Stock Data Visualization Documentation

By- Ankur Baijal

1. Introduction

The **Stock Data Visualization** project is a web-based application that allows users to visualize stock market data for a selection of companies. This project aims to provide an intuitive interface for users to explore various metrics related to stock performance. The application supports dynamic charting, enabling users to select different metrics to view their stock data graphically.

Goals of the Project

- **User Engagement**: To provide a simple and effective way for users to interact with stock data.
- **Data Visualization**: To present data in an easily digestible format through charts.
- **Downloadable Resources**: To allow users to save visualized data for offline analysis.

---

2. Features

2.1 Company Selection

- A user-friendly list of companies is presented for selection.
- Users can click on a company name to view its stock data.

2.2 Dynamic Charting

- Displays the stock performance in a line chart format.
- Users can select various metrics to visualize (e.g., Closing Value, Volume).

2.3 Download Chart

- Users have the option to download the displayed chart as a PNG image.
- This feature allows for easy sharing and offline analysis.

2.4 Search Functionality

- A search input is provided to filter through the company list.
- Users can quickly find companies by typing their names.

2.5 Responsive Design

- The application interface is designed to adjust to various screen sizes.
- Ensures usability across different devices (desktops, tablets, mobile).

---

3. Technologies Used

3.1 Frontend Technologies

- **HTML**: Used to structure the web application, creating the layout and defining elements.
- **CSS**: Utilized for styling the application, ensuring a visually appealing interface.
- **JavaScript**: Adds interactivity and functionality, enabling dynamic updates based on user input.
- **Chart.js**: A powerful library for rendering interactive charts.
- **Bootstrap**: A framework that provides responsive design components, facilitating a better user experience.

---

4. Project Structure

The project is organized in a straightforward manner, making it easy to navigate and understand. Here's a breakdown of the project structure:

graphql
Copy code

```
Stock-Data-Visualization/
/project-root
    /frontend
        - index.html
        - styles.css
        - script.js
    /backend
        - server.js
    /data
        - dump.csv
    - package.json
```

---

5. Setup Instructions

5.1 Frontend Setup

Follow these steps to set up the frontend of the project on your local machine:

Step 1: Clone the Repository

1. Open your terminal or command prompt.

Run the following command to clone the repository:
bash

```
git clone
https://github.com/ankurbaijal123/Stock-Data-Visualization.git
```

2.

Change into the project directory:
bash

```
cd Stock-Data-Visualization
```

3.

Step 2: Open the Project

1. Locate the `index.html` file in the project folder.
2. Open it in your web browser. You can do this by double-clicking the file or by using the terminal command based on your operating system:
   ○

For Windows:

```
start index.html
```

5.2 Backend Setup

If your project includes a backend service (like Node.js) to serve the data, follow these instructions:

Step 1: Navigate to the Backend Directory
If your project has a separate backend folder, navigate to it:
bash
Copy code

```
cd backend  # Adjust the path if necessary
```

1.

Step 2: Install Dependencies

Install the necessary dependencies required for the backend service:
bash
Copy code
```
npm install
```

1.

Step 3: Start the Server
Launch the backend server to serve data to the frontend:
bash
```
node server.js
```

1.

---

6. Usage

6.1 Company List

- Upon opening the application, users will see a list of available companies on the left side of the interface.
- Clicking on a company name will load its corresponding stock data in the chart area.

6.2 Chart Display

- The right side of the application features a chart displaying the stock data for the selected company.
- Users can choose different metrics (such as Closing Value, Opening Value, etc.) from a dropdown menu to visualize different data sets.

6.3 Download Chart

- Below the chart, there is a button labeled "Download Chart."
- Clicking this button will download the current chart as a PNG image for offline use or analysis.

6.4 Search Functionality

- A search box is provided above the company list.
- Users can type in the search box to filter the displayed list of companies in real-time.

---

7. Functionality Explained

7.1 Company List

- The application fetches a unique list of company names from the backend and displays them in a scrollable list.
- The list is designed to be intuitive, allowing users to click on a company name to load its stock data.

7.2 Dynamic Charting

- When a user clicks on a company, the application retrieves the stock data associated with that company.
- The chart updates to reflect the selected company's stock performance, offering a visual representation of trends over time.

7.3 Download Feature

- The download button utilizes HTML5 canvas features to convert the chart into an image format (PNG).
- This allows users to save the visualized data for reports or presentations.

7.4 Search Implementation

- The search input listens for user input and filters the list of companies dynamically.
- As users type, the list updates in real-time, hiding companies that do not match the search query.

Each functionality and there Design choices

9. HTML Structure (`index.html`)

The HTML file serves as the backbone of the application, providing the structure and layout for the user interface.

Key Sections:

- **Head Section**:
  - Imports Bootstrap for responsive design.
  - Links the custom CSS file (`style.css`) for styling.
  - Sets the character encoding and viewport settings for mobile responsiveness.
- **Body Section**:
  - Contains a container for the main content.
  - Includes a header for the page title.
  - Divides the layout into two main columns: one for the company list and another for the chart.

Design Choices:

- **Responsive Layout**: Utilizes Bootstrap's grid system for responsiveness, ensuring the application looks good on all devices.
- **Semantic HTML**: Uses headings and lists to clearly define content sections, improving accessibility and SEO.

---

10. CSS Styles (`style.css`)

The CSS file contains styles that enhance the appearance and layout of the application.

Key Styles:

- **Container and Layout**:
  - Sets padding, margin, and background colors to create a clean, organized layout.
- **Company List**:
  - Styles for the company list to enhance visibility and interaction.

Design Choices:

- **Visual Hierarchy**: Uses contrasting colors to distinguish different sections and highlight important elements (like the selected company).
- **Usability**: Ensures the layout is visually appealing and easy to navigate, enhancing user experience.

---

11. JavaScript Functionality (`script.js`)

The JavaScript file manages the logic and interactivity of the application.

Key Functions:

- **Initializing the Application**:
  - Fetches company data and populates the company list on page load.
- **Populating Company List**:
  - Dynamically generates list items for each company and attaches event listeners for interactivity.
- **Loading Company Data**:
  - Retrieves and displays stock data for the selected company in a chart.
- **Updating the Chart**:
  - Uses Chart.js to render the selected company's stock data on a line chart.

Design Choices:

- **Modular Code**: Functions are broken down into specific tasks for clarity and maintainability.
- **Dynamic Updates**: The use of `fetch` and event listeners allows the UI to react to user interactions without requiring a full page refresh.

---

12. Conclusion

This documentation provides a comprehensive overview of the **Stock Data Visualization** project's codebase, covering the structure, styles, and functionality. By following this documentation, developers can understand the design choices and logic behind the implementation, making it easier to maintain, enhance, or expand the application in the future