Comprehensive Project Report: Automatic Generation of Visualizations and Infographics using Large Language Models (LLMs)

Ceren Celik 101457127

---

1. Project Overview

Title: Automatic Generation of Visualizations and Infographics using Large Language Models (LLMs)

Summary: This project aims to develop a platform that allows users to upload datasets (Excel, CSV) and ask questions in natural language. The system dynamically generates visualizations (plots, graphs) and provides textual insights using Large Language Models (LLMs). The platform makes data analysis accessible for non-technical users by combining Flask for backend processing and HTML, JavaScript, and CSS for the frontend.

Key Features:

- File upload functionality for CSV and Excel datasets.
- Dynamic visualization generation using LangChain Agents.
- Text summaries and insights using LIDA (Large-scale Interactive Data Analysis).
- OAuth-based authentication for secure access.
- Interactive frontend for user queries and visualization downloads.

---

2. System Architecture

Technologies and Tools:

- **Frontend**: HTML, JavaScript, CSS (with Bootstrap 5)
- **Backend**: Flask (Python)
- **Visualization**: LangChain Agent, Plotly, Matplotlib
- **Text Summarization**: LIDA
- **Authentication**: OAuth (Microsoft Authentication Library - MSAL)
- **Database**: SQLAlchemy (Flask SQLAlchemy)
- **APIs**: OpenAI API

Components:

1. **Frontend**:
    - **Files**: base.html, home.html, upload.html, visualize.html

- **Features**:
  - **File Upload Section**: Users can upload datasets.
  - **Query Input Section**: Textbox to input natural language queries.
  - **Visualization Display**: Interactive area to view generated plots and download them in various formats.

2. **Backend**:
   - **Files**: views.py, auth.py, models.py, chatbot_utils.py
   - **Features**:
     - **Flask Routes**:
       - /upload: Handle file uploads and process datasets.
       - /visualize: Generate visualizations based on user queries.
       - /login and /logout: Handle OAuth-based authentication.
     - **Processing Logic**:
       - **LangChain**: For generating dynamic visualizations.
       - **LIDA**: For text summarization and insights.
3. **Database**:
   - **Tables**:
     - User: Stores user information.
     - FileUpload: Tracks uploaded files.
     - FileData: Stores file metadata and summaries.
     - Question: Stores generated questions related to datasets.

---

3. File Upload and Processing Pipeline

File Handling

- **File Upload**:
  - **Endpoint**: /upload
  - **Supported Formats**: CSV, Excel (XLSX)
  - **Logic**:
    - Files are uploaded via a form in upload.html.

        - The backend processes the file and extracts metadata (column names, data types).
        - The file is stored in a user-specific directory.

- **File Parsing**:
    - **Function**: process_large_file(filepath) in views.py
    - **Handles Large Files** by processing in chunks (for CSV) or reading directly (for Excel).
- **Database Storage**:
    - **Model**: FileData in models.py
    - Stores filename, file path, and a JSON representation of the file data.

---

4. Documentation

4.1 Technical Documentation

System Architecture

Provide a high-level overview of the system, including component diagrams and data flow diagrams. This helps in understanding the overall structure of your project.

Data Processing

Detail your data sources, collection methods, cleaning and preprocessing steps, and any feature engineering techniques used. This section should give a clear picture of how you prepared your data for use in your model.

Model Training

Explain your model selection rationale, the hyperparameter tuning process, and specifications of your training environment. This helps in understanding your approach to solving the problem at hand.

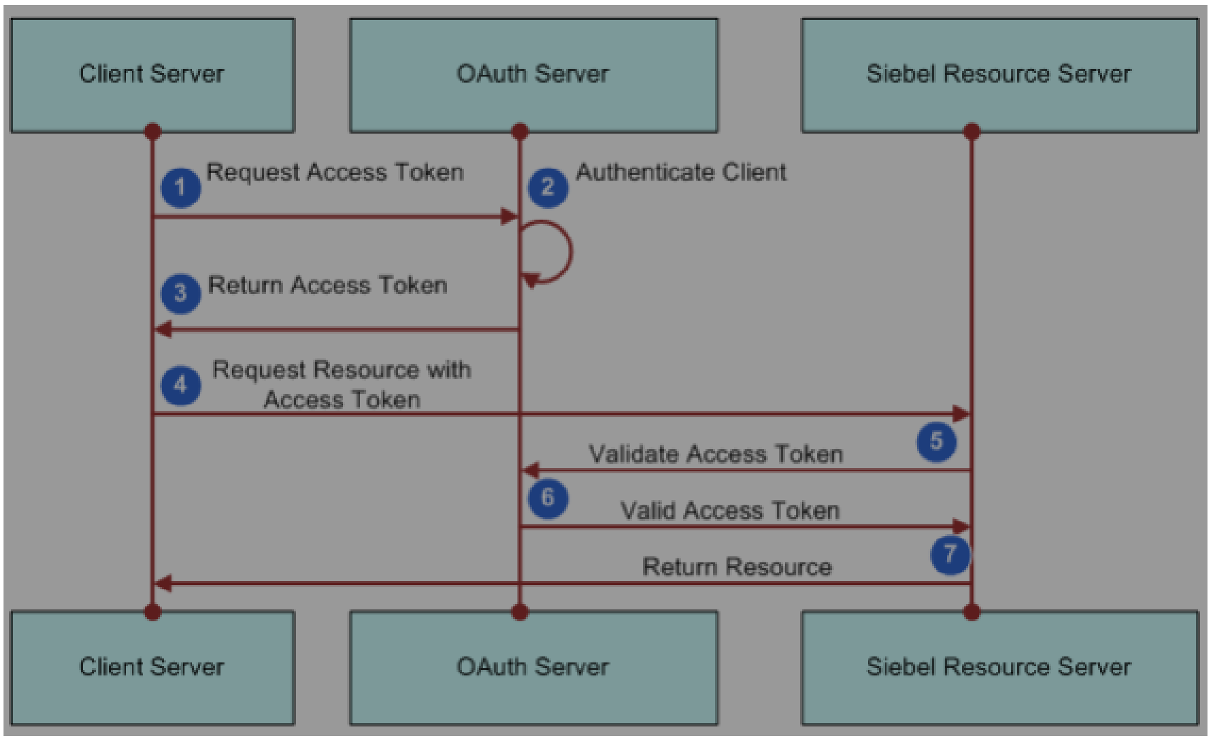
Evaluation

Describe your evaluation metrics and the rationale behind choosing them. Include baseline model performance and document iterative improvements and their results.

4.2 OAuth Authentication

OAuth Integration with MSAL

Authentication Workflow:



1. **Login Route (/login)**:
   - Redirects the user to the Microsoft login page with the necessary scopes.
   - Uses the Microsoft Authentication Library (MSAL) to initiate the login process.
2. **Callback Route (/auth/callback)**:
   - Handles the OAuth callback and exchanges the authorization code for an access token.
   - Stores the access token in the session for future API calls.
3. **Protected Routes**:
   - Routes such as /upload and /visualize are protected using an OAuth authentication decorator to ensure only authenticated users can access them.

API Data Authentication:

- Uses the access token to securely fetch data from Microsoft Graph API.
- Handles token expiry and ensures tokens are refreshed when necessary.

Error Handling:

- Provides user-friendly error messages for authentication failures (e.g., InvalidAuthenticationToken).
- Ensures environment variables are correctly set for client ID, client secret, and redirect URI.

4.3 Project Journal

Maintain a project journal throughout the development process. This should include:

- **Key Decision Points**: Using Flask instead of React because it is hard to get the data.
- **Challenges Encountered**: Creating a chatbot that can handle both LangChain Agent and LIDA.
- **Solutions Implemented**: Flask was used for the backend due to its simplicity and flexibility. Jinja2 templates were used for rendering dynamic content on the frontend. Django was also considered for certain components to facilitate rapid development and maintainability.

---

5. Conclusion

This project effectively combines modern web technologies, LangChain Agent, and LIDA to create an accessible data analysis platform. By allowing users to upload datasets and generate insights via natural language, the platform bridges the gap between technical and non-technical users. Secure authentication, dynamic visualizations, and a user-friendly interface ensure a seamless user experience.

Future Improvements

- **Enhance Visualization Options**: Support for more complex chart types.
- **Scalability**: Optimize for handling larger datasets.
- **User Management**: Additional roles and permissions.
