**Financial Data Visualization WebApp**

This project is a web application that allows users to upload financial data in CSV format, visualize the data through bar and pie charts, and dynamically display the chart by year. The application is built using Flask for the backend and React for the frontend.

**Code Walkthrough**

**Project Structure**

financial-app/

├── Backend/

│ └── app.py # Flask backend script

├── node\_modules/ # Node.js dependencies (generated by npm)

├── public/ # Public files for the frontend (auto-generated)

├── src/

│ ├── components/

│ │ ├── ChartDisplay.jsx # Component for displaying charts

│ │ └── FileUpload.jsx # Component for file upload

│ ├── App.css # CSS for styling the frontend

│ ├── App.js # Main React component

│ ├── index.css # Global CSS

│ ├── index.js # Entry point for React

│ ├── logo.svg # React logo

│ ├── reportWebVitals.js # optional

│ └── setupTests.js # optional

│

├── .gitignore # optional

├── package-lock.json # NPM lock file (auto-generated)

├── package.json # NPM package configuration (auto-generated)

└── README.md # Project documentation

**Backend (Flask) - app.py**

The backend of the application is built using Flask, it handles file uploads and processes the CSV data into JSON format suitable for visualization.

**Endpoint**: **/upload (upload\_file())**

* + Accepts POST requests containing the CSV file.
  + Processes the CSV data using the Pandas library.
  + Aggregates the data by year and category.
  + Returns the aggregated data as JSON.

**Data Processing**: (**aggregate\_data(df)**)

* + Converts the 'Date' column to datetime format and extracts the year.
  + Groups the data by 'Year' and 'Category' and sums the 'Amount' column.
  + Converts the grouped data into a dictionary format suitable for JSON serialization.

**Frontend (React)**

The frontend of the application is built using React. It provides a simple user-friendly interface for uploading files and visualizing the data.

**Main Components**

1. **App.js**

* Manages the state of the application, including the uploaded data (chartData) and the currently selected year (selectedYear).
* Contains the *handleFileUpload* function, which is triggered when a file is successfully uploaded. This function sets the chartData state with the processed data from the backend.
* Provides a dropdown for selecting the year, which updates the pie chart accordingly.

1. **FileUpload.jsx**

* This component is responsible for handling file uploads.
* Contains an <input> element of type file that allows users to select a CSV file.
* Upon successful upload, it triggers the *onUploadSuccess* callback function (passed as a prop) with the data returned from the backend.

1. **ChartDisplay.jsx**

* This component is responsible for rendering the bar and pie charts.
* Receives the processed data and the selected year as props and renders the charts accordingly.
* Bar and Pie charts are generated using the react-chartjs-2 and Chart.js library.

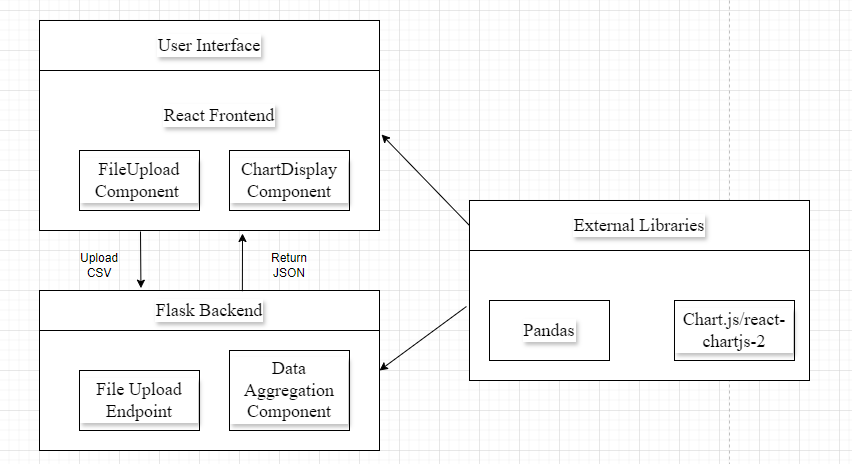
**Styles (CSS)**

* **App.css**: Contains styling rules for the application, including layout, typography, and component-specific styles.

**External Libraries**

* **Pandas**: Used in the backend for data manipulation and aggregation.
* **Flask-CORS**: To allow communication between the React frontend and Flask backend.
* **Chart.js / React-Chartjs-2**: Used in the frontend for rendering bar and pie charts.

**Architecture diagram**



**Prerequisites**

To run this project, we will need:

* **Node.js** and **npm**: For running the React frontend.
* **Python 3.12**: For running the Flask backend.
* **pip**: Python package manager to install Flask and other dependencies.

**Setup Instructions**

**1. Download the Repository**

*WebApp\_Project.zip*

*cd financial-app*

**2. Install Backend Dependencies**

Navigate to the Backend/ directory and install the required Python packages:

*cd Backend*

*pip install flask pandas flask-cors*

**3. Install Frontend Dependencies**

Navigate to the root directory (financial-app/) and install the Node.js dependencies:

*npm install*

**4. Run the Backend**

Start the Flask server:

*cd Backend*

*python app.py*

The server should start on *http://127.0.0.1:5000*.

**5. Run the Frontend**

In a new terminal window, start the React application:

*npm start*

The React app should automatically open in your default browser at *http://localhost:3000*.

**Using the Application**

1. **Upload a CSV File**: Click on the "Choose File" button, select a CSV file containing financial data, and upload it.
2. **View Charts**: After uploading, the bar and pie charts will display the financial data.
3. **Select Year**: Use the dropdown to select a different year, and the pie chart will update accordingly.

**Troubleshooting**

**CORS Issues -** If you encounter Cross-Origin Resource Sharing (CORS) errors, ensure the Flask server has CORS enabled using the flask-cors package.

**Module Not Found -** Ensure all dependencies are installed using pip install for Python packages and npm install for Node.js packages.

**Data Not Displaying -** Ensure your CSV file is correctly formatted with the required columns. Check the console for any error messages.