Project Synopsis

Multi-Asset Financial Analytics Dashboard (Stocks, Crypto, Forex)

---

1. Problem Statement

Real-World Problem Definition

Financial markets are diverse, involving not just stocks but also **Cryptocurrency and Currency Exchange (Forex)**. Different market participants have vastly different needs: **Active Traders** require dense, real-time technical data and complex charts, whereas **Long-term Investors** (especially beginners) are often overwhelmed by such complexity and prefer simplified, fundamental insights. Current platforms rarely cater to both personas effectively within a single interface, forcing users to juggle multiple fragmented tools.

Industry Relevance

Fintech platforms today are moving towards **"Super Apps"** that consolidate various asset classes. The ability to toggle between a **"Pro/Trader" interface** and a **"Lite/Investor" interface** is a key differentiator in modern UX design (e.g., Binance Lite vs. Pro), allowing platforms to capture a wider demographic.

Target Users

- **Active Traders:** Require technical depth, volatility metrics, and advanced charting.
- **Retail Investors:** Require simplified growth views and fundamental data.
- **Crypto & Forex Enthusiasts:** Users tracking digital assets and currency pairs.

---

2. Requirement Analysis

Functional Requirements

- **Dual-Mode Interface:** A toggle in the user profile to switch between:
  - **Trader Mode:** High-frequency data, candlestick charts, MACD/RSI indicators, order depth.
  - **Investor Mode:** Simplified line charts, growth percentages, fundamental summaries, clean UI.
- **Multi-Asset Support:** Fetch and display data for Stocks, Cryptocurrencies, and Forex pairs.

- **Search & Watchlist:** Unified search bar supporting ticker symbols across all three asset classes.
- **Real-time Updates:** Live price ticking for high-volatility assets (Crypto).

Non-Functional Requirements

- **State Persistence:** Remember user's preferred mode (Trader vs. Investor) across sessions.
- **Low Latency:** Critical for Trader mode updates.
- **Modular UI Design:** Architecture that supports conditional rendering of complex vs. simple components.

---

3. System Architecture

High-Level Architecture

The system utilizes a **Client-Server architecture** with a specialized logic layer to handle the "Dual Mode" rendering. The backend aggregates data from multiple specialized endpoints (Stock APIs, Crypto APIs, Forex APIs) and normalizes it for the frontend.

[Include System Architecture Diagram Here]

Data / API Flow

1. **User Preference Check:** Frontend checks User Profile settings to load "Trader" or "Investor" layout.
2. **Unified API Request:** React sends a request (e.g., `GET /asset/BTC-USD`).
3. **Backend Aggregation:** Node.js identifies the asset type and routes the request to the specific provider (e.g., CoinGecko for Crypto, Alpha Vantage for Forex).
4. **Data Normalization:** Backend standardizes the response format so the frontend can render it regardless of asset source.
5. **Conditional Rendering:**
   - If **Trader Mode**: Renders candlestick charts and depth tables.
   - If **Investor Mode**: Renders simple area charts and summary cards.

---

4. Technology Stack

Languages

- JavaScript (ES6+)
- HTML5
- CSS3 (with CSS Grid/Flexbox for layout switching)

Frameworks

- **Frontend:** React.js (using Context API for Theme/Mode state)
- **Backend:** Node.js, Express.js

Libraries

- **Axios:** Handling requests to multiple distinct APIs.
- **Recharts / Lightweight-charts:** For rendering Candlestick (Trader) and Area (Investor) charts.
- **Mongoose:** Database modeling.

Tools Used

- MongoDB Compass
- Postman (for testing multi-asset endpoints)
- Git & GitHub
- VS Code

---

5. Implementation

Module-Wise Explanation

Frontend Modules (React): * **Profile Manager:** Switch toggle logic (Update User Preference). * **Trader View Container:** Components for Technical Analysis, Order Book visualizations, and Bollinger Bands. * **Investor View Container:** Components for Portfolio Growth, P/E Ratios, and "At a Glance" metrics. * **Universal Search:** Logic to distinguish between `$AAPL` (Stock), `#BTC` (Crypto), and `€EURUSD` (Forex).

Backend Modules (Node + Express): * **Multi-Asset Controller:** Logic to switch API providers based on asset type. * **User Preference Controller:** Stores and retrieves the "Mode" setting.

Core Logic / APIs

- **Mode-Based Data Fetching:** "Trader" mode requests fetch more granular historical data (1-min intervals), while "Investor" mode fetches daily/weekly aggregates to reduce load.
- **Normalization Layer:** Converts diverse API responses (Crypto vs. Forex) into a standard JSON structure for the UI.

Database Schema (MongoDB)

- **User Collection:** Fields for `username`, `password`, `preferredLayout` ("TRADER" | "INVESTOR").

- **Watchlist Collection:** Supports mixed arrays of assets (e.g., [AAPL, BTC, EUR/USD]).

---

6. Testing & Evaluation

API Testing

- Verified distinct endpoints for Stocks, Crypto, and Forex.
- Tested latency differences between heavy "Trader" data payloads and light "Investor" payloads.

UI Testing

- **Toggle Stress Test:** Verified seamless switching between Trader/Investor views without page reload.
- **Responsiveness:** Ensured complex Trader tables collapse neatly on mobile devices.

Edge Cases Handled

- **Market Hours:** Handling closed stock markets vs. 24/7 crypto markets.
- **API Rate Limits:** Caching frequent requests to prevent lockout from external providers.

---

7. Deployment (Bonus)

Deployment Type

- Local deployment (Node.js/React).

Bonus Scope

- **Cloud Deployment:** Render/Vercel.
- **Dockerization:** Containerizing the backend for consistent multi-environment behavior.

---

8. Outcome & Learnings

What Worked

- **Dynamic UX:** Successfully implemented a distinct visual feel for Traders (Dark mode default, dense data) vs. Investors (Clean, spacious layout).
- **Unified Backend:** Abstracting the complexity of three different asset APIs into one internal API.

Challenges Faced

- **Data Normalization:** Mapping Forex "Bid/Ask" prices to Stock "Last Traded Price" structures.
- **State Management:** Keeping the selected mode persistent across page refreshes.

Future Enhancements

- **AI Advisory:** "Investor" mode could offer AI-generated summaries of complex "Trader" charts.
- **Wallet Integration:** Allow actual crypto wallet connection for the Trader mode.

Final Note

This project demonstrates **advanced full-stack capabilities**, specifically in **User-Centric Architecture (Personalization)** and **API Aggregation**. By catering to two distinct user personas (Trader vs. Investor) and integrating multiple financial markets (Stocks, Crypto, Forex), it simulates a complex, real-world Fintech application.