StockPulse: Stock Analyser

Front End Development (22CS021)

Submitted by

Dewansh – 2310993812
Mohit – 2310993878

Semester: 5

BE-CSE (Artificial Intelligence)

Guided by

Dr. Swati Malik

CHITKARA UNIVERSITY INSITUTE OF ENGINEERING & TECHNOLOGY

CHITKARA UNIVERSITY, RAJPURA

AUGUST 2025

ACKNOWLEDGEMENTS

With immense pleasure, we, **Mohit (2310993878), Dewansh Jha (2310993812)**, present the project report titled **"Stock Pulse"** as a part of the curriculum of **B.E. CSE (Artificial Intelligence).**We would like to express our heartfelt gratitude to our project guide, **Dr. Swati**, for his valuable guidance, continuous support, and encouragement throughout the development of this project. His insightful feedback and technical mentorship were instrumental in shaping this work.

We are also thankful to our respected Dean, **Dr. Sushil Narang**, for providing us with this opportunity to work on an impactful and real-world project in the domain web development. His support and leadership helped create an environment that fostered innovation and learning.

This project has been a significant milestone in our academic journey, and we are grateful to everyone who supported us directly or indirectly.

Signature……

Name: Dewansh Jha

Roll No: 2310993812

Signature……..

Name: Mohit

Roll No: 2310993812

Abstract

In today's dynamic stock market, investors and analysts often struggle to identify early trends and track portfolio performance efficiently. This process typically involves visiting multiple financial websites, manually aggregating data, and using separate tools, which can be time-consuming, complex, and lack a unified view.

StockPulse is a web-based financial analysis platform designed to address these challenges. It offers a unified solution that integrates **real-time market data**, **dynamic portfolio management**, and **stock analysis tools**—all within a single, user-friendly interface.

The platform uses **React.js** to build a dynamic and component-based front end, ensuring scalability and ease of maintenance. Navigation between different pages is handled using **React Router DOM**, enabling a seamless browsing experience. To enhance user decision-making, the application connects to external financial APIs (like the Financial Modeling Prep API) to provide live stock prices and market data. This allows for powerful features like real-time portfolio valuation, where a user's holdings, gains, and losses are updated instantly.

In addition to its core market data functionalities, StockPulse also provides a comprehensive **portfolio management system** and **secure user authentication**. These tools not only simplify investment tracking but also provide a secure, personalized dashboard for each user. The user interface is styled with **Tailwind CSS** to maintain a clean, modern, and data-focused theme suitable for financial analysis.

By combining live market data with essential portfolio tools in one platform, StockPulse aims to offer a smarter, faster, and more insightful stock analysis experience. The system is **modular**, **responsive**, and designed to be **scalable**, with the potential for future integration of advanced charting libraries, news sentiment analysis, and AI-powered trend detection features.

Table of Contents

Table of Figures

1. Introduction

1.1. Objective

StockPulse is an innovative online stock analysis platform developed to simplify and enhance the way investors track the market and manage their portfolios. The project brings together all essential analysis tools on a single, dynamic website where users can monitor real-time stock prices, manage personal portfolios, and visualize market data. One of the key highlights of StockPulse is its direct integration with live financial data APIs, allowing it to provide up-to-the-minute stock quotes and calculate portfolio performance dynamically, making the experience highly relevant and powerful for making informed decisions.

The platform offers a variety of features accessible from a secure user dashboard, including a market overview, a search function for individual stocks, and a detailed portfolio tracker. From the main navigation, users can quickly access modules for viewing stock details and managing their holdings. Each section is designed to be intuitive, responsive, and visually clean to ensure smooth interaction across all devices. The application flow enables users to first get a broad market view, then analyze specific stocks, and finally track their own investments seamlessly in one place.

Technically, StockPulse has been built using a modern stack with React.js for the frontend, and React Router DOM managing navigation between pages. Styling is implemented through Tailwind CSS to keep the design clean, modern, and consistent. The development environment was Visual Studio Code, which supports a component-based architecture that makes the platform modular and easy to maintain. By combining this dynamic frontend with a powerful Node.js backend and MongoDB database, the platform ensures efficient and secure data handling for all user and market information.

The primary goal of StockPulse is to reduce the complexity of market analysis by offering an all-in-one solution where users can discover, analyze, and track their investments without the need to switch between multiple websites. The integration of live, third-party data is crucial, making the experience highly practical and relevant for helping investors make better, more timely decisions. Whether for a seasoned trader or a new investor, StockPulse provides the essential tools to manage every detail of a portfolio with confidence and clarity.

1.2. Technology Stack

•React (with Vite): Core framework used to build the application with component-based architecture for scalable development.

- `App.jsx` handles page routing and global structure.
- `main.jsx` sets up the root render and wraps the app with `BrowserRouter`.

• **React Router DOM:** Powers navigation between pages such as **Home**, **Login**, **Signup**, **Results** and **Add Portfolio** without page reloads.

• **HTML5:** Provides semantic structure and elements for all pages and components.

• **CSS3 (Modular + Global):**

- `App.css` and `index.css` manage base styles, layout, and responsiveness.
- Custom `style.css` adds component-specific styling for branding and theme consistency.

• **JavaScript (ES6+):** Handles client-side logic, dynamic rendering, and event-driven functionality across components.

UI Components & Effects

- **Intersection Observer API**: Enables animation-on-scroll functionality for cards and features to enhance user engagement.
- **Font Awesome**: Provides icons used in navigation bars and other interactive elements.

- **Responsive Design**: Achieved using flexible box layouts (Flexbox) and media queries, ensuring mobile and desktop compatibility.
- **Data Management**
  - **MongoDB**: Used to store all the latest stocks data , stocks news and the portfolio
- **Hosting**
  - The project is suitable for static hosting platforms such as GitHub Pages or Vercel, as it does not require server-side scripting or databases.

2. Project Description

In the current digital era, investors face a range of challenges while analyzing the stock market online. Most platforms either focus on providing raw data without personalized insights or require users to switch between multiple websites to gather news, view charts, and track their portfolios. This fragmented experience often leads to information overload, wasted time, and difficulty in making well-informed decisions. Additionally, traditional financial platforms rarely tailor the data presentation to a user's specific holdings or investment interests.

Another critical problem is the lack of integrated tools that can support the end-to-end process of analysis and monitoring—from discovering new opportunities to tracking existing assets. Users typically have to rely on separate apps or manual spreadsheets to monitor their portfolios, which increases the likelihood of missing critical market movements. For example, a user may know the total value of their portfolio but must manually check the news and performance for each individual stock. StockPulse aims to address these issues by providing a unified financial dashboard that combines essential services with real-time data streams. By integrating live stock quotes, portfolio performance tracking, and market news into a single application, StockPulse solves the problem of fragmented analysis while delivering a smoother, more personalized experience.

Furthermore, many existing financial solutions have cluttered, user-unfriendly interfaces that can be overwhelming, especially for investors who are not professional traders. Without clear guidance and data visualization, users can get lost in the numbers and may overlook trends relevant to their financial goals. StockPulse seeks to overcome these challenges by offering an intuitive, easy-to-navigate platform that adapts to individual portfolios and provides all essential analysis tools in one place.

In addition to personalization and integration, StockPulse also emphasizes clarity and accessibility. Modern investing can be complex, and understanding the performance of a diverse portfolio can be difficult. StockPulse provides features that allow users to see their performance at a glance, manage watchlists, and analyze stocks with clean, understandable visuals, ensuring they stay on top of their investments. By incorporating real-time updates and secure access to portfolio data, the platform enhances convenience and builds confidence. This holistic approach positions StockPulse as more than just a data feed—it becomes a trusted financial companion for the evolving needs of today's investors.

3. System Architecture & Design

To overcome the limitations of traditional financial analysis platforms, StockPulse proposes an all-in-one web-based solution that combines real-time data analysis, dynamic portfolio management, and essential market visualization tools in a single, user-friendly platform. The methodology focuses on a modular architecture, allowing for the seamless integration of features such as live stock tracking, portfolio performance calculation, data visualization, and secure user authentication.

The platform is developed using the **React.js** framework, enabling a responsive and dynamic user interface. Routing between pages is managed by **React Router DOM**, which ensures smooth navigation without full page reloads. Each feature, such as the portfolio dashboard or the stock search functionality, is implemented as an individual React component to promote reusability and maintainability. Data inputs are handled through controlled forms, while incoming market data and calculated portfolio values are managed through internal state and logic, populated by asynchronous API calls.

A key highlight of the methodology is the integration of **live financial data from third-party APIs**. User requests for stock information trigger backend calls to these external services, fetching up-to-the-minute price and company data. This real-time information is then used to power the platform's core functionalities. Other features, like the portfolio dashboard and performance calculators, are implemented using conditional logic and dynamic rendering based on this fetched data and the user's saved portfolio details.

To ensure visual consistency and a professional feel, styling is managed with the **Tailwind CSS** framework, applying a clean, dark-themed interface across all pages that is well-suited for data visualization. The entire application is hosted locally for testing and is architected for easy deployment to live cloud servers like Vercel and Render. The proposed methodology emphasizes simplicity, modularity, personalization, and scalability, making StockPulse a modern solution to the challenges of personal financial analysis.

3.1. High-Level System Architecture

The StockPulse application follows a multi-tier client-server architecture, comprising a React-based frontend, a RESTful backend built with Java (Spring Boot), and a MongoDB NoSQL database. The architecture ensures modularity, scalability, and ease of maintenance while enabling seamless communication between components via HTTP-based APIs.

Client (React) ⇄ REST API (Spring Boot) ⇄ MongoDB Database

- **Frontend (Client)**: React.js handles the user interface, routing, and state management.
- **Backend (Server)**: Spring Boot handles business logic, API endpoints, and authentication.

Database**: MongoDB stores user profiles, portfolio holdings, and cached stock information.**

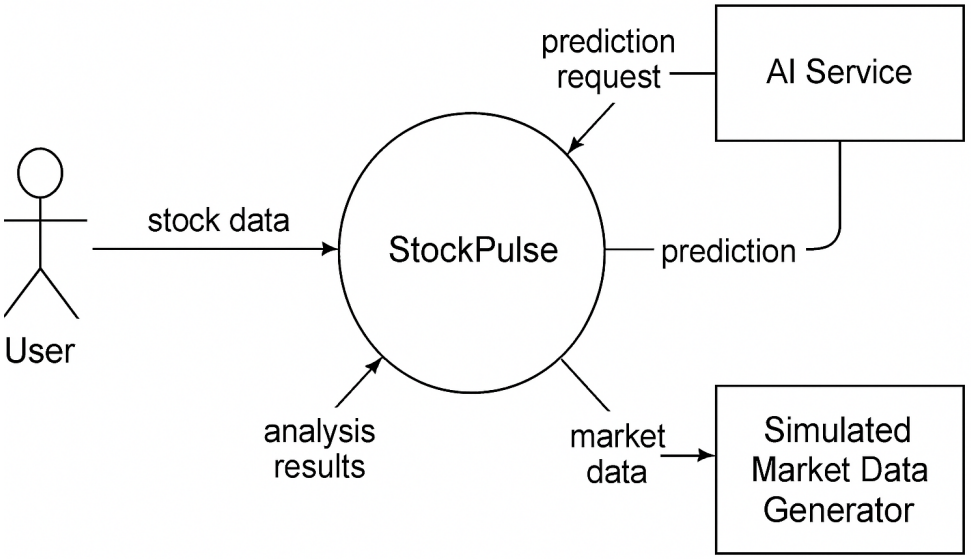
3.2 Detailed System Design

The system is structured into the following modular components:

- **Authentication & Landing Page**: Acts as the primary entry point for users. This module includes the landing page that highlights the platform's features, along with secure forms for user registration and login.
- **User Dashboard**: This is the central hub for authenticated users. It displays a high-level market overview, such as top gainers and losers, and provides navigation to all other features of the application.
- **Stock Search & Details**: A powerful component that allows users to search for specific stocks by their ticker symbol (e.g., AAPL, GOOGL). Selecting a stock directs the user to a detailed view with real-time price data, historical charts, and other relevant financial metrics fetched from the API.
- **Portfolio Management**: This is the core logic module of the application. It handles all CRUD (Create, Read, Update, Delete) operations for a user's personal stock holdings. It dynamically calculates the portfolio's total value, individual gains/losses, and overall performance using live market data.
- **API & Data Layer (Backend)**: The Node.js/Express server acts as the intermediary between the frontend, the database, and external financial APIs. It exposes secure RESTful endpoints, processes requests, and serves structured JSON data to the React client.
- **Responsive UI & User Experience**: The entire front-end is built with **React** and styled with **Tailwind CSS** to ensure a clean, modern, and fully responsive user interface. Interactive charts and cards are used to enhance the user experience and make complex data easy to understand across all devices.

3.3UML

FlowChart:



4. Database Design

The **StockPulse** application uses **MongoDB**, a NoSQL document database, to store and manage application data in collections. Each document in a collection is stored in a JSON-like format and can contain nested structures, which provides significant flexibility for handling financial data such as **user profiles, portfolio holdings, and cached stock information**.
The database design is centered around the core entities of the application, primarily the **Users**, **Portfolio**, and **Stocks** collections. This schema ensures a logical separation of data, allowing for efficient queries to retrieve a user's portfolio, populate it with the latest market data, and maintain a secure record of user information. This structure is designed to be both scalable and performant, supporting the application's real-time data requirements.

User

├── _id (PK)

├── name

├── password (Hashed)

└── createdAt

Portfolio

├── _id (PK)

├── userId (FK → User._id)

├── stockSymbol (String)

├── avgPrice (Number)

├── createdAt (Date) └── createdAt

Stock

├── _id (PK)

├── name (String)

├── exchange (String)

├── industry (String)

├── price (Number)

├── marketCap (Number)

5. Data Flow Diagrams (DFD)

To develop the StockPulse platform efficiently, several software tools, libraries, and APIs have been utilized. The goal was to build a fast, responsive, and modular web application that integrates **real-time financial data** for a powerful stock analysis experience. Since the platform is built using **React.js**, it benefits from a component-based structure, making it easier to manage features like portfolio management, real-time stock tracking, and data visualization.
To support seamless navigation between different pages (Dashboard, Portfolio, etc.), **React Router DOM** has been used. For data integration, a **third-party financial API** is utilized, allowing the system to provide live, up-to-the-minute market data. Styling is handled through the **Tailwind CSS** framework, ensuring the user interface is consistent, modern, and responsive. The entire project is developed and tested using **Visual Studio Code**, with **Node.js** and **npm** used for managing dependencies.

Below is a detailed list of the software tools and technologies used:

- **Frontend Framework**: **React.js** – used to create reusable components and manage the dynamic behavior of the UI.
- **Routing Library**: **React Router DOM** – manages routing between multiple pages without full page reloads.
- **Styling**: **Tailwind CSS** – a utility-first CSS framework for clean, theme-based styling across the application.
- **JavaScript Runtime & Dependency Manager**: **Node.js and npm** – used to run the development server and install third-party packages.
- **Web Browser**: **Google Chrome / Firefox** – for testing, debugging, and rendering the front-end interface.
- **Data Integration**: **External Financial APIs** (e.g., Financial Modeling Prep) – used to fetch live stock prices, company profiles, and market data.
- **Version Control**: **Git and GitHub** – for version tracking, backup, and collaboration.

5.1 Level 1 DFD (Detailed Level)

This diagram shows the flow of data when a user interacts with the platform.

[User]

├── Login/Register → [Authentication Module] → MongoDB

├── Search for Stock → [Stock Data Module] → (MongoDB Cache & External API)

├── Add Stock to Portfolio → [Portfolio Controller] → MongoDB

├── View/Update Portfolio → [Portfolio Controller] → (MongoDB & External API for prices)

└── Fetch Market Overview → [Stock Data Module] → External API

Module Descriptions:

- **Authentication Module**: Handles user registration and secure JWT-based login.
- **Stock Data Module**: Fetches market data and specific stock information, utilizing an external financial API and a local MongoDB collection for caching.
- **Portfolio Controller**: Manages all portfolio-related actions, including adding, updating, deleting, and retrieving a user's stock holdings from the database.

6. Features Implemented

The **StockPulse** platform has been developed to provide a user-centric, data-driven stock analysis experience. The system is designed to help users efficiently identify, visualize, and understand emerging trends and unusual activity within specific stock market sectors or themes. It allows users to define and monitor niche market areas, leveraging AI to analyze simulated market data for statistically significant spikes or anomalies. Below is a detailed list of the features implemented in the application:

- **Secure User Authentication**

A full registration and login system is implemented using JWT for session management and bcrypt for password hashing, ensuring that user data is secure.

- **Niche Market/Keyword Selection**

Users have the ability to select or define specific keywords (e.g., 'rubber' sector) or simulated stock groups to monitor for trends and unusual activity.

- **AI-Powered Trend & Anomaly Detection**

The Python backend employs statistical models to automatically analyze market data. It flags keywords or stock metrics that show unusual spikes or significant deviations from historical norms.

- **Interactive Trend Dashboard**

A dashboard built with React.js visually presents the detected trends to the user. This includes displaying "Hot Buzz" keywords and "Anomaly Alerts" for stocks and sectors in a clear and interactive format.

- **Protected Routes**

Frontend routes are protected, preventing unauthenticated or unverified users from accessing sensitive pages such as the main analysis dashboard.

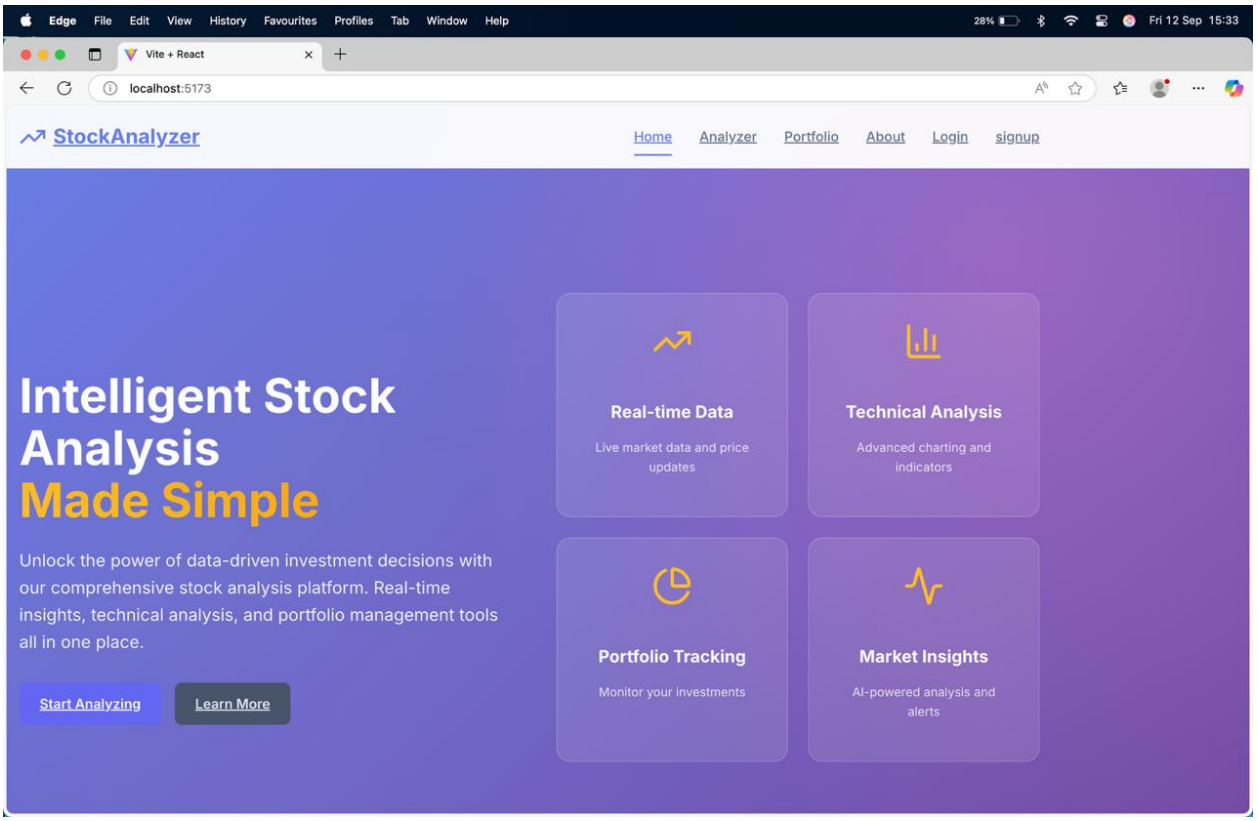
- **Responsive UI**

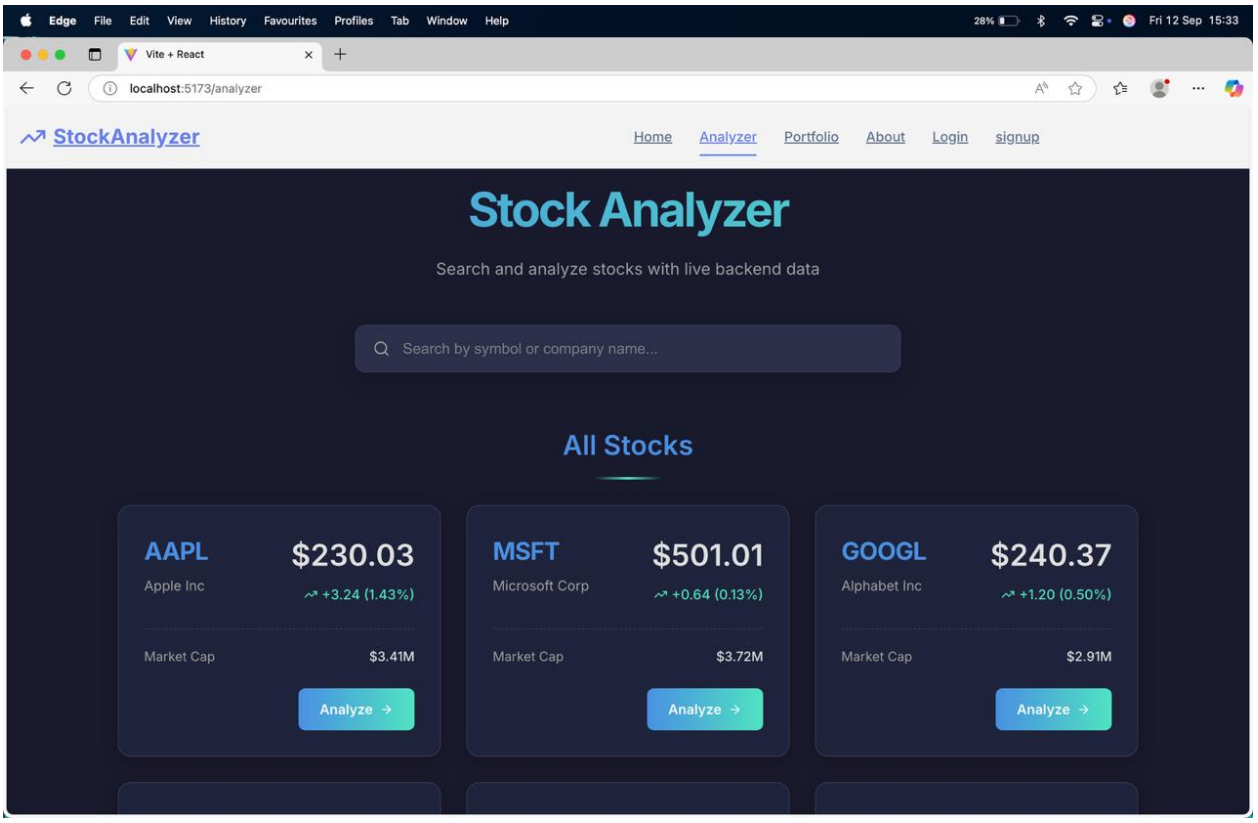
The user interface is fully responsive, providing an optimal viewing experience on desktops, tablets, and mobile devices. It features an animated, mobile-friendly navbar for seamless navigation.

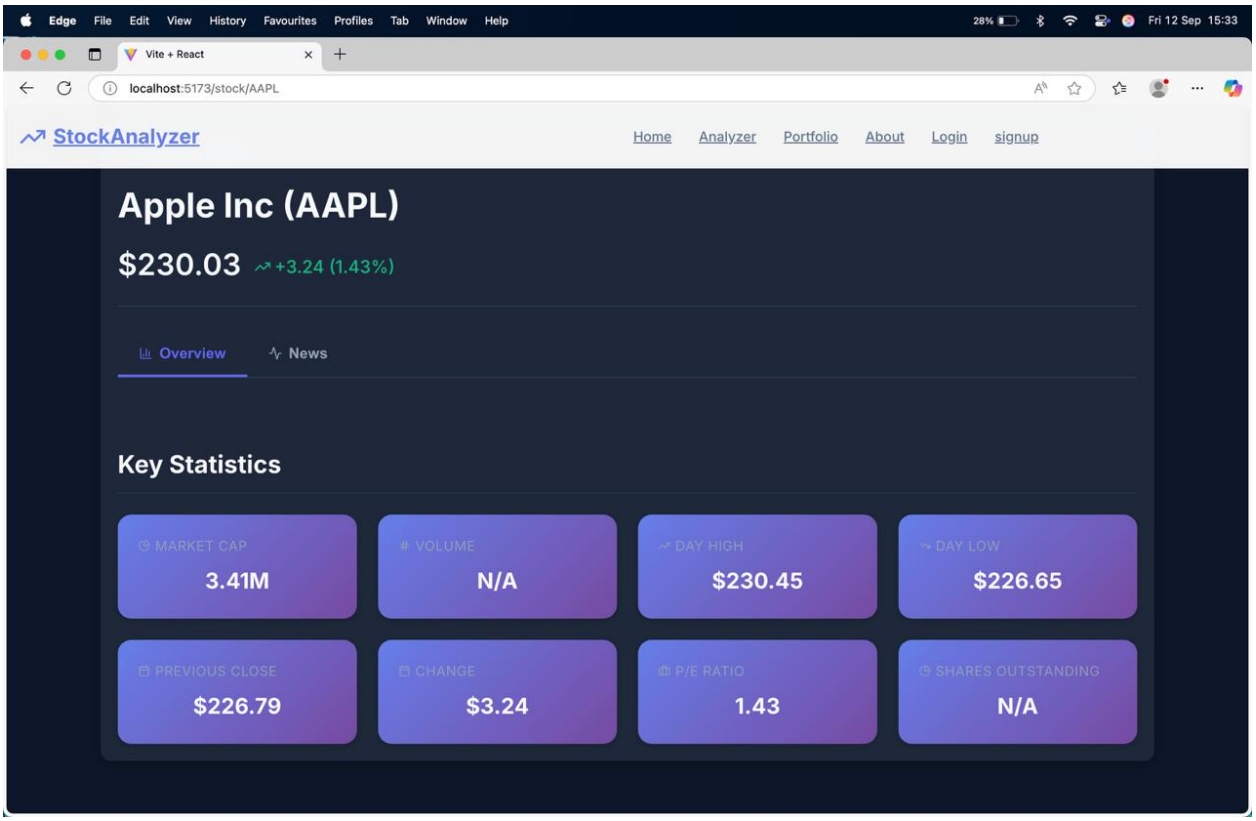
- **Simulated Market Data Generator**

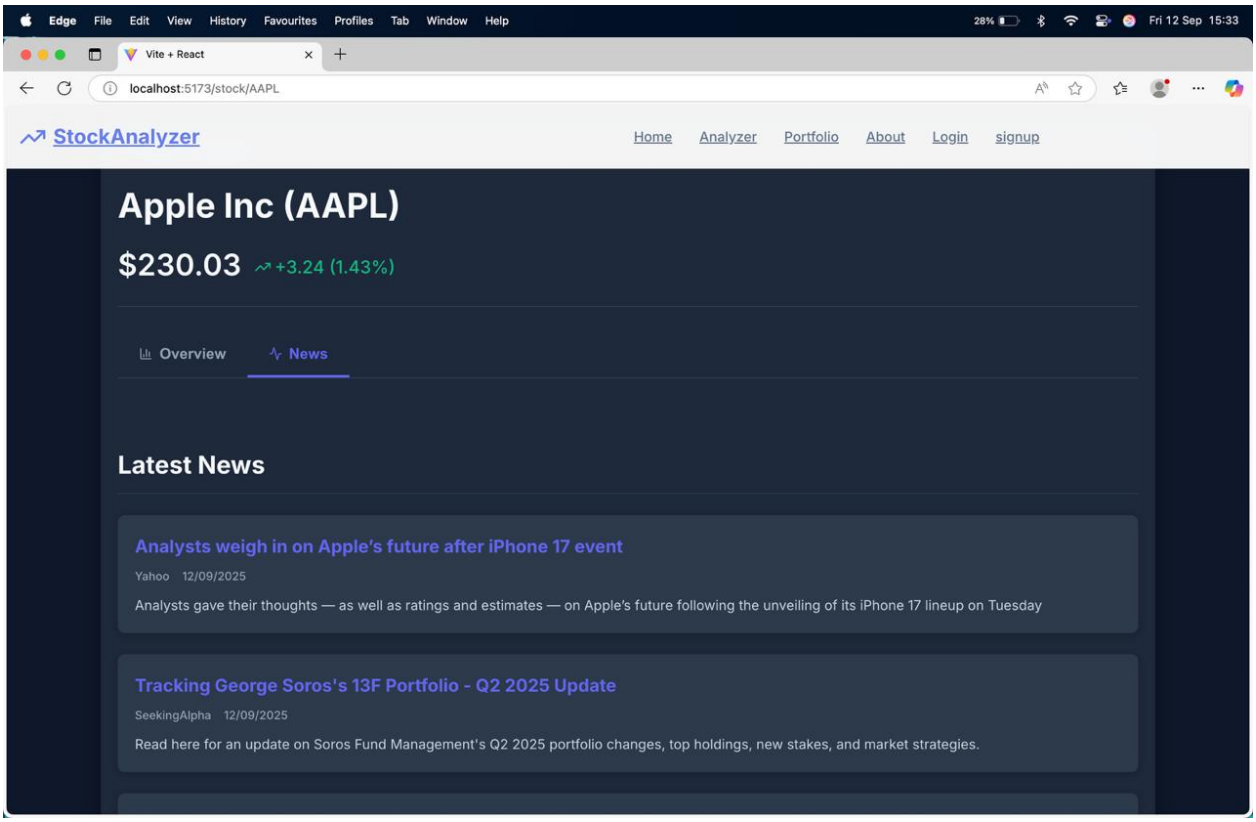
To ensure the platform can be demonstrated effectively, backend scripts create realistic, time-series data for keyword mentions and stock metrics. This data includes engineered "trends" for the AI to detect.

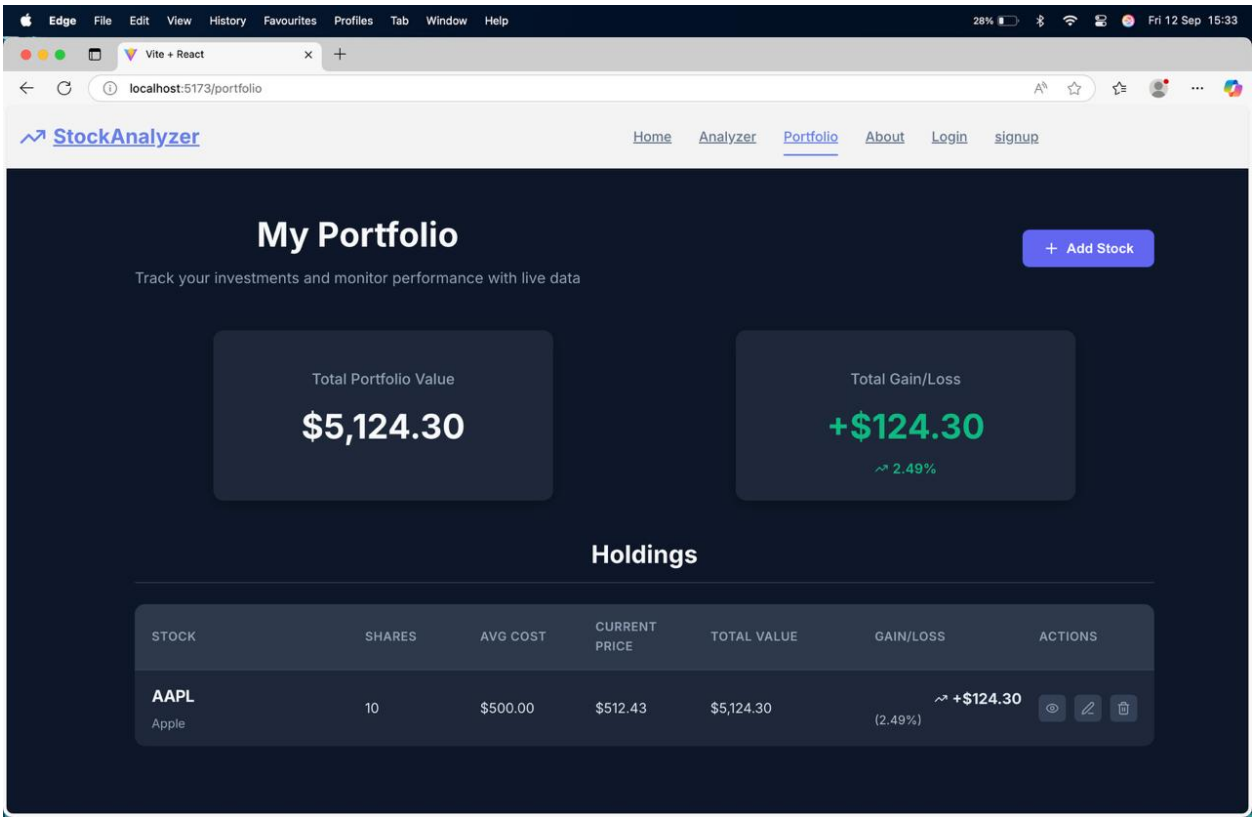
7. Screenshot

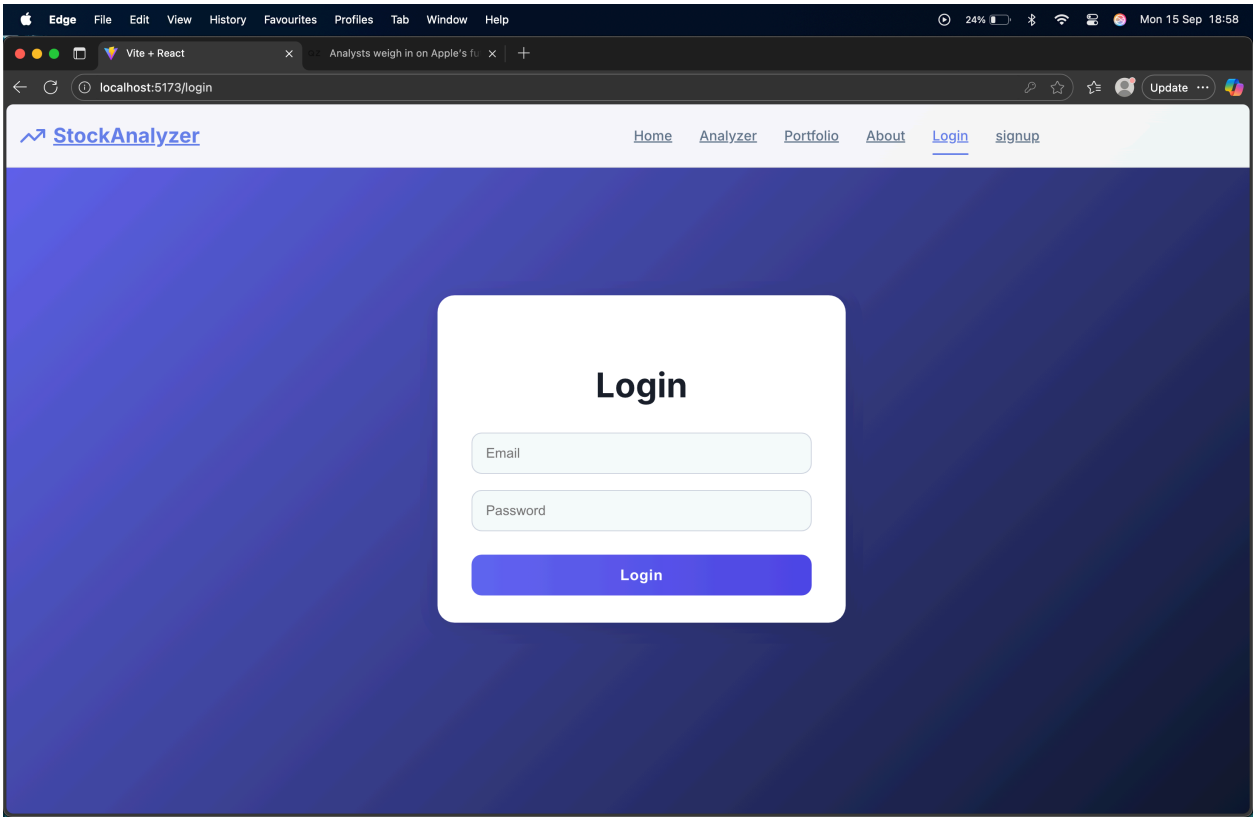


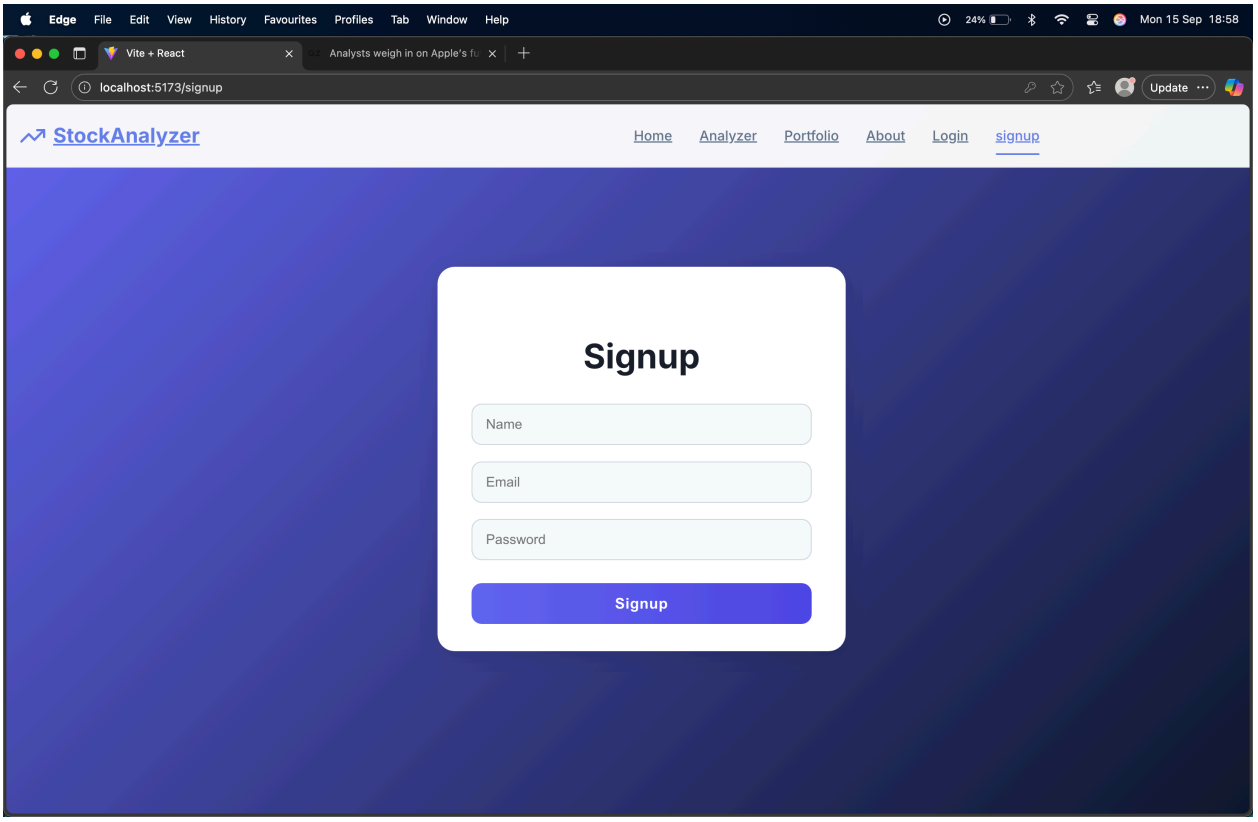



Edge File Edit View History Favourites Profiles Tab Window Help
28% Fri 12 Sep 15:33
Vite + React
localhost:5173/stock/AAPL
StockAnalyzer
Home Analyzer Portfolio About Login signup
Apple Inc (AAPL)
$230.03 +3.24 (1.43%)
Overview News
Key Statistics
MARKET CAP
3.41M
VOLUME
N/A
DAY HIGH
$230.45
DAY LOW
$226.65
PREVIOUS CLOSE
$226.79
CHANGE
$3.24
P/E RATIO
1.43
SHARES OUTSTANDING
N/A


Edge File Edit View History Favourites Profiles Tab Window Help
28% Fri 12 Sep 15:33
Vite + React
localhost:5173/stock/AAPL
StockAnalyzer
Home Analyzer Portfolio About Login signup
Apple Inc (AAPL)
$230.03 +3.24 (1.43%)
Overview News
Latest News
Analysts weigh in on Apple's future after iPhone 17 event
Yahoo 12/09/2025
Analysts gave their thoughts — as well as ratings and estimates — on Apple's future following the unveiling of its iPhone 17 lineup on Tuesday
Tracking George Soros's 13F Portfolio - Q2 2025 Update
SeekingAlpha 12/09/2025
Read here for an update on Soros Fund Management's Q2 2025 portfolio changes, top holdings, new stakes, and market strategies.


Edge File Edit View History Favourites Profiles Tab Window Help
28% Fri 12 Sep 15:33
Vite + React
localhost:5173/portfolio
StockAnalyzer
Home Analyzer Portfolio About Login signup
My Portfolio
Track your investments and monitor performance with live data
+ Add Stock
Total Portfolio Value
$5,124.30
Total Gain/Loss
+$124.30
2.49%
Holdings
STOCK SHARES AVG COST CURRENT PRICE TOTAL VALUE GAIN/LOSS ACTIONS
AAPL
Apple
10
$500.00
$512.43
$5,124.30
+$124.30
(2.49%)


Edge File Edit View History Favourites Profiles Tab Window Help
24% Mon 15 Sep 18:58
Vite + React
Analysts weigh in on Apple's fu
localhost:5173/login
Update
StockAnalyzer
Home Analyzer Portfolio About Login signup
Login
Email
Password
Login


Edge File Edit View History Favourites Profiles Tab Window Help
Mon 15 Sep 18:58
Vite + React
Analysts weigh in on Apple's fu
localhost:5173/signup
Update
StockAnalyzer
Home
Analyzer
Portfolio
About
Login
signup
Signup
Name
Email
Password
Signup

8. Challenges Faced

Building the StockPulse platform involved several technical challenges, particularly in managing real-time data and ensuring a secure, seamless user experience between the frontend and backend. The key challenges and their solutions include:

1. **Complex Frontend State Management**
    - **Challenge**: Managing user authentication status, security tokens, and dynamic profile data across the entire React application proved to be complex.
    - **Solution**: This was solved by implementing React's state management capabilities to create a centralized and predictable way of handling application-wide data.
2. **Asynchronous Error Handling**
    - **Challenge**: Ensuring that specific backend error messages (like "User already exists" or "Stock not found") were correctly caught and displayed on the frontend required careful implementation.
    - **Solution**: This was handled by using try...catch blocks within asynchronous fetch or axios calls in the React components, allowing the UI to show user-friendly error messages based on the server's response.
3. **Secure Authentication Flow**
    - **Challenge**: Creating a secure connection between the decoupled frontend and backend for user login and protected data access.
    - **Solution**: A secure system was built using JWT (JSON Web Tokens) for session management and creating protected routes on both the client and server to prevent unauthorized access to sensitive dashboard pages.
4. **Real-time Data Synchronization**
    - **Challenge**: Fetching user portfolio data from our database and simultaneously fetching live market prices from an external API, then merging them accurately for the user.
    - **Solution**: Asynchronous JavaScript patterns were used to handle multiple concurrent API calls. The data was then processed and combined on the backend before being sent to the client, ensuring the UI always displayed synchronized, up-to-date information.
5. **Responsive Data Visualization**
    - **Challenge**: Making sure that data-heavy components like portfolio tables and trend charts were both readable and functional on a wide range of devices, from desktops to mobile phones.
    - **Solution**: A mobile-first design approach was adopted using Tailwind CSS. Flexbox and Grid were utilized to create a fully responsive layout that adapts to different screen sizes, ensuring an optimal user experience everywhere.

9. Future Scope

While StockPulse currently provides a robust framework for market analysis, there are several key areas where the platform can be significantly enhanced to improve its usability, analytical power, and real-world utility for investors. The planned enhancements are not merely additive; they aim to transform the application from a powerful demonstration into a comprehensive and indispensable tool for any serious market participant. These updates focus on deepening the data integration, incorporating more advanced artificial intelligence, and refining the user experience with personalized, proactive features.

The project can be expanded by incorporating more

- **Advanced artificial intelligence and machine learning models**. One major step would be to implement
- **Natural Language Processing (NLP) for sentiment analysis** on financial news articles and social media mentions related to specific stocks or sectors. This would allow the platform to quantify market sentiment, providing users with a crucial layer of qualitative context to complement the quantitative price data. Beyond NLP, the platform could explore more sophisticated
- **time-series analysis and deep learning models** to identify complex, non-linear trend patterns that are invisible to simpler statistical methods, giving users a significant competitive edge.

Finally, a suite of **user-centric features** would be developed to make the platform more interactive and personalized. The addition of

- **customizable alert features**, delivered via email or push notifications, would transform StockPulse into a proactive monitoring tool that actively informs users of significant market events or detected trends without requiring them to be constantly logged in. Additionally, allowing users to create, save, and manage their own
- **custom watchlists of real stocks** would greatly enhance the tool's utility for monitoring potential investments and organizing research. With these updates, StockPulse has the potential to evolve from an analysis tool into a complete digital investment ecosystem, offering real-time support and hyper-personalized recommendations.

10.Conclusion

The development of the StockPulse application was fundamentally driven by the need to address the inefficiency and complexity inherent in modern stock market analysis. In a financial world saturated with data, the core challenge for investors is to identify meaningful signals amidst the noise—a phenomenon often leading to "paralysis by analysis." Retail investors, in particular, struggle to track niche sectors and emerging trends that are often overlooked by mainstream financial media. StockPulse was conceived as a practical, innovative solution to this problem, designed to

significantly improve the efficiency of early trend identification by providing a unified platform with clear, actionable visualizations. The project successfully navigates this challenge by integrating essential analysis services into a single, cohesive user experience, thereby reducing the friction and fragmentation users typically face.

From a technical standpoint, the project represents a successful implementation of a modern, full-stack web application. The architectural choice to decouple the React frontend from the Node.js backend API was deliberate and crucial. This design not only allows for independent development, testing, and scaling of the client and server but also creates a more flexible and future-proof system. The API-first approach means that the same backend logic could eventually serve other clients, such as a native mobile application, with minimal rework. This demonstrates a thoughtful and maintainable approach to software engineering, geared towards long-term viability.

Furthermore, the project showcases a

complete development lifecycle, from the initial architectural design and database schema to the implementation of complex features like AI-driven trend analysis. The result is not merely a proof-of-concept but a

scalable and production-ready platform. This holistic execution proves the viability of the concept and the robustness of the implementation, establishing a strong technical foundation upon which the extensive future enhancements can be confidently built.

The core value of StockPulse lies in its intelligent analysis capabilities. The platform

leverages AI to systematically analyze simulated market data, specifically focusing on identifying statistically significant keyword mention spikes and stock metric anomalies. This AI-driven engine acts as a sophisticated filtering mechanism, moving beyond simple lagging indicators to identify potential leading indicators from market buzz and performance metrics. Rather than simply presenting raw data, the application processes it to automatically flag unusual activity and potential trends, thereby empowering users to make more informed and timely decisions. This functionality, combined with its

intuitive data visualizations, makes complex market analysis more accessible and understandable for a wider range of users.

In a broader context, StockPulse stands as a testament to the growing convergence of web development, data science, and financial technology. It reflects a practical approach to solving real-world problems in the investment domain through the thoughtful application of technology. It contributes to the ongoing democratization of financial tools, bringing powerful analytical capabilities, once the exclusive domain of large institutions, to the individual investor. As the expectations of digital tools continue to grow, platforms must evolve beyond mere data presentation to offer genuine, intelligent insights. With its modular design and

powerful analytical engine, StockPulse serves as an excellent foundation for building even more sophisticated, user-aware applications that can navigate the complexities of the financial markets.

11.Appendix

GitHub Repo: https://github.com/jangramohit/StockPortfolio