Excel Table Viewer and Row Sum Calculator

Project Overview

The **Excel Table Viewer and Row Sum Calculator** is a Streamlit-based web application designed to process Excel files, display the contents of tables (sheets), and calculate row sums. The user can upload an Excel file and interact with it via a simple web interface to view tables, select rows, and calculate the sum of numerical data within the selected rows.

Features

1. **Table Viewer**: Displays the tables (sheets) within an uploaded Excel file.
2. **Row Sum Calculator**: Allows users to select rows from a chosen table and calculate the sum of numerical values in the row.
3. **Interactive User Interface**: Built with Streamlit, providing a responsive and user-friendly interface.
4. **Error Handling**: Handles cases like empty Excel files, missing tables, or invalid row selections.

Installation Instructions

To run this project locally, follow these steps:

1. **Clone the repository:**

bash

CopyEdit

git clone https://github.com/your-username/excel-table-viewer.git

cd excel-table-viewer

2. **Create a virtual environment (optional but recommended):**

bash

CopyEdit

python -m venv venv

source venv/bin/activate # On Windows, use 'venv\Scripts\activate'

3. **Install the required dependencies:**

bash

CopyEdit

pip install -r requirements.txt

The requirements.txt file should include:

txt

CopyEdit

streamlit==1.10.0

pandas==1.4.2

numpy==1.22.3

4. **Run the Streamlit app:**

bash

CopyEdit

streamlit run app.py

This will start the application at http://localhost:8501.

How to Use the Application

1. **Upload the Excel file**:
   Upon starting the application, the user will be asked to upload an Excel file that contains tables (sheets).
2. **Select a table**:
   After uploading, the user can select which table (sheet) they want to view. The table's content will be displayed.
3. **Select a row to calculate the sum**:
   After viewing the table, the user can select a row. The sum of all numerical values in the selected row will be calculated and displayed.

Example Workflow

1. Start the app by running streamlit run app.py.
2. Upload an Excel file containing tables.
3. Select a table from the dropdown.

4. Choose a row from the selected table.
5. The sum of the row's numerical values will be displayed.

Libraries Used

- **Streamlit**: A Python library to create interactive web applications. It helps in building the UI quickly and easily.
    - *Why Streamlit?* It simplifies the creation of web applications without the need for extensive frontend knowledge.
- **Pandas**: A powerful data manipulation library in Python. It is used to load the Excel file and handle the tabular data.
    - *Why Pandas?* It makes it easy to work with structured data like Excel files, and offers built-in methods to read and manipulate data.
- **NumPy**: A library for numerical computations, used here to handle numeric operations on the data.
    - *Why NumPy?* It efficiently handles numerical operations, which are crucial for calculating row sums.

Potential Improvements

- **Handle different Excel file formats**: Currently, the application only processes .xls files. It could be extended to support .xlsx and other formats.
- **Enhance error handling**: Handle edge cases such as empty sheets, missing data, or incorrect file formats more gracefully.
- **Add advanced operations**: Extend the functionality to include more complex Excel operations, such as filtering, sorting, or data aggregation.
- **Add user authentication**: For larger-scale applications, user authentication can be added to restrict access.

Missed Edge Cases

- **Empty Excel Sheets**: If a sheet is empty, the app could fail to display any rows or columns.

- **Non-Numerical Data**: If a row contains non-numeric data (e.g., text or symbols), it may not be included in the sum calculation.
- **Large Excel Files**: Very large files may take longer to process or could cause performance issues.

Testing

- You can test the app by using the Postman collection to call the API endpoints (if extended to include REST endpoints for table operations).
- Unit tests for the core functionality (data loading, processing, etc.) can be added in the future.