Group 10 – Second Results:

Specification Document for Django Project

https://github.com/amithacoon/Project-BeeHive

Overview

This Django project is designed to handle file uploads from clients (CSV and Excel files), process the data using an NLP module (such as the Gemini API) to generate tags and summarized phrases, and save the processed data back into a new table with additional columns. The processed data can be downloaded by the user in either CSV or Excel format.

Generic Nature of the Code

The codebase is designed with a high degree of modularity and flexibility, making it adaptable for various use cases beyond the specific task it currently performs. Here are some aspects that highlight its generic nature:

1. **Modular Functions**
    1. Functions like `clear_media_directory`, `handle_uploaded_file`, `read_csv_file`, and `read_excel_file` are written to perform specific tasks that can be reused in different contexts without modification.
2. **Extensible Data Processing**
    1. The `process_dataframe` function is designed to process any DataFrame column using a provided prompt. By changing the prompt or the column number, the same function can handle different types of data processing tasks.
3. **Configurable API Integration**

    1. The `gemini` function can be configured with different model names and API keys, making it adaptable for various content generation tasks using different models or APIs.

4. **Progress Tracking**

    1. The progress tracking mechanism is generic enough to be applied to any long-running task. The use of threading and locks ensures that it can be safely used in a multi-threaded environment.

5. **File Handling**

    1. The ability to handle both CSV and Excel files allows for flexibility in the types of data sources that can be processed. Adding support for more file types would require minimal changes.

6. **Template and Frontend**

    1. The HTML template is designed with generic form elements and progress indicators, making it easy to adapt the frontend for different types of file processing tasks.

7. **Data Saving Options**

    1. The `save_dataframe` function supports saving DataFrames in both CSV and XLSX formats with an optional RTL orientation. This makes it versatile for various output requirements.

Project Components and Flow

Forms (`forms.py`)

1. **UploadCSVForm**

    1. **Purpose**: A form class that handles file uploads, specifically for CSV files.

    2. **Fields**: A file field to upload CSV files.

Clear Media (`clear_media.py`)

1. **clear_media_directory**

    1. **Purpose**: Clears and recreates the media directory to ensure a clean state for new file uploads.
    2. **Functionality**: Deletes all files in the media directory and recreates the directory.

URLs (`urls.py`)

1. **URL Patterns**
    1. **Purpose**: Defines URL patterns for the Django project.
    2. **Main URLs**:
        1. `/upload/`: Maps to `csv_upload_view` for handling file uploads.
        2. `/progress/`: Maps to `get_progress_view` for tracking the processing progress.
        3. `/display/<filename>/`: Maps to `display_csv` for displaying the processed file.

Views (`views.py`)

1. **gemini**
    1. **Purpose**: Uses the Gemini API to generate content based on a provided prompt and text.
    2. **Input**: `prompt` (str), `text` (str)
    3. **Output**: Generated text (str)
2. **csv_upload_view**
    1. **Purpose**: Handles CSV file uploads through a form.
    2. **Input**: HTTP request
    3. **Output**: Renders `home.html` with the form or calls `display_csv` to show the uploaded file.
3. **display_csv**
    1. **Purpose**: Displays the contents of the uploaded CSV or Excel file.

    2. **Input**: HTTP request, file object
    3. **Output**: Renders `display.html` with the file data.
4. **read_csv_file**
    1. **Purpose**: Reads a CSV file and returns its content as a list of lists.
    2. **Input**: `file_path` (str)
    3. **Output**: Data (list of lists)
5. **read_excel_file**
    1. **Purpose**: Reads an Excel file and returns its content as a list of lists.
    2. **Input**: `file_path` (str)
    3. **Output**: Data (list of lists)
6. **load_excel_to_dataframe**
    1. **Purpose**: Loads an Excel file into a pandas DataFrame.
    2. **Input**: `file_path` (str)
    3. **Output**: DataFrame
7. **process_dataframe**
    1. **Purpose**: Processes each row of the DataFrame using the Gemini API.
    2. **Input**: DataFrame, column number, new column name, prompt
    3. **Output**: Updated DataFrame with new content
8. **update_progress** and **get_progress**
    1. **Purpose**: Functions for progress tracking.
    2. **Input/Output**: Updates and retrieves the current progress value.
9. **get_progress_view**
    1. **Purpose**: Returns the current progress as a JSON response.

2. **Input**: HTTP request
3. **Output**: JSON response with progress value

10. **asis_df**
    1. **Purpose**: Extracts text from a specified column and adds it to the result DataFrame without any modifications.
    2. **Input**: DataFrame, column number, new column name
    3. **Output**: Updated DataFrame
11. **save_dataframe**
    1. **Purpose**: Saves the DataFrame in the specified format (CSV or XLSX) with optional RTL orientation.
    2. **Input**: DataFrame, format (str), rtl (bool)
    3. **Output**: Saves the file in the media directory

HTML Templates

Home Template (`home.html`)

1. **File Upload Form**
    1. **Purpose**: Allows users to upload CSV or Excel files and choose the output format.
    2. **Components**:
        1. Form with file input and radio buttons for format selection.
        2. Buttons for uploading and running the process.
2. **Progress Bar**
    1. **Purpose**: Displays the progress of the file processing.
    2. **Components**: Progress bar, text for percentage and estimated time.
3. **JavaScript**

    1. **Purpose**: Handles form submission, progress bar updates, and displaying the processed file.
    2. **Functionality**: Sends the file to the server, updates progress bar and estimated time of arrival (ETA), and shows the download link for the processed file.

4. **Display Updated Table Button**
    1. **Purpose**: Allows users to view the processed and updated table.
    2. **Components**: Button to show the updated table in an iframe.

Display Template (`display.html`)

1. **Table**
    1. **Purpose**: Renders the uploaded and processed data in a tabular format.
    2. **Components**:
        1. Headers and rows to display the data.
        2. Styling to enhance readability.

Usage

1. **File Upload and Processing:**
    1. Users upload CSV or Excel files through a web form. The files are saved in the media directory, and their contents are displayed.
    2. The data is processed using the Gemini API, and progress is tracked and displayed.
2. **Data Output:**
    1. The processed data can be saved in the desired format with optional RTL orientation for CSV files.
    2. Users can download the processed file or view the updated table directly on the webpage.

Deployment

- **Deployment**: This application can be deployed on a cloud platform for accessibility.