AI-Powered Stock Analysis and Trading Platform

Yiwei Li, Yanbo Tong, Haoyang Guo

December 3, 2025

Application Overview

- **Goal**: Revolutionize investment strategies using advanced AI algorithms and data analysis.
- **Core Function**: Provides users with tailored stock recommendations, real-time market insights, **various trading strategies, synchronous detailed portfolios, and mock of authentic stock trading**.
- **User Experience**: Features a user-friendly interface to help individuals make informed decisions.
- **Benefit**: Promotes efficient trading practices for novice and experienced investors alike.
- **Key Enabler**: Combines Artificial Intelligence with robust data sources.

Outline

- System Architecture
- Primary Functionality
- Technical Details
- Contribution

System Architecture - Distributed Design

Front-end

- **Original Page**
  - Login & Registration
  - Simple Stock
  - Multiple Stocks
  - User Management (Adding Balance, Password Change)
  - AI-driven Chart Analysis & Investment Advice
- **New Functionalities**
  - Virtual Portfolio for Mock Trading
    - Real-time Stocks & Prices
    - Trading Strategies (Coupled and Basket)
    - Brief Overview of Holding
    - Winner & Loser

Back-end

- **Server**
  - Express / Node Server
  - Route Management
- **API Design**
  - RESTFul API
    - **New API**
      - virtualPortfolio.js
  - Data Validation
  - Error Handling

System Architecture

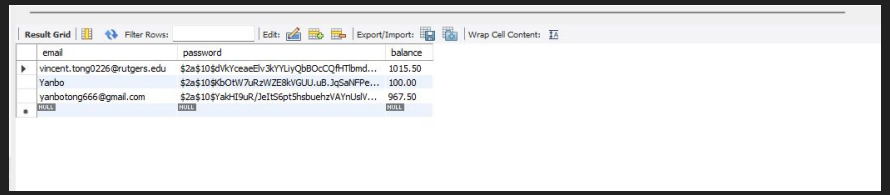
System Architecture - Database Description

User table

Column name	Type	Description
email	VARCHAR(255)	PRIMARY KEY, Store users' email
password	VARCHAR(255) NOT NULL	Store user password
balance	DECIMAL(10, 2) DEFAULT 0.00	Store user balance in profile

```sql
CREATE TABLE Users (
    email VARCHAR(255) PRIMARY KEY,
    password VARCHAR(255) NOT NULL,
    balance DECIMAL(10, 2) DEFAULT 0.00
);
```

```sql
UPDATE Users
SET virtual_balance = 100000.00
WHERE email = 'vincent.tong0226@rutgers.edu';
```




```sql
ALTER TABLE Users
ADD COLUMN virtual_balance DECIMAL(10, 2) DEFAULT 100000.00;
```

System Architecture - Database Description

Stock transaction table

Column name	Type	Description
timestamp	TIMESTAMP DEFAULT CURRENT_TIMESTAMP PRIMARY KEY,	To store the timestamp of every transactions. Avoid primary key conflict
email	VARCHAR(255) NOT NULL,	Store the user's email of the transaction
stock_name	VARCHAR(255)	
number	INT	Store the number of stock user bought in this transaction
current_price	DECIMAL(10, 2) NOT NULL,	
Is_sold	BOOLEAN DEFAULT FALSE	It will change to 0 if the user sold all of the stock
bought__price	DECIMAL(10, 2) NOT NULL	It will get the price of the stock when it is bought

```
CREATE TABLE stock_transactions (
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP PRIMARY KEY,

    email VARCHAR(255) NOT NULL,
    `stock_name` VARCHAR(255),
    number INT,
    current_price DECIMAL(10, 2) NOT NULL,

    `is_sold` BOOLEAN DEFAULT FALSE,

    FOREIGN KEY (email) REFERENCES Users(email)
);
```

```
ALTER TABLE stock_transactions
ADD COLUMN is_virtual BOOLEAN DEFAULT TRUE;
```

```
ALTER TABLE stock_transactions DROP PRIMARY KEY;

ALTER TABLE stock_transactions
ADD COLUMN id INT NOT NULL AUTO_INCREMENT PRIMARY KEY FIRST;
```

System Architecture - API interface

Function	API	HTTP Method
User Login	/api/login	POST
User Registration	/api/register	POST
Password Change	/api/change-password	PUT
Deposit Money	/api/deposit	POST
Sell Stock	/api/sellStock	POST
See Holding Stocks	/api/buyAndViewStocks	GET
Get Stock Trend	/api/stock-trend	GET

API Interface (Continued)

Function	API	HTTP Method
Chart Analysis	/api/chartAnalysis	GET
AI Portfolio Prediction	/api/aiPortfolioPrediction	GET/POST
AI Personal Advice	/api/aiPersonalAdvice	GET/POST
AI Advice on Single Stock	/api/singleAdvice	GET
AI Advice on Multiple Stocks	/api/multipleAdvice	GET
Stock Q&A	/api/stockQA	POST

New API

- virtualPortfolio.js
  - Achieve new functionalities in the virtualPortfolio.html, including the instant trading, portfolio summary, batch trading, and basket orders.

Function	API	HTTP Method
Single Instant Buy (Virtual)	api/virtual-buy	POST
Single Sell (Virtual)	api/virtual-sell	POST
Get Portfolio Summary/Holdings	api/virtual-portfolio	GET
Get Single Stock Price Timeline	api./price-timeline/:symbol	GET

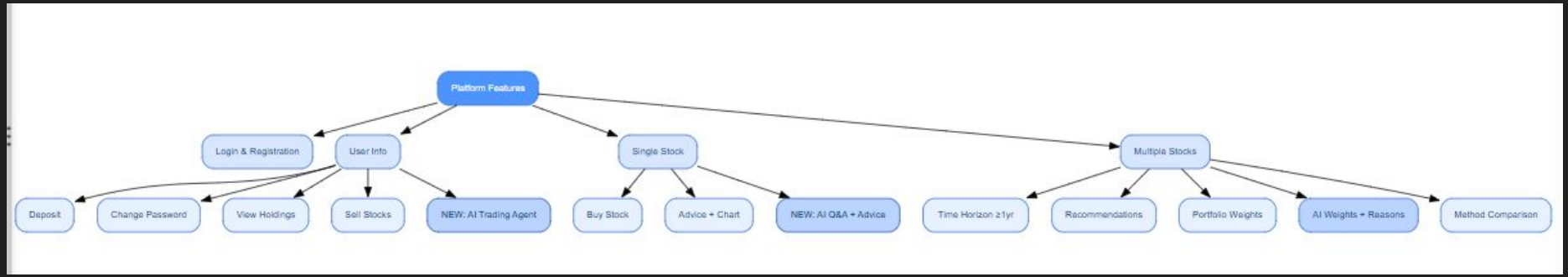
New API (Continued)

Function	API	HTTP Method
Get Total Portfolio Value Timeline	api//portfolio-timeline	GET
Backtesting	api/backtest	POST
Batch Buy (Multi-Stock)	api/batch-buy	POST
Batch Sell (Multi-Stock)	api/batch-sell	POST
Get Aggregated Basket Orders	api/basket-orders	GET
Join Basket Order	api/basket-join	POST
Execute Basket Order	api/basket-execute	POST

Primary Functionality

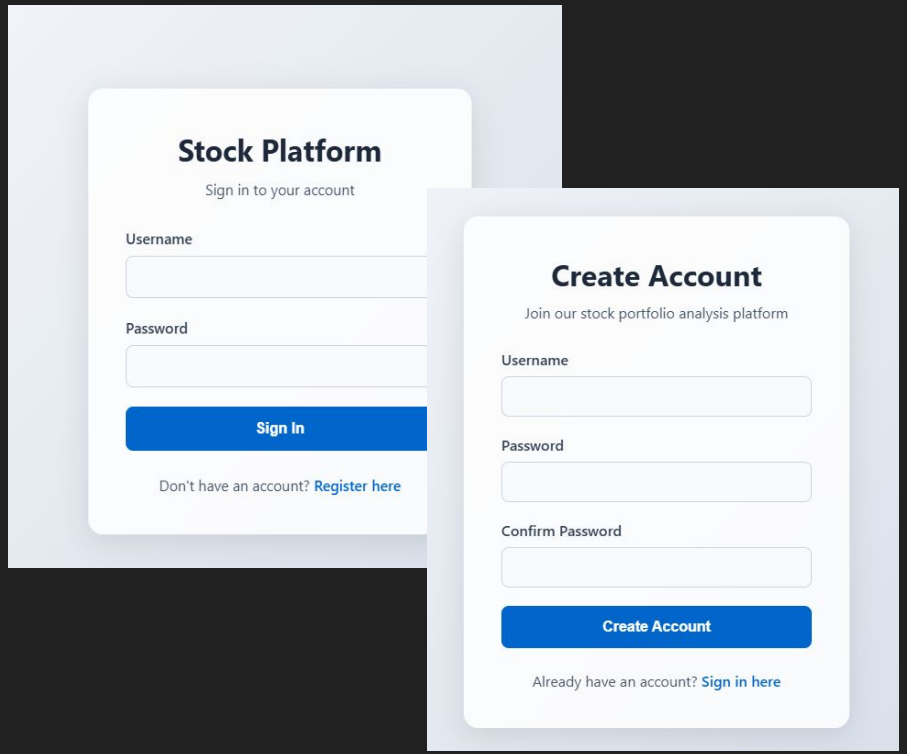
Single stock prediction
Multiple stock prediction
User Information
AI agent for stock forecasting

Primary Functionality - Module Overview



Key Functionality - Login logistics

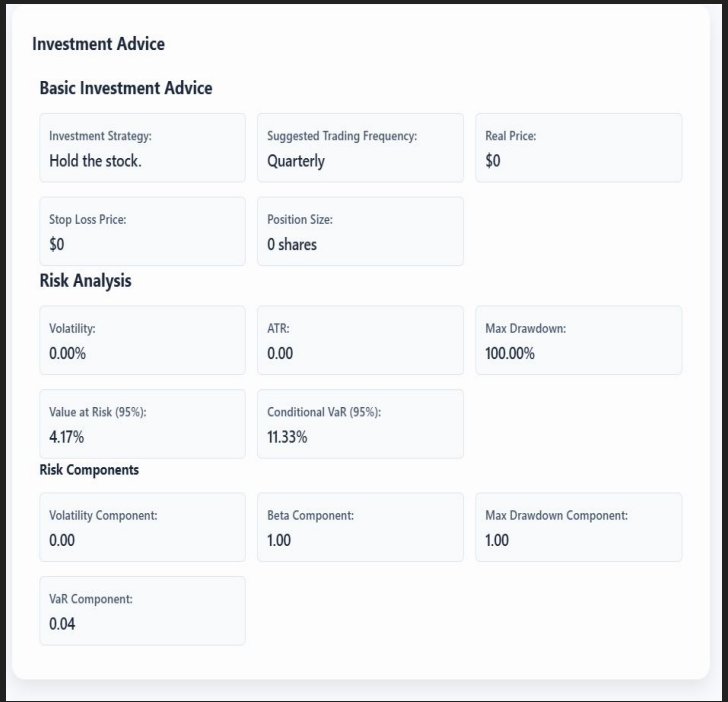
- **Register: first time into the stock analysis system**
  - Username: user defines
  - Password: user defines
- **Login:after registration, register information are stored in the database**
  - Username: user input
  - Password: user input
- **Error handling:**
  - If user inputs a wrong password or nonexistent username. It will give alert



Single Stock Module - Single Stock Analysis

- **User input**
  - **Stock ticker:** User chooses the ticker symbol
  - **Initial capital:** User inputs the amount and it can get suggestions about the stock ticker
  - **Real price:** User can view the real-time-price of the stock ticker

- **Trend chart:** User can view the historical price of the stock ticker and can click on specific date to check the closing price



Single Stock Module - Single Stock Prediction Logic

SMA(10) vs. LMA(horizon)	Investment Horizon (Risk Factor)	Stop-Loss Percentage (Control)	Suggested Action
SMA(10) > LMA(horizon)	Quarterly (Lower Risk)	Smaller	BUY
SMA(10) > LMA(horizon)	Annually (Higher Risk)	Larger	BUY
SMA(10) = LMA(horizon)	Any	Proportional to Horizon	SELL
SMA(10) < LMA(horizon)	Any	Proportional to Horizon	HOLD

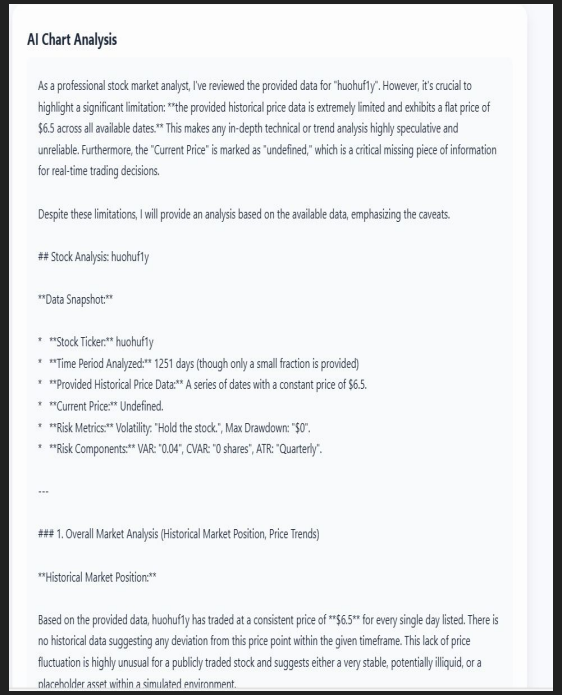
Single Stock Module - Single Stock AI Analysis

Analysis Metrics

- Analysis metrics
- Strategy
- Frequency
- currentPrice
- maxDrawdown
- Var
- Cvar
- atr
- riskMetrics
- Volatility

Analysis Content

- Overall Market Analysis
- Risk Assessment
- Technical Analysis
- Investment Strategy
- Final Recommendation



AI Chart Analysis

As a professional stock market analyst, I've reviewed the provided data for "huohuf1y". However, it's crucial to highlight a significant limitation: **the provided historical price data is extremely limited and exhibits a flat price of $6.5 across all available dates.** This makes any in-depth technical or trend analysis highly speculative and unreliable. Furthermore, the "Current Price" is marked as "undefined," which is a critical missing piece of information for real-time trading decisions.

Despite these limitations, I will provide an analysis based on the available data, emphasizing the caveats.

Stock Analysis: huohuf1y

Data Snapshot:

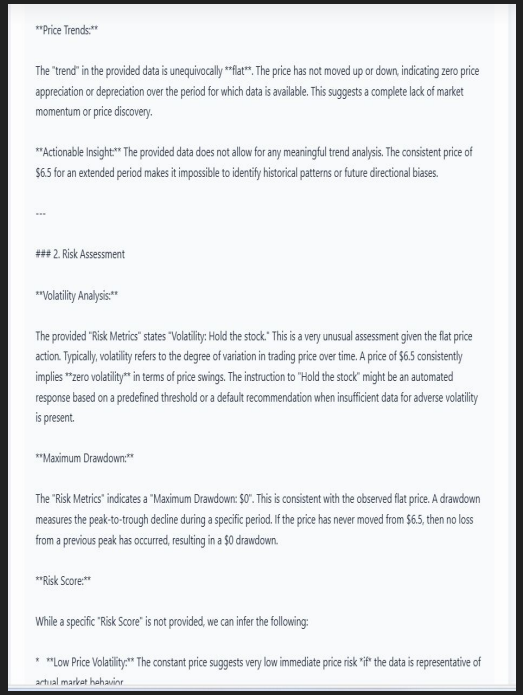
* **Stock Ticker:** huohuf1y
* **Time Period Analyzed:** 1251 days (though only a small fraction is provided)
* **Provided Historical Price Data:** A series of dates with a constant price of $6.5.
* **Current Price:** Undefined.
* **Risk Metrics:** Volatility: "Hold the stock.", Max Drawdown: "$0".
* **Risk Components:** VAR: "0.04", CVAR: "0 shares", ATR: "Quarterly".

---

1. Overall Market Analysis (Historical Market Position, Price Trends)

Historical Market Position:

Based on the provided data, huohuf1y has traded at a consistent price of **$6.5** for every single day listed. There is no historical data suggesting any deviation from this price point within the given timeframe. This lack of price fluctuation is highly unusual for a publicly traded stock and suggests either a very stable, potentially illiquid, or a placeholder asset within a simulated environment.



Price Trends:

The "trend" in the provided data is unequivocally **flat**. The price has not moved up or down, indicating zero price appreciation or depreciation over the period for which data is available. This suggests a complete lack of market momentum or price discovery.

Actionable Insight: The provided data does not allow for any meaningful trend analysis. The consistent price of $6.5 for an extended period makes it impossible to identify historical patterns or future directional biases.

---

2. Risk Assessment

Volatility Analysis:

The provided "Risk Metrics" states "Volatility: Hold the stock." This is a very unusual assessment given the flat price action. Typically, volatility refers to the degree of variation in trading price over time. A price of $6.5 consistently implies **zero volatility** in terms of price swings. The instruction to "Hold the stock" might be an automated response based on a predefined threshold or a default recommendation when insufficient data for adverse volatility is present.

Maximum Drawdown:

The "Risk Metrics" indicates a "Maximum Drawdown: $0". This is consistent with the observed flat price. A drawdown measures the peak-to-trough decline during a specific period. If the price has never moved from $6.5, then no loss from a previous peak has occurred, resulting in a $0 drawdown.

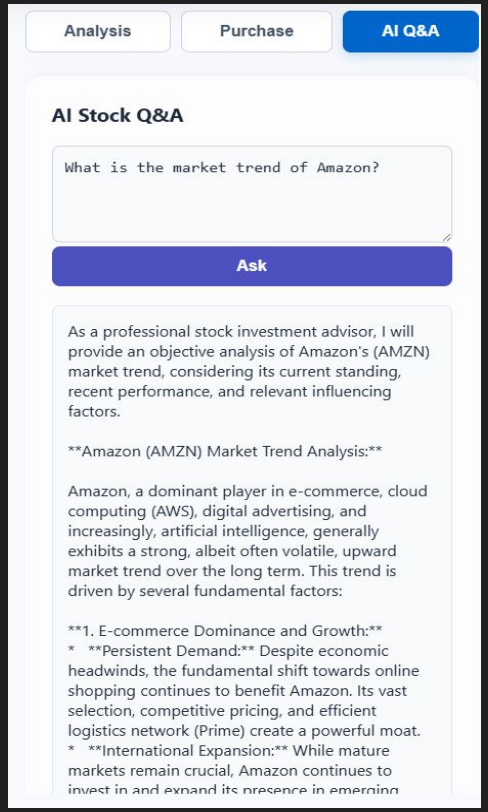
Risk Score:

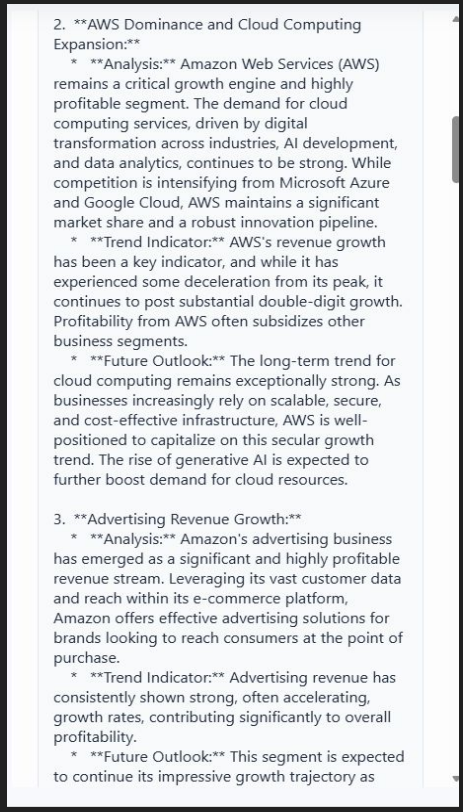
While a specific "Risk Score" is not provided, we can infer the following:

* **Low Price Volatility:** The constant price suggests very low immediate price risk *if* the data is representative of actual market behavior.

Single Stock Module - AI for general questions

- Check API Key
- Initialize Google Gemini with Gemini API
- Define /Chat API Endpoint
- Receive User Message
- Send Message to Gemini API
- Extract AI Response
- Print AI Response with Typing Effect
- Send AI Response to Client
- Handle Errors Gracefully





Single Stock Module - Code for AI Q&A

```
// CHANGED: Import Google Generative AI
const { GoogleGenerativeAI } = require("@google/generative-ai");

// CHANGED: Initialize Gemini AI Client
const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
const model = genAI.getGenerativeModel({ model: "gemini-2.5-flash-lite" });

router.post('/ask', async (req, res) => {
    try {
        const { question } = req.body;
        if (!question) {
            return res.status(400).json({ error: "Question is required" });
        }

        const prompt = `
You are a professional stock investment advisor. Please answer the following question in English, ensuring your answer is professional, accurate and in English.

Question: ${question}

Please consider the following aspects:
1. If the question is about a specific stock, provide a detailed analysis.
2. If the question is about investment strategy, provide concrete advice.
3. If the question is about market trends, provide an objective analysis.
4. If the question is about risk control, provide practical suggestions.

Make sure your answer is:
1. Professional and accurate
2. Easy to understand
3. Specific and actionable
4. Includes necessary risk warnings

Please answer ONLY in English, no matter what language the question is.
`;

        // CHANGED: Call Gemini API
        const result = await model.generateContent(prompt);
        const response = await result.response;
        const answer = response.text();

        res.json({
            answer,
            timestamp: new Date().toISOString()
        });

    } catch (error) {
        console.error("Error generating AI answer:", error);
        res.status(500).json({
            error: "Failed to generate answer",
            details: error.message
        });
    }
});

module.exports = router;
```

Single Stock Module - Code for AI Personal & Investment Advice

```
require("dotenv").config();
const express = require("express");
const router = express.Router();
// CHANGED: Import Google Generative AI
const { GoogleGenerativeAI } = require("@google/generative-ai");
const { pool } = require('../utils/db');

// CHANGED: Initialize Gemini AI Client
const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
const model = genAI.getGenerativeModel({ model: "gemini-2.5-flash-lite" });

router.get('/ai-personal-advice', (req, res) => {
    res.status(405).json({ error: 'Method Not Allowed' });
});

router.post('/ai-personal-advice', async (req, res) => {
    console.log('[aiPersonalAdvice.js] POST /api/ai-personal-advice called, body:', req.body);
    try {
        const { email } = req.body;
        if (!email) {
            return res.status(400).json({ error: "Email is required" });
        }

        const [transactions] = await pool.query('SELECT * FROM stock_transactions WHERE email = ?', [email]);
        const [user] = await pool.query('SELECT * FROM users WHERE email = ?', [email]);
        if (!user[0]) {
            return res.status(404).json({ error: "User not found" });
        }

        const prompt = `
You are a professional stock investment advisor. Here are the user's historical stock transactions and current account balance. Please analyze the user's investm

User's transaction history:
${JSON.stringify(transactions, null, 2)}

User's current balance: ${user[0].balance}
`;
```

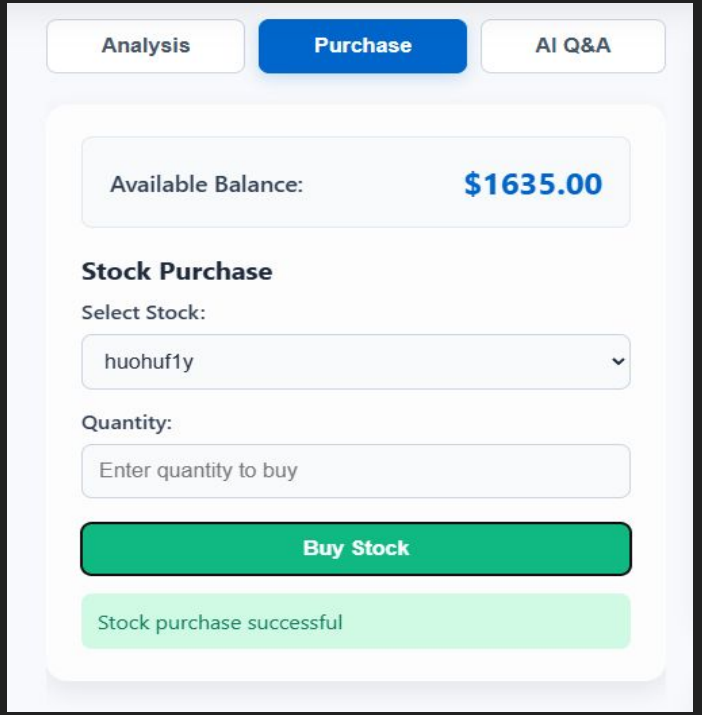
```
require("dotenv").config();
const express = require("express");
const router = express.Router();
// CHANGED: Import Google Generative AI
const { GoogleGenerativeAI } = require("@google/generative-ai");

if (!process.env.GEMINI_API_KEY) { // CHANGED: Check for Gemini key
    console.error("❌ ERROR: Missing GEMINI_API_KEY in .env file!");
    process.exit(1);
}

// CHANGED: Initialize Gemini AI Client
const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
const model = genAI.getGenerativeModel({ model: "gemini-2.5-flash-lite" });

router.post('/chat', async (req, res) => {
    try {
        const userMessage = req.body.message;
        if (!userMessage) {
            return res.status(400).json({ error: "Message is required" });
        }

        // CHANGED: Call Gemini API
        const result = await model.generateContent(userMessage);
        const response = await result.response;
        const aiResponse = response.text();

        res.json({ response: aiResponse });
    } catch (error) {
        console.error("❌ Error calling AI API:", error.message);
        res.status(500).json({ error: "Internal Server Error" });
    }
});

module.exports = router;
```

Single Stock Module - Single Stock Purchase

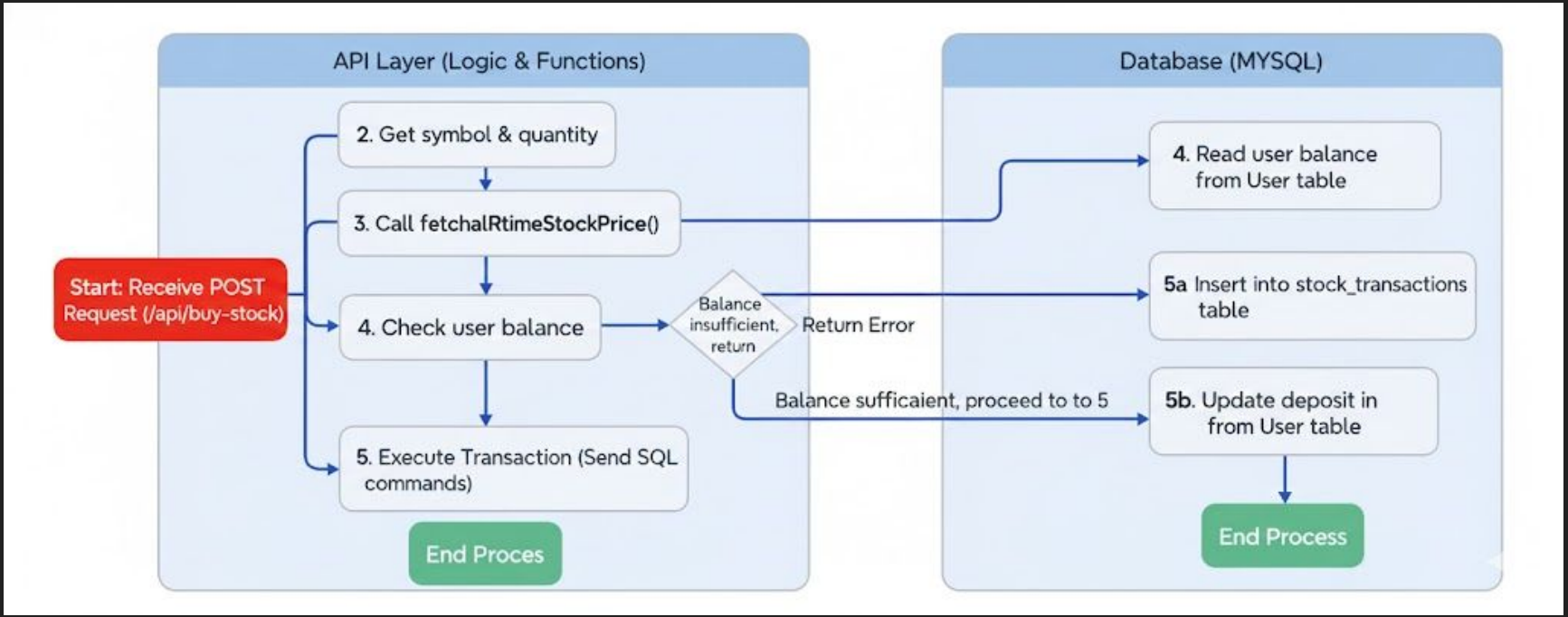
Input

- Stock Symbol: user input
- Quantity: user input

If user purchases successfully, it will give the message: Stock purchase successful



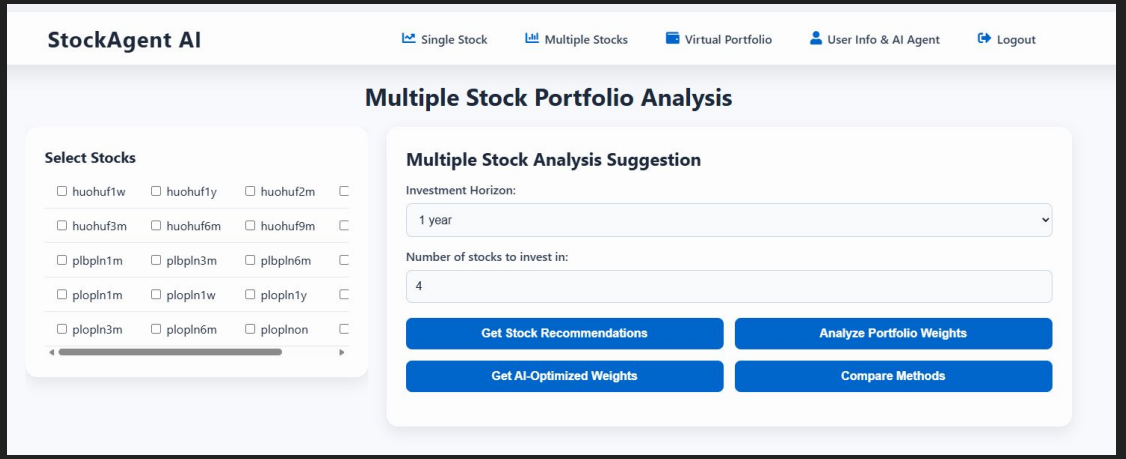
Single Stock Module - Single Stock purchase workflow



Multiple Stock Module

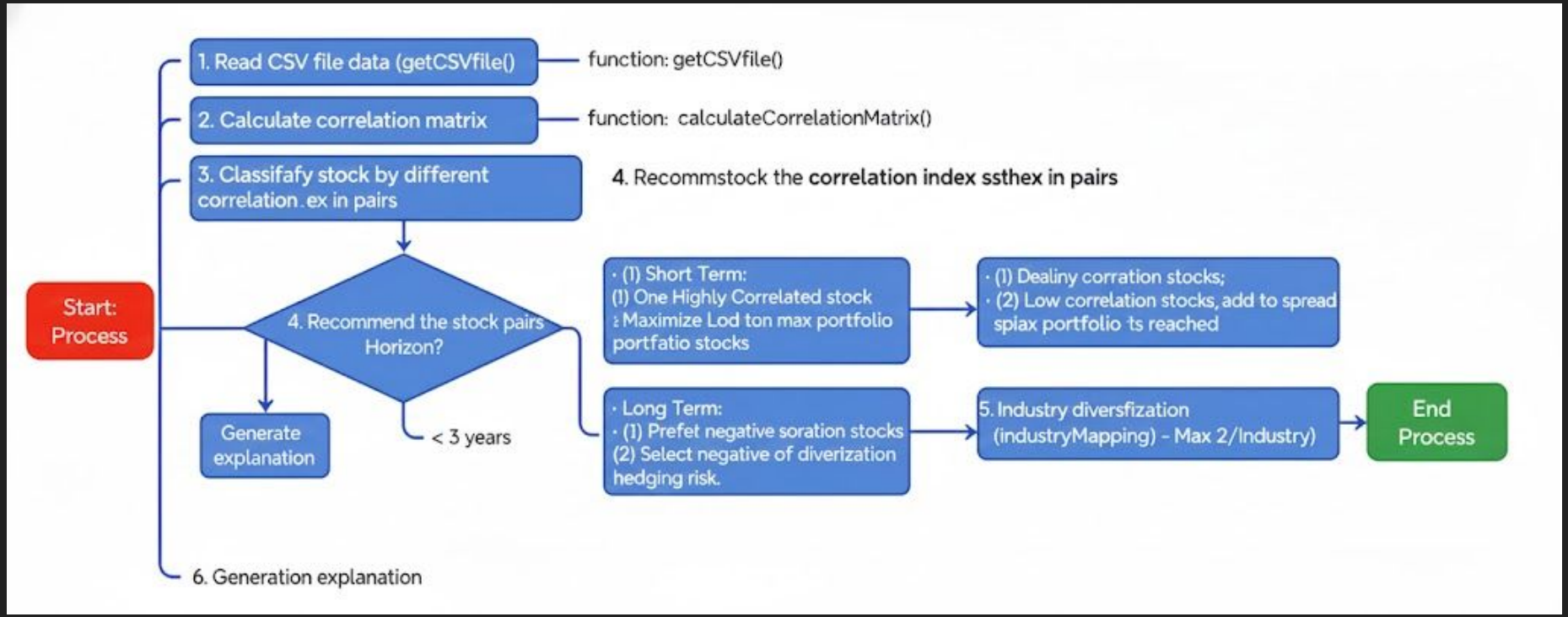
Multiple stock module - Multiple Stock Prediction

- **Input**
  - Investment horizon
  - The multiple of stocks you want to input
- **Output:**
  - Recommend portfolios
  - Portfolio weights
  - AI optimized weights(traditional and AI method)



Multiple stock module - Prediction process

Key points - Correlation matrix!



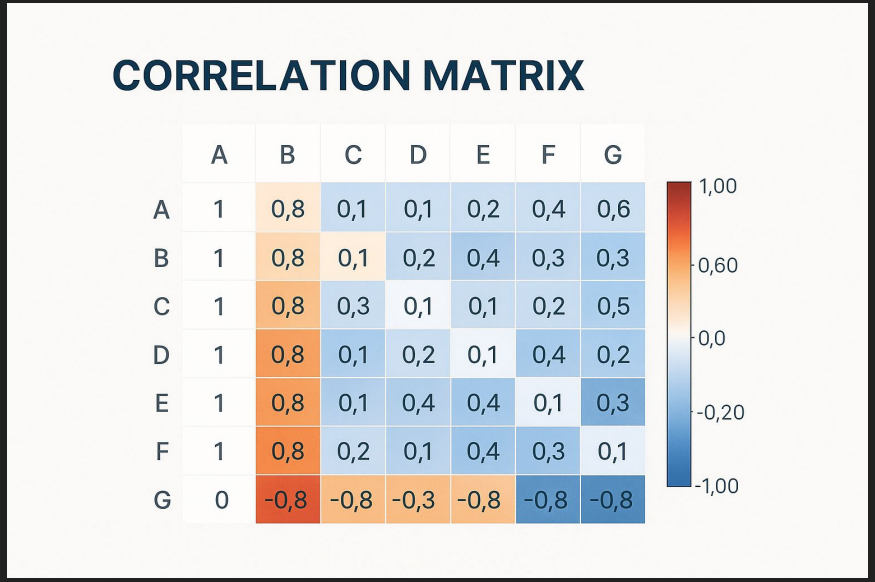
Multiple stock module - Prediction Correlation Matrix

Correlation matrix is a symmetric matrix used to represent pair-to-pair correlations between a set of variables,

- **1: Perfectly positive correlation.**
- **-1: Completely negative correlation.**
- **0: No correlation.**

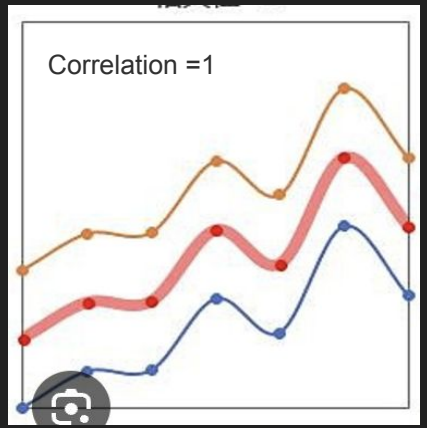
$$\text{Correlation}(A, B) = \frac{\text{Cov}(A, B)}{\sqrt{\text{Var}(A) \times \text{Var}(B)}}$$

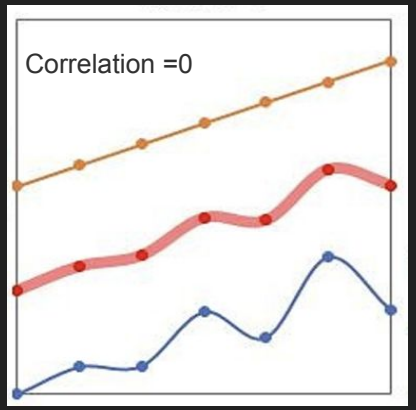
• **Covariance**: Measures the degree to which the prices of two stocks move in tandem.
• **Variance**: A measure of how much the price of a single stock fluctuates.

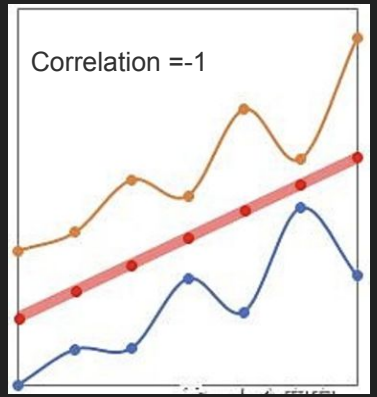


Multiple stock module - Prediction Correlation Matrix

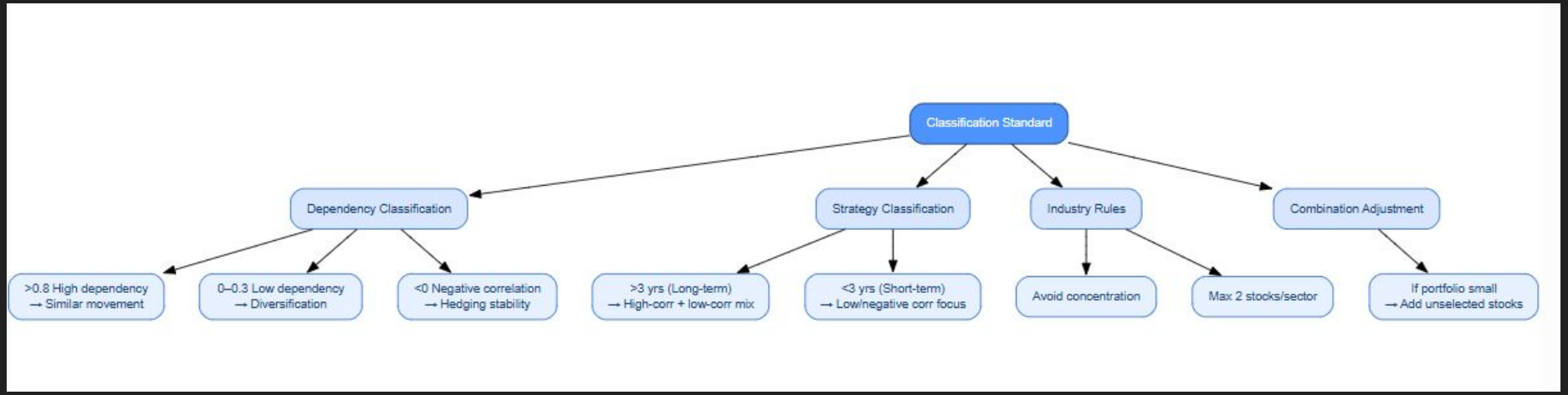
- If two stocks are **highly correlated** their price changes may be similarly affected by the same market event, and risk cannot be spread.
- If two stocks are **negatively correlated**, price movements can be partially hedged, reducing the volatility of the portfolio.







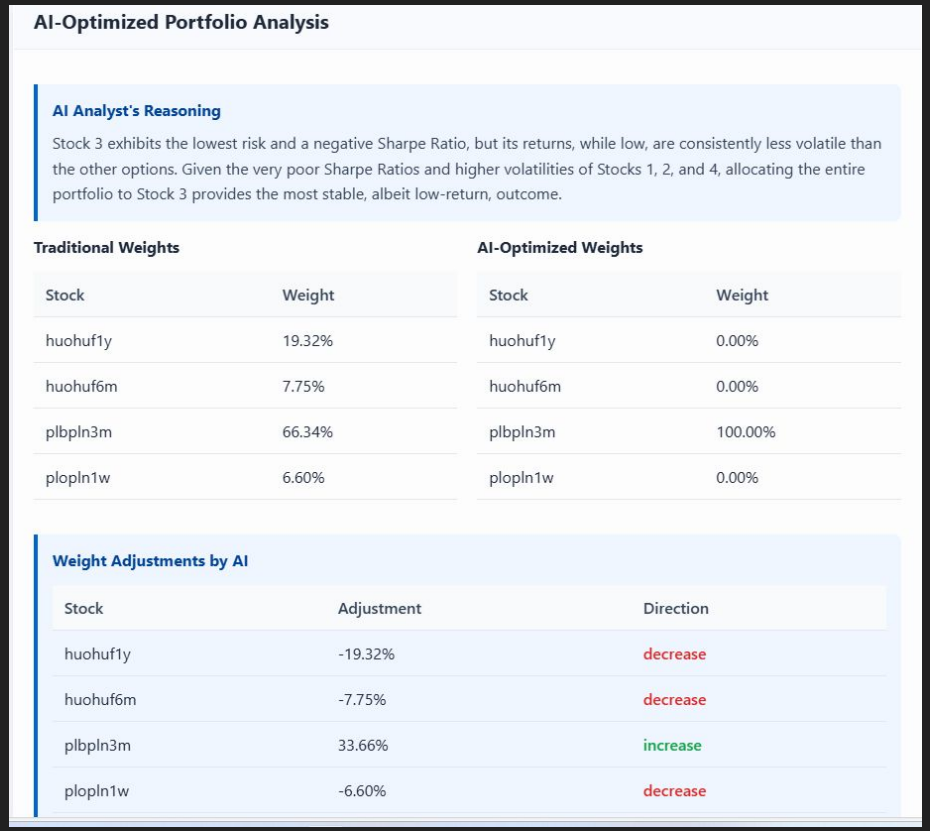
Multiple stock module - Prediction Classification Standard



Multiple stock module - Logistics of Module Portfolio Weights

- **Initial Weight Calculation and Adjustments**
  - The system first calculates an Initial Weight for each stock by adjusting it based on its individual risk and risk-adjusted return (Sharpe ratio).
  - **Risk-based Adjustment**: Risk Adjustment = risk / total risk
  - **Sharpe Ratio Adjustment**: SharpeAdjustment = 1 + (sharpeRatio / 2)
  - **Combined Initial Weigh**t: InitialWeight = RiskAdjustment * SharpeAdjustment
- **Correlation Adjustment**
  - Next, the initial weights are adjusted based on the Correlation Matrix.
  - Logic: Stocks that are highly correlated will have their weights reduced because they do not effectively spread risk. Stocks that are negatively correlated offer a hedging benefit, reducing the portfolio's volatility.
- **Final Normalization**
  - After the initial and correlation adjustments, the sum of all calculated weights will generally not equal 1. The Normalization step ensures the total allocation equals 100%.
  - Procedure: The system takes the final adjusted weight for each stock and divides it by the sum of all adjusted weights.
  - Validation: The system must ensure the sum of all weights equals 1.
  - **Final Normalized Weight of Stock i =** Adjusted Weight of Stock i / Sum of All Stocks' Adjusted Weight
  - Ensures that **100%** of the designated investment capital is allocated across the portfolio.

Multiple stock module - Calculate Weights by AI



AI-Optimized Portfolio Analysis

AI Analyst's Reasoning

Stock 3 exhibits the lowest risk and a negative Sharpe Ratio, but its returns, while low, are consistently less volatile than the other options. Given the very poor Sharpe Ratios and higher volatilities of Stocks 1, 2, and 4, allocating the entire portfolio to Stock 3 provides the most stable, albeit low-return, outcome.

Traditional Weights

Stock	Weight
huohuf1y	19.32%
huohuf6m	7.75%
plbpln3m	66.34%
plopln1w	6.60%

AI-Optimized Weights

Stock	Weight
huohuf1y	0.00%
huohuf6m	0.00%
plbpln3m	100.00%
plopln1w	0.00%

Weight Adjustments by AI

Stock	Adjustment	Direction
huohuf1y	-19.32%	decrease
huohuf6m	-7.75%	decrease
plbpln3m	33.66%	increase
plopln1w	-6.60%	decrease

Input:
Multiple stocks

Output:
Weights of multiple stocks(Adjustment and AI analysis)

Multiple stock module - Calculate weights by AI

Import modules: Uses Express and Google Gemini.

Initialize Google Gemini API with apiKey in the env file.

Define POST endpoint /ai-predict:

- Accepts stock features from client.
- Constructs a prompt based on these features.
- Sends the prompt to Google Gemini for portfolio weight suggestions.
- Parses and normalizes AI's response.
- Returns the weights as a JSON array summing to 1.

Multiple stock module - AI Consideration Advantages

AI Analysis Preparation

- Constructs structured prompts
- Incorporates all stock feature information
- Sets analysis objectives and constraints

AI Model Processing

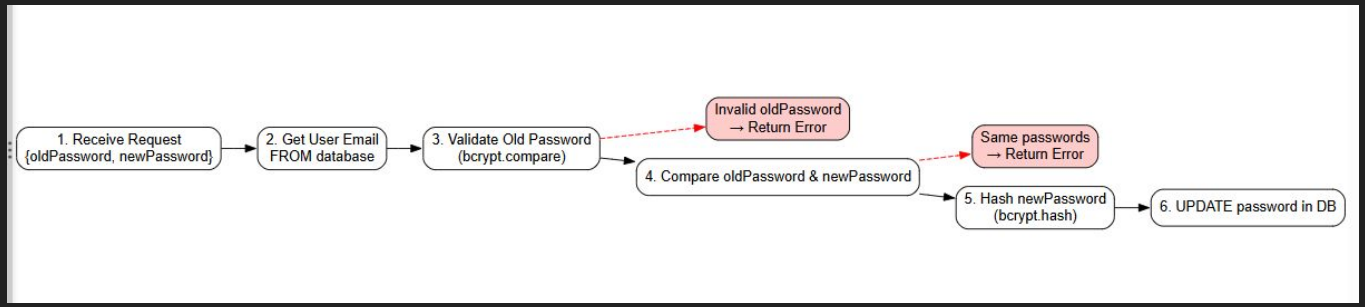
- Utilizes Gemini model for analysis
- Considers risk-adjusted returns
- Evaluates stock correlations
- Analyzes overall portfolio risk
- Takes market conditions into account

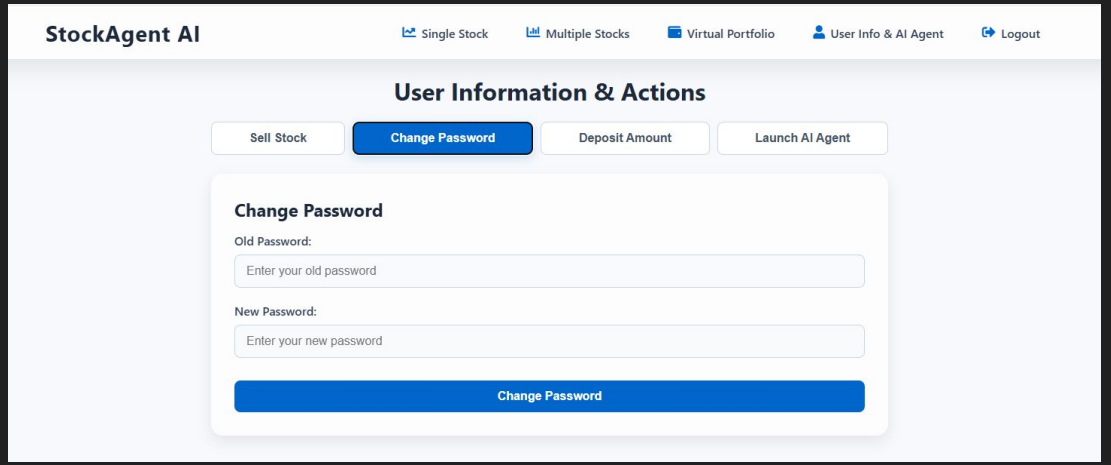
Traditional vs AI Optimization Methods

Factor	Traditional Method	AI Method
Returns Analysis	✓ Basic return calculation	✓ Advanced return analysis with market context
Risk Assessment	✓ Standard deviation	✓ Multiple risk metrics (VaR, Drawdown)
Market Conditions	✗ Not considered	✓ Current market environment analysis
Correlation Analysis	✓ Basic correlation matrix	✓ Advanced correlation with market trends
Dynamic Adjustment	✗ Static weights	✓ Dynamic weight adjustment

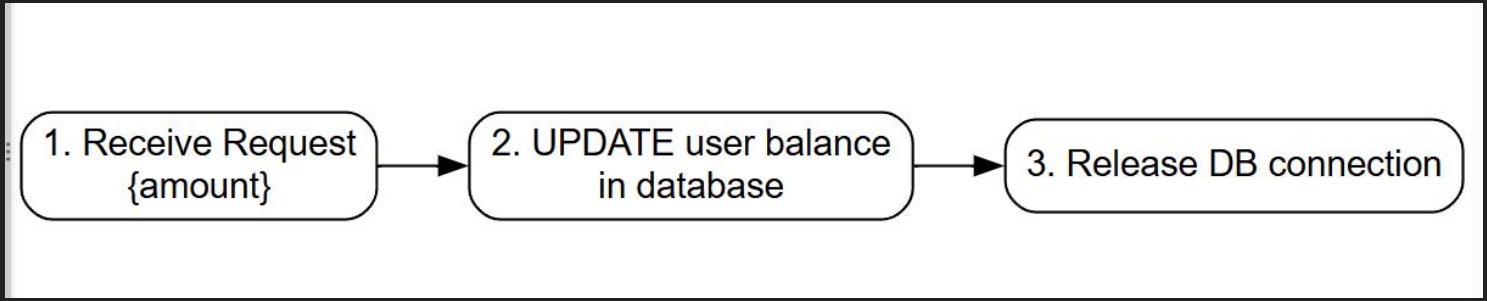
User Information Module

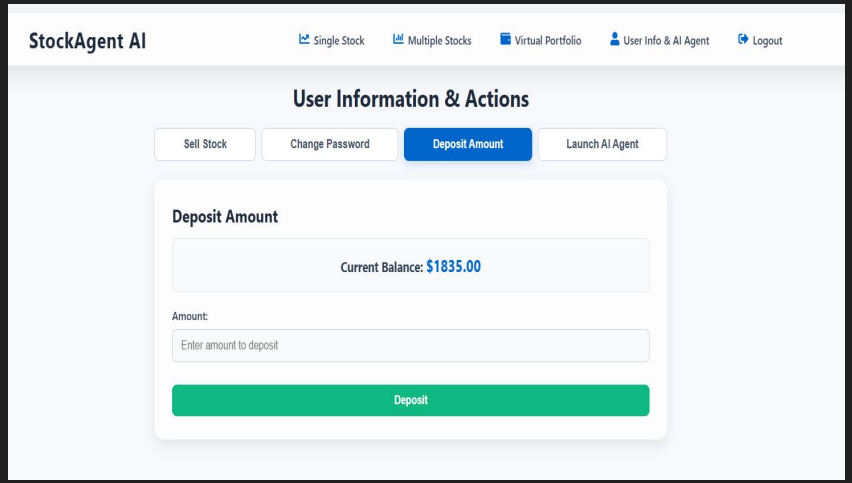
User Information Module - Change Password

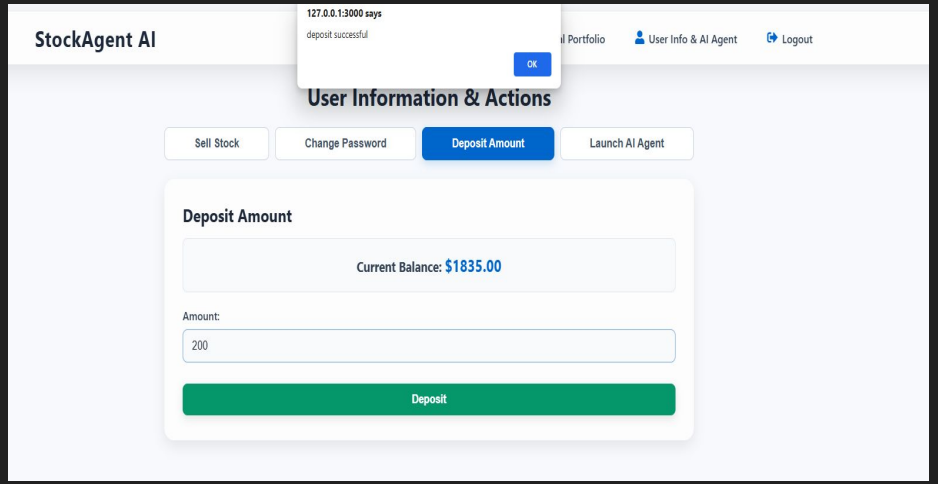




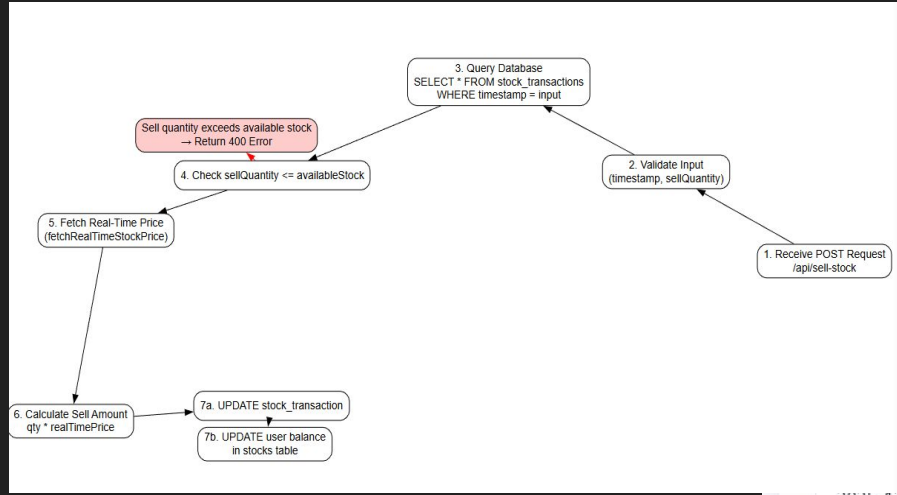
User Information Module - Deposit Money







User Information Module - Sell Stock



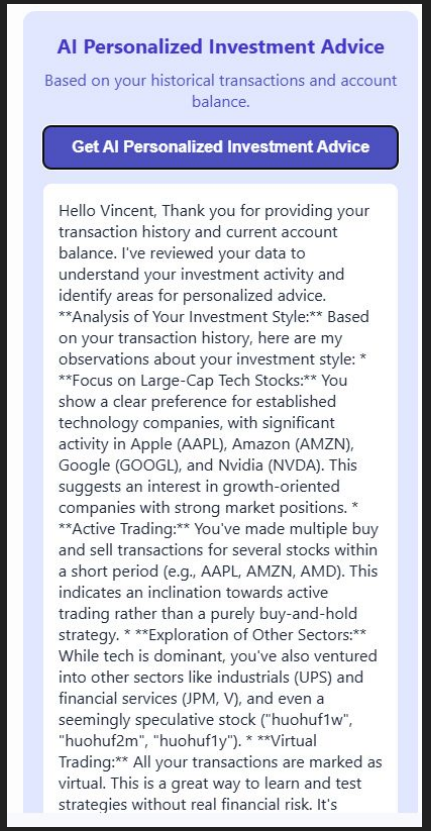
AI personalized Investment Advice

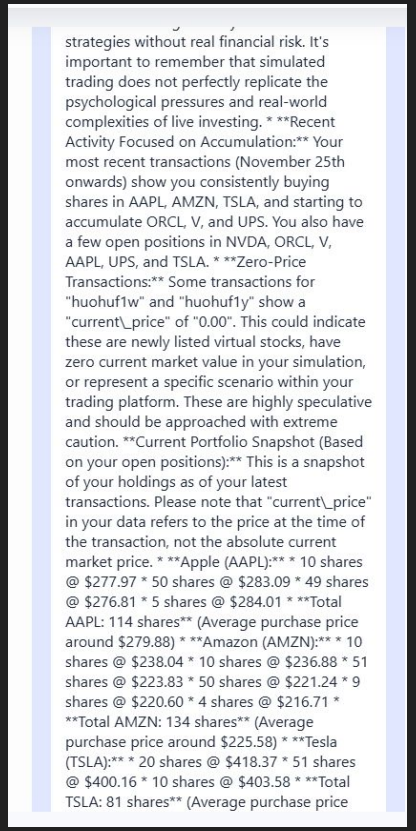
User input

- Uer email
- User transaction history
- User current balance

Analysis Content

- Investment style
- Analysis of Trading Patterns
- Risk analyze
- Recommendations for action steps

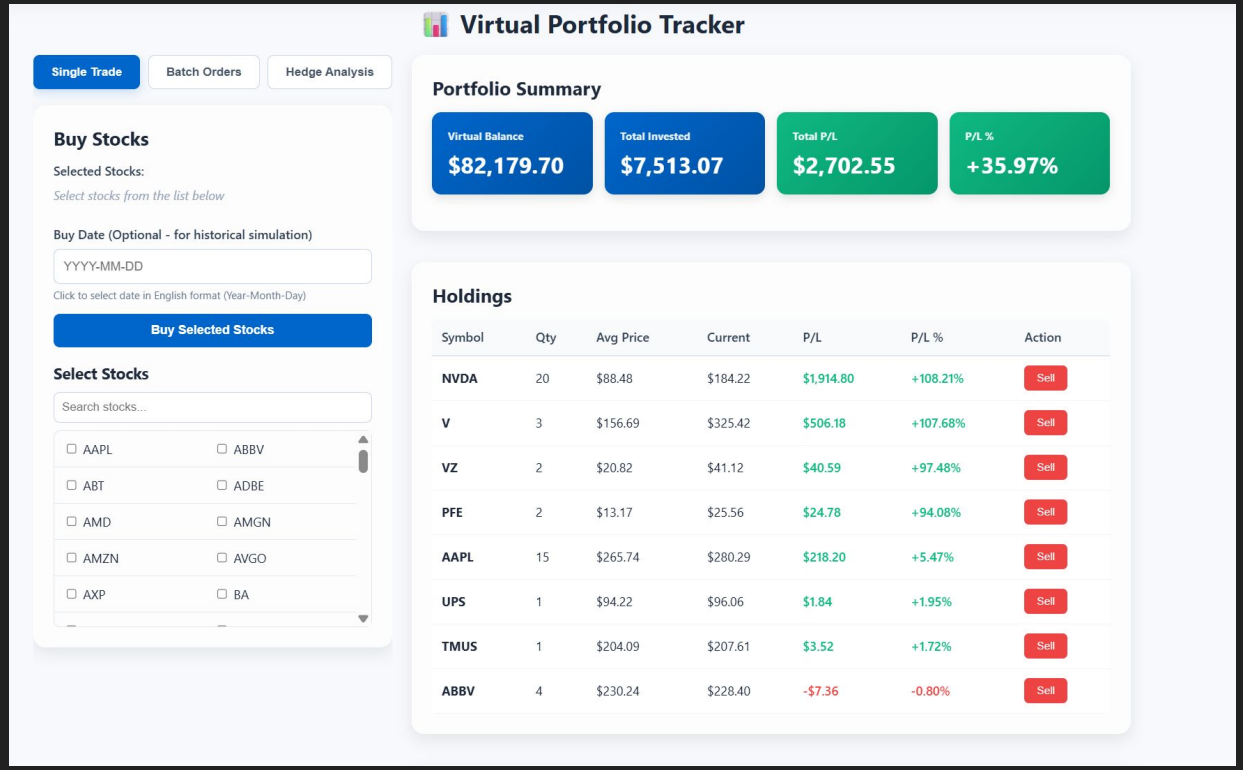




Virtual Portfolio

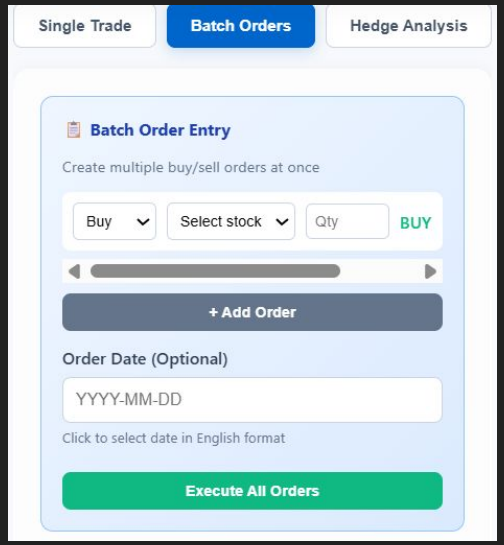
Core Features

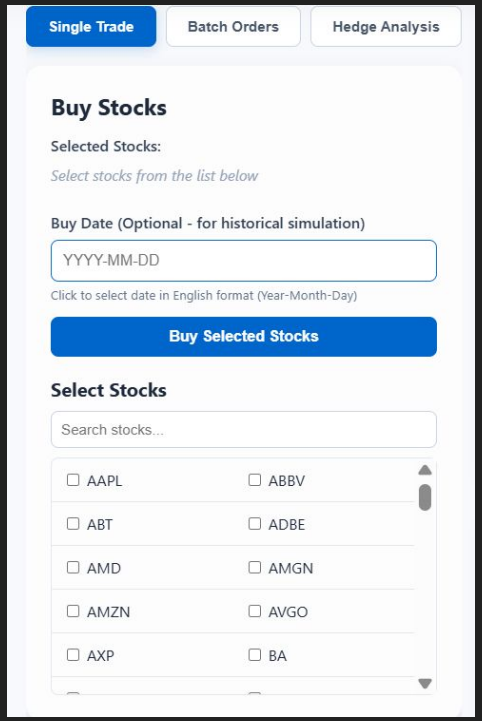
- Virtual stock trading with $100,000 initial balance
- Real-time price tracking via Finnhub API
- Automatic portfolio updates every 30 seconds
- Virtual balance management

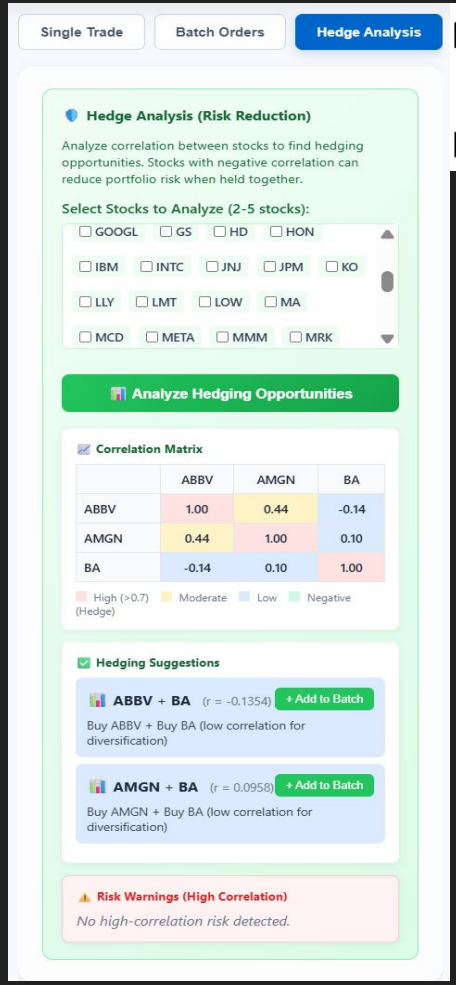


Trading Features

- Single Trade: Buy individual stocks with quantity selection
- Batch Orders: Execute multiple buy/sell orders simultaneously
- Hedge Analysis: Analyze stock correlation to find hedging opportunities and reduce portfolio risk

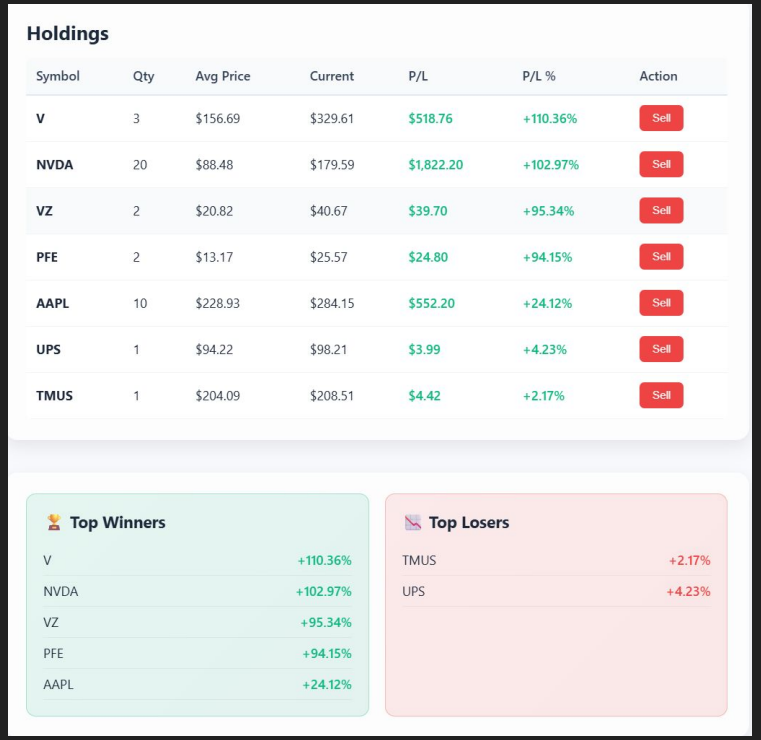






Portfolio Analysis

- Portfolio Summary
  - Virtual balance, total invested, current value
- Profit/Loss Tracking
  - Absolute and percentage calculations
- Winners & Losers:
  - Top 5 performing and worst performing stocks
- Holdings Details:
  - Symbol, quantity, avg buy price, current price, P/L
- Backtesting:
  - Test strategies on historical data with metrics (max drawdown, volatility)



Holdings

Symbol	Qty	Avg Price	Current	P/L	P/L %	Action
V	3	$156.69	$329.61	$518.76	+110.36%	Sell
NVDA	20	$88.48	$179.59	$1,822.20	+102.97%	Sell
VZ	2	$20.82	$40.67	$39.70	+95.34%	Sell
PFE	2	$13.17	$25.57	$24.80	+94.15%	Sell
AAPL	10	$228.93	$284.15	$552.20	+24.12%	Sell
UPS	1	$94.22	$98.21	$3.99	+4.23%	Sell
TMUS	1	$204.09	$208.51	$4.42	+2.17%	Sell

Top Winners

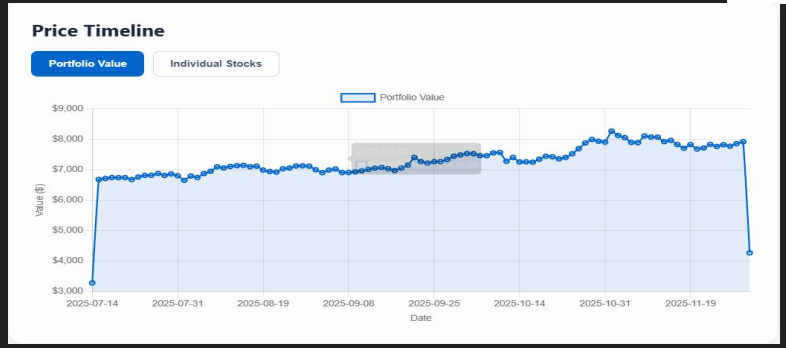
V	+110.36%
NVDA	+102.97%
VZ	+95.34%
PFE	+94.15%
AAPL	+24.12%

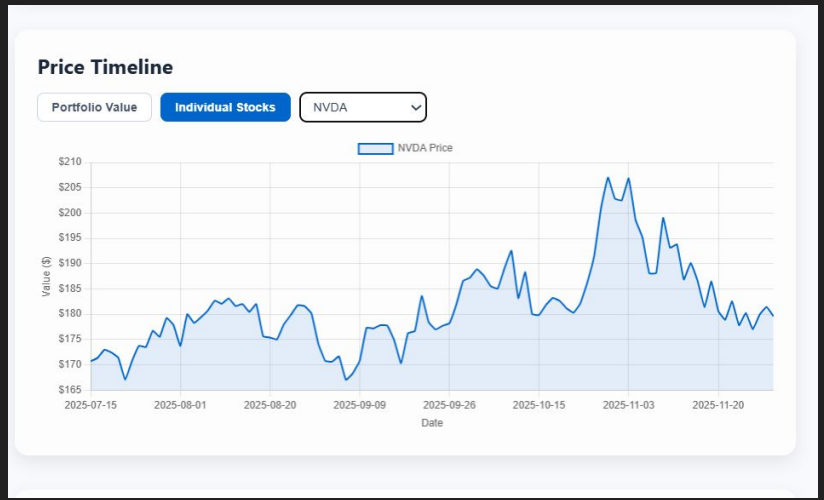
Top Losers

TMUS	+2.17%
UPS	+4.23%

Visualization

- Portfolio Value Timeline Chart
  - Track total portfolio value over time
- Individual Stock Price Chart
  - View price movements for specific stocks
- Real-time Data Updates
  - Charts refresh automatically with latest prices





API Usage

- Finnhub API
  - Real-time stock quotes (current price, previous close)
- Alpha Vantage API
  - Historical daily data (OHLCV)

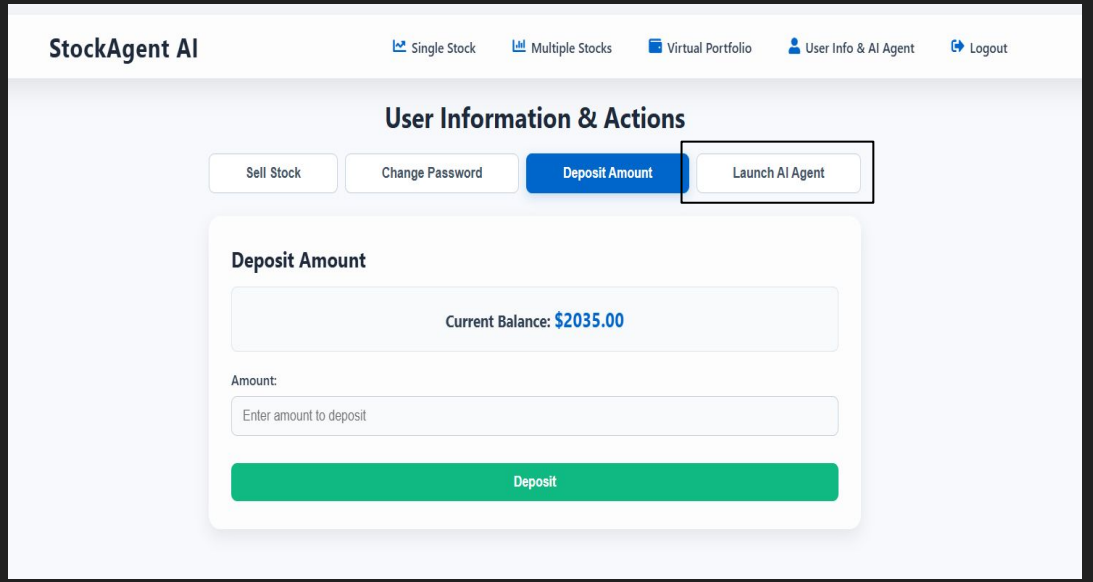
```
FINNHUB_API_KEY=d4m9fh1r01qjidhudfe0d4m9fh1r01qjidhudfeg
ALPHA_VANTAGE_API_KEY=4G7GFSDBJDDTJ4B7
```

AI Simulation Agent

User Information Module - AI Agent

AI Agent (Google Gemini)

- Inside the Stockagent of PromptCoder, use python app.py to run the agent
- click the button and it will go to the http:localhost:5001

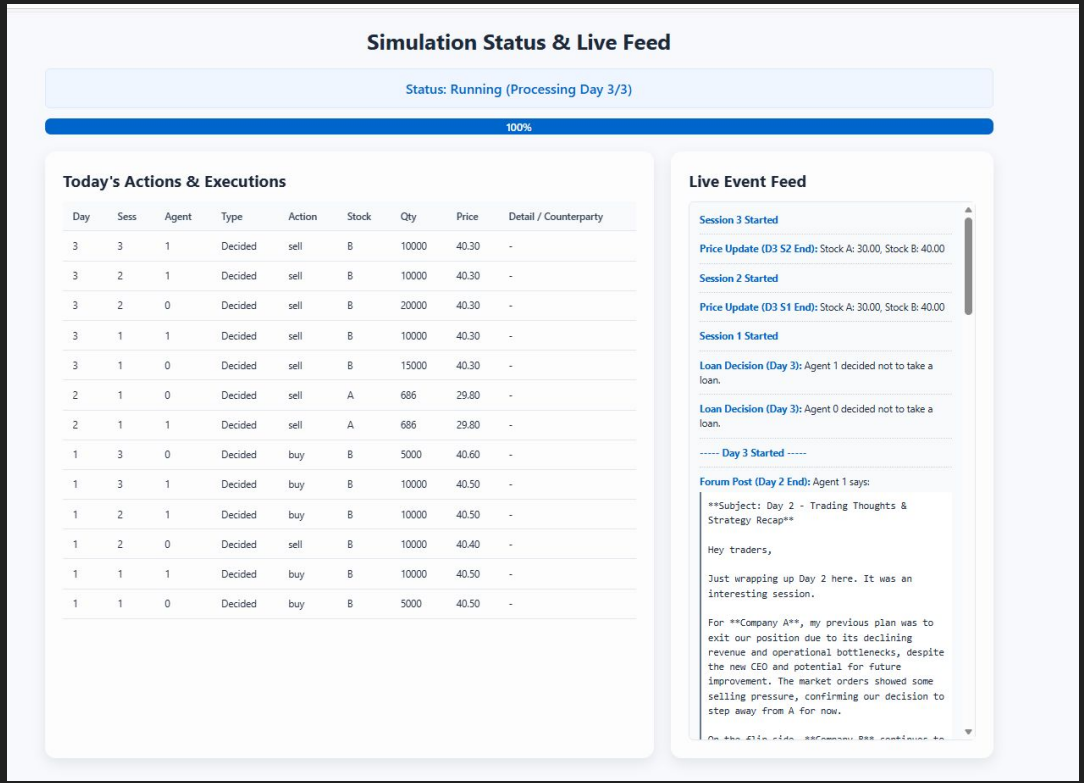


User Information module - AI Agent

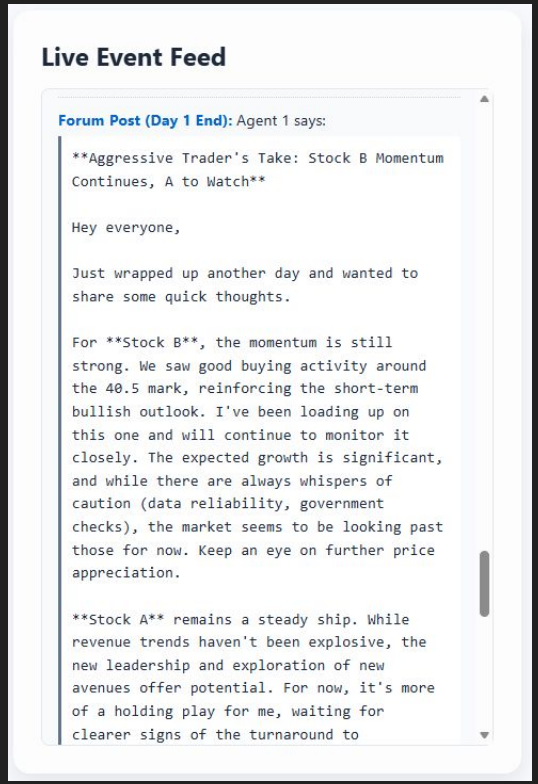
- **API Configuration:**
  - Google Gemini API key
  - Model Name
- **Basic Settings:**
  - Number of Agents
  - Total Simulation Days
  - Daily Trading Sessions
- **Stock Initial Settings**
  - Stock A/B initial Price
- **Agent Initial Property**
  - Max/Min initial property
- **Loan Settings**
  - It will analyze two stocks
  - It will make a more comprehensive inference based on the financial reports and the market

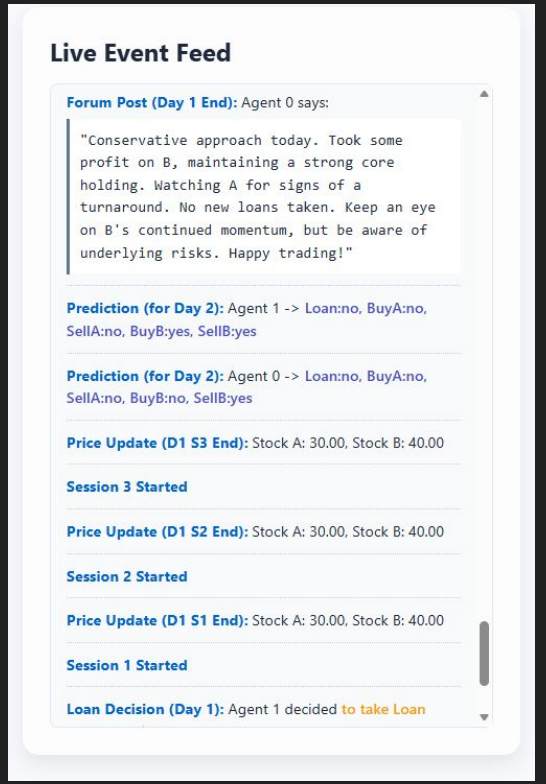
User Information Module - AI Agent

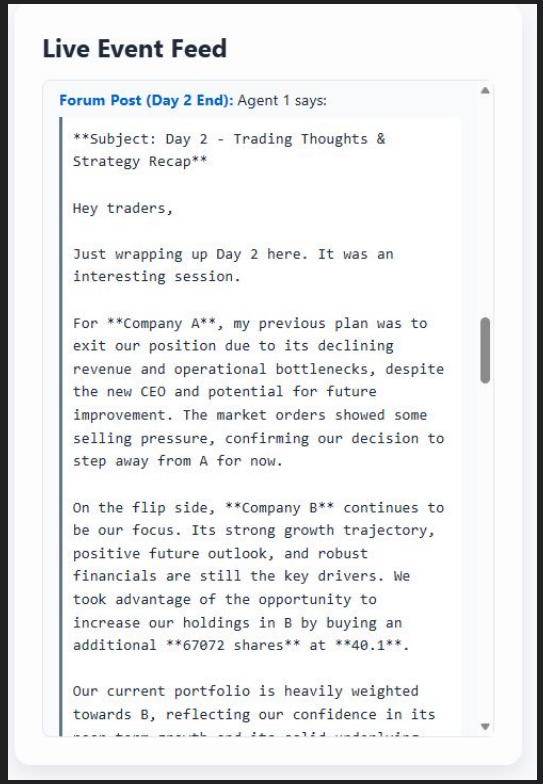
Click “Run Simulation”



User Information Module - AI Agent







Frontend Design

- HTML: overall layout
- Two forms:
  - Login form and registration form, with the login form displayed by default, and the registration form shown when a button is clicked.
- Element breakdown:
  - Input fields: Used for the user to enter their username and password (<input> tag).
  - Buttons: Used for the user to perform login or registration actions (<button> tag).

Technical Details

- Frontend
  - HTML, CSS, JavaScript for user interaction.
- Backend
  - Node.js with Express.js for server-side logic.
- Database: MySQL for storing user and stock data.
- APIs: Facilitate communication between frontend and backend.
- Utility Libraries:
  - bcrypt.js (Hash users’ password)
  - Csv-parser (Read CSV file)
  - Moment.js
  - Math.js (Used to calculate some functions for stock broadcast)

Demo

Contribution

- **Yiwei Li**
  - Backend Design and programming
  - Ai Agent Design and programming
- **Yanbo Tong**
  - Google Slides Design
  - Program Report (Part 5 - System Evaluation)
  - Frontend Design : Client-Friendly UI Page
- **Haoyang Guo**
  - Implement the frontend and backend of virtual portfolio system
  - using real-time stock API

Thank You!