**Visualizer Documentation By Muhluri Mhlongo**

**Overview**

This project is a full-stack web application designed to process and visualize data. The solution allows users to upload files containing data (in JSON or CSV format), stores it in a PostgreSQL database, and visualizes it using different types of graphs such as line charts, bar charts, and pie charts. Additionally, the application includes a filter function that allows users to filter data by date and category.

**Features:**

* **File Upload**: Users can upload JSON or CSV files.
* **Data Processing**: The backend processes the uploaded data and stores it in a PostgreSQL database.
* **Data Visualization**: The frontend visualizes the data using various types of graphs (line chart, bar chart, pie chart).
* **Filtering**: Users can filter the data by date and category, or apply both filters simultaneously.

**Tech Stack**

* **Frontend**: React
  + axios: To make HTTP requests to the backend.
  + react-chartjs-2: For rendering various types of charts (line, bar, and pie).
  + file-reader: To read uploaded files in the frontend.
* **Backend**: Spring Boot
  + Spring Web: To build RESTful APIs for the frontend to interact with.
  + Spring Data JPA: For database operations (CRUD operations).
  + Apache POI: For reading CSV files (if needed).
  + PostgreSQL: A relational database used to store the uploaded data.

**How It Works**

1. **Frontend (React)**:
   * The user uploads a file (either JSON or CSV).
   * The file is sent to the Spring Boot backend via an HTTP POST request (using axios).
   * After the file is successfully uploaded and processed, the frontend fetches the data from the backend and displays it as a graph using react-chartjs-2.
2. **Backend (Spring Boot)**:
   * The backend exposes RESTful APIs to handle file uploads and data retrieval.
   * The backend uses Spring Data JPA to store the processed data into a PostgreSQL database.
   * If the file is in JSON format, it is parsed using the ObjectMapper from the Jackson library. For CSV files, the backend uses Apache POI or OpenCSV to parse the data.
   * The backend provides APIs for the frontend to query the stored data, including support for filtering by date and category.
3. **Data Visualization**:
   * The frontend uses react-chartjs-2 to display data in different types of charts:
     + **Line Chart**: To show trends over time (e.g., total confirmed cases).
     + **Bar Chart**: To compare daily cases or other metrics.
     + **Pie Chart**: To show proportions (e.g., active cases vs. recovered cases).
4. **Filtering**:
   * Users can filter data by a date range (start date to end date).
   * Users can also filter data by specific categories (e.g., total active cases).
   * Both filters can be applied simultaneously to refine the data visualization.

**Dependencies**

**Backend:**

* Spring Web: To build REST APIs.
* Spring Data JPA: For database interaction and CRUD operations.
* Apache POI: To read CSV/Excel files if needed.
* PostgreSQL Driver: For connecting to PostgreSQL.
* Jackson ObjectMapper: For parsing JSON files.

**Frontend:**

* axios: For making API calls to the backend.
* react-chartjs-2: To render charts on the frontend.
* file-reader: For reading files on the frontend.

**Steps to Run the Application**

**Backend:**

1. Clone the repository.
2. Set up your PostgreSQL database.
3. Update application.properties with your PostgreSQL connection details.
4. Run the backend Spring Boot application (mvn spring-boot:run).
5. The backend should now be running at http://localhost:8080.

**Frontend:**

1. Navigate to the frontend folder.
2. Install dependencies by running npm install.
3. Start the frontend React app with npm start.
4. The frontend should now be running at http://localhost:3000.

**Sample Data Processing**

* If the file is in JSON format, it will be processed using the ObjectMapper from the Jackson library in Spring Boot.
* If the file is in CSV format, it will be processed using either Apache POI or OpenCSV, depending on the file type.
* Once the file is parsed, the data is stored in the PostgreSQL database, and the frontend can retrieve it for visualization.

**Conclusion**

This solution effectively addresses the requirements of the assessment. It provides a full-stack application with the ability to upload, process, store, and visualize data. By using React for the frontend, Spring Boot for the backend, and PostgreSQL for the database, the application is modular, scalable, and easy to maintain. The use of various chart types and filtering options enhances the data visualization and allows users to gain insights from the data in a flexible way.