**Create authentication service that returns JWT**   
  
As part of first step of JWT process, the user credentials needs to be sent to authentication service request that generates and returns the JWT.  
  
Ideally when the below curl command is executed that calls the new authentication service, the token should be responded. Kindly note that the credentials are passed using -u option.

package com.cognizant.spring\_learn.util;

import org.springframework.stereotype.Component;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

import java.util.function.Function;

@Component

public class JwtUtil {

    private final String secret = "secret"; // keep secret safe in production

    private final long expiration = 1000 \* 60 \* 60; // 1 hour

    // 1. Generate Token

    public String generateToken(String username) {

        return Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

                .setExpiration(new Date(System.currentTimeMillis() + expiration))

                .signWith(SignatureAlgorithm.HS256, secret)

                .compact();

    }

    // 2. Extract Username

    public String extractUsername(String token) {

        return extractClaim(token, Claims::getSubject);

    }

    // 3. Extract any claim

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {

        final Claims claims = extractAllClaims(token);

        return claimsResolver.apply(claims);

    }

    // 4. Extract claims internally

    private Claims extractAllClaims(String token) {

        return Jwts.parser()

                .setSigningKey(secret)

                .parseClaimsJws(token)

                .getBody();

    }

    // 5. Validate Token

    public boolean validateToken(String token, String username) {

        final String extractedUsername = extractUsername(token);

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

    }

    // 6. Check token expiration

    private boolean isTokenExpired(String token) {

        final Date expiration = extractClaim(token, Claims::getExpiration);

        return expiration.before(new Date());

    }

}

package com.cognizant.spring\_learn.config;

import com.cognizant.spring\_learn.filter.JwtRequestFilter;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration

public class SecurityConfig {

    private final JwtRequestFilter jwtRequestFilter;

    public SecurityConfig(JwtRequestFilter jwtRequestFilter) {

        this.jwtRequestFilter = jwtRequestFilter;

    }

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http.csrf().disable()

            .authorizeHttpRequests()

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

            .anyRequest().authenticated()

            .and()

            .addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();

    }

    @Bean

    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

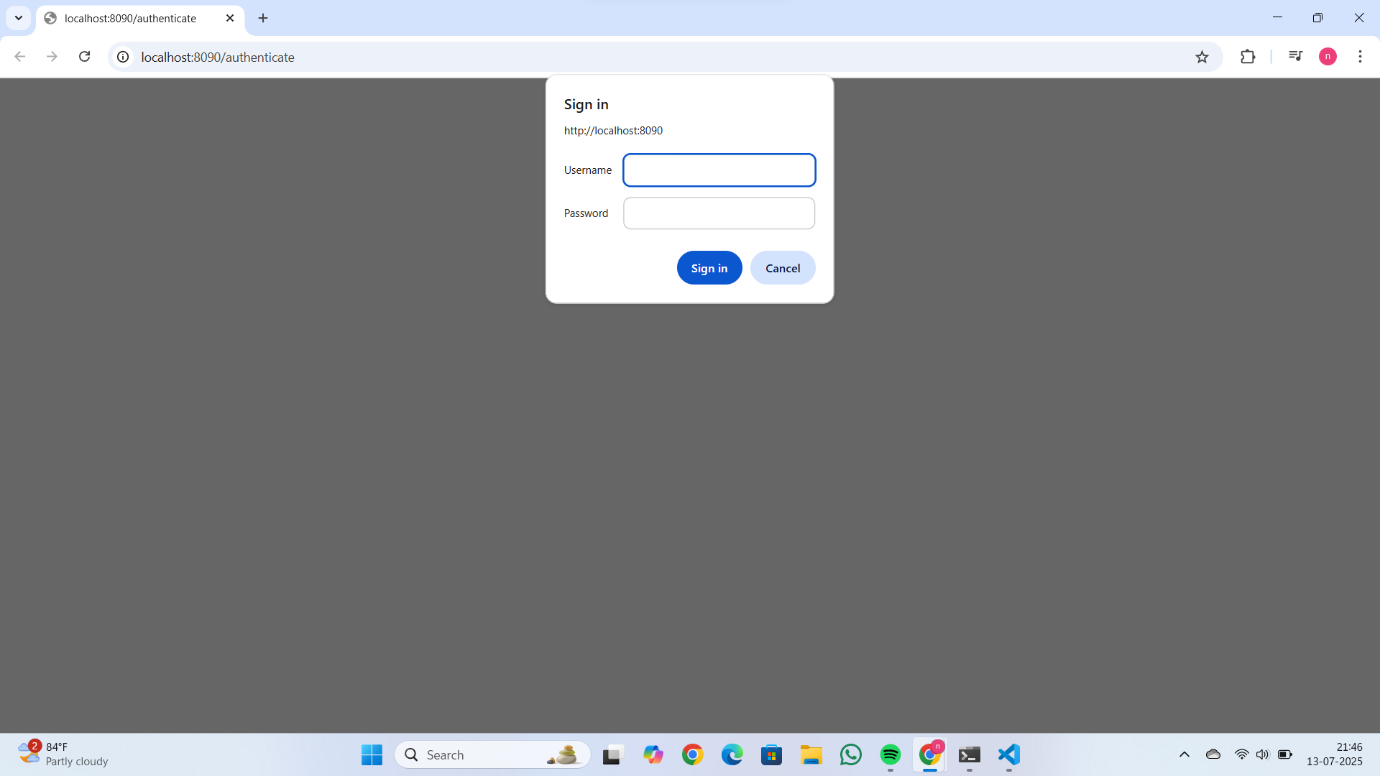
        return config.getAuthenticationManager();

    }

}

**Request**

curl -s -u user:pwd http://localhost:8090/authenticate

****

**Response**

{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

