StockMarket Dashboard

Development Approach

The project was developed in two main phases — backend first, followed by the frontend. For the backend, I used Node.js with Express.js to build modular API routes for companies, historical data, metrics, and predictions. SQLite (via `better-sqlite3`) was integrated for persistent storage of historical stock data. To implement stock price prediction, I created a Python script using scikit-learn's LinearRegression model, reading JSON input from stdin and returning predictions in JSON format.Once the backend API was functional, I moved to the frontend using HTML, CSS, and JavaScript. I built a responsive dashboard UI that dynamically fetches data from the backend APIs and uses Chart.js to visualize historical prices and metrics. The design incorporated a lite theme, a responsive sidebar, and a hamburger menu for smaller screens.

Technologies Used

- **Backend:** Node.js, Express.js, better-sqlite3, yahoo-finance2, Python 3, scikit-learn, NumPy
- **Frontend:** HTML, CSS, JavaScript, Chart.js
- **Database:** SQLite
- **Tools:** Nodemon, body-parser, CORS

Challenges Encountered

One of the key challenges was integrating the frontend and backend in the same project, ensuring smooth API communication and correct file serving from the public directory. Another major issue arose with **yahoo-finance2** — many older Yahoo Finance API functions (like `historical()`) are now deprecated, requiring me to adapt the code to use the newer `chart()` method for fetching stock data. Fetching and processing this data efficiently while keeping the code maintainable was critical. Additionally, ensuring real-time chart updates, handling asynchronous data loading, and keeping the UI responsive required careful planning and event handling. These challenges ultimately strengthened the architecture and made the project more robust.