# Architectural designs

### ****1. Conceptual Architecture****

The conceptual architecture provides a high-level view of the system's key functional modules and their interactions.

#### Key Modules:

* **User Interface (UI)**:
  + Displays stock information, user portfolio, and market trends.
  + Offers interactive features like search, filtering, and trading.
* **Stock Market Data Aggregator**:
  + Fetches real-time stock prices and historical data from APIs (e.g., Yahoo Finance, Alpha Vantage).
  + Applies the **pipe-and-filter** pattern to process raw data (e.g., currency conversion, price normalization).
* **Trading Engine**:
  + Manages buy/sell orders.
  + Handles order matching and execution.
* **Portfolio Manager**:
  + Tracks user assets, holdings, and performance.
  + Sends updates when portfolio value changes.
* **Notifications and Alerts**:
  + Pushes alerts about stock price changes, news, or portfolio milestones.
* **Analytics Engine**:
  + Provides insights like trends, stock predictions, and market analytics.
* **Authentication and User Management**:
  + Secures the system with login, registration, and account management.

### ****2. Execution Architecture****

The execution architecture focuses on the runtime interactions between these components.

#### Design:

* **Pipe-and-Filter Architecture**: Used in the data pipeline for stock market data processing:
  1. **Source Input (Pipe)**: Connects to APIs or databases to fetch raw data.
  2. **Filters**:
     + Data Parsing: Converts raw data into usable formats (JSON, XML, etc.).
     + Data Normalization: Ensures uniform currency, units, and time zones.
     + Aggregation: Combines data from multiple sources for accuracy.
  3. **Output (Pipe)**: Sends processed data to the analytics engine or UI.
* **Hybrid Client-Server**:
  1. **Client**: A web or mobile app that retrieves data and displays it.
  2. **Server**: Processes data, executes trades, and communicates with third-party APIs.
  3. **GraphQL/REST APIs**: Provide seamless communication between the client and server.

#### Runtime Execution:

* Requests flow from the **UI** to the backend (via APIs).
* The **Trading Engine** interacts with brokers or exchanges for execution.
* Processed data from the **Analytics Engine** feeds back into the **UI** and **Portfolio Manager**.

### ****3. Implementation Architecture****

This view focuses on the technologies, frameworks, and deployment structure.

#### Technology Stack:

* **Frontend (UI)**:
  + Frameworks: React, Vue.js, or Angular.
  + Hybrid Platforms: Flutter or React Native (for cross-platform mobile apps).
* **Backend**:
  + Programming Language: Python (Django, Flask) or Node.js.
  + Frameworks for Pipe-and-Filter: Apache Kafka (for data streams), or custom pipelines using Python.
* **Data Storage**:
  + Relational Databases: PostgreSQL or MySQL (for user data and transactions).
  + NoSQL Databases: MongoDB or Redis (for caching and real-time data).
* **APIs**:
  + Stock Data APIs: Yahoo Finance, Alpha Vantage, or Quandl.
  + Communication APIs: GraphQL for efficient querying or REST for simplicity.
* **Analytics**:
  + Libraries: Pandas, NumPy.
* **Deployment**:
  + Cloud Services: AWS, Azure, or Google Cloud for scalability.
  + CI/CD: Jenkins, GitHub Actions for continuous deployment.

#### Sub-Architectures:

* **Pipe-and-Filter**: Used for data processing and analytics.
* **Microservices**: Each core function (trading, data aggregation, user management) runs as an independent service.
* **Event-Driven Architecture**: For notifications, triggered by events like price changes.

### Final Diagram Views:

#### ****Conceptual View****:

css

Copy code

[User Interface] <---> [Backend API]

/ | \

[Trading Engine] [Portfolio Manager] [Analytics Engine]

|

[Stock Market Data Aggregator] <--> [External APIs]

#### ****Implementation View****:

1. **Frontend**: React/Flutter hosted on AWS Amplify.
2. **Backend**: Python Flask/Node.js with Kubernetes for scalability.
3. **Data Processing**: Kafka pipelines for stock market data flow.