**Chirag H M-(6373750)**

**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.  
  
**Request**

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

**Response**

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

This can be incorporated as three major steps:

* Create authentication controller and configure it in SecurityConfig
* Read Authorization header and decode the username and password
* Generate token based on the user retrieved in the previous step

**Solution:**

Add all the necessary dependencies in the pom.xml and create a JwtUnit class inside the util package .

package com.cognizant.spring\_learn.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import org.springframework.stereotype.Component;

import java.security.Key;

import java.util.Date;

*@Component*

public class JwtUtil {

private static final Key ***key*** = Keys.*secretKeyFor*(*SignatureAlgorithm*.***HS256***);

private static final long ***EXPIRATION\_TIME*** = 3600000; // 1 hour

public String generateToken(String username) {

return Jwts.*builder*()

.setSubject(username)

.setIssuedAt(new Date())

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

.signWith(***key***)

.compact();

}

}

**Create a authController class in the controller package**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.util.JwtUtil;

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

import org.springframework.http.ResponseEntity;

import org.springframework.util.Base64Utils;

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

import jakarta.servlet.http.HttpServletRequest;

*@RestController*

public class AuthController {

*@Autowired*

private JwtUtil jwtUtil;

*@GetMapping*("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

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

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

return ResponseEntity.*status*(401).body("Missing Authorization header");

}

// Decode base64 encoded username:password

String[] creds = new String(Base64Utils.decodeFromString(header.substring(6))).split(":");

String username = creds[0];

String password = creds[1];

// For now, use a hardcoded validation (replace with DB/LDAP check in real apps)

if ("user".equals(username) && "pwd".equals(password)) {

String token = jwtUtil.generateToken(username);

return ResponseEntity.*ok*().body("{\"token\":\"" + token + "\"}");

} else {

return ResponseEntity.*status*(401).body("Invalid credentials");

}

}

}

**Create a SecurityConfig class inside the config package.**

package com.cognizant.spring\_learn.config;

import org.springframework.context.annotation.Configuration;

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

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.context.annotation.Bean;

*@Configuration*

public class SecurityConfig {

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests()

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

.anyRequest().authenticated()

.and().httpBasic();

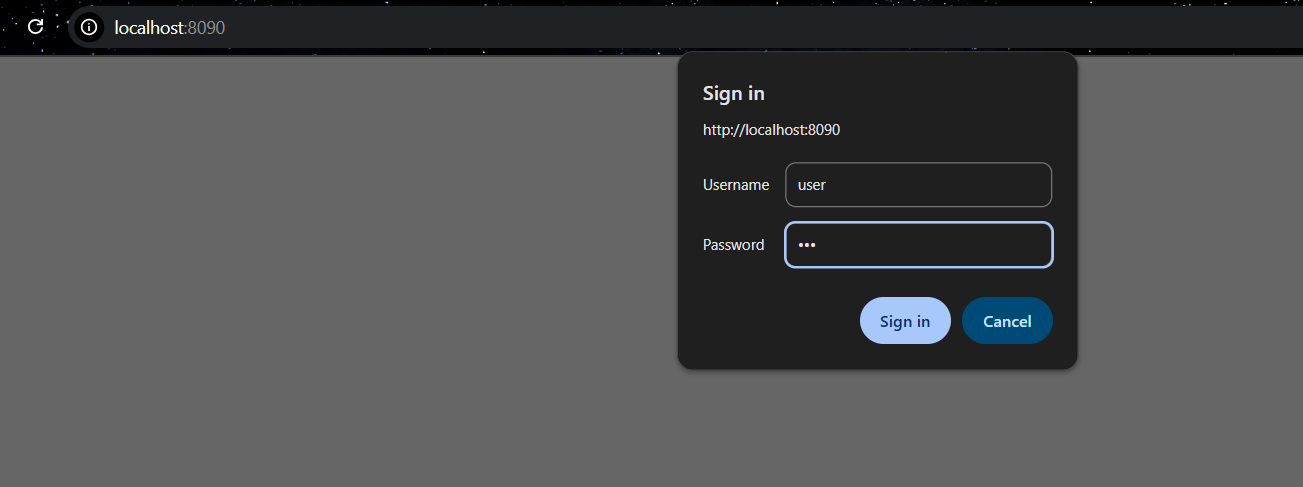
return http.build();

}

}

**Set the server port to :**

server.port=8090



Add the password popped in in your console

Output:

