InsightStox — Object Design

This document defines the **Object-Oriented Design (OOD)** for the InsightStox system. It focuses on: - Identifying classes and objects - Their attributes and responsibilities - Relationships between objects - Interaction models - Internal object-level reasoning of the system.

1. Object Model Overview

The system is built around several core domains: - **User domain** (accounts, authentication) - **Portfolio domain** (collections of holdings) - **Transaction domain** (buy/sell operations) - **Market Data domain** (quotes, historical OHLC) - **Analytics domain** (allocation, P&L, valuation)

Each domain is represented through classes, objects, and supporting data structures.

2. Object Identification

Below are the primary objects in the system, organized by domain.

2.1 User Domain Objects

Class: User

Attributes: - userId - name - email - passwordHash

Responsibilities: - Authenticate and manage sessions - Hold references to portfolios.

Class: Queries

Attributes: - userId - Queries

Responsibilities: - Hold User Queries for further reviews.

Class: Suggestions

Attributes: - userId - Suggestions

Responsibilities: - Holds all the Suggestions given by the user.

2.2 Portfolio Domain Objects

Class: Holding

Attributes: - symbol - quantity - averagePrice - realizedPnL

Responsibilities: - Represent current investment position - Handle adjustments from new transactions - Provide valuation methods

2.3 Transaction Domain Objects

Class: Transaction

Attributes: -UserId - symbol - type (BUY / SELL) - quantity - price - fees - executedAt

Responsibilities: - Represent atomic trading action - Modify holdings - Feed into analytics calculations

2.4 Stocks Domain Objects

Class: Stocks

Attributes: -Symbol -assetType -Currency -name -Sector -Country

Responsibilities: - Show all stock related details.

2.5 Analytics Domain Objects

Class: PortfolioValuation

Attributes: - -TotalValuation -Created At

Responsibilities: - Combine holdings and quotes into a final Valuation

2.6 Watchlist Objects

Class: Watchlist

Attributes: - -UserId -Stocks

Responsibilities: - Stores User Watchlist

2.6 Session Objects

Class: Session

Attributes: - -UserId -token -expiry

Responsibilities: - Stores User Session details.

3. Class Responsibilities & Collaborations

This section explains the behavior and collaboration between objects.

3.1 User → Stocks

- User owns multiple Stocks.
- User class retrieves and manages portfolio list.

3.2 User → Transaction

- Users make multiple transactions.
- Each transaction contributes to portfolio holdings.

3.3 User → Suggestions

- User interacts with Suggestions to give his/her Suggestions for the website.
- Each Suggestion contributes for the system development.

3.4 User → Queries

- User interacts with Queries to give his/her Queries for the website.
- Each Query contributes the system for solving user Problems.

3.5 Transaction → Holdings

- Each Transaction interacts with Holdings to update user holding.

3.6 User → Holdings

- The user interacts with Holdings to get the current holding and calculate required data.

3.6 User → Watchlist

- The user interacts with Holdings to add or remove a stock from his/her Watchlist.

3.7 User → Session

- The user interacts adds and remove his/her sessions

4. Object Behaviors

4.1 User Behaviors

- login()
- register()
- updateProfile()
- fetchPortfolios()

4.2 Transaction Behaviors

- addTransaction()

4.3 Holding Behaviors

- updateHoldings()
- addHoldings()
- getHoldings()

4.4 Stock Behaviors

- addStock()

4.5 Queries Behaviors

- addQuery()

4.6 Suggestions Behaviors

- addSuggestion()

4.7 Watchlist Behaviors

- addStock()
- removeStock()
- getWatchlist()

4.8 Session Behaviors

- addSession()

- removeSession()
- getSessions()

4.9 PotfolioValuation Behaviors

- addValuation()
- getValuation()

5. Object Relationships

All relationships described in pure text form.

User ↔ Holdings (One-to-Many)

- One User has many Holdings.
- Each Holding belongs to exactly one User.

User ↔ Transaction (One-to-Many)

- A User consists of multiple Transactions.
- Each transaction is done by only 1 user.

User ↔ Watchlist (One-to-Many)

- A User has multiple stocks in the watchlist.
- A watchlist belongs to only 1 user.

User ↔ Queries (One-to-Many)

- A User can give multiple queries.
- A Query can be given by only 1 user.

User ↔ Suggestions (One-to-Many)

- A User can give multiple suggestions.
- A Suggestion can be given by only 1 user.

User ↔ Sessions (One-to-max Five)

- A User can have multiple sessions.
- A Session can be given of only 1 user.

User ↔ PortfolioValuation (One-to-Many)

- A user can have many portfolio valuations based on days.
- A portfolio valuation belongs to only one user.

6. Object Interaction Scenarios

6.1 Scenario: Adding a Transaction

1. The user selects a Stock.
2. The user sends a request to add a BUY or SELL transaction.
3. Backend creates a Transaction object.
4. Holdings loads existing holdings.
5. Transaction updates the respective Holding.
6. Updated Holding recalculates average price & PnL.
7. PortfolioSummary recalculates total value.

6.2 Scenario: Viewing Portfolio Dashboard

1. The user requests portfolio details.
2. Portfolio retrieves Holdings.
3. Holdings fetch latest Quote.
4. PortfolioSummary computes valuation.
5. AllocationBreakdown computes sector weights.
6. API returns complete dashboard summary.

6.3 Scenario: Viewing Watchlist

1. User requests to view the watchlist.
2. The system retrieves the watchlist and fetches real time data.
3. API returns complete watchlist data.

6.4 Scenario: Adding Stock to Watchlist

1. The user selects a stock to add to watchlist.
2. System verify if stock is already present in watchlist or not.
3. Add stock to the watchlist.
4. API return success message.

6.5 Scenario: Adding Stock to Watchlist

1. The user selects a stock to remove from the watchlist.
2. System verify if stock is present in watchlist or not.
3. Remove stock from the watchlist.
4. API return success message.