Flask Application Documentation-Follow Steps Sandeep Rajak step by step development.

Table of Contents

1. Overview

This Flask application is designed to perform data processing on Excel files, allowing users to upload two files, perform an inner join based on a common column, and view/download the result. The application also supports the generation of a PDF report.

2. Installation

Clone the repository:

bash

Copy code

git clone https://github.com/yourusername/your-flask-app.git

cd your-flask-app

Create a virtual environment:

bash

Copy code

python -m venv venv

Activate the virtual environment:

On Windows:

bash

Copy code

venv\Scripts\activate

On Unix or MacOS:

bash

Copy code

source venv/bin/activate

Install dependencies:

bash

Copy code

pip install -r requirements.txt

3. Usage

Run the Flask application:

bash

Copy code

python app.py

Visit http://localhost:5000 in your web browser.

4. File Structure

lua

Copy code

your-flask-app/

|-- app.py

|-- templates/

| |-- index.html

| |-- result.html

|-- venv/

|-- .gitignore

|-- README.md

|-- requirements.txt

app.py: Main Flask application file.

templates/: Folder containing HTML templates.

venv/: Virtual environment folder (created after running python -m venv venv).

.gitignore: File specifying files and directories to be ignored by version control.

README.md: Project documentation.

requirements.txt: List of Python dependencies.

5. Dependencies

Flask: Web framework for Python.

pandas: Data manipulation library.

Flask-WTF: Flask integration for WTForms.

6. Routes

/: Home page displaying the file upload form.

/process: Route to process uploaded files and perform the inner join.

/download_excel: Route to download the result as an Excel file.

7. HTML Templates

index.html: Template for the home page with the file upload form.

result.html: Template for displaying the result and the download button.

8. Data Processing

The application uses the pandas library to read and process Excel files. The inner join is performed based on the user-selected common column.

9. Download Functionality

The result set can be downloaded in Excel format with a single click using the "Download Result" button on the result page.

bash

Copy code

pip install pdfkit

Then, in your Flask app, create a route that generates a PDF report and serves it for download.

Example code:

python

Copy code

```
import pdfkit
from flask import render_template, make_response

@app.route('/generate_pdf_report')
def generate_pdf_report():
    # Assume 'result_set' is your DataFrame
    html_content = render_template('result.html', result=result_set.to_html(index=False))
```

```
    # Configure PDF options if needed
    pdf_options = {
        'page-size': 'A4',
        'margin-top': '0mm',
        'margin-right': '0mm',
        'margin-bottom': '0mm',
        'margin-left': '0mm',
    }

    pdf = pdfkit.from_string(html_content, False, options=pdf_options)

    # Serve the PDF for download
    response = make_response(pdf)
    response.headers['Content-Type'] = 'application/pdf'
    response.headers['Content-Disposition'] = 'attachment; filename=result_report.pdf'

    return response
```

This example assumes you have set up pdfkit and your result set is stored in the variable result_set. Adjust the code according to your specific needs.

11. Troubleshooting

Issue: The application doesn't start.

Solution: Double-check that all dependencies are installed, the virtual environment is activated, and there are no syntax errors in your code.