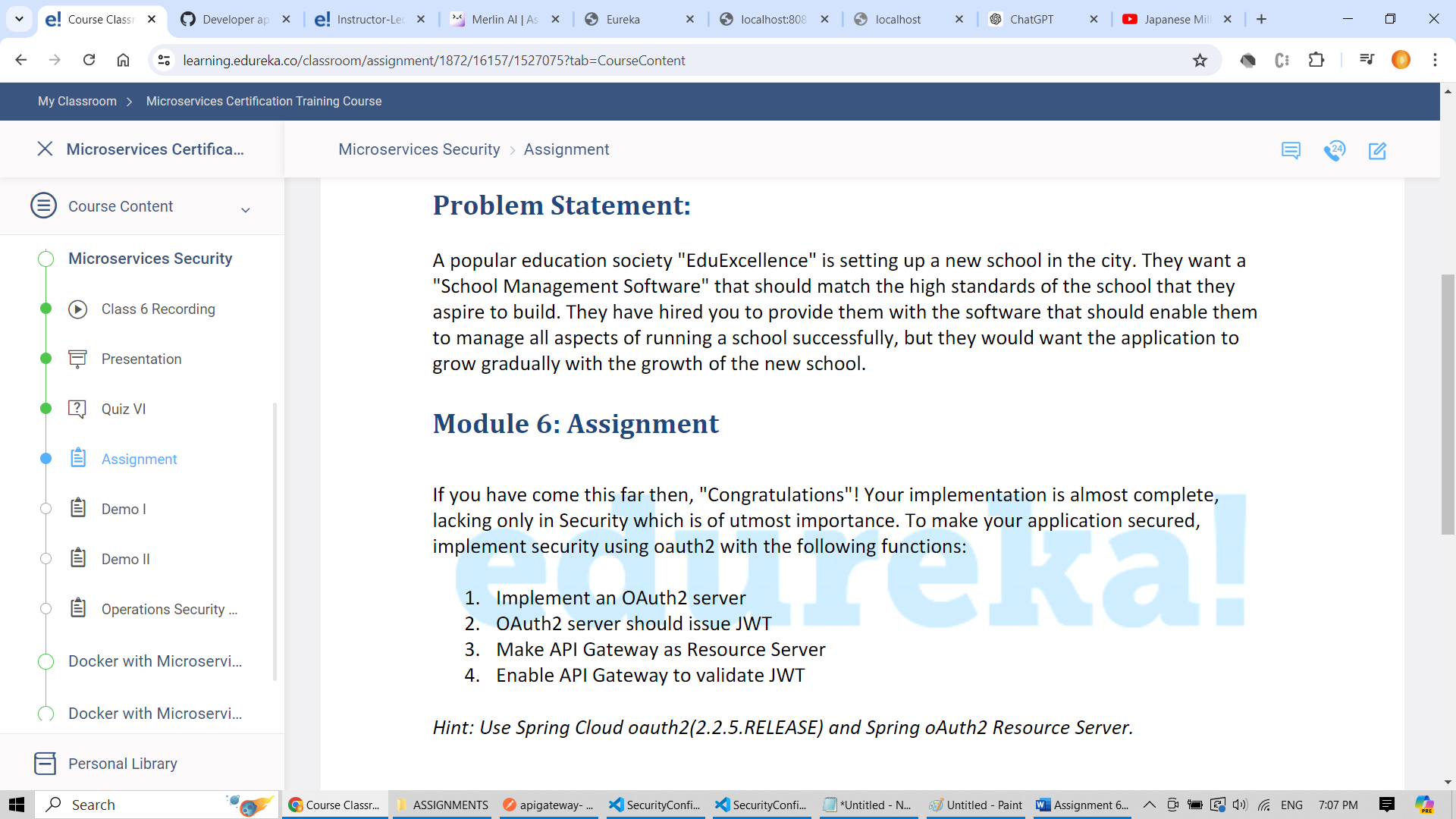
Problem 6:



Step 1: Dependencies for making api gateway as resource server. Added it in pom.xml

        <dependency>

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

            <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>

        </dependency>

Step 2: Added following configuration in apigateway microservice to validate jwt token

package com.example.apigateway;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.annotation.Order;

import org.springframework.security.config.Customizer;

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

import org.springframework.security.config.web.server.ServerHttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.server.SecurityWebFilterChain;

@Configuration

public class ResourceServerConfig {

    @Bean

    @Order(1)

    public SecurityWebFilterChain securityWebFilterChain(ServerHttpSecurity httpSecurity) throws Exception {

        httpSecurity.authorizeExchange().anyExchange().authenticated(); // every request should be authenticate

        httpSecurity.oauth2ResourceServer(c -> c.jwt(j -> j.jwkSetUri("http://localhost:9000/oauth2/jwks")));

        return httpSecurity.build();

    }

}

Step 3: feesms , studentms, edureka registry services already submitted as part of Assignment 5. No change in it.

Step 4: Dependencise needed for oauth server

        <dependency>

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

            <artifactId>spring-boot-starter-oauth2-authorization-server</artifactId>

        </dependency>

        <dependency>

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

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

        </dependency>

Step 5: Added boiler plate code needed in oauth server to support 1. Human users, 2.Non-human users / micro services (only added studentms under oauth surveillance as part of this assignment) . Also added code to generate jwt token, updated security filter chain to support oauth, added code to validate all the requests.

package com.example.oauth\_server;

import com.nimbusds.jose.jwk.JWKSet;

import com.nimbusds.jose.jwk.RSAKey;

import com.nimbusds.jose.jwk.source.JWKSource;

import com.nimbusds.jose.proc.SecurityContext;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.annotation.Order;

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.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.oauth2.core.AuthorizationGrantType;

import org.springframework.security.oauth2.core.ClientAuthenticationMethod;

import org.springframework.security.oauth2.core.oidc.OidcScopes;

import org.springframework.security.oauth2.server.authorization.client.InMemoryRegisteredClientRepository;

import org.springframework.security.oauth2.server.authorization.client.RegisteredClient;

import org.springframework.security.oauth2.server.authorization.client.RegisteredClientRepository;

import org.springframework.security.oauth2.server.authorization.config.annotation.web.configuration.OAuth2AuthorizationServerConfiguration;

import org.springframework.security.oauth2.server.authorization.config.annotation.web.configurers.OAuth2AuthorizationServerConfigurer;

import org.springframework.security.oauth2.server.authorization.settings.AuthorizationServerSettings;

import org.springframework.security.oauth2.server.authorization.settings.ClientSettings;

import org.springframework.security.oauth2.server.authorization.settings.TokenSettings;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

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

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.NoSuchAlgorithmException;

import java.security.interfaces.RSAPrivateKey;

import java.security.interfaces.RSAPublicKey;

import java.time.Duration;

import java.util.UUID;

@Configuration

public class SecurityConfig {

    @Bean

    @Order(1)

    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {

        httpSecurity.csrf().disable();

        OAuth2AuthorizationServerConfiguration.applyDefaultSecurity(httpSecurity);

        httpSecurity.getConfigurer(OAuth2AuthorizationServerConfigurer.class)

                .oidc(Customizer.withDefaults());

        httpSecurity.exceptionHandling(e -> e.authenticationEntryPoint(new LoginUrlAuthenticationEntryPoint("/login")));

        return httpSecurity.build();

    }

    @Bean

    @Order(2)

    public SecurityFilterChain defaultFilterChain(HttpSecurity httpSecurity) throws Exception {

        httpSecurity.formLogin(Customizer.withDefaults());

        httpSecurity.authorizeHttpRequests(c -> c.anyRequest().authenticated()); // every request should be authenticate

        return httpSecurity.build();

    }

    @Bean // Human user

    public UserDetailsService userDetailsService() {

        UserDetails john = User.withUsername("john")

                .password("123")

                .roles("write", "read")

                .build();

        return new InMemoryUserDetailsManager(john);

    }

    @Bean // Non-Human user

    public RegisteredClientRepository registeredClientRepository() {

        RegisteredClient registeredClient = RegisteredClient.withId(UUID.randomUUID().toString())

                .clientId("client")

                .clientSecret("secret")

                .clientAuthenticationMethod(ClientAuthenticationMethod.CLIENT\_SECRET\_BASIC)

                .authorizationGrantType(AuthorizationGrantType.AUTHORIZATION\_CODE)

                .authorizationGrantType(AuthorizationGrantType.REFRESH\_TOKEN)

                .authorizationGrantType(AuthorizationGrantType.CLIENT\_CREDENTIALS)

                .tokenSettings(TokenSettings.builder().accessTokenTimeToLive(Duration.ofHours(2)).build())

                .redirectUri("http://localhost:8081/students")

                .scope(OidcScopes.OPENID)

                .scope("myscope")

                .clientSettings(ClientSettings.builder().requireProofKey(false).build())

                .build();

        return new InMemoryRegisteredClientRepository(registeredClient);

    }

    @Bean

    public JWKSource<SecurityContext> jwkSource() throws NoSuchAlgorithmException {

        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");

        keyPairGenerator.initialize(2048);

        KeyPair keyPair = keyPairGenerator.generateKeyPair();

        RSAPrivateKey privateKey = (RSAPrivateKey) keyPair.getPrivate();

        RSAPublicKey publicKey = (RSAPublicKey) keyPair.getPublic();

        RSAKey rsaKey = new RSAKey.Builder(publicKey)

                .privateKey(privateKey)

                .keyID(UUID.randomUUID().toString())

                .build();

        JWKSet jwkSet = new JWKSet(rsaKey);

        return (jwkSelector, securityContext) -> jwkSelector.select(jwkSet);

    }

    @Bean

    public AuthorizationServerSettings authorizationServerSettings() {

        return AuthorizationServerSettings.builder().build();

    }

    @Bean

    public PasswordEncoder passwordEncoder() {

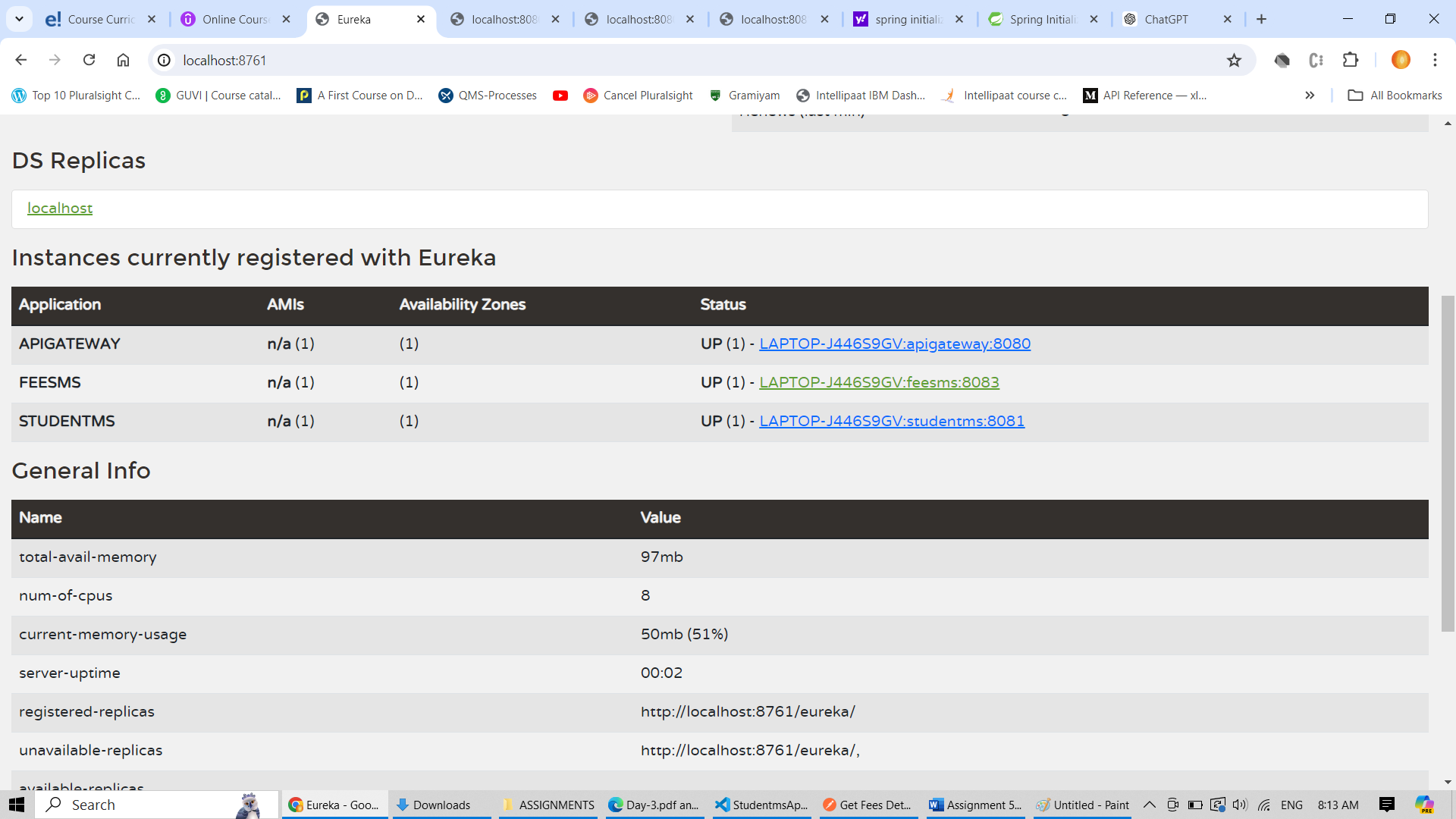
        return NoOpPasswordEncoder.getInstance();

    }

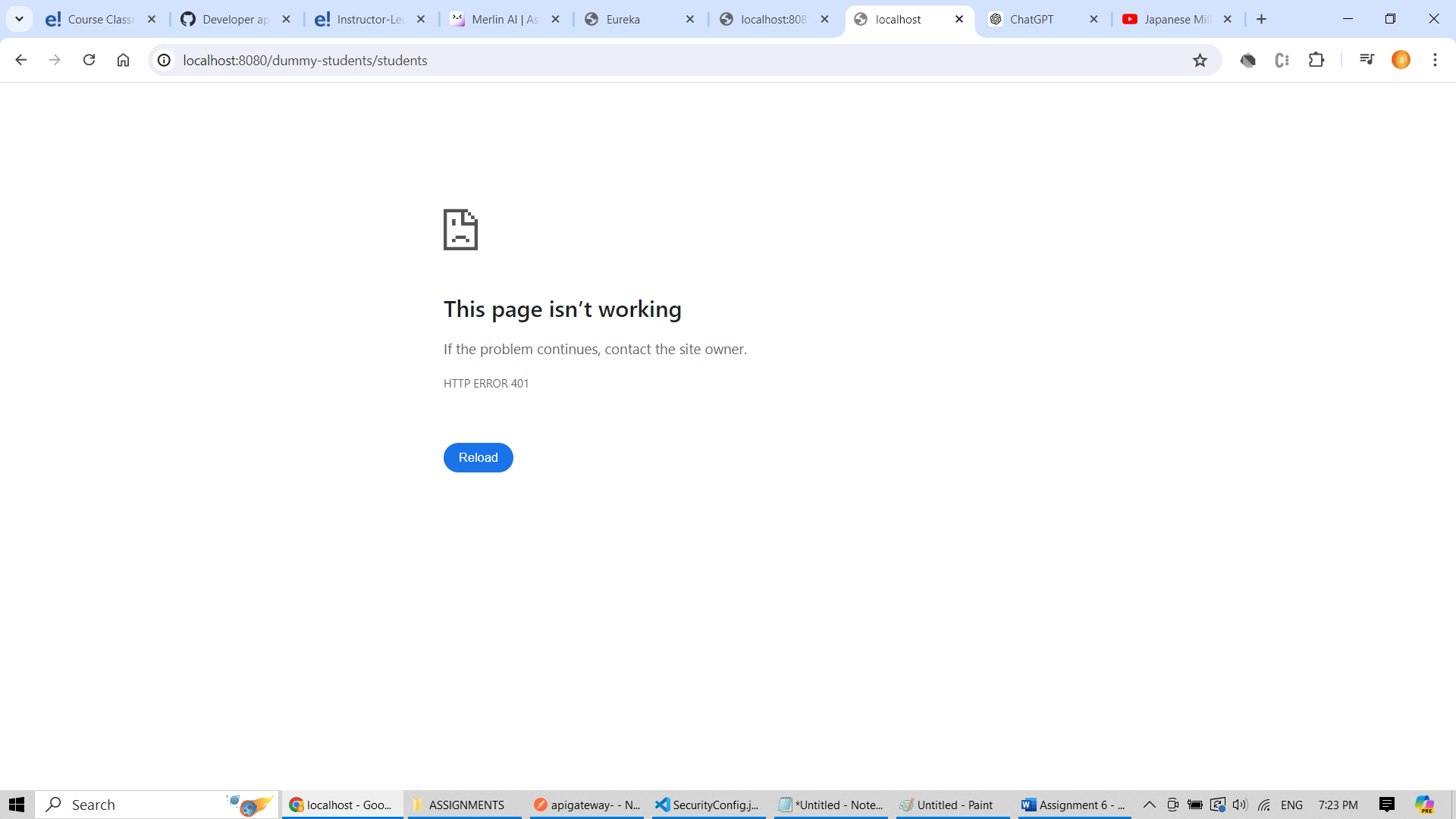
}

Output screenshots

Service discovery could able to identify apigateway, studentms, feesms



Accessing studenms from api gateway without token leads to 401 (unauthorized). This error is due to the addition of authentication , authorization from aouth securit



If Bearer token is added, then the same call is authorized and can get the desired results.

