How to Authenticate API Data Using MSAL for Login

Microsoft Graph requires an OAuth 2.0 access token issued by Microsoft's identity platform (Azure AD).

What is Microsoft Graph API

Microsoft Graph is considered an external API of Microsoft. It is a **RESTful API** that provides a unified interface to interact with a wide variety of Microsoft 365 services and data. It acts as the gateway to access Microsoft services like:

- **Outlook (emails, calendar)**
- **OneDrive (files)**
- **SharePoint**
- **Teams (chat, meetings, collaboration)**
- **Excel**
- **Planner**
- **Azure Active Directory (users, groups, and organizational data)**
- **Microsoft 365 usage reports**

Key Features of Microsoft Graph:

1. **Unified Endpoint:**
   The base URL https://graph.microsoft.com allows you to access multiple services through a single API.
2. **Authentication:**
   Microsoft Graph uses OAuth 2.0 for secure authentication and authorization.
3. **SDK Support:**
   Available SDKs for different languages like **Python, JavaScript, C#, and Java** make integration easier.
4. **Rich Query Capabilities:**
   It supports filters, paging, and batch requests to optimize data retrieval.

Examples of Use Cases:

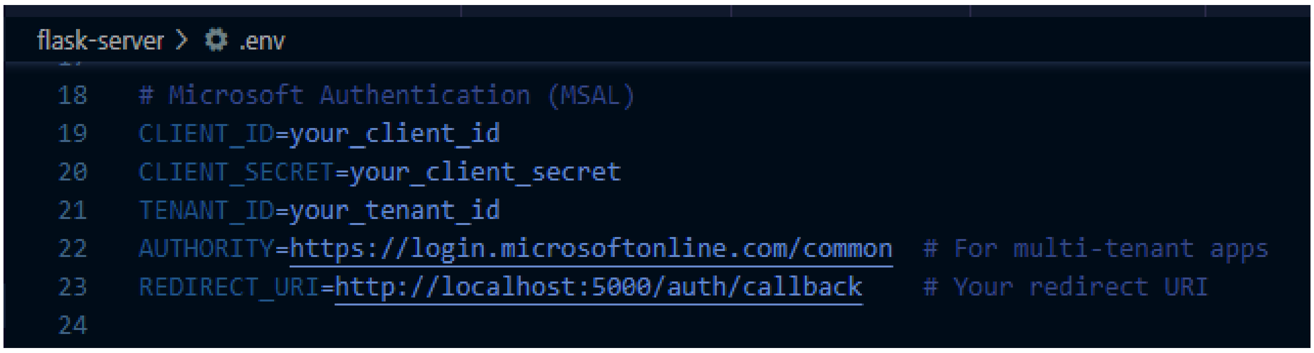
- Automating email notifications.
- Syncing calendar events.
- Managing user profiles and groups in Azure Active Directory.
- Accessing files stored in OneDrive or SharePoint.
- Creating reports or dashboards based on Microsoft 365 data.

If you get the following InvalidAuthenticationToken error, you try to call the Microsoft Graph API with your application's JWT instead of an OAuth access token.

Steps to Obtain an OAuth Access Token

1. Ensure Environment Variables Are Set Correctly

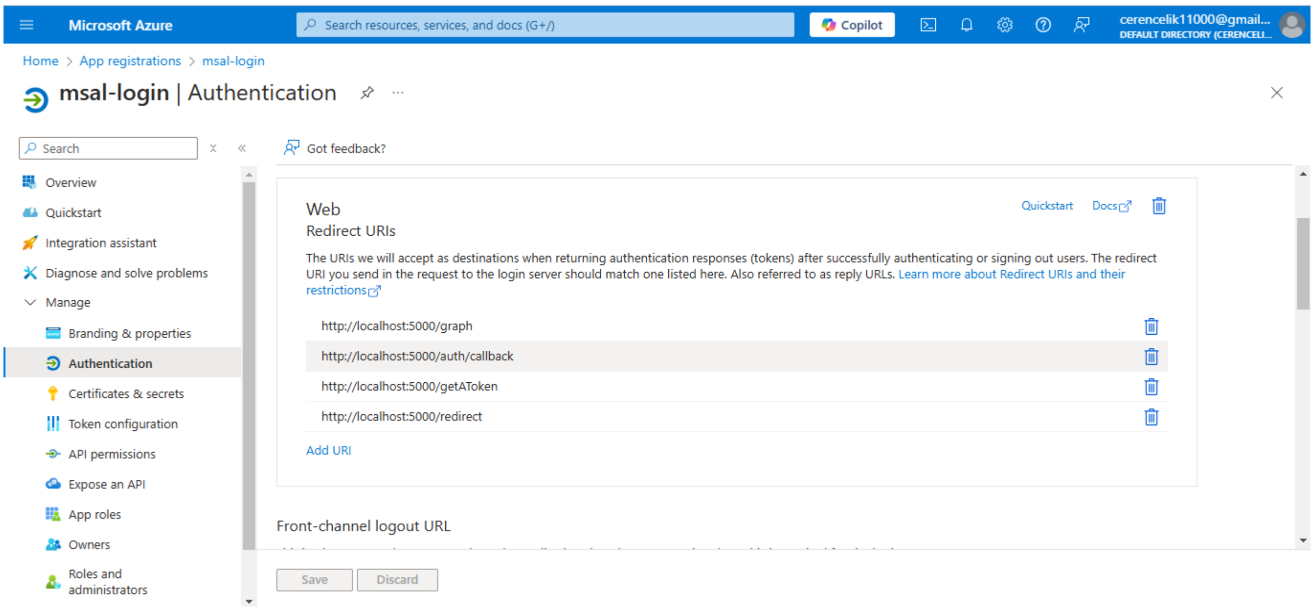
   Make sure your .env file contains the following variables:



flask-server > .env

```
# Microsoft Authentication (MSAL)
CLIENT_ID=your_client_id
CLIENT_SECRET=your_client_secret
TENANT_ID=your_tenant_id
AUTHORITY=https://login.microsoftonline.com/common  # For multi-tenant apps
REDIRECT_URI=http://localhost:5000/auth/callback    # Your redirect URI

```

You can add your redirect URI from Azure Portal – App Registrations – Authentication – Web – Redirect URIs



Web

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. Learn more about Redirect URIs and their restrictions

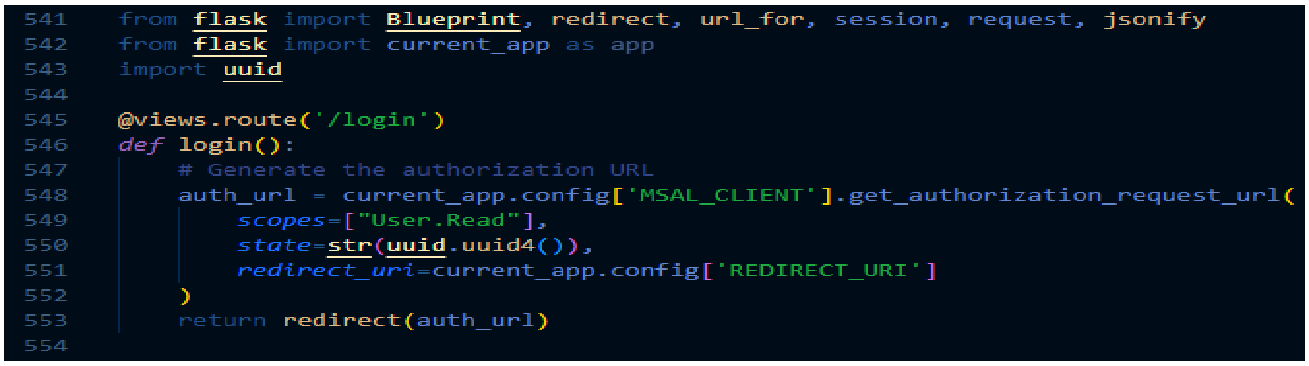
http://localhost:5000/graph
http://localhost:5000/auth/callback
http://localhost:5000/getAToken
http://localhost:5000/redirect

Add URI

Front-channel logout URL

2. Add OAuth Routes to views.py

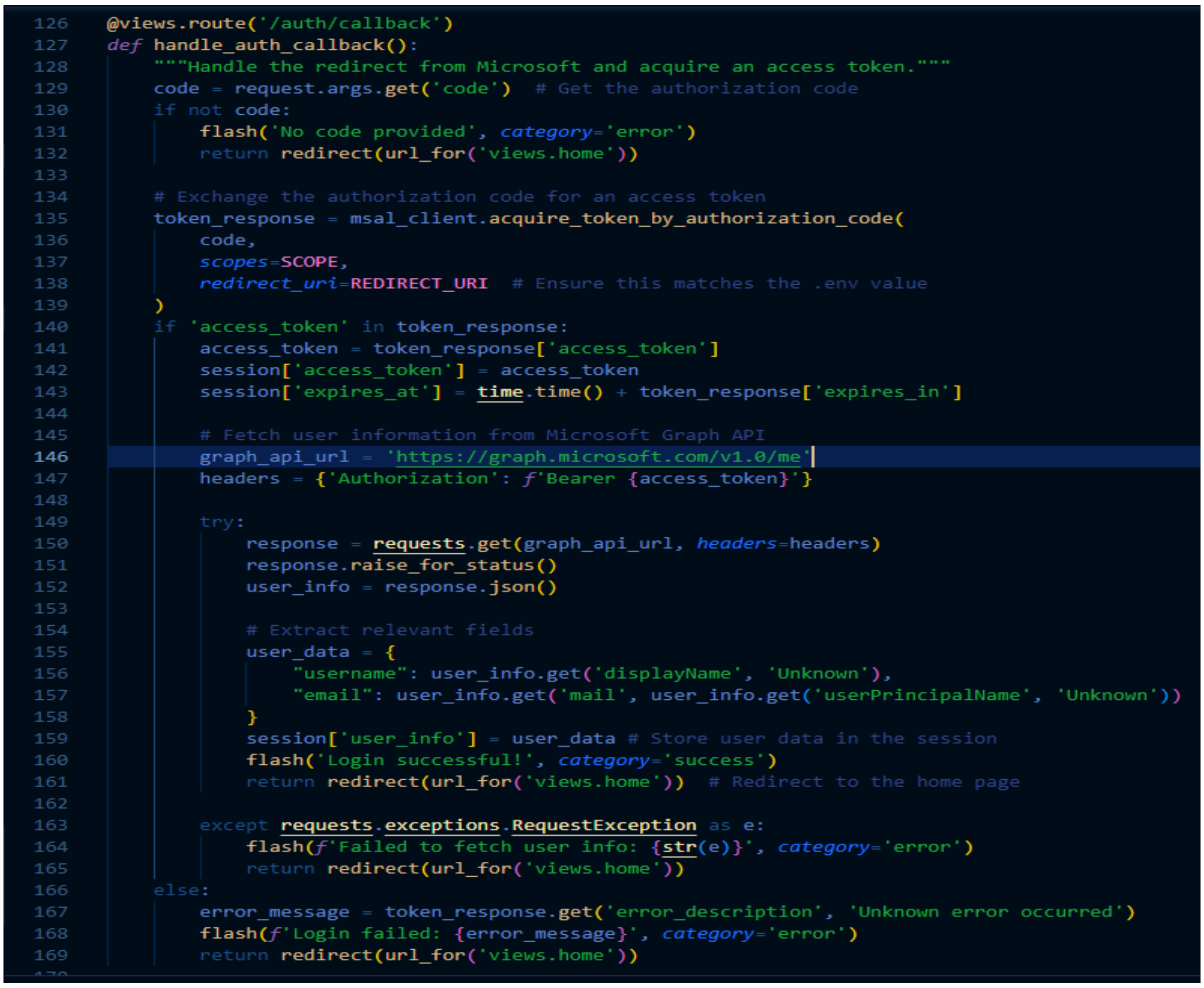
   Create or modify routes to handle login and token retrieval.



```
from flask import Blueprint, redirect, url_for, session, request, jsonify
from flask import current_app as app
import uuid

@views.route('/login')
def login():
    # Generate the authorization URL
    auth_url = current_app.config['MSAL_CLIENT'].get_authorization_request_url(
        scopes=["User.Read"],
        state=str(uuid.uuid4()),
        redirect_uri=current_app.config['REDIRECT_URI']
    )
    return redirect(auth_url)

```

/login Route:

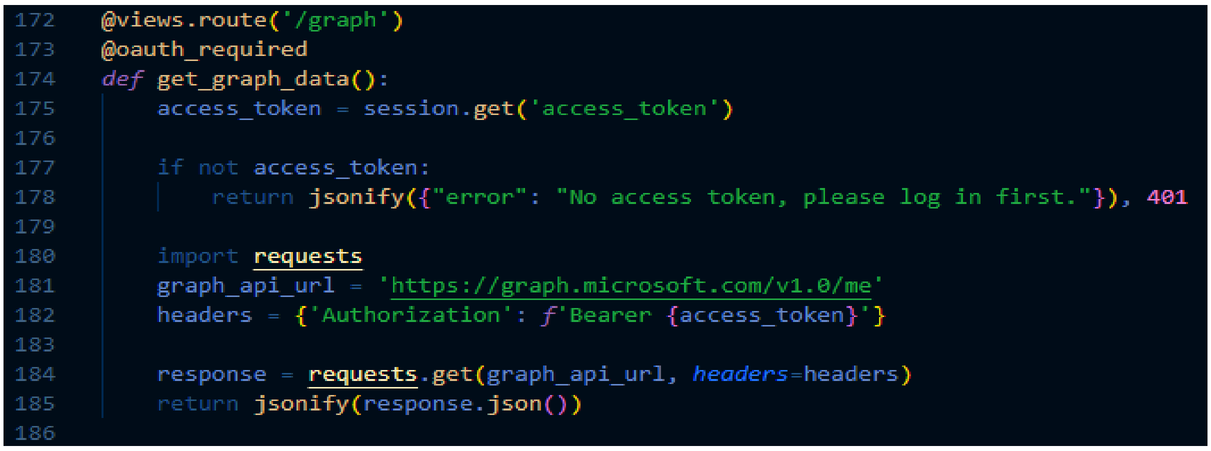
Initiates the login process by redirecting the user to the Microsoft login page with the necessary scopes.



```python
@views.route('/auth/callback')
def handle_auth_callback():
    """Handle the redirect from Microsoft and acquire an access token."""
    code = request.args.get('code')  # Get the authorization code
    if not code:
        flash('No code provided', category='error')
        return redirect(url_for('views.home'))

    # Exchange the authorization code for an access token
    token_response = msal_client.acquire_token_by_authorization_code(
        code,
        scopes=SCOPE,
        redirect_uri=REDIRECT_URI  # Ensure this matches the .env value
    )
    if 'access_token' in token_response:
        access_token = token_response['access_token']
        session['access_token'] = access_token
        session['expires_at'] = time.time() + token_response['expires_in']

        # Fetch user information from Microsoft Graph API
        graph_api_url = 'https://graph.microsoft.com/v1.0/me'
        headers = {'Authorization': f'Bearer {access_token}'}

        try:
            response = requests.get(graph_api_url, headers=headers)
            response.raise_for_status()
            user_info = response.json()

            # Extract relevant fields
            user_data = {
                "username": user_info.get('displayName', 'Unknown'),
                "email": user_info.get('mail', user_info.get('userPrincipalName', 'Unknown'))
            }
            session['user_info'] = user_data # Store user data in the session
            flash('Login successful!', category='success')
            return redirect(url_for('views.home'))  # Redirect to the home page

        except requests.exceptions.RequestException as e:
            flash(f'Failed to fetch user info: {str(e)}', category='error')
            return redirect(url_for('views.home'))
    else:
        error_message = token_response.get('error_description', 'Unknown error occurred')
        flash(f'Login failed: {error_message}', category='error')
        return redirect(url_for('views.home'))
```

/auth/callback Route:

Handles the OAuth callback, exchanges the authorization code for an access token, and stores the token in the session.

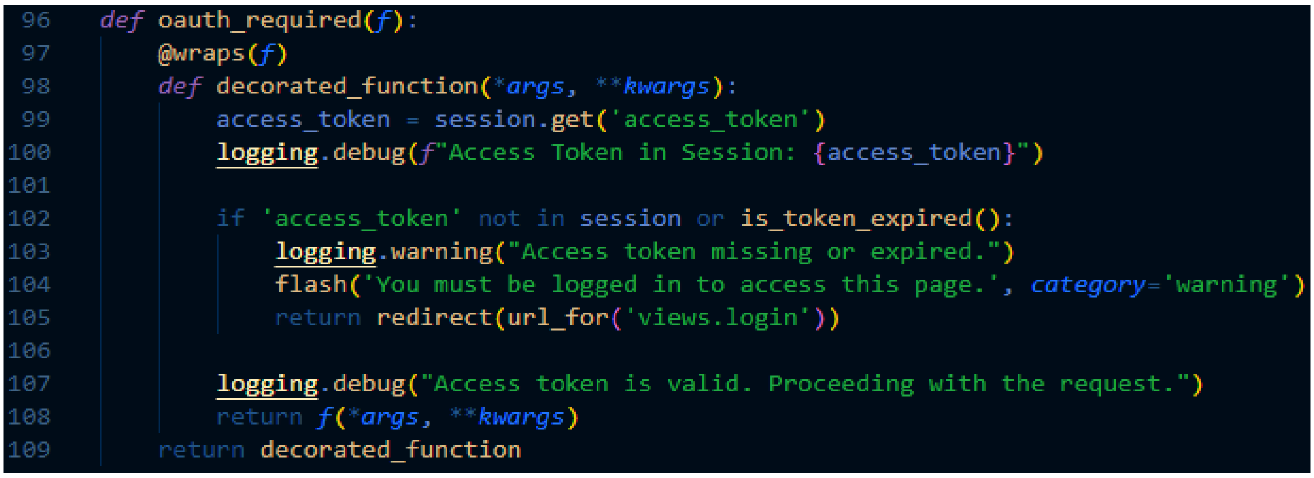


```python
@views.route('/graph')
@oauth_required
def get_graph_data():
    access_token = session.get('access_token')

    if not access_token:
        return jsonify({"error": "No access token, please log in first."}), 401

    import requests
    graph_api_url = 'https://graph.microsoft.com/v1.0/me'
    headers = {'Authorization': f'Bearer {access_token}'}

    response = requests.get(graph_api_url, headers=headers)
    return jsonify(response.json())
```

/graph Route:

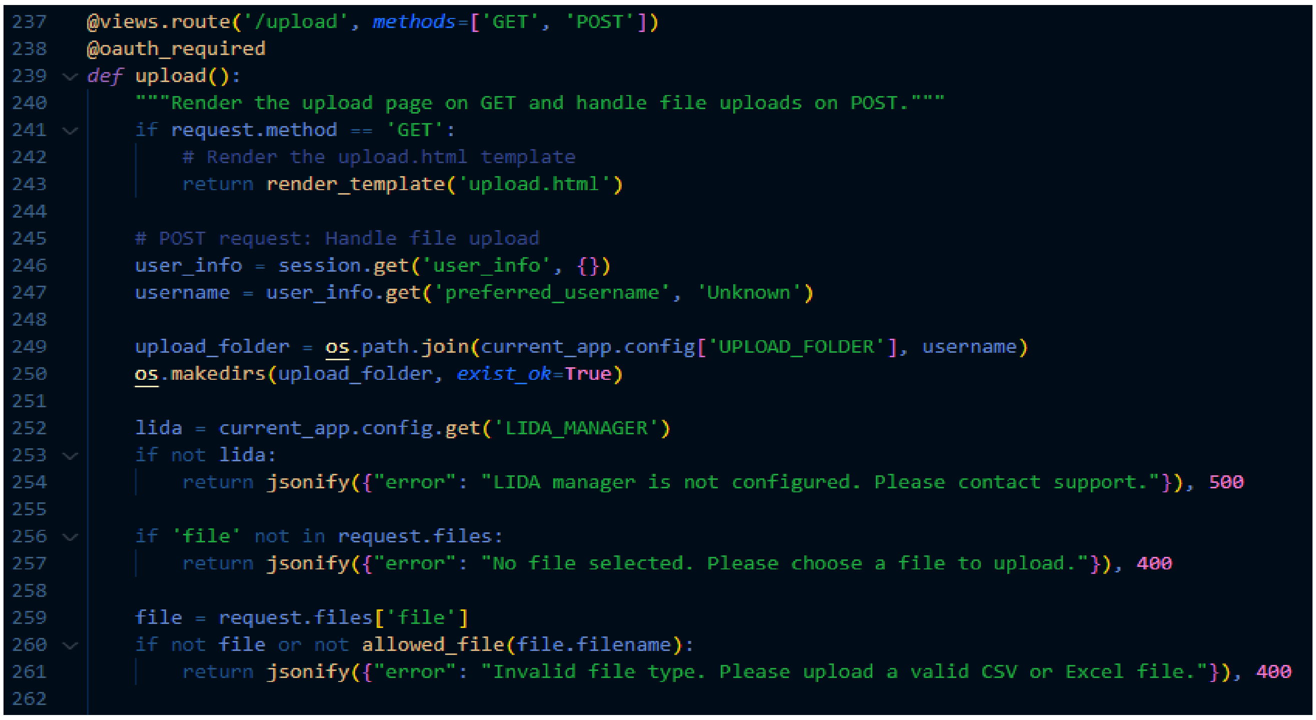
Fetches user data from Microsoft Graph using the access token stored in the session. This setup should help you obtain a valid OAuth access token and use it to call the Microsoft Graph API successfully. After that, it is time to protect necessary routes so that only authenticated users with a valid OAuth access token can access them. Additionally, we also need to ensure that the protection integrates smoothly with our MSAL-based OAuth authentication.

1. Create the OAuth Authentication Decorator



```python
def oauth_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        access_token = session.get('access_token')
        logging.debug(f"Access Token in Session: {access_token}")

        if 'access_token' not in session or is_token_expired():
            logging.warning("Access token missing or expired.")
            flash('You must be logged in to access this page.', category='warning')
            return redirect(url_for('views.login'))

        logging.debug("Access token is valid. Proceeding with the request.")
        return f(*args, **kwargs)
    return decorated_function
```

2. Apply the Decorator to the Routes

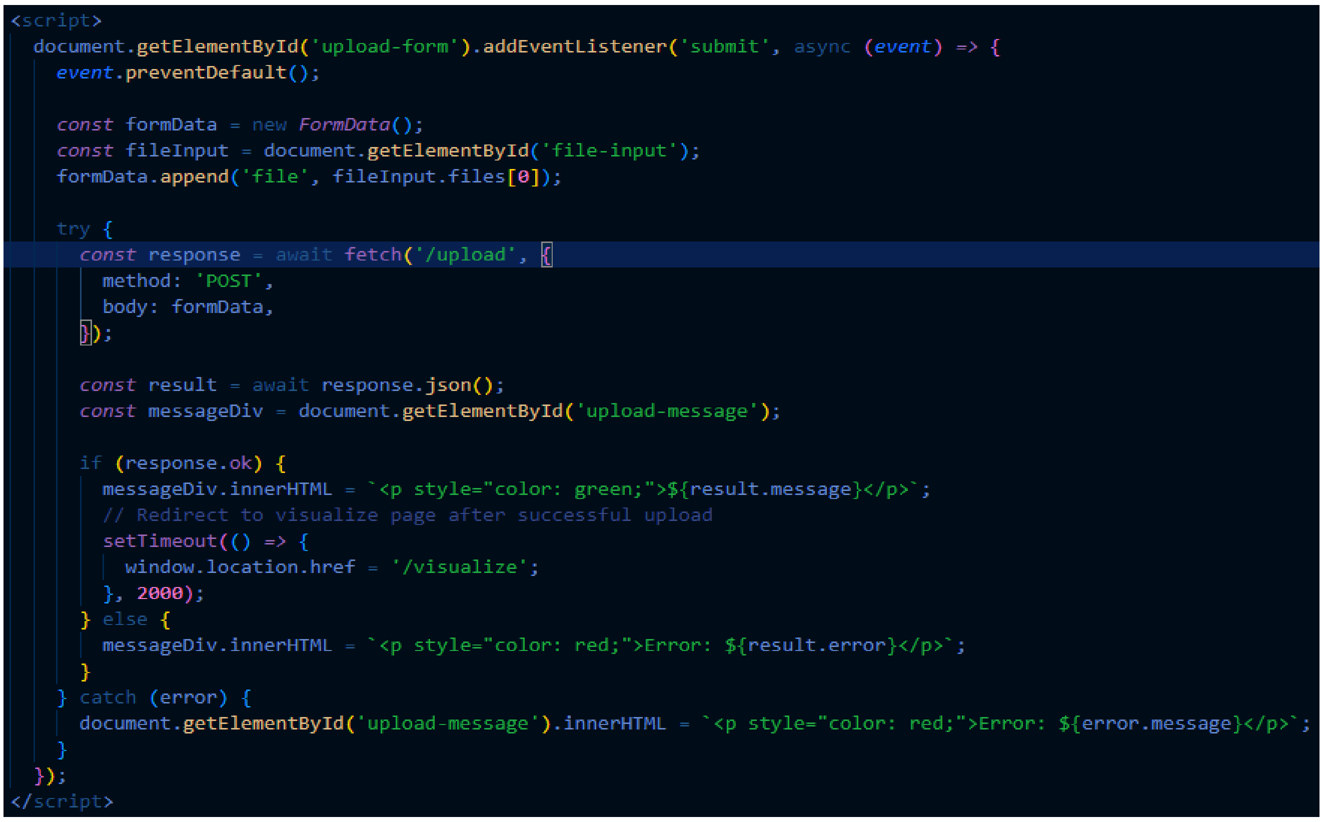


```python
@views.route('/upload', methods=['GET', 'POST'])
@oauth_required
def upload():
    """Render the upload page on GET and handle file uploads on POST."""
    if request.method == 'GET':
        # Render the upload.html template
        return render_template('upload.html')

    # POST request: Handle file upload
    user_info = session.get('user_info', {})
    username = user_info.get('preferred_username', 'Unknown')

    upload_folder = os.path.join(current_app.config['UPLOAD_FOLDER'], username)
    os.makedirs(upload_folder, exist_ok=True)

    lida = current_app.config.get('LIDA_MANAGER')
    if not lida:
        return jsonify({"error": "LIDA manager is not configured. Please contact support."}), 500

    if 'file' not in request.files:
        return jsonify({"error": "No file selected. Please choose a file to upload."}), 400

    file = request.files['file']
    if not file or not allowed_file(file.filename):
        return jsonify({"error": "Invalid file type. Please upload a valid CSV or Excel file."}), 400

```

GET Request to /upload → Renders upload.html.

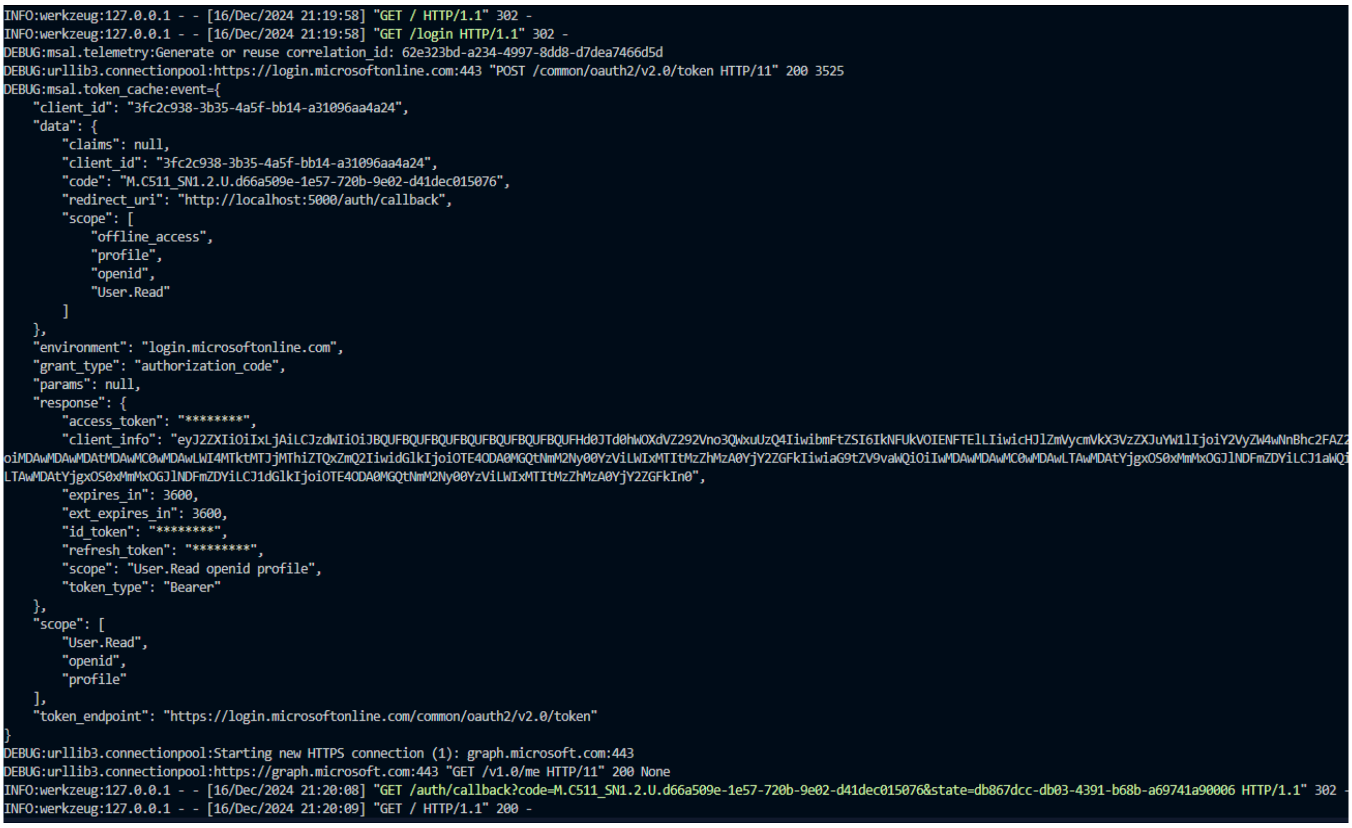
POST Request to /upload → Handles the file upload and returns a JSON response.

Frontend – upload.html

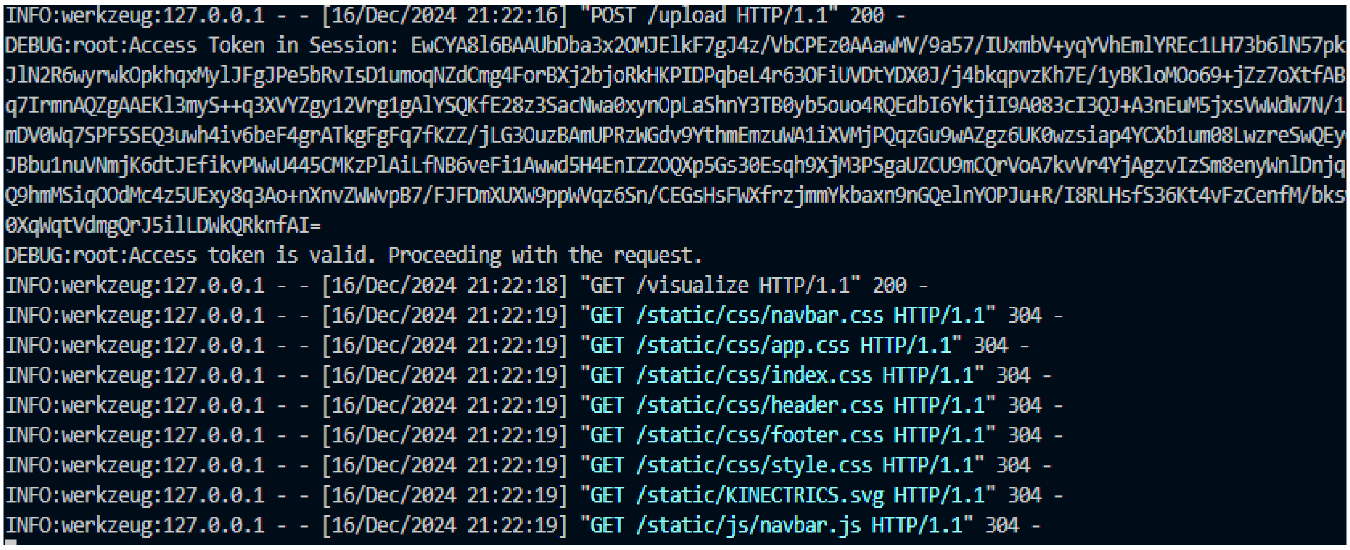


```
<script>
  document.getElementById('upload-form').addEventListener('submit', async (event) => {
    event.preventDefault();

    const formData = new FormData();
    const fileInput = document.getElementById('file-input');
    formData.append('file', fileInput.files[0]);

    try {
      const response = await fetch('/upload', {
        method: 'POST',
        body: formData,
      });

      const result = await response.json();
      const messageDiv = document.getElementById('upload-message');

      if (response.ok) {
        messageDiv.innerHTML = `<p style="color: green;">${result.message}</p>`;
        // Redirect to visualize page after successful upload
        setTimeout(() => {
          window.location.href = '/visualize';
        }, 2000);
      } else {
        messageDiv.innerHTML = `<p style="color: red;">Error: ${result.error}</p>`;
      }
    } catch (error) {
      document.getElementById('upload-message').innerHTML = `<p style="color: red;">Error: ${error.message}</p>`;
    }
  });
</script>
```

Terminal output after logging in using MSAL



```
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:19:58] "GET / HTTP/1.1" 302 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:19:58] "GET /login HTTP/1.1" 302 -
DEBUG:msal.telemetry:Generate or reuse correlation_id: 62e323bd-a234-4997-8dd8-d7dea7466d5d
DEBUG:urllib3.connectionpool:https://login.microsoftonline.com:443 "POST /common/oauth2/v2.0/token HTTP/11" 200 3525
DEBUG:msal.token_cache:event={
    "client_id": "3fc2c938-3b35-4a5f-bb14-a31096aa4a24",
    "data": {
        "claims": null,
        "client_id": "3fc2c938-3b35-4a5f-bb14-a31096aa4a24",
        "code": "M.C511_SN1.2.U.d66a509e-1e57-720b-9e02-d41dec015076",
        "redirect_uri": "http://localhost:5000/auth/callback",
        "scope": [
            "offline_access",
            "profile",
            "openid",
            "User.Read"
        ]
    },
    "environment": "login.microsoftonline.com",
    "grant_type": "authorization_code",
    "params": null,
    "response": {
        "access_token": "********",
        "client_info": "eyJ2ZXIiOiIxLjAiLCJzdWIiOiJBQUFBQUFBQUFBQUFBQUFBQUFBQUFHd0JTd0hWOXdVZ292Vno3QWxuUzQ4IiwibmFtZSI6IkNFUkVOIENFTElLIiwicHJlZmVycmVkX3VzZXJuYW1lIjoiY2VyZW4wNnBhc2FAZ2
oiMDAwMDAwMDAtMDAwMC0wMDAwLWI4MTktMTJjMThiZTQxZmQ2IiwidGlkIjoiOTE4ODA0MGQtNmM2Ny00YzViLWIxMTItMzZhMzA0YjY2ZGFkIiwiaG9tZV9vaWQiOiIwMDAwMDAwMC0wMDAwLTAwMDAtYjgxOS0xMmMxOGJlNDFmZDYiLCJ1aWQi
LTAwMDAtYjgxOS0xMmMxOGJlNDFmZDYiLCJ1dGlkIjoiOTE4ODA0MGQtNmM2Ny00YzViLWIxMTItMzZhMzA0YjY2ZGFkIn0",
        "expires_in": 3600,
        "ext_expires_in": 3600,
        "id_token": "********",
        "refresh_token": "********",
        "scope": "User.Read openid profile",
        "token_type": "Bearer"
    },
    "scope": [
        "User.Read",
        "openid",
        "profile"
    ],
    "token_endpoint": "https://login.microsoftonline.com/common/oauth2/v2.0/token"
}
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): graph.microsoft.com:443
DEBUG:urllib3.connectionpool:https://graph.microsoft.com:443 "GET /v1.0/me HTTP/11" 200 None
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:20:08] "GET /auth/callback?code=M.C511_SN1.2.U.d66a509e-1e57-720b-9e02-d41dec015076&state=db867dcc-db03-4391-b68b-a69741a90006 HTTP/1.1" 302 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:20:09] "GET / HTTP/1.1" 200 -
```



```
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:16] "POST /upload HTTP/1.1" 200 -
DEBUG:root:Access Token in Session: EwCYA8l6BAAUbDba3x2OMJElkF7gJ4z/VbCPEz0AAawMV/9a57/IUxmbV+yqYVhEmlYREc1LH73b6lN57pk
JlN2R6wyrwkOpkhqxMylJFgJPe5bRvIsD1umoqNZdCmg4ForBXj2bjoRkHKPIDPqbeL4r63OFiUVDtYDX0J/j4bkqpvzKh7E/1yBKloMOo69+jZz7oXtfAB
q7IrmnAQZgAAEKl3myS++q3XVYZgy12Vrg1gAlYSQKfE28z3SacNwa0xynOpLaShnY3TB0yb5ouo4RQEdbI6YkjiI9A083cI3QJ+A3nEuM5jxsVwWdW7N/1
mDV0Wq7SPF5SEQ3uwh4iv6beF4grATkgFgFq7fKZZ/jLG3OuzBAmUPRzWGdv9YthmEmzuWA1iXVMjPQqzGu9wAZgz6UK0wzsiap4YCXb1um08LwzreSwQEy
JBbu1nuVNmjK6dtJEfikvPWwU445CMKzPlAiLfNB6veFi1Awwd5H4EnIZZOQXp5Gs30Esqh9XjM3PSgaUZCU9mCQrVoA7kvVr4YjAgzvIzSm8enyWnlDnjq
Q9hmMSiqOOdMc4z5UExy8q3Ao+nXnvZWWvpB7/FJFDmXUXW9ppWVqz6Sn/CEGsHsFWXfrzjmmYkbaxn9nGQelnYOPJu+R/I8RLHsfS36Kt4vFzCenfM/bks
0XqWqtVdmgQrJ5ilLDWkQRknfAI=
DEBUG:root:Access token is valid. Proceeding with the request.
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:18] "GET /visualize HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:19] "GET /static/css/navbar.css HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:19] "GET /static/css/app.css HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:19] "GET /static/css/index.css HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:19] "GET /static/css/header.css HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:19] "GET /static/css/footer.css HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:19] "GET /static/css/style.css HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:19] "GET /static/KINECTRICS.svg HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [16/Dec/2024 21:22:19] "GET /static/js/navbar.js HTTP/1.1" 304 -
```