الجامعة الأورومتوسطية بفاس
EUROMED UNIVERSITY OF FES
UNIVERSITÉ EUROMED DE FÈS

École d'Ingénierie Digitale
et d'Intelligence Artificielle

PROJECT REPORT:

Dynamic Data Visualization from CSV Files

Aya Hassan
1ere année cycle ingénieur Big Data

1. Introduction

The project aims to develop a web application that allows users to upload a CSV file, view a preview of the data, display descriptive statistics, and generate an interactive scatter plot,Line Chart et Bar Graph using the Plotly library. This project is developed using the Flask framework, with tools like Pandas for data analysis and Plotly for graphical visualization.

2. Technologies Used

- Flask: A lightweight web framework used to create the backend of the application.
- Pandas: A Python library used for data manipulation and analysis in the form of DataFrames.
- Plotly: A Python library for creating interactive and dynamic plots.
- HTML, CSS: For building the user interface and styling the pages.

3. Process Description

a. Uploading the CSV File

The user can upload a CSV file through a form on the interface. The file is then validated to ensure it meets the required extension criteria (i.e., .csv). If the file is valid, it is saved to a specified folder on the server.

b. Data Processing

Once the file is uploaded, Pandas is used to read the CSV content and perform analysis. The first five rows of data are extracted to provide a quick preview for the user.

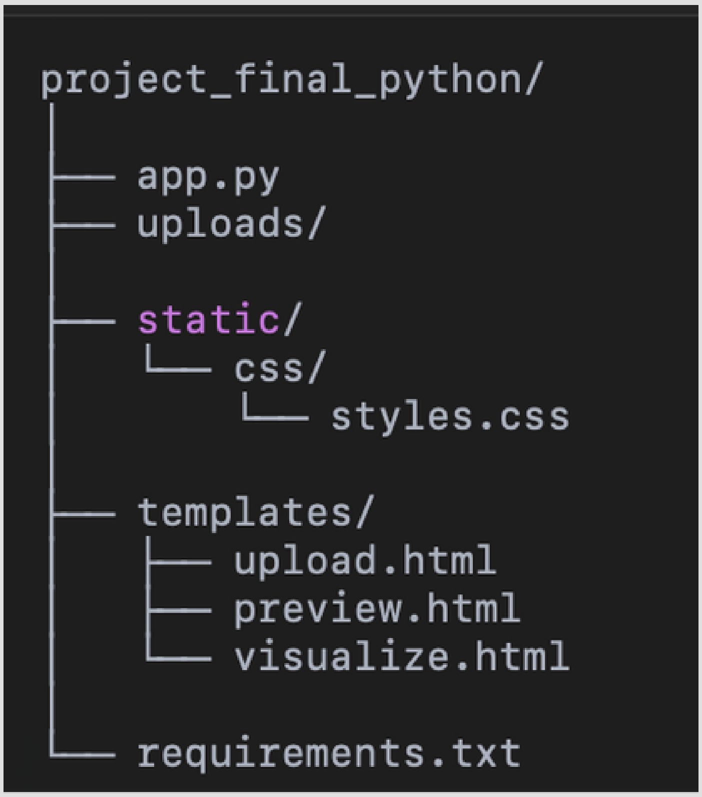
c. Descriptive Statistics

The CSV file is analyzed to extract descriptive statistics such as mean, standard deviation, minimum, and maximum values for numeric columns. These statistics are then displayed as a table on the page.

d. User Interface

The interface is simple and user-friendly, allowing the user to upload the CSV file, see a preview of the data, view descriptive statistics, and see an interactive graph. It is designed to be responsive and visually appealing using CSS.

4. Code Structure



```
project_final_python/
│
├── app.py
├── uploads/
│
├── static/
│   └── css/
│       └── styles.css
│
├── templates/
│   ├── upload.html
│   ├── preview.html
│   └── visualize.html
│
└── requirements.txt
```

a. Python File (app.py)

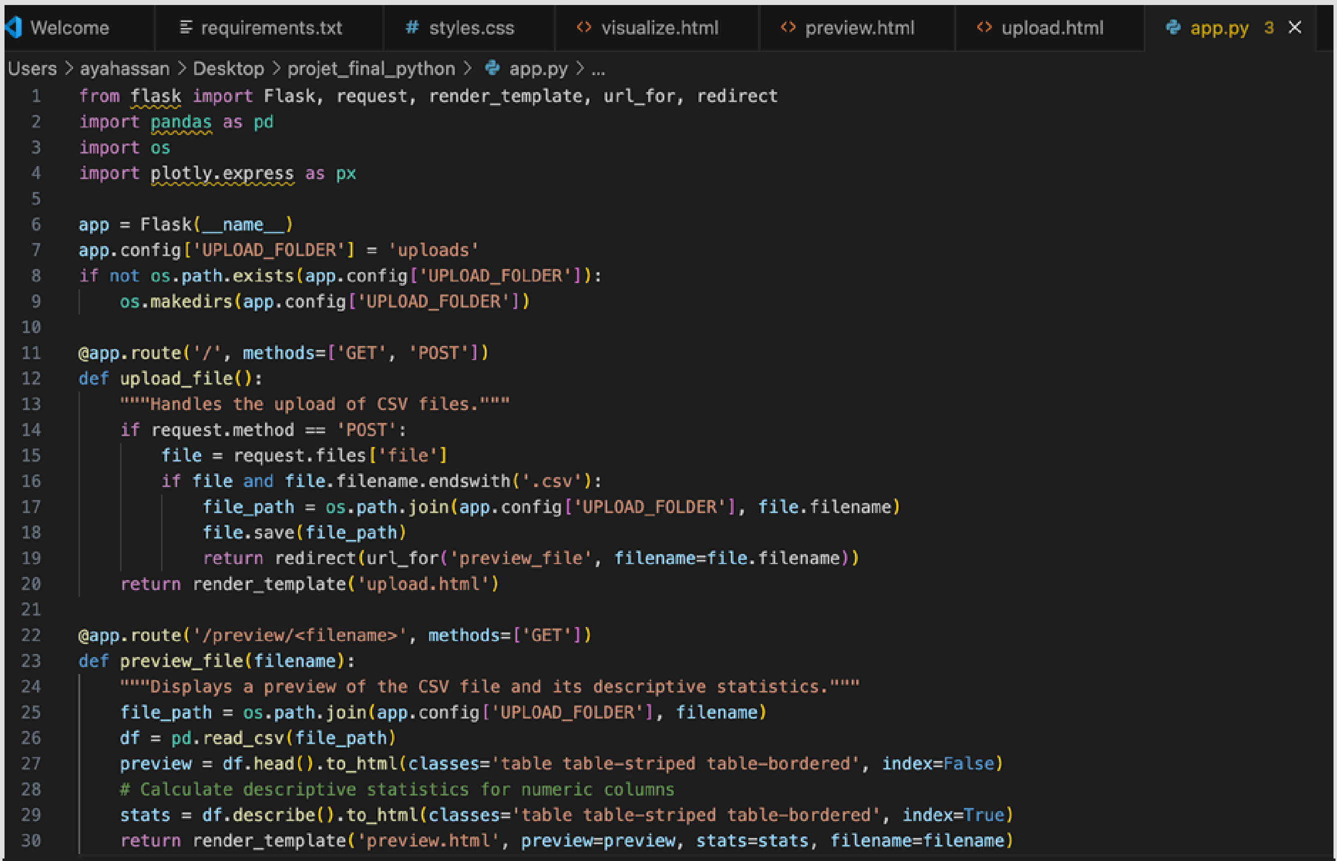
The app.py file contains the main logic for the Flask application. It manages the routes, file upload and validation, graph generation, and statistical computations.

• Upload Route (/):
Handles uploading of .csv files and redirects to the preview page.

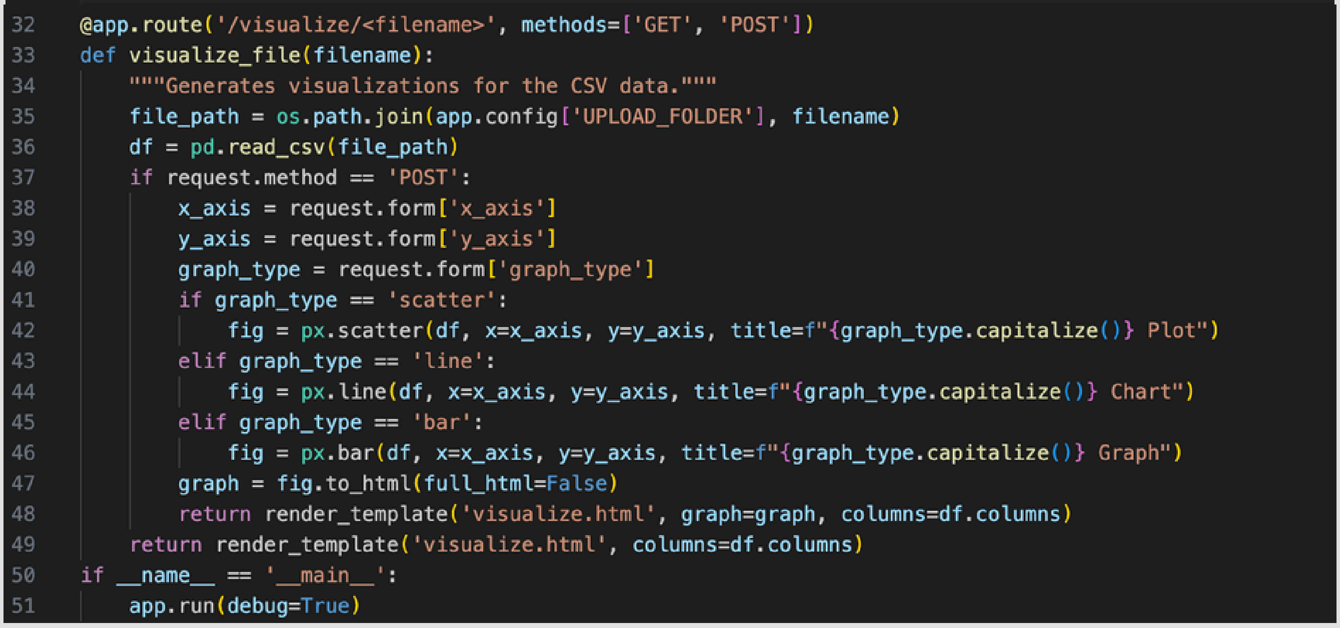
• Preview Route (/preview/<filename>):
Displays the first few rows of the uploaded CSV file and computes descriptive statistics such as mean, standard deviation, min, and max values for numeric columns.

• Visualize Route (/visualize/<filename>):
Allows the user to create different types of visualizations (scatter, line, bar) for the uploaded data.

Here is the code:



```python
from flask import Flask, request, render_template, url_for, redirect
import pandas as pd
import os
import plotly.express as px

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'
if not os.path.exists(app.config['UPLOAD_FOLDER']):
    os.makedirs(app.config['UPLOAD_FOLDER'])

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    """Handles the upload of CSV files."""
    if request.method == 'POST':
        file = request.files['file']
        if file and file.filename.endswith('.csv'):
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
            file.save(file_path)
            return redirect(url_for('preview_file', filename=file.filename))
    return render_template('upload.html')

@app.route('/preview/<filename>', methods=['GET'])
def preview_file(filename):
    """Displays a preview of the CSV file and its descriptive statistics."""
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    df = pd.read_csv(file_path)
    preview = df.head().to_html(classes='table table-striped table-bordered', index=False)
    # Calculate descriptive statistics for numeric columns
    stats = df.describe().to_html(classes='table table-striped table-bordered', index=True)
    return render_template('preview.html', preview=preview, stats=stats, filename=filename)
```



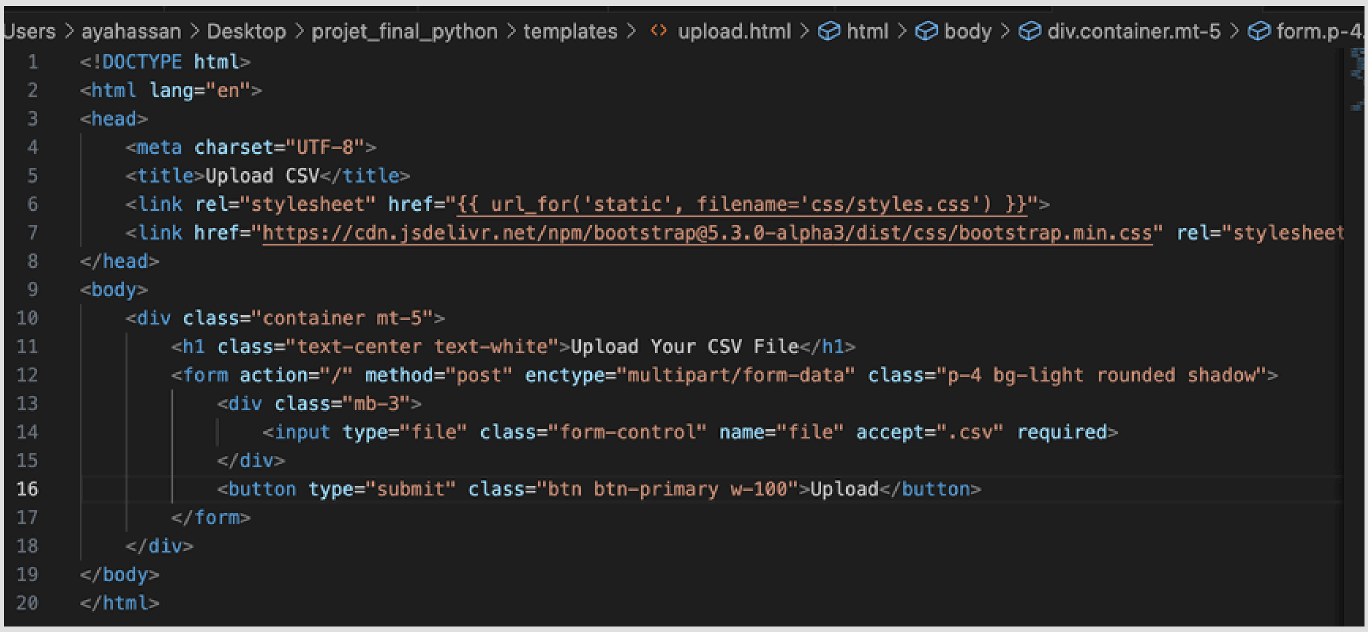
```python
@app.route('/visualize/<filename>', methods=['GET', 'POST'])
def visualize_file(filename):
    """Generates visualizations for the CSV data."""
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    df = pd.read_csv(file_path)
    if request.method == 'POST':
        x_axis = request.form['x_axis']
        y_axis = request.form['y_axis']
        graph_type = request.form['graph_type']
        if graph_type == 'scatter':
            fig = px.scatter(df, x=x_axis, y=y_axis, title=f"{graph_type.capitalize()} Plot")
        elif graph_type == 'line':
            fig = px.line(df, x=x_axis, y=y_axis, title=f"{graph_type.capitalize()} Chart")
        elif graph_type == 'bar':
            fig = px.bar(df, x=x_axis, y=y_axis, title=f"{graph_type.capitalize()} Graph")
        graph = fig.to_html(full_html=False)
        return render_template('visualize.html', graph=graph, columns=df.columns)
    return render_template('visualize.html', columns=df.columns)
if __name__ == '__main__':
    app.run(debug=True)
```

b.HTML Templates

Why it's used:

HTML templates are used to render the user interface (UI) for uploading files, previewing data, and visualizing data. Flask uses the Jinja2 template engine, which allows you to embed dynamic content (like file data or user inputs) directly into your HTML files.

upload.html (File Upload Form)

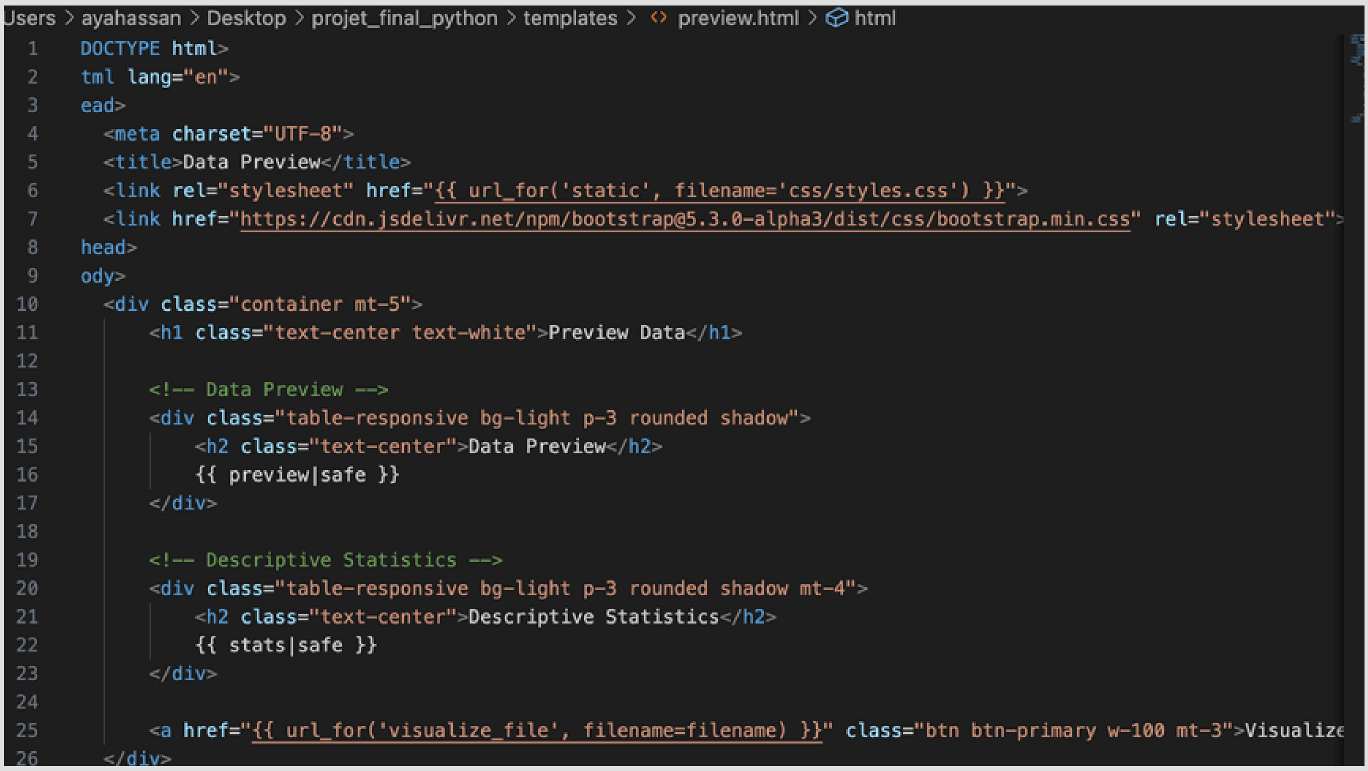


Users > ayahassan > Desktop > projet_final_python > templates > upload.html > html > body > div.container.mt-5 > form.p-4.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Upload CSV</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet
</head>
<body>
    <div class="container mt-5">
        <h1 class="text-center text-white">Upload Your CSV File</h1>
        <form action="/" method="post" enctype="multipart/form-data" class="p-4 bg-light rounded shadow">
            <div class="mb-3">
                <input type="file" class="form-control" name="file" accept=".csv" required>
            </div>
            <button type="submit" class="btn btn-primary w-100">Upload</button>
        </form>
    </div>
</body>
</html>
```

This form allows users to upload their CSV files. It uses the HTML <input type="file"> element for file selection. The enctype="multipart/form-data" attribute ensures that the file is sent correctly to the server when the form is submitted.

preview.html (Data Preview and Stats)

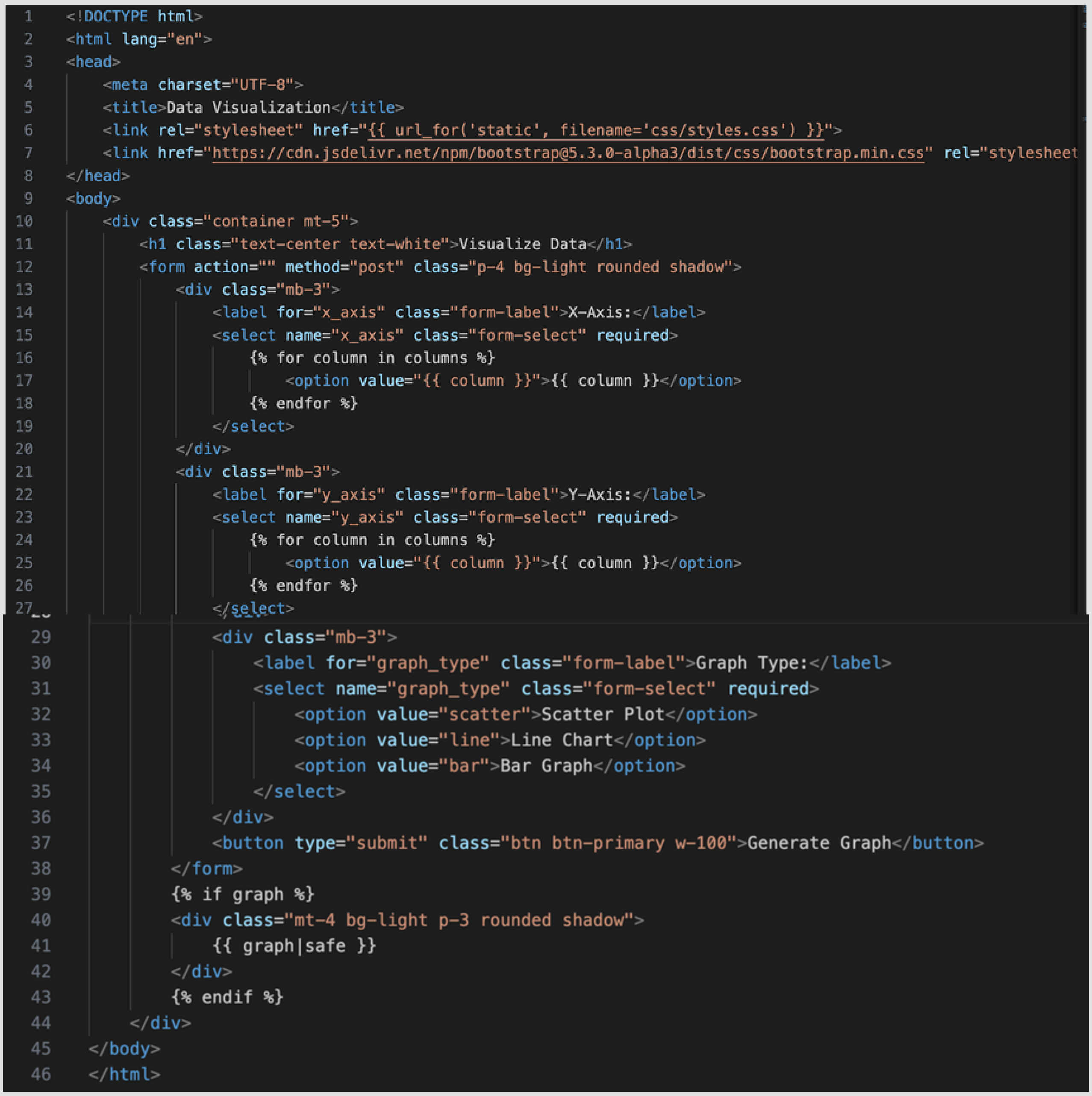


Users > ayahassan > Desktop > projet_final_python > templates > preview.html > html

```html
DOCTYPE html>
tml lang="en">
ead>
  <meta charset="UTF-8">
  <title>Data Preview</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet">
head>
ody>
  <div class="container mt-5">
      <h1 class="text-center text-white">Preview Data</h1>

      <!-- Data Preview -->
      <div class="table-responsive bg-light p-3 rounded shadow">
          <h2 class="text-center">Data Preview</h2>
          {{ preview|safe }}
      </div>

      <!-- Descriptive Statistics -->
      <div class="table-responsive bg-light p-3 rounded shadow mt-4">
          <h2 class="text-center">Descriptive Statistics</h2>
          {{ stats|safe }}
      </div>

      <a href="{{ url_for('visualize_file', filename=filename) }}" class="btn btn-primary w-100 mt-3">Visualize
  </div>
```

This page renders the preview of the data (first few rows) and the descriptive statistics. The {{ preview|safe }} and {{ stats|safe }} insert the generated HTML tables from the Flask app into the page.

visualize.html (Graph Visualization Form)

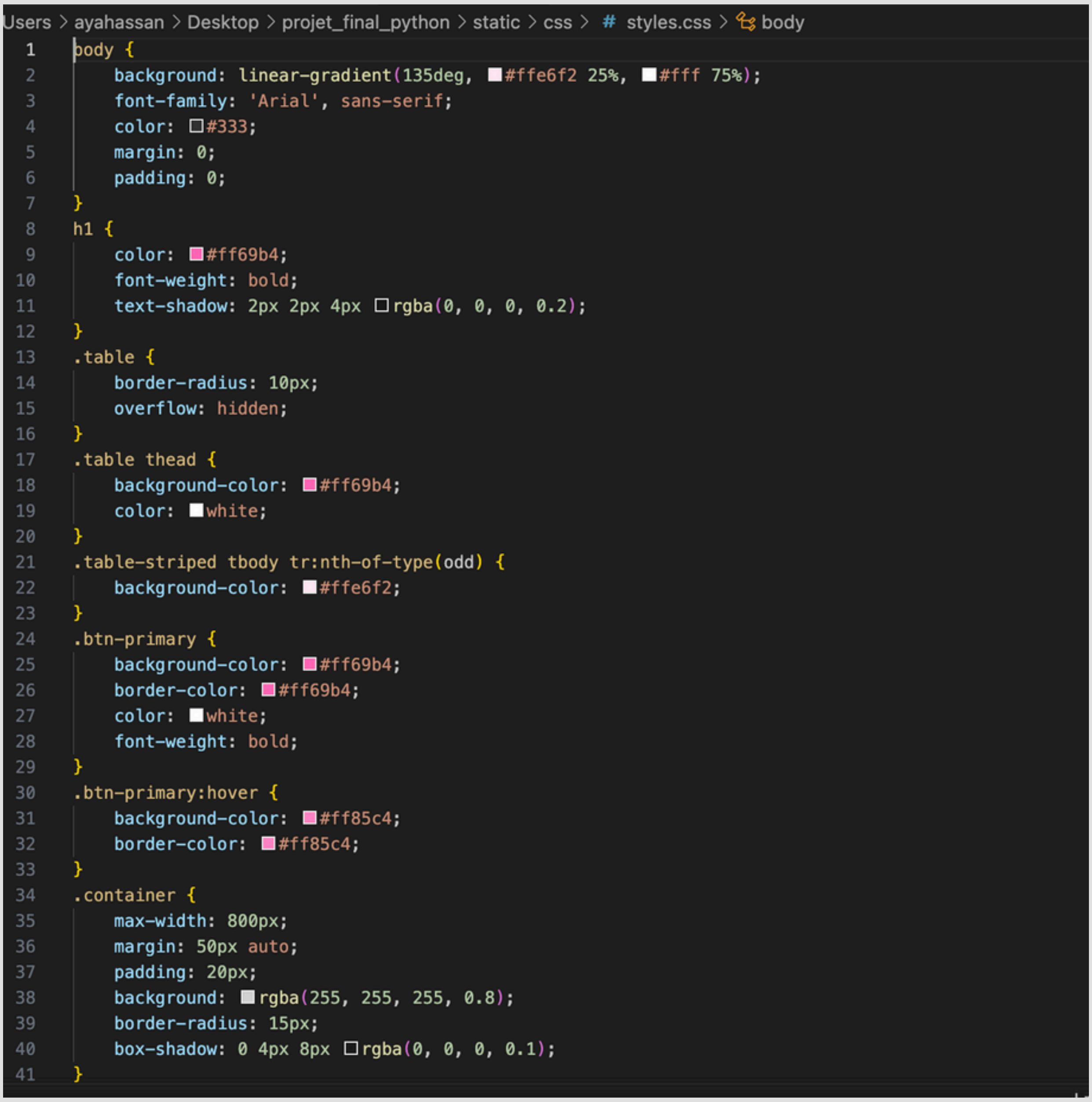


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Data Visualization</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet
</head>
<body>
    <div class="container mt-5">
        <h1 class="text-center text-white">Visualize Data</h1>
        <form action="" method="post" class="p-4 bg-light rounded shadow">
            <div class="mb-3">
                <label for="x_axis" class="form-label">X-Axis:</label>
                <select name="x_axis" class="form-select" required>
                    {% for column in columns %}
                        <option value="{{ column }}">{{ column }}</option>
                    {% endfor %}
                </select>
            </div>
            <div class="mb-3">
                <label for="y_axis" class="form-label">Y-Axis:</label>
                <select name="y_axis" class="form-select" required>
                    {% for column in columns %}
                        <option value="{{ column }}">{{ column }}</option>
                    {% endfor %}
                </select>
            <div class="mb-3">
                <label for="graph_type" class="form-label">Graph Type:</label>
                <select name="graph_type" class="form-select" required>
                    <option value="scatter">Scatter Plot</option>
                    <option value="line">Line Chart</option>
                    <option value="bar">Bar Graph</option>
                </select>
            </div>
            <button type="submit" class="btn btn-primary w-100">Generate Graph</button>
        </form>
        {% if graph %}
        <div class="mt-4 bg-light p-3 rounded shadow">
            {{ graph|safe }}
        </div>
        {% endif %}
    </div>
</body>
</html>
```

This form allows the user to choose which columns to use for the X and Y axes in the selected graph type. The for loop dynamically generates dropdown options for each column in the CSV file.

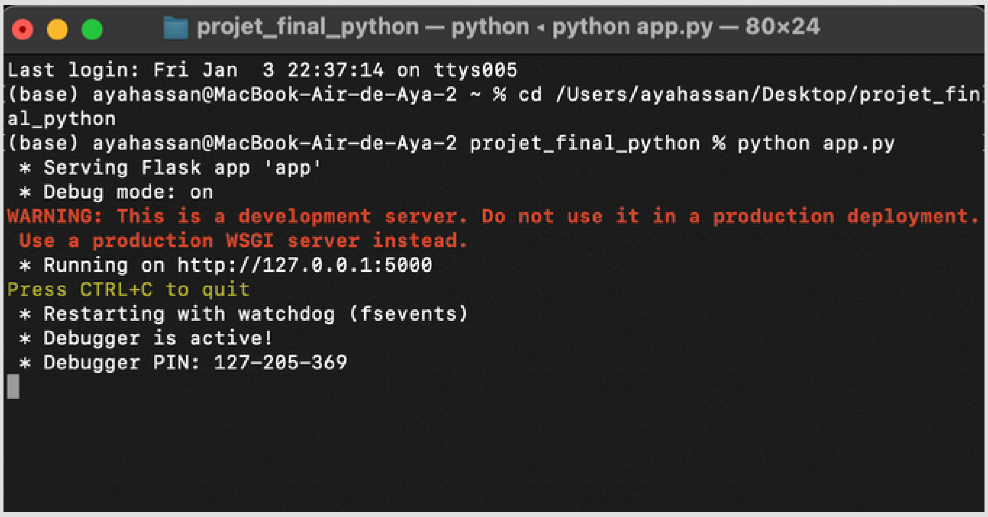
c.CSS (Styling)

The CSS file is designed to make the interface visually appealing and user-friendly. It styles buttons, tables, and general layout.



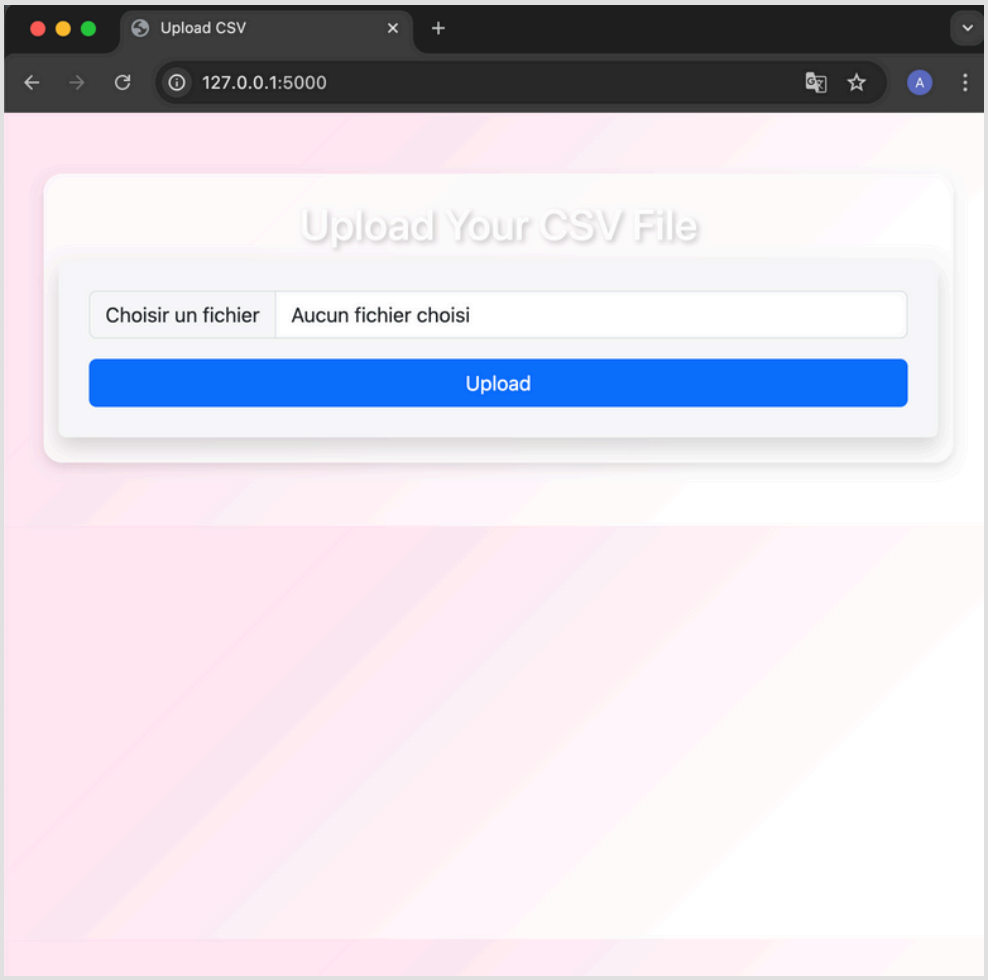
```
Users > ayahassan > Desktop > projet_final_python > static > css > # styles.css > body
body {
    background: linear-gradient(135deg, #ffe6f2 25%, #fff 75%);
    font-family: 'Arial', sans-serif;
    color: #333;
    margin: 0;
    padding: 0;
}
h1 {
    color: #ff69b4;
    font-weight: bold;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2);
}
.table {
    border-radius: 10px;
    overflow: hidden;
}
.table thead {
    background-color: #ff69b4;
    color: white;
}
.table-striped tbody tr:nth-of-type(odd) {
    background-color: #ffe6f2;
}
.btn-primary {
    background-color: #ff69b4;
    border-color: #ff69b4;
    color: white;
    font-weight: bold;
}
.btn-primary:hover {
    background-color: #ff85c4;
    border-color: #ff85c4;
}
.container {
    max-width: 800px;
    margin: 50px auto;
    padding: 20px;
    background: rgba(255, 255, 255, 0.8);
    border-radius: 15px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
```

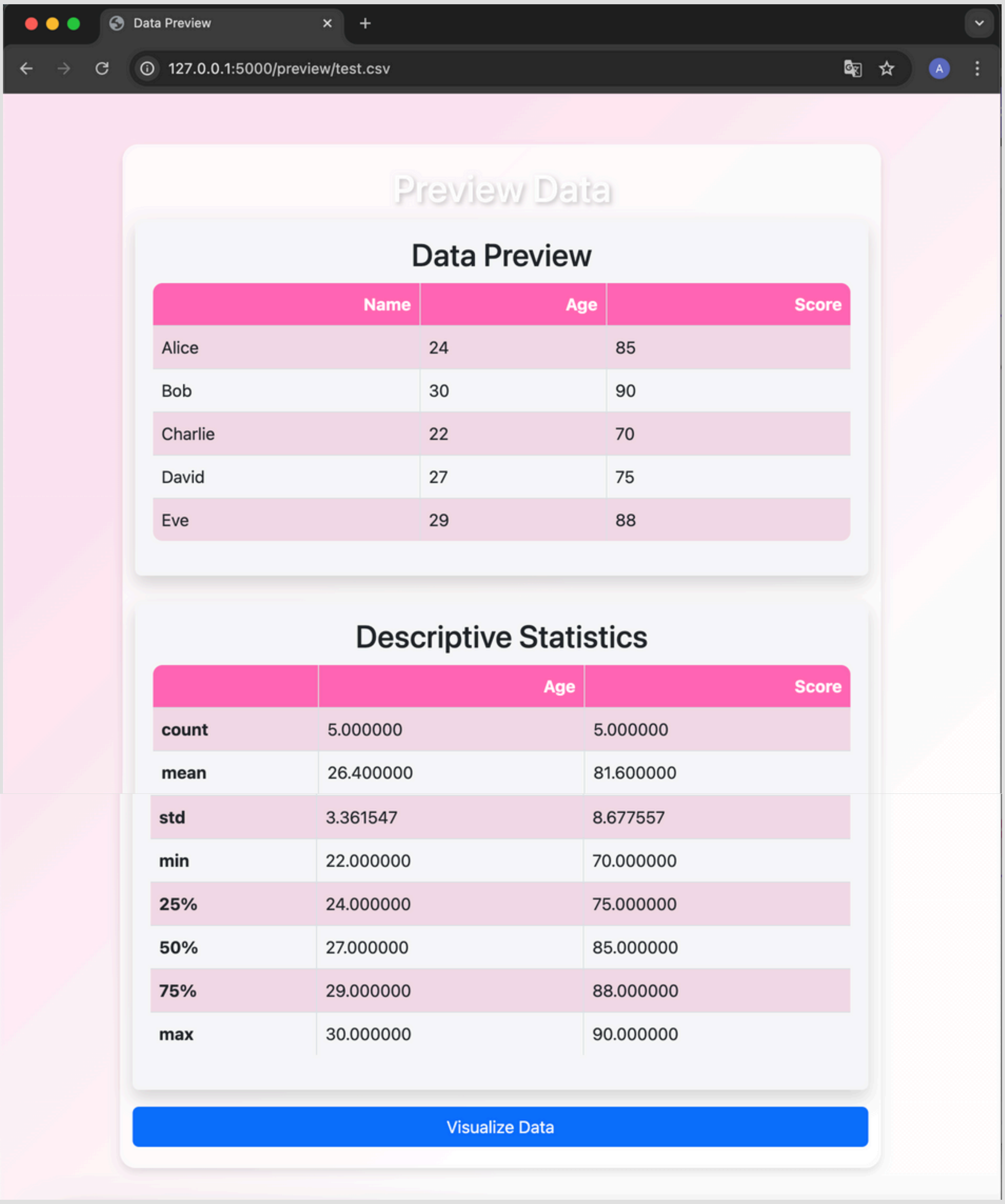
Test:



```
projet_final_python — python ‹ python app.py — 80×24
Last login: Fri Jan  3 22:37:14 on ttys005
(base) ayahassan@MacBook-Air-de-Aya-2 ~ % cd /Users/ayahassan/Desktop/projet_fin
al_python
(base) ayahassan@MacBook-Air-de-Aya-2 projet_final_python % python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (fsevents)
 * Debugger is active!
 * Debugger PIN: 127-205-369
```

*Running on http://127.0.0.1:5000/
This indicates the server is running. Open your browser and navigate to http://127.0.0.1:5000/.





Data Preview

127.0.0.1:5000/preview/test.csv

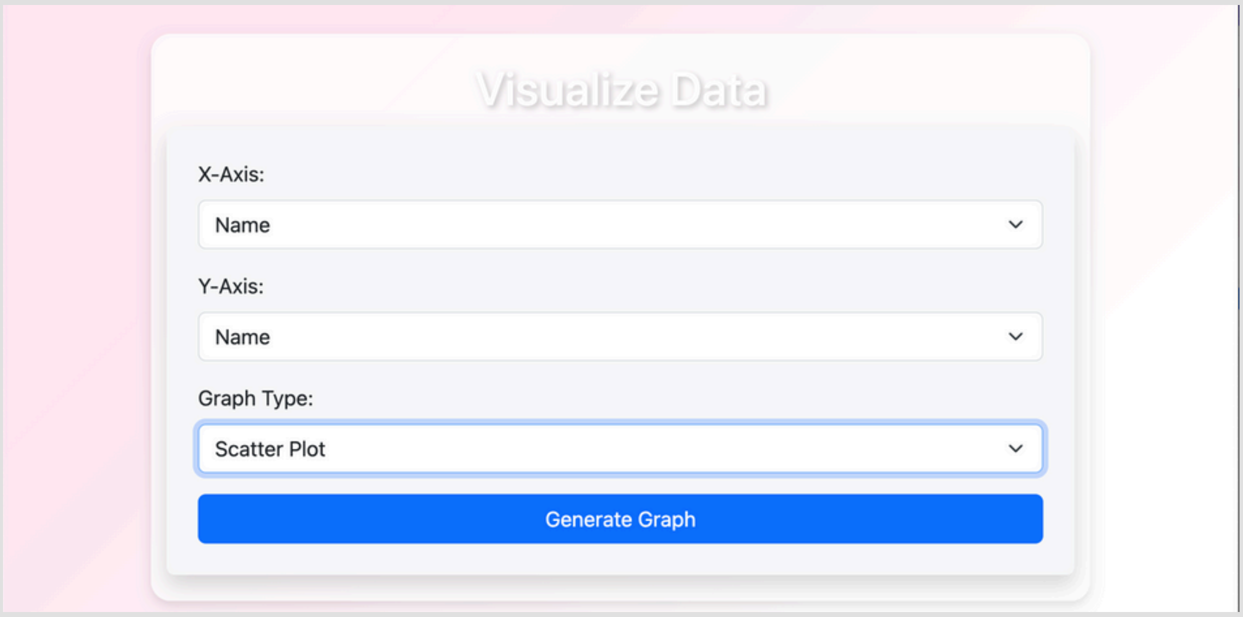
Preview Data

Data Preview

Name	Age	Score
Alice	24	85
Bob	30	90
Charlie	22	70
David	27	75
Eve	29	88

Descriptive Statistics

	Age	Score
count	5.000000	5.000000
mean	26.400000	81.600000
std	3.361547	8.677557
min	22.000000	70.000000
25%	24.000000	75.000000
50%	27.000000	85.000000
75%	29.000000	88.000000
max	30.000000	90.000000

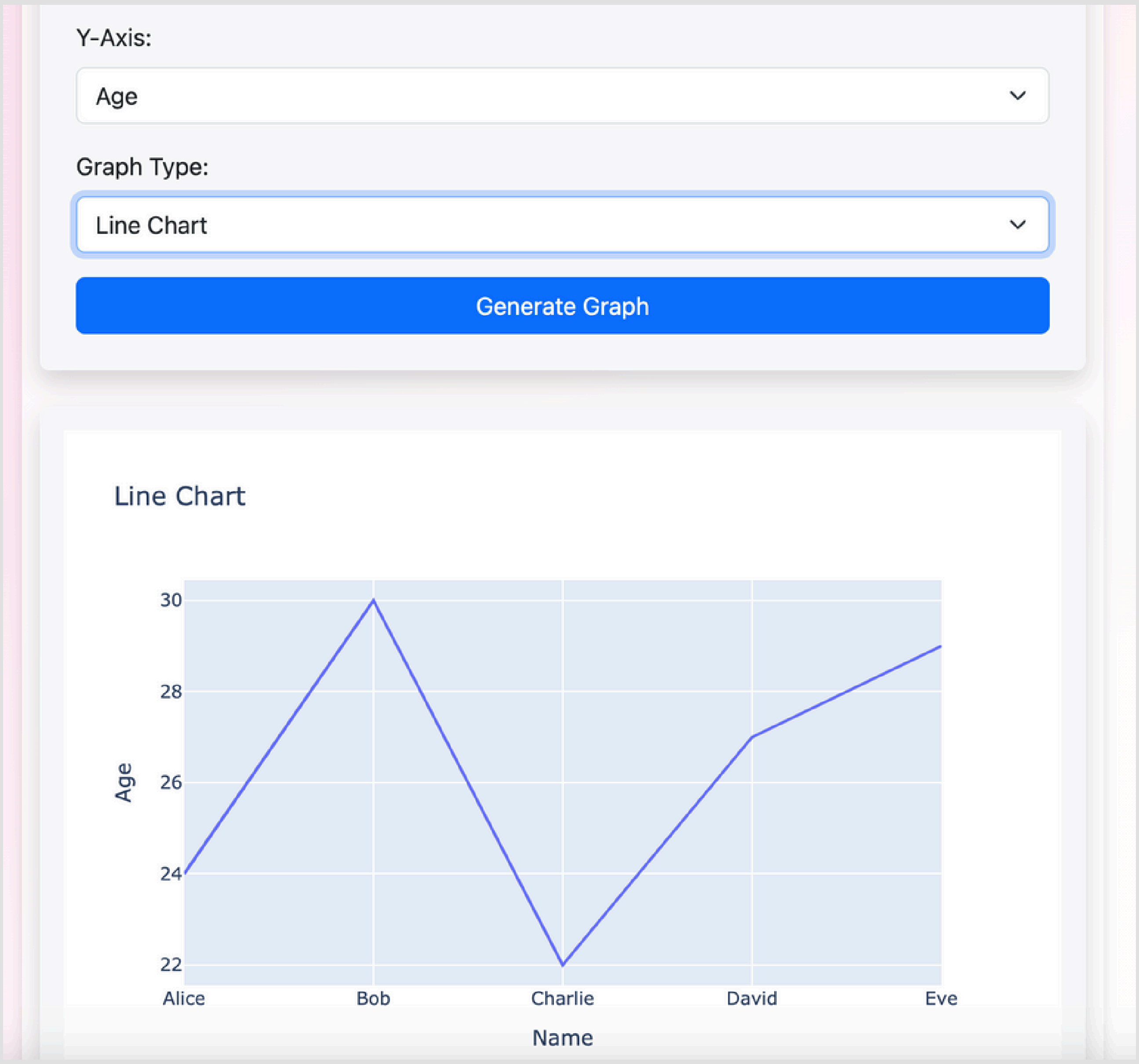


Visualize Data

X-Axis: Name

Y-Axis: Name

Graph Type: Scatter Plot


Y-Axis: Age

Graph Type: Line Chart

Conclusion

This project demonstrates how to use Flask, Pandas, and Plotly to create an interactive web application for data analysis. It meets the needs of a user who wants to upload CSV files, get descriptive statistics, and visualize the data. With this project, users can easily analyze simple datasets without needing to work with complex data processing tools.