# Description

Our stock trading game simulates the experience of real stock market trading, but in an online environment. Players utilize real-time data from the New York Stock Exchange (NYSE) to make strategic decisions about buying and selling stocks. The objective is to grow their portfolios and accumulate wealth by the end of the game. Admins oversee the game's operations to ensure fairness and adherence to rules.

Administrators have the ability to customize game parameters such as initial capital and duration, as well as monitor player activity for compliance. Our game offers an engaging platform for learning about stock markets while providing a competitive arena for players to test their skills and strategies.

# Screens

A screenshot of a screenshot of a website

Description automatically generated

This is the UI for the portfolio of player where he can see the total profit/loss for the individual stocks and overall portfolio as well. Also you can see quantity, price you bought it for.

A screenshot of a computer screen

Description automatically generated

This is the UI for buying a stock. Where you will enter the number of shares to be bought and total amount is generated with the current rates.

# Features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Name** | **Access By** | **Short Description** | **Expected Implementation** | **Source of Idea** |
| 01 | Player registration | Player | Register players for the game | Must implement | Project instructions |
| 02 | Game creation | Admin | Admins create new game instances | Must implement | Project instructions |
| 03 | Starting cash account | Player | Players receive initial capital in portfolios | Must implement | Project instructions |
| 04 | Real-time stock data access | Player | Access to current stock market data | Must implement | Project instructions |
| 05 | Buy stocks | Player | Players can purchase stocks | Must implement | Project instructions |
| 06 | Sell stocks | Player | Players can sell owned stocks | Must implement | Project instructions |
| 07 | Portfolio management | Player | View and manage owned stocks | Must implement | Project instructions |
| 08 | Portfolio value tracking | Player | Monitor portfolio value changes | Must implement | Project instructions |
| 09 | Game leaderboard | Player | View rankings of players' portfolios | Must implement | Project instructions |
| 10 | Admin dashboard | Admin | Monitor game progress and player activity | Must implement | Project instructions |
| 11 | Player authentication | Player | Secure login for players | Must implement | Project instructions |
| 12 | Admin authentication | Admin | Secure login for admins | Must implement | Project instructions |
| 13 | Game customization | Admin | Set game parameters (duration, starting cash) | Must implement | Project instructions |
| 14 | Transaction fees | Admin | Set fees for buy/sell transactions | Likely to be done | Industry regulations |
| 15 | Viewing competitor portfolios | Player | Optional view of other player’s portfolios | Likely to be done | User feedback |
| 16 | Trade history tracking | Player | View past buy/sell actions | Likely to be done | User feedback |
| 17 | Time and date settings | Admin | Configure game start and end times | Likely to be done | Project requirements |
| 18 | Email notifications | Player | Receive alerts for important events | Probably not unless easy | User feedback |
| 19 | Chat functionality | Player | Communicate with other players in-game | Probably not unless easy | User feedback |
| 20 | News feed integration | Player | Access to financial news updates | Likely to be done | User feedback |
| 21 | Achievements | Player | Unlockable milestones for player accomplishments | Probably not unless easy | User feedback |
| 22 | Tutorial | Player | Guided introduction to game mechanics | Probably not unless easy | User feedback |
| 23 | Advanced trading options | Player | Additional trading actions beyond buy/sell | Probably not unless easy | User feedback |
| 24 | Multi-game support | Admin | Ability to manage multiple concurrent games | Probably not unless easy | User feedback |
| 25 | Social media integration | Player | Share game progress on social platforms | Probably not unless easy | User feedback |
| 26 | Data analytics dashboard | Admin | Insights into player behavior and trends | Probably not unless easy | Project requirements |
| 27 | Stock market simulator | Player | Practice trading without real money | Probably not unless easy | Project requirements |
| 28 | Language localization | Player | Support for multiple languages | Probably not unless easy | User feedback |
| 29 | User support | Player | Access to help resources and FAQs | Probably not unless easy | User feedback |
| 30 | Real-time market charts | Player | Interactive chart to display data | Probably not unless easy | User feedback |

# Tools and Packages

* Node.js for server-side JavaScript runtime environment.
* Express.js for web application framework for Node.js to handle HTTP requests.
* MongoDB for storing player data, game settings, and transactions.
* React.js for building user interfaces, for the front end.
* JWT for secure authentication and authorization.

# App API:

* GET /portfolio?player=playername&game=gameid: Retrieves the current portfolio of a player for a specific game.
* POST /sell?player=playername&game=gameid&stock=tickersymbol&quant=nnn: Initiates a sell action for a specified quantity of a stock in the game's context. Responds with the success or failure of the transaction and the selling price.
* POST /buy?player=playername&game=gameid&stock=tickersymbol&quant=nnn: Initiates a buy action for a specified quantity of a stock in the game's context. Responds with the success or failure of the transaction and the buying price.
* POST /register: Registers a new player for the game with provided details.
* POST /create\_game: Creates a new game instance with customizable parameters.

# Stock API:

I’m going to use Alpha Vantage API for the stock price. Here are some example API calls:

* GET /query?function=GLOBAL\_QUOTE&symbol=MSFT&apikey=demo: Retrieves the latest stock quote for Microsoft (MSFT), including open, high, low, and current price.
* GET /query?function=TIME\_SERIES\_DAILY&symbol=MSFT&apikey=demo: Retrieves daily stock prices for Microsoft (MSFT).
* GET /query?function=SYMBOL\_SEARCH&keywords=Microsoft&apikey=demo: Searches for stocks based on keywords, in this case, "Microsoft".

I’m going to use FinnHub API for the stock news. Here are some example API cals:

* GET/company-news?symbol=AAPL&from=2023-08-15&to=2023-08-20=demo : Retrieves 1 year of historical news and new updates of AAPL company.

# Attribution

* Web research: Utilizing information obtained from online sources, such as documentation, tutorials, and forums, to inform decision-making and implementation strategies.
* Images sourced are from Wealthsimple and Jainam Pro for reference and design inspiration.