QuantIQ

PoC (Proof of Work) Plan

PoC Project Plan: Simple Quantitative Finance App

This PoC will focus on the following basic features:

1. **User Authentication (Basic)**
   - A simple login/signup system.
2. **Fetching Financial Data**
   - Fetch real-time stock or cryptocurrency data from a free API (e.g., **Alpha Vantage**, **Yahoo Finance**, or **CoinGecko** for crypto).
3. **Portfolio Management (Basic)**
   - Users can input a few stocks/cryptos they own and track their current value.
4. **Basic Visualization**
   - Display stock prices in a simple line graph over time using a library like **Matplotlib** or **Plotly**.

Step-by-Step Breakdown

1. Set Up the Project

- **Backend**: Use **Django** (for simplicity, and it will help you get familiar with Django REST framework if you plan to scale).
- **Frontend**: Keep it simple, using **HTML, CSS, and JavaScript** (with React if you're comfortable).

- **Database**: Use a simple SQLite (default with Django) or PostgreSQL for later scalability.

2. User Authentication

- Implement **Django’s built-in authentication system**.
- Allow users to sign up, log in, and view a dashboard with their portfolio.
- Use Django templates or React for the frontend to display this.

3. Fetching Financial Data

- Use an API like **Alpha Vantage** or **CoinGecko** to get real-time stock prices.
- Fetch the stock price data using **Python’s `requests` module** (for backend) and show it on the frontend.

4. Portfolio Management

- Allow users to add stocks or crypto they own (with quantity and purchase price).
- Use the stock data fetched in step 3 to display current market value and the user’s portfolio's performance (gain/loss).
- For example:
  - User owns 10 shares of **AAPL** at $100 each.
  - Real-time AAPL price is fetched.

    - The system will show the portfolio value, current price, and total gain/loss.

5. Basic Visualization

- Plot a simple line graph showing the historical price of a stock over time using **Matplotlib** or **Plotly**.
- For **frontend** (React):
- Use **Chart.js** or **Plotly.js** to visualize the stock price trends in an interactive chart.

Minimal PoC Structure

- **Frontend (React)**:
    - Basic layout (login, portfolio dashboard).
    - Stock price display and graph rendering.
- **Backend (Django)**:
    - Basic user authentication and portfolio data management.
    - API to fetch real-time stock prices and portfolio data.
- **Database**:
    - Users' portfolios (stocks/cryptos they own).
    - Simple SQLite database to store portfolio data.

Development Timeline (2-3 Weeks)

1. **Week 1:**
   - Set up the Django backend with user authentication.
   - Implement fetching of stock/crypto data using an API.
   - Store the user’s portfolio in the database.
2. **Week 2:**
   - Display the portfolio and real-time stock data on the frontend.
   - Add basic stock data visualization (chart).
   - Test the core functionality (fetch data, display it, manage portfolio).
3. **Week 3:**
   - Refine the UI and make it user-friendly.
   - Perform some additional tests.
   - Add basic error handling and validation.

Conclusion

This **Proof of Concept** project is manageable and lets you focus on the core elements of your larger vision without overwhelming yourself. You can then **gradually scale** it by adding more complex features, like machine learning predictions, algorithmic trading, or financial modeling.

Starting with this simplified version will help you get comfortable with:

- Django’s authentication system.
- Handling APIs and real-time data.
- Building a simple portfolio tracker.
- Implementing data visualization.