Project Report: Data Visualization Dashboard

Overview

This project is a full-stack data visualization dashboard built to meet the requirements outlined in the assignment. The goal was to ingest a provided JSON data file, store the data in a database, build a backend API to serve the data, and create an interactive dashboard that visualizes key insights using various charts and filters.

Technologies & Tools

- **Backend**
  - **Flask (Python):** Used to develop a RESTful API.
  - **Flask-PyMongo:** Integrated Flask with MongoDB for database operations.
  - **MongoDB:** A NoSQL database used to store the JSON data.
  - **Flask-CORS:** Enabled Cross-Origin Resource Sharing (CORS) to allow the React frontend to communicate with the Flask backend.
- **Frontend**
  - **React.js:** Used for building a dynamic, single-page dashboard.
  - **Create React App:** Provided a boilerplate for the React application.
  - **Bootstrap:** Utilized for responsive styling and layout.
  - **react-chartjs-2 / Chart.js:** Used to render interactive charts (e.g., bar charts) that display data insights.
- **Development Environments**
  - **PyCharm:** Used for backend (Python) development.
  - **VS Code:** Used for frontend (React) development.

Architecture & Implementation

Backend Implementation (Flask + MongoDB)

1. **Data Ingestion:**
   - The provided `jsondata.json` file is loaded into MongoDB.

    - A check ensures data is only loaded if the database is empty, preventing duplicate entries.
2. **API Endpoints:**
    - **`/data`:**
        - Returns all records from MongoDB.
        - Supports query parameters (such as `country`, `topic`, `sector`, `region`, and `end_year`) for filtering data.
    - **`/stats`:**
        - Returns aggregated statistics (e.g., average intensity per sector) using MongoDB aggregation.
    - **`/data/<string:topic>`:**
        - Provides a filtered view based on a specific topic.
3. **Database:**
    - MongoDB is used to store the data in a collection named `data` within the `jsondata_db` database.
4. **Running the Backend:**

The Flask server runs on `http://127.0.0.1:5000` and is started via the command: `python app.py`

Frontend Implementation (React)

1. **Project Setup:**
    - Created using **Create React App**.
    - The project structure separates components from the main application logic.
2. **Dashboard Component:**
    - **Filter Section:**
        - Users can input filter criteria (e.g., country, sector, topic, region, end year).
        - Upon applying filters, the component calls the Flask API with the query parameters to fetch the corresponding data.
    - **Data Table:**
        - Displays the raw data in a table format.
        - Columns include End Year, Intensity, Sector, Topic, Region, Country, Relevance, Pestle, Source, and Likelihood.
    - **Chart Visualization:**
        - A bar chart (implemented using Chart.js via react-chartjs-2) shows the average intensity per sector.
        - This visual insight helps to quickly understand which sectors are most prominent according to the data.
3. **Running the Frontend:**

The React development server runs on `http://localhost:3000` and is started via the command:
`npm start`

4. **Data Flow:**
    - The frontend makes HTTP requests to the Flask backend to retrieve data and statistics.
    - Data is dynamically rendered based on user input through filters.

Project Output

When the project is running:

- **Dashboard Landing Page:**
    - A clean and responsive layout displays the dashboard title.
- **Filter Controls:**
    - Users can enter filter criteria to refine the dataset displayed.
- **Data Table:**
    - The table lists all records (or the filtered subset) retrieved from the backend.
- **Interactive Bar Chart:**
    - A chart visualizes the average intensity per sector, providing an overview of key trends.
- **Interactivity:**
    - Real-time updates occur as users apply filters, demonstrating dynamic data fetching and visualization.

Conclusion

This project demonstrates full-stack development capabilities by:

- Efficiently handling data ingestion from a JSON file.
- Storing and querying data using MongoDB.
- Building a robust Flask API to serve dynamic data.
- Creating an interactive, user-friendly dashboard using React and Chart.js.

It addresses all the key requirements provided in the assignment and serves as a comprehensive example of integrating backend and frontend technologies to produce meaningful data visualizations.
