**Ex :   
  
Solution:**

AuthenticateController.java –

package com.cognizant.jwt.jwt\_auth\_demo.controller;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

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

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

import java.util.Date;

@RestController

public class AuthenticationController {

    private final String SECRET\_KEY = "mysecretkey123";

    @RequestMapping("/authenticate")

    public String generateToken(HttpServletRequest request) {

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

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

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

        }

        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];

        // basic chek

        if (!"user".equals(username) || !"pwd".equals(password)) {

            throw new RuntimeException("Invalid credentials.");

        }

        String token = Jwts.builder()

                .setSubject(username)

                .setIssuedAt(new Date())

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

                .signWith(SignatureAlgorithm.HS256, SECRET\_KEY)

                .compact();

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

    }

}

SequrityConfig.java –

package com.cognizant.jwt.jwt\_auth\_demo.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

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.core.userdetails.UserDetailsService;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

    @Bean

    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http

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

            .authorizeHttpRequests(auth -> auth

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

                .anyRequest().authenticated()

            )

            .httpBasic(Customizer.withDefaults());

        return http.build();

    }

    @Bean

    public UserDetailsService userDetailsService() {

        // user:pwd (this matches your curl)

        UserDetails user = User.withUsername("user")

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

                .roles("USER")

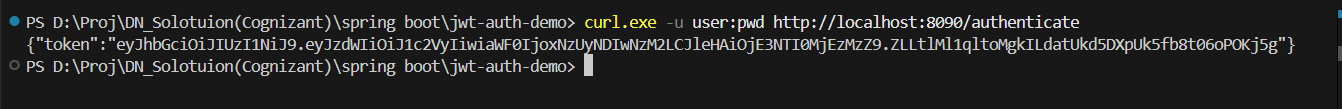
                .build();

        return new InMemoryUserDetailsManager(user);

    }

}

**Output:**

****