader(HttpHeaders.AUTHORIZATION) String authHeader) {

        if (authHeader != null && authHeader.toLowerCase().startsWith("basic")) {

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

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

            String credentials = new String(credDecoded, StandardCharsets.UTF\_8);

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

            if ("user".equals(values[0]) && "pwd".equals(values[1])) {

                String token = jwtUtil.generateToken(values[0]);

                return ResponseEntity.ok(Map.of("token", token));

            }

        }

        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();

    }

}

**JwtUtil.java**

package com.example.jwtauth.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 = "mysecretkey";

    private static final long EXPIRATION\_TIME = 1000 \* 60 \* 10; // 10 minutes

    public String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

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

                .signWith(SignatureAlgorithm.HS256, SECRET\_KEY.getBytes())

                .compact();

    }

}

**JavaAuthApplication.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);

    }

}

**SecurityConfig.class**

package com.example.jwtauth.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.config.annotation.web.configurers.AuthorizeHttpRequestsConfigurer;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

   public SecurityConfig() {

   }

   @Bean

   public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

      ((AuthorizeHttpRequestsConfigurer.AuthorizedUrl)((HttpSecurity)http.csrf().disable()).authorizeHttpRequests().anyRequest()).permitAll();

      return (SecurityFilterChain)http.build();

   }

}

**AuthenticationController.class**

package com.example.jwtauth.controller;

import com.example.jwtauth.util.JwtUtil;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.Map;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestHeader;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class AuthenticationController {

   @Autowired

   private JwtUtil jwtUtil;

   public AuthenticationController() {

   }

   @GetMapping({"/authenticate"})

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

      if (authHeader != null && authHeader.toLowerCase().startsWith("basic")) {

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

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

         String credentials = new String(credDecoded, StandardCharsets.UTF\_8);

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

         if ("user".equals(values[0]) && "pwd".equals(values[1])) {

            String token = this.jwtUtil.generateToken(values[0]);

            return ResponseEntity.ok(Map.of("token", token));

         }

      }

      return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();

   }

}

**JavaUtil.class**

package com.example.jwtauth.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

import org.springframework.stereotype.Component;

@Component

public class JwtUtil {

   private static final String SECRET\_KEY = "mysecretkey";

   private static final long EXPIRATION\_TIME = 600000L;

   public JwtUtil() {

   }

   public String generateToken(String username) {

      return Jwts.builder().setSubject(username).setIssuedAt(new Date()).setExpiration(new Date(System.currentTimeMillis() + 600000L)).signWith(SignatureAlgorithm.HS256, "mysecretkey".getBytes()).compact();

   }

}

**JavaAuthApplication.class**

package com.example.jwtauth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication {

   public JwtAuthApplication() {

   }

   public static void main(String[] args) {

      SpringApplication.run(JwtAuthApplication.class, args);

   }

}

**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>jwt-auth-service</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>jwt-auth-service</name>

    <description>JWT Auth Service</description>

    <packaging>jar</packaging>

    <parent>

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

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

        <version>3.1.2</version>

        <relativePath/>

    </parent>

    <properties>

        <java.version>17</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-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>

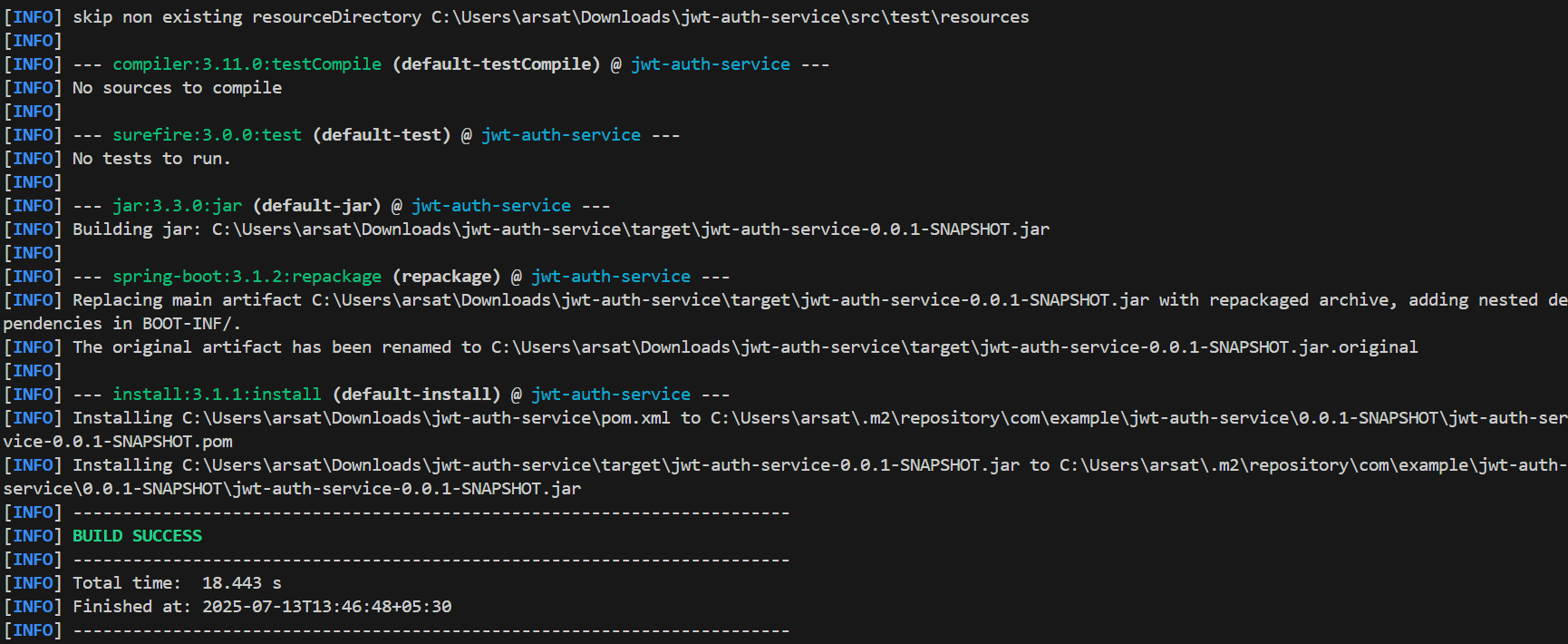
            </plugin>

        </plugins>

    </build>

</project>

**OUTPUT :**

****