To The Moon

Capstone Project Report

COMP3900-W18A-TO-THE-MOON

Project Report Submission Date: 23/04/2021

Austin Landry
Developer
z5234010
z5234010@student.unsw.edu.au

Jun Moey
Developer
z5112826
z5112826@student.unsw.edu.au

Karim Saad
Developer
z5209523
z5209523@student.unsw.edu.au

Sean Smith
Developer
z3415419
z3415419@student.unsw.edu.au

Sarah Tan
Scrum Master
z5166081
z5166081@student.unsw.edu.au

Overview

Problem Domain

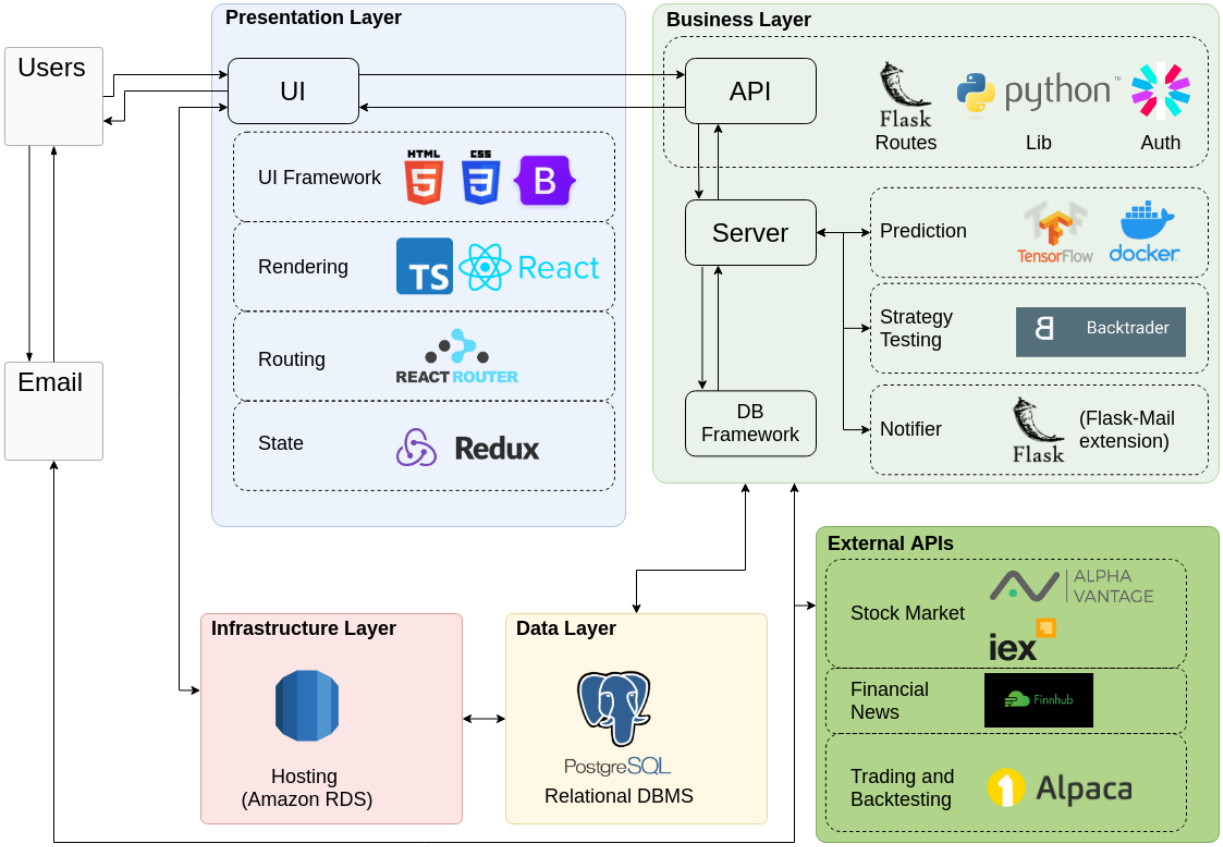
“To develop a web application which leverages machine learning methods, social forums, and intuitive UI features, to aid our users in researching, managing, and initiating new investment portfolios.”

Our product, “To The Moon” (TTM), is a web application for stock portfolio management which features machine learning predictive models, social features, news and automated trading. The platform provides users with an intuitive, all-in-one investment application. In preparation for its development, we undertook extensive market research to identify our key competitors and their critical features, with the commitment to provide a lightweight implementation of these successful features, in addition to our new and exciting investment functionality.

We believe TTM delivered upon that commitment by matching the basic functionality of our competitors, while also providing users with a refinement of their winning features. Some of this basic functionality includes portfolio creation and management, investment research and execution capabilities, news and current affairs resources, and stock screening and visualisation features. By streamlining the UI and removing irrelevant features, we created an intuitive investment research and management platform for the novice investor.

However, TTM strives to be an all-inclusive investment platform, so we also decided to include additional technical features to attract intermediate and advanced investors. Some of these technical features include our machine learning prediction models, interactive graphing features, and automated backtesting. We believe this provides the optimal balance between a simple and intuitive UI for novice users, while never sacrificing our technical functionality for advanced investors.

Architecture



To The Moon employs a three-layer web application architecture to implement the presentation, business logic and data functionality in separate layers communicating over a REST API.

1. The presentation layer displays processed data and handles user inputs.
2. The business layer implements the application functionality, divided into a server module for core features (authentication, user profiles, information retrieval, portfolio management, forum comments) and two microservices to implement auxiliary features (share price prediction and trading simulation).
3. The data layer stores application data on a cloud-hosted service for user consumption, including company information, market data, user-created content e.g. portfolios, forum comments.
4. The business layer is the first point of connection for retrieval requests to the data layer and also handles communication with third-party APIs to retrieve required application data where necessary. The separate processing arrangement ensures loose coupling between the business and presentation/data layers.
5. Microservices are containerised to ensure they can run independently of the server for maintenance and testing purposes and to support software-level dependencies that the server may not be equipped to accommodate.

Frontend

Presentation Layer

The presentation layer was built on the React framework. React implements its own XML-like language for rapid rendering and testing. It then converts this to HTML, CSS and Javascript for use on the web.

React uses fast, event-driven DOM rendering which should allow for a performant and responsive UI (React, 2021). React also has better support for external libraries and third party integration compared to frameworks for ready-made applications like Angular.

We used React's Typescript flavour to enforce better code structure and minimise type errors by enforcing explicit data types at compile time (SitePoint, 2021). Typescript also appears to be better supported than Flow (React, 2021), the other type checking option supported by React.

Our team also used React-Bootstrap to construct ready-made UI elements for its responsive performance, and Highcharts for visualising stock data due to its native support for graphing time-series financial data (Highcharts, 2021). For handling routing within our Single Page Application, we used React Router, which can easily handle nested views and the progressive resolution of views. To predictably handle frontend state within our application, our team used React Redux to centralise state management.

Backend

Business Layer

The project will implement a Python-based backend. The Python language was selected for its clear syntax and easy to understand structure. More importantly, it has well integrated support for the data science and machine learning libraries required to implement the prediction features. It is not as well suited for concurrent processing compared to languages like Golang or Java, but performance issues can be mitigated and supporting machine learning models on those other platforms would require more effort on integration.

To expose backend functions to the UI we will implement a REST API, accepting requests and responses over HTTP. We opted for the Flask application framework due to its accessibility and greater flexibility compared to Django, a more rigid and resource-heavy framework (Singh, 2021). Flask also supports endpoint namespaces, schemas for data validation and integration with the Swagger interface description language. We opted to use the RestX framework to implement these schemas as it offers a comprehensive selection of data types and integrates well with Swagger, which can automatically generate documentation from the schemas read. While Flask does not yet support asynchronous requests, the capacity offered by the library and

its lightweight nature was judged sufficient for usage by moderate volumes of users. Flask is also modular and supports the above mentioned extensions.

The server endpoints and prediction and trading systems are also run as separate microservices to better support upscaling of the application for higher request volumes. In particular, services can be deployed with better optimised libraries, e.g. Quant, which supports async requests, or can be set up to accept queries from the presentation layer and divert network traffic from the server, without disrupting the entire application.

Login and authentication functionality was implemented with RFC 7519 JSON web tokens (JWT) as a lightweight alternative to session based (OAuth) authentication. Both offer similar levels of security but JWT is easier to set up and implement for the volume of users intended for testing.

Data Layer

The data layer was implemented with a PostgreSQL database. PostgreSQL uses an object relational model which can suitably express the relationship between stocks, portfolios and user interactions and user profiles better than document models. It also has balanced performance on read and write operations suitable for the scale of the application and good concurrency, data typing and third party library support. The performance improvement of recent PostgreSQL implementations compares favourably to lightweight SQL flavours such as MySQL that have historically performed better on read-only operations (Hristozov, 2019). Our database is hosted in the cloud utilising AWS's Relational Database Service (RDS) which aided in provided a resizable and cost-efficient setup for our data layer.

Machine Learning Layer

To implement prediction-based features, AI models were built with libraries appropriate to prototype a variety of model types, including 'scikit-learn' for traditional machine learning models (SVM regressor), 'statstools' for statistical models (ARIMA and derivatives) and 'Tensorflow' for deep learning models. Ultimately, the deep learning models were selected due to their performance, so the final implementation uses exclusively Tensorflow. Tensorflow also offers the Keras high-level API which was employed to prototype and test different model architectures. For production, Tensorflow also features low-level performance optimisations during model building and testing that would be critical in a high-use production environment.

Trading Backtesting Layer

The trading simulation and strategy component was implemented with Backtrader using a broker account provided by Alpaca Markets. Of the backtesting engines available to simulate trading strategies and return formatted results, Backtrader is the most widely supported alternative for local running and was chosen over Zipline which is now unsupported since its parent project Quantopian became defunct, and QuantConnect Lean, which does not offer local running of strategies to minimise data upload overheads.

Functionality

Project Objectives

Our core and novel features have successfully achieved the proposed objectives for the project. The functionality of each feature will be exposed in the next section with reference to the project objectives.

A. **User profiles:** *The system will allow sellers to create user profiles to store personalised content and preferences generated from the use of the system. Users can register for a profile and login to the system with one to use the full selection of the system's features.*
B. **Portfolio management:** *The system will allow the creation of portfolios to let users aggregate their investments into a list and monitor their performance. Users can create portfolios and add, edit, delete and view investments.*
C. **Visualisation of share information:** *The system will display visualisations of share price data that is both interactive and rich.*
D. **Stock information:** *The system will display information related to a stock such as company statistics, company financials and share price/value data.*
E. **Investment news and current affairs:** The system will provide financial-related news items for general consumption and for use in analysing a specific company or investment.
F. **Opportunities to find new potential investments:** *The system will provide opportunities for users to explore investments that may not be currently held in the user's portfolio by providing recommendations for new potential investments.*
G. **Investment predictions and online trading:** *The system will allow users to submit stock price data and receive predictions on the future stock price for a selected company. Users will be able to select models to use during prediction and request predictions with provided parameters.*
H. **Social interaction:** *The system will allow users to interact with each other to facilitate meaningful discussion and build knowledge on investing within the community.*
I. **Customisable dashboard:** *The system will provide users with a highly customisable dashboard to track the content that they most care about. The dashboard should support viewing of personal investments, stock graphs and financial news.*

Core Features

Landing Page

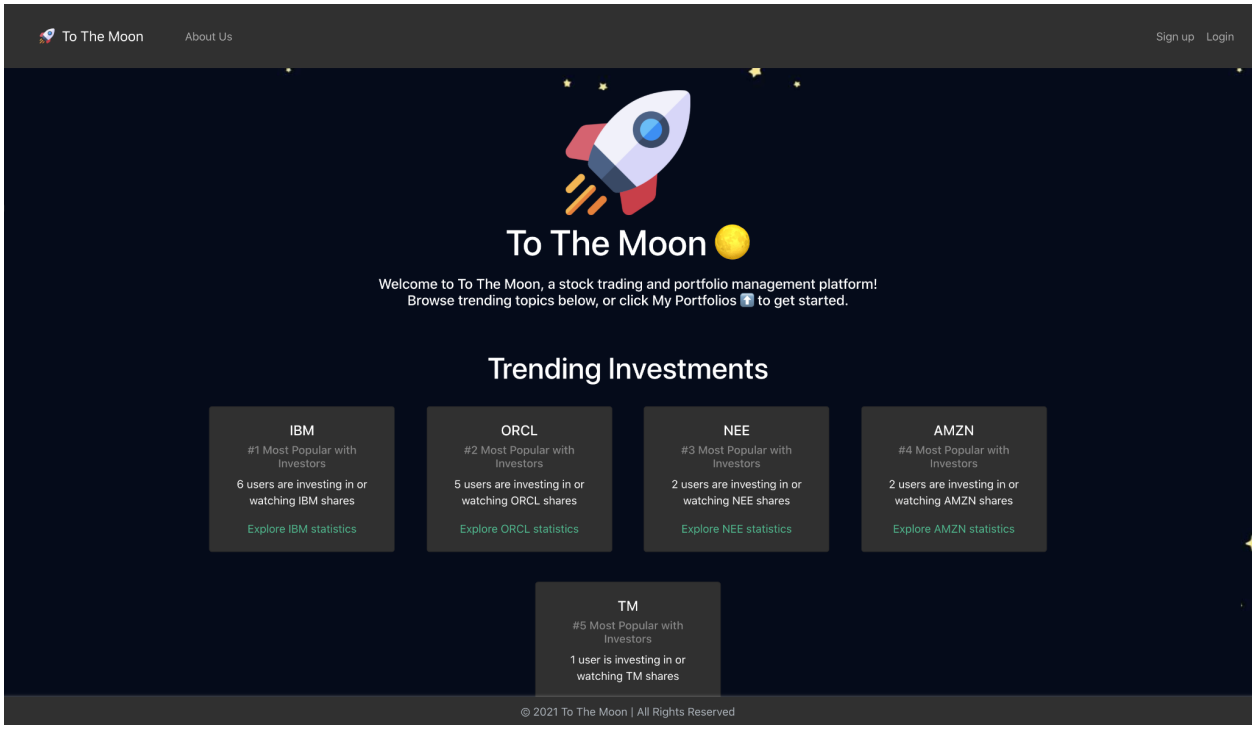
Description

The landing page acts as the ‘home page’ of our web application and serves as a means to attract news users. Users are greeted with an overview of the web application, including a trending investments section and current financial news. Users are able to sign up through the Sign up menu button and login, where they can explore the application’s further functionality.

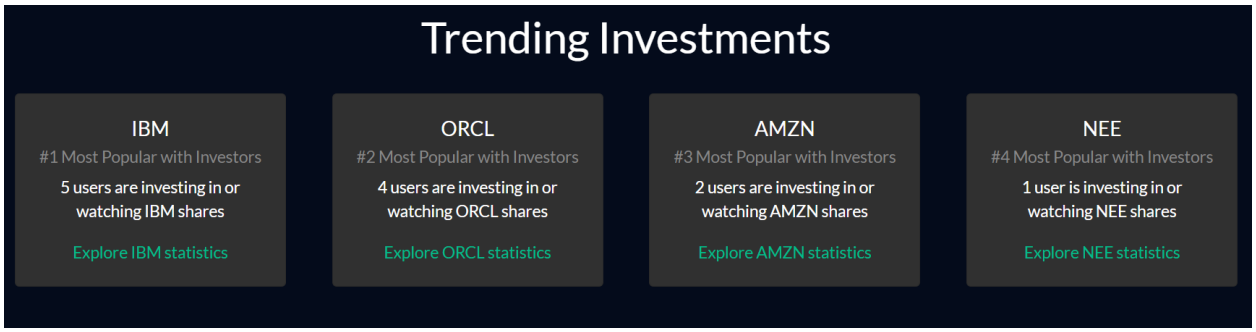
This feature satisfies requirements pertaining to objective(s):

- **A. User profiles**
- **E. Investment news and current affairs**
- **F. Opportunities to find new potential investments**

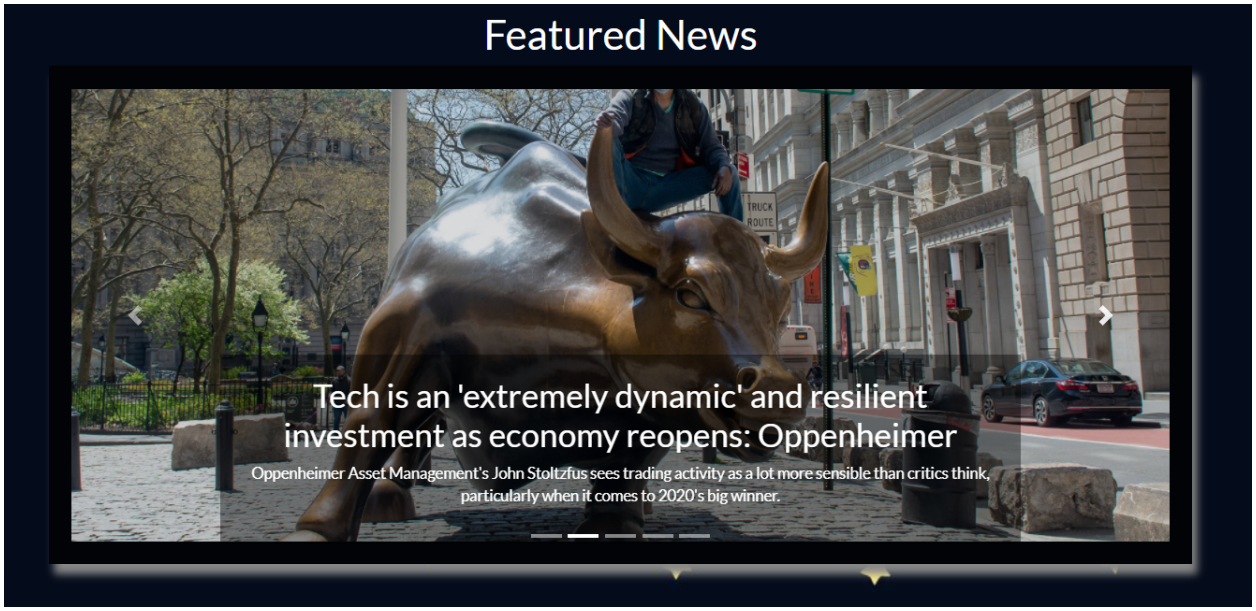
Exposition



Upon launch or clicking *To The Moon* in the navigation bar, users are brought to the landing page.



The Trending Investments section features the top invested stocks on the platform by its users. Clicking the ‘Explore X statistics’ link brings authenticated users to the stock page for X. Unauthenticated users will instead be redirected to the login page.



The Featured News section contains the latest general financial news. A user is able to click the left and right arrows to scroll through the carousel to view different articles. The carousel will also scroll automatically and clicking on an article will open the news article in a new tab.

About Us

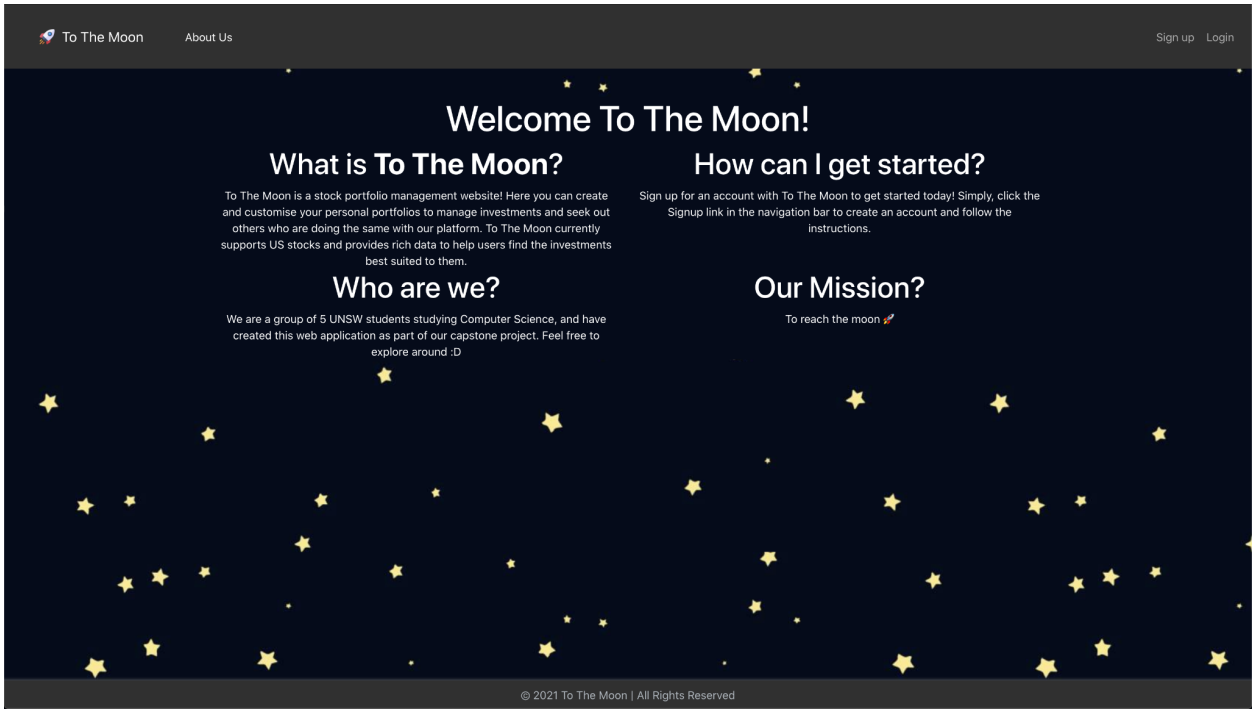
Description

A brief page to give the user contextual information on To The Moon and its makers.

This feature satisfies requirements pertaining to objective(s):

- **F. Opportunities to find new potential investments**

Exposition



Clicking on *About Us* in the navigation bar will bring the user to a page where they can learn about our product, how they can get started, and who we are.

Authentication

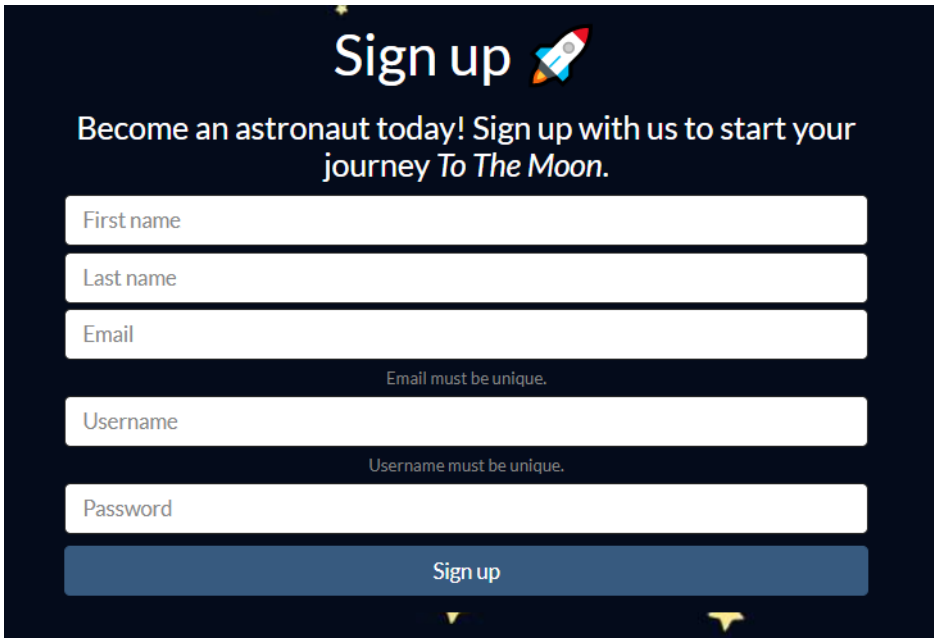
Description

Users are able to create accounts to track their activity on the platform, using their registered email and password as handles for authentication.

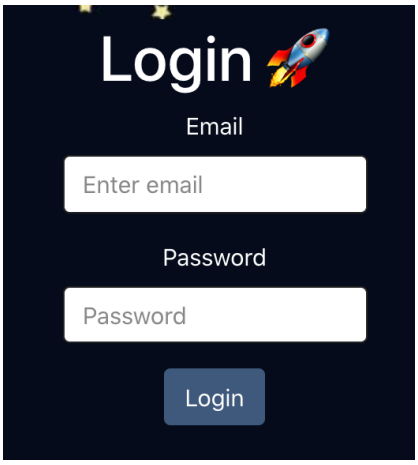
This feature satisfies requirements pertaining to objective(s):

- **A. User profiles**

Exposition



Clicking *Sign up* in the navigation bar will redirect users to a page where they are able to create an account on our platform. After providing their first and last name, and a valid email, username, and password, they will be signed in and have access to all of To The Moon's features.



To log in to an existing account, users can click *Login* in the navigation bar (top right). Upon providing their account email and password, they will be redirected to the Landing Page.

Portfolio Management

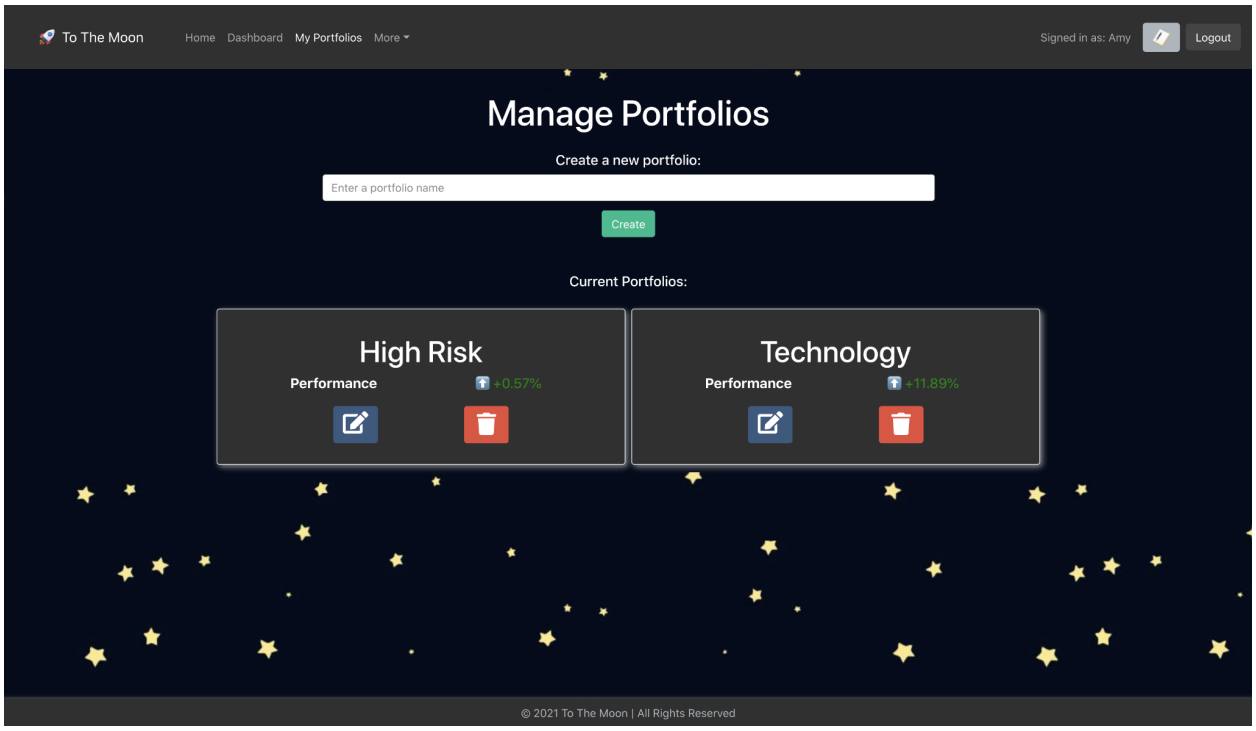
Description

Portfolios are collections of investments that authenticated users can create, view, edit and delete. All portfolios are individual, unless published publicly as a watchlist.

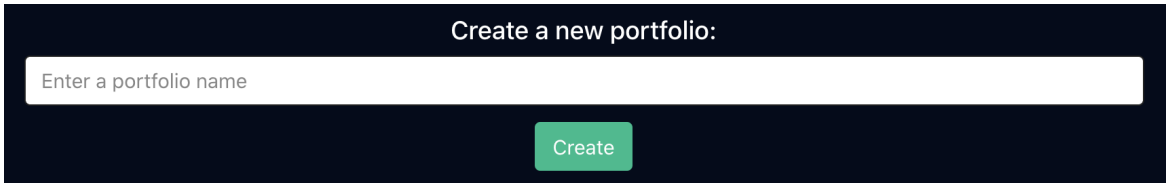
This feature satisfies requirements pertaining to objective(s):

- **F. Opportunities to find new potential investments**

Exposition

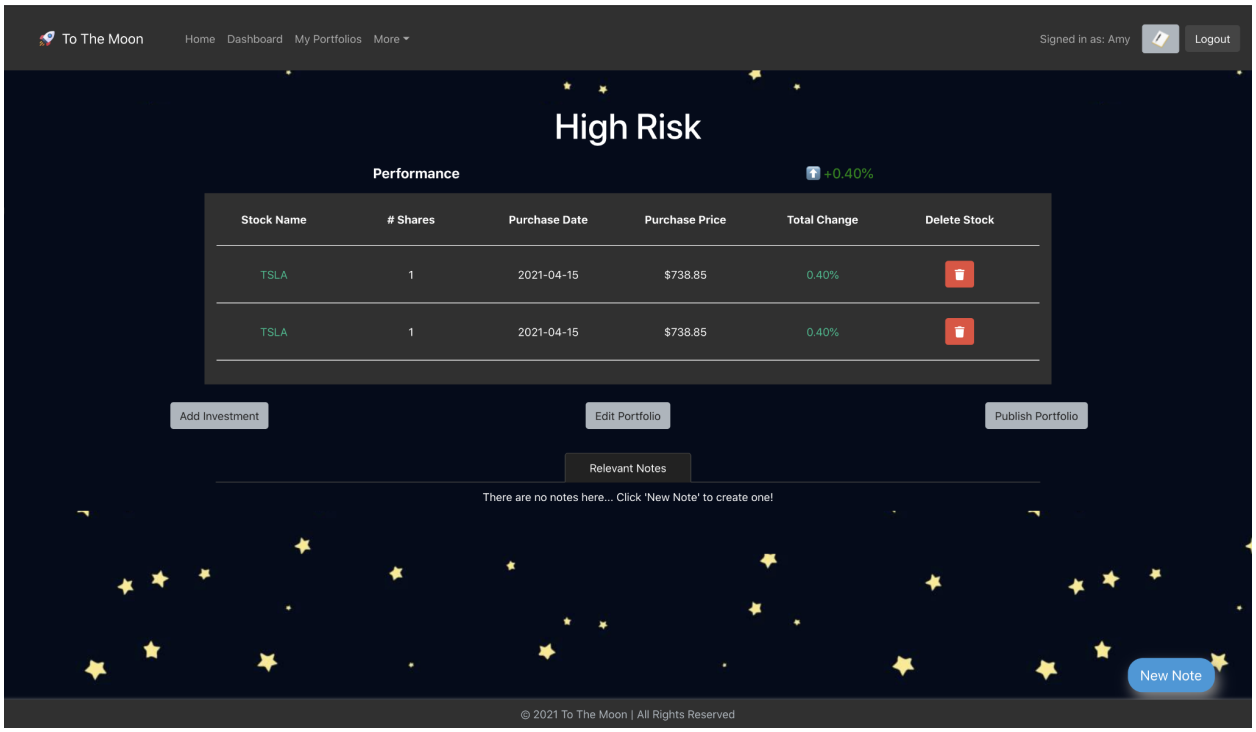


To manage stock portfolios, users can click *My Portfolios* in the navigation bar to be directed to the relevant page.

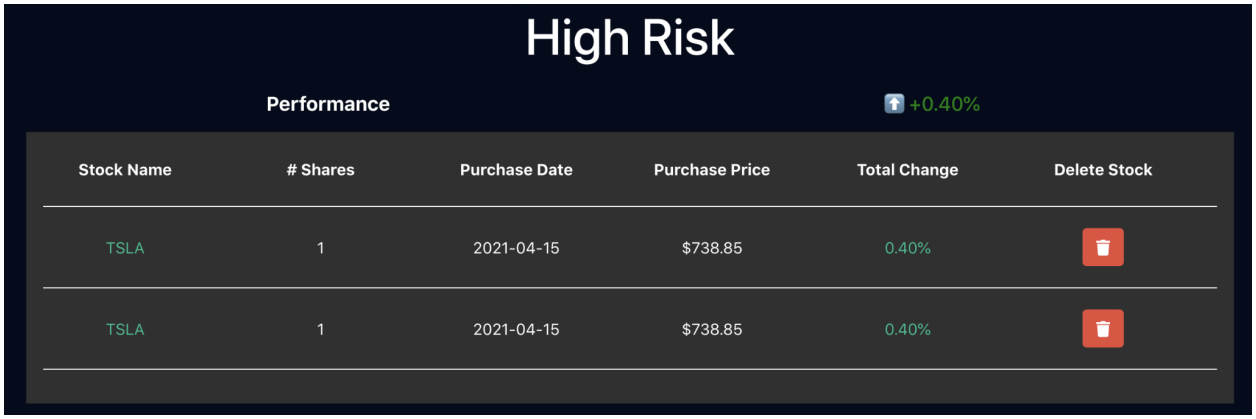


Here, users can add new portfolios by providing a name and clicking ‘Create’. The newly created portfolio will be empty.

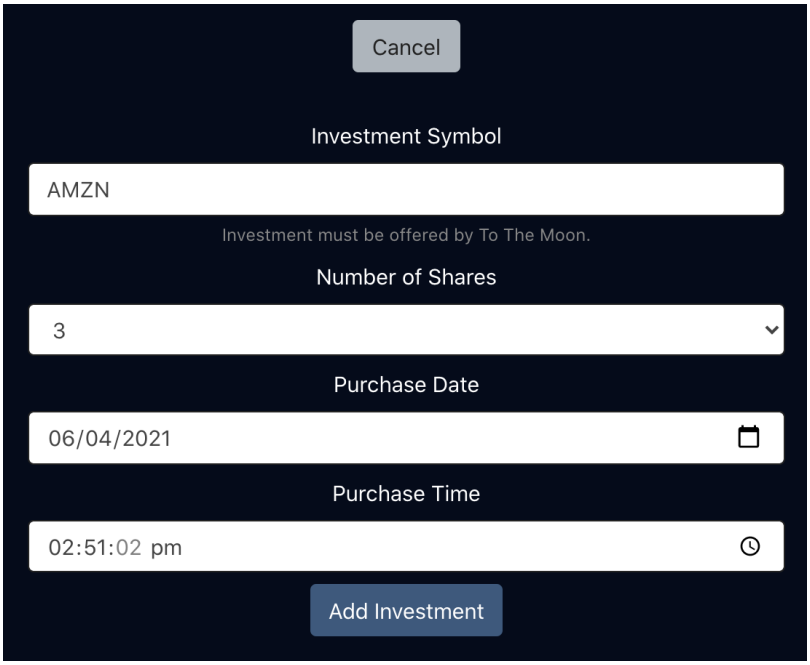
Users can click the (blue) edit icon to view a portfolio's details which will allow them to further edit the portfolio's name or add investments to it. Clicking the (red) trash icon will also remove unwanted portfolios.



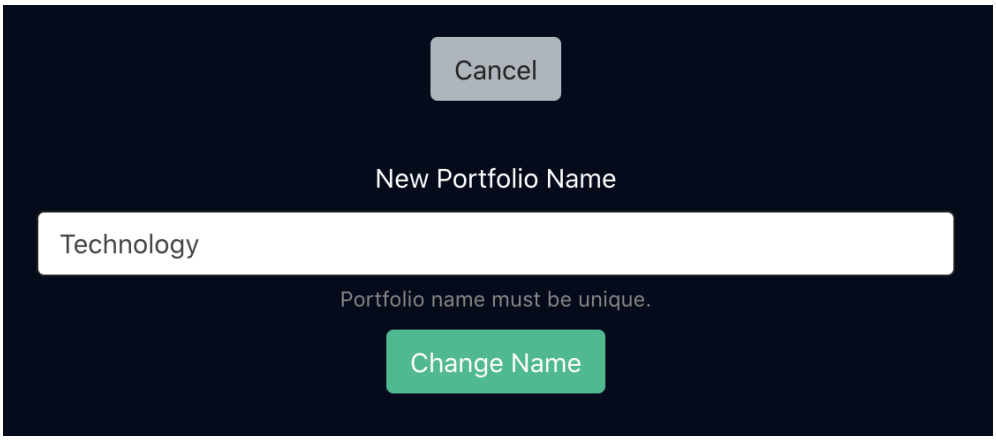
Clicking the edit icon will direct the user to the portfolio's page where they will be able to see the details of each investment.



Each row in the table indicates separate purchases under a particular stock name, the amount of shares, the date purchased and their purchase price, and the total change from value at purchase. Clicking the (red) trash icon on the right will remove unwanted investments.



Clicking the ‘Add Investment’ button produces a form where users can input a stock symbol, number of shares and a purchase date/time to add an investment to the portfolio. If the user wants to record a past purchase, they are able to alter the date of purchase and purchase time to add share purchases at past prices.



Clicking the ‘Edit Portfolio’ button produces a form where users can input a name to change the name of the current portfolio. The name must be unique within the user’s portfolios.

Watchlists

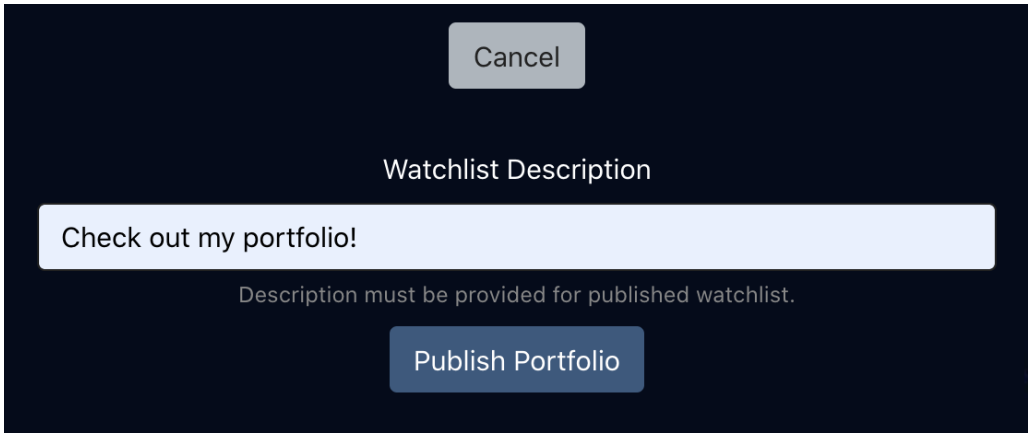
Description

Watchlists are curated lists of investments which users are able to publish via their portfolios. Published watchlists enable users to follow their progress and facilitate user interaction and knowledge sharing on the platform.

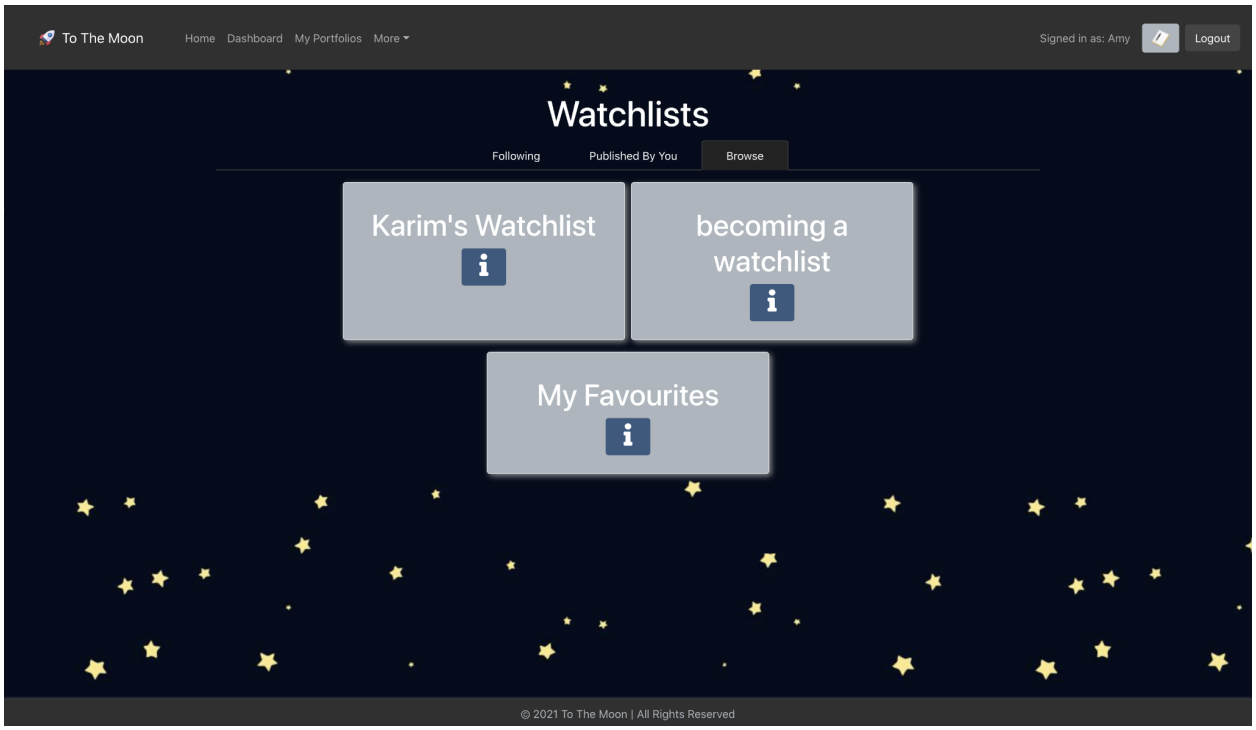
This feature satisfies requirements pertaining to objective(s):

- **H. Social interaction**
- **F. Opportunities to find new potential investments**

Exposition



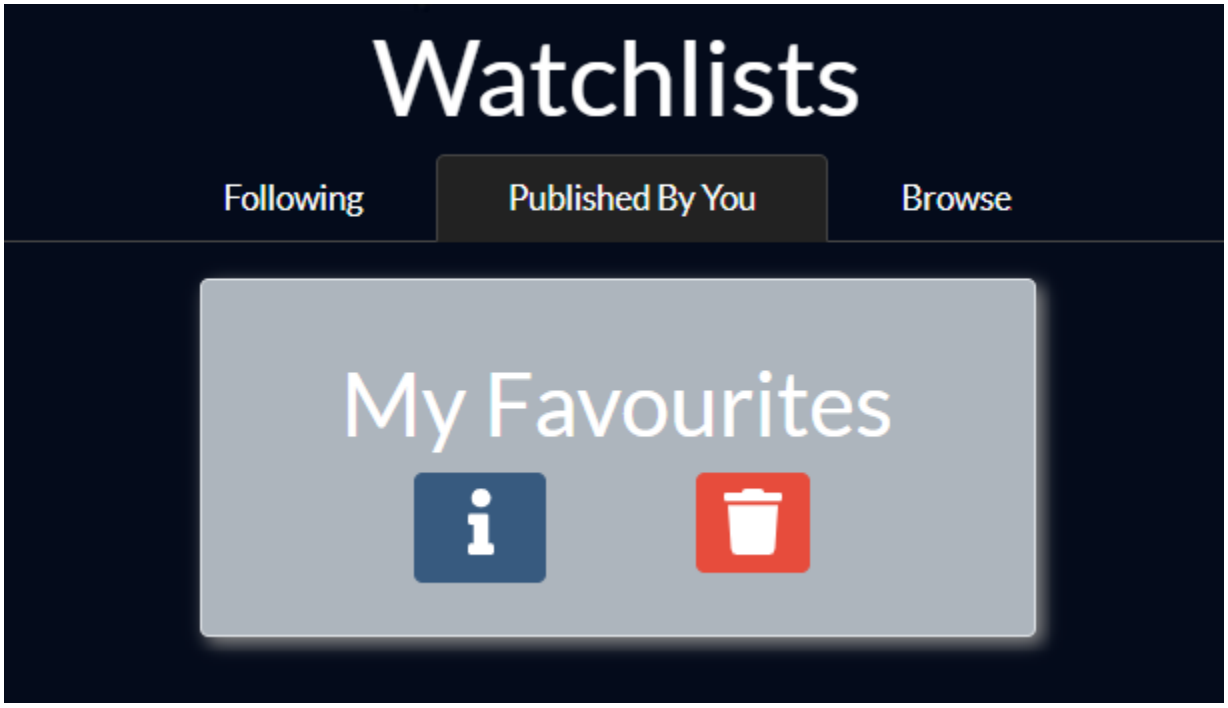
Users can also publish their portfolios to make them watchlists, which other users are able to view and follow on the ‘Watchlists’ page. To publish a portfolio, users click the ‘Publish Portfolio’ button in the bottom right of a portfolio’s page. Once clicked, the user will be asked to input a description for their watchlist in the form before publishing it by clicking ‘Publish Portfolio’ again.



Published portfolios are immediately added to the 'Watchlists' page where all users can browse them. Users can navigate to this page by clicking *Watchlists* under the *More* dropdown in the navigation bar. The 'Browse' tab displays all public watchlists that can be followed by any authenticated user on the platform.

Clicking the (blue) information icon will direct the user to the page for the chosen watchlist. Watchlists aggregate the stock positions from the original portfolio and save only the stocks and

their proportion of the portfolio. When viewing watchlists, we also provide summary information about the company.



Users also have the option of deleting their published watchlists by clicking the (red) trash icon on the ‘Published By You’ section of the watchlist page. This will unpublish the portfolio, but it will remain in the user’s portfolio section.

Stock Information

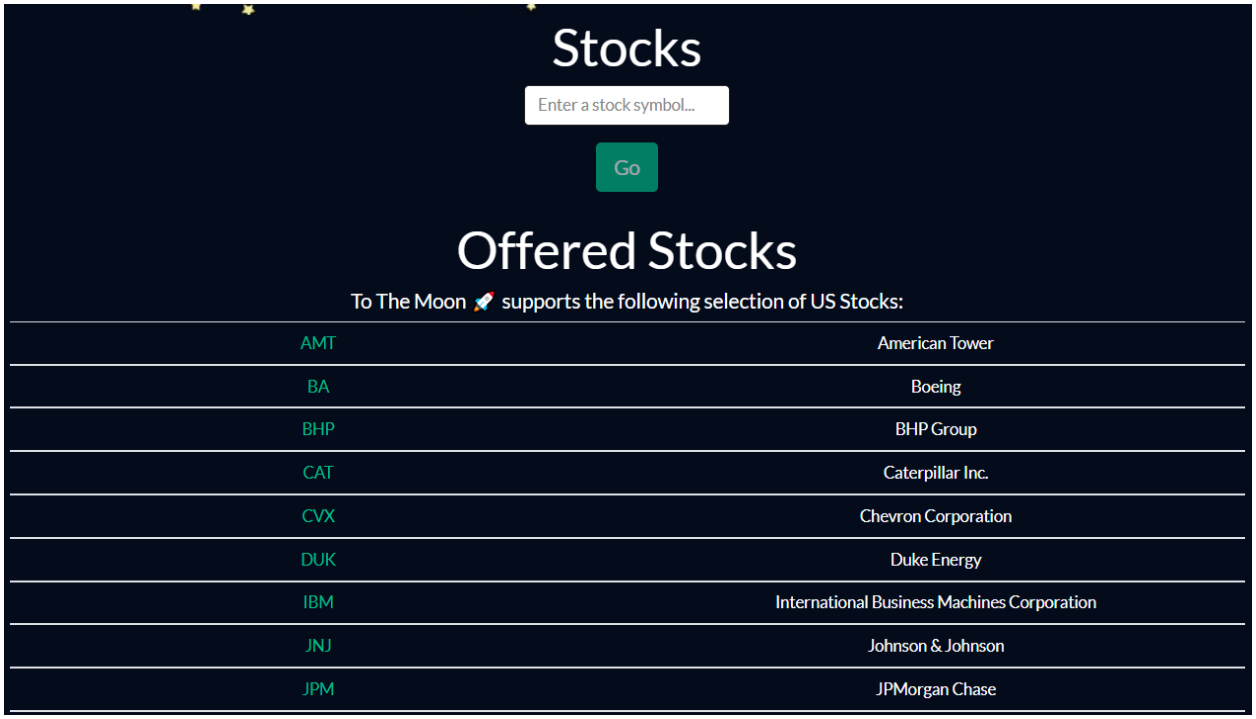
Description

The stock information feature allows users to browse information about stocks that are currently offered. The page for a stock shows rich information about a company's financial data, share price and related news. An interactive graph lets users find the data points they seek by adjusting the time period and zoom.

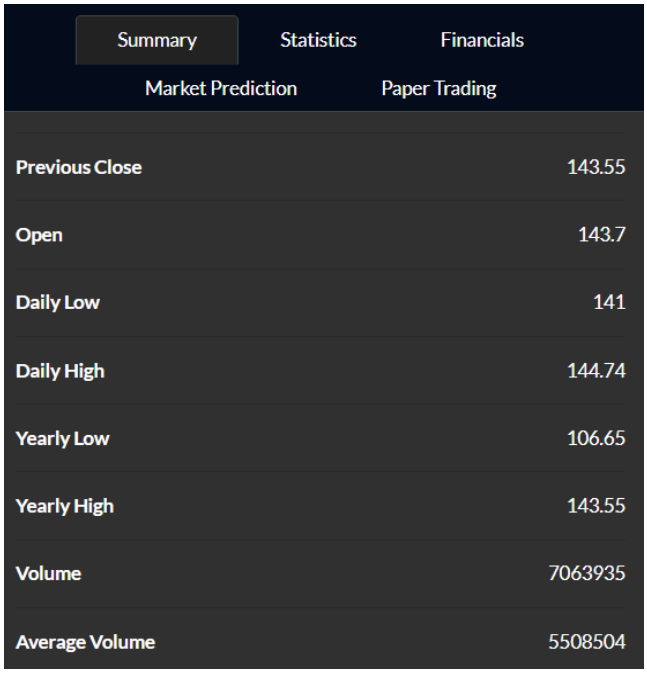
This feature satisfies requirements pertaining to objective(s):

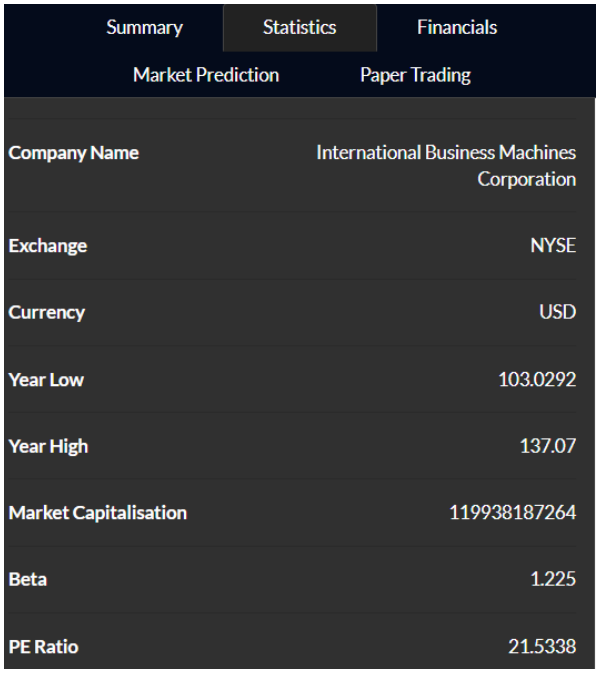
- **C. Visualisation of share information**
- **D. Stock information**
- **E. Investment news and current affairs**

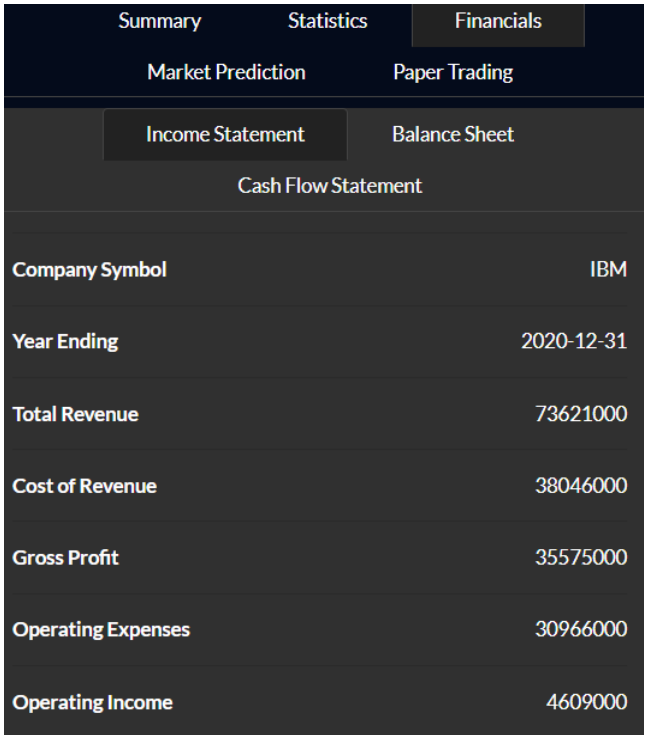
Exposition

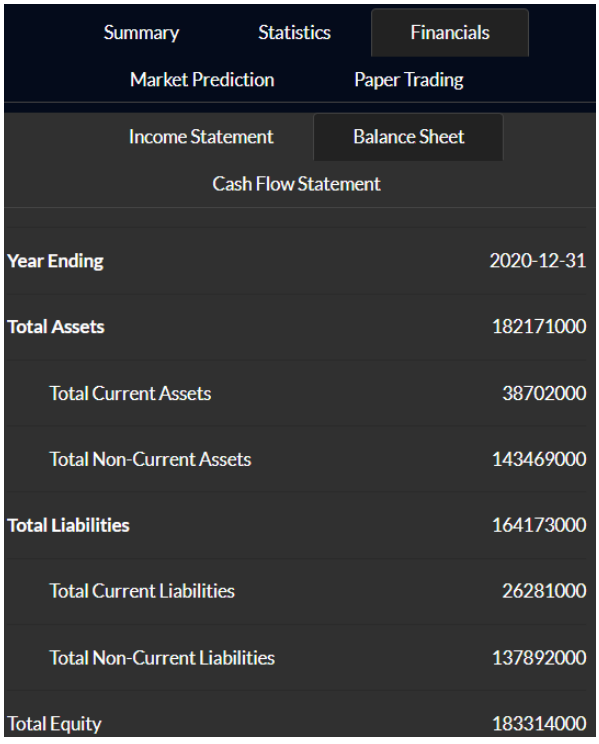


When users are ready to search for potential investments, they may click *Stocks* under *More* in the navigation bar to be brought to this page. Here, they are able to scroll through a complete list of the stocks offered on our platform (full list not shown in image), or they can search for a specific stock.



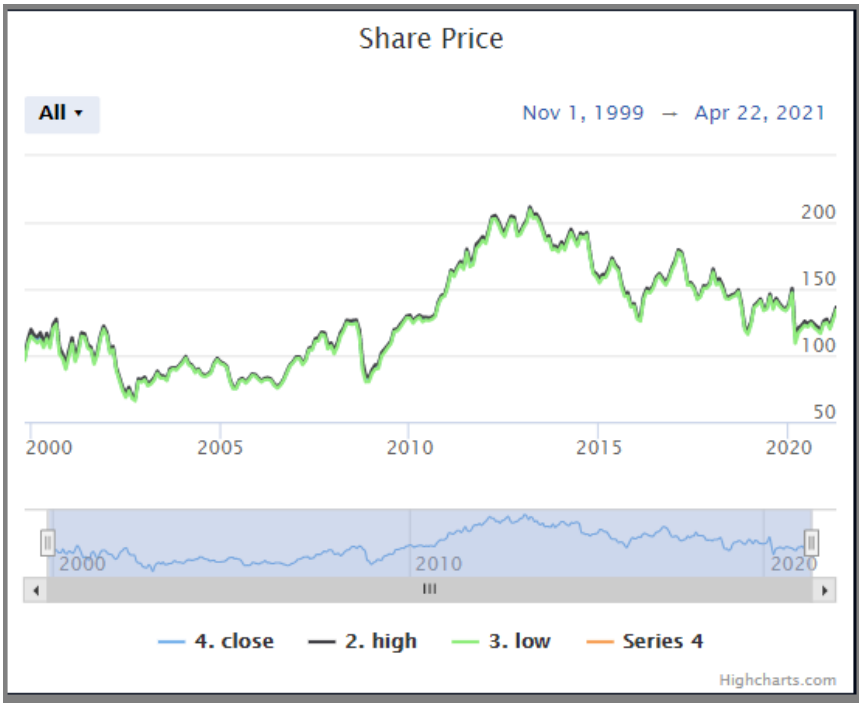




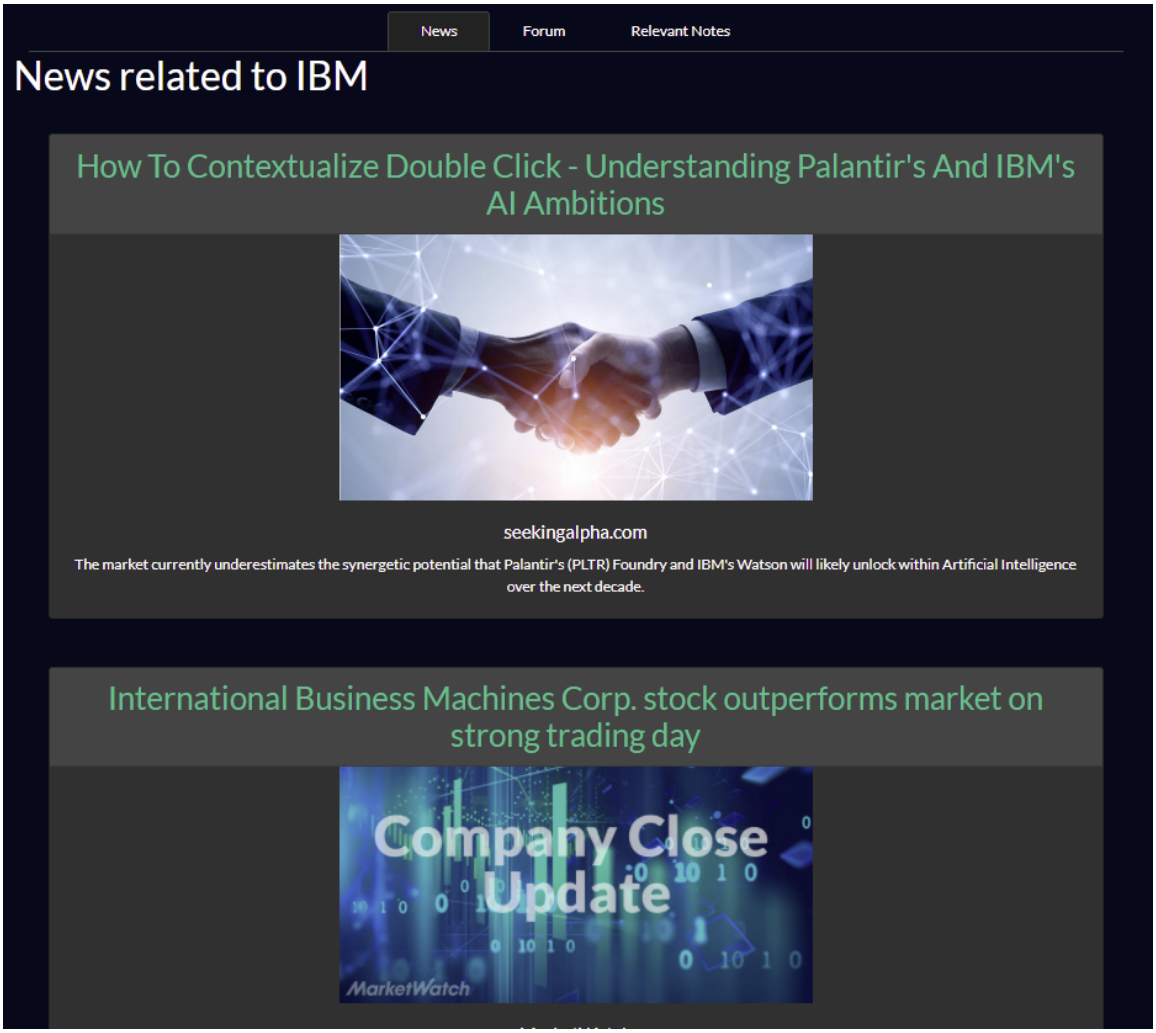


Searching for a supported stock, or clicking on a link to a supported stock will direct the user to the stock's page. On the stock page, a stock information table displays the following sections:

- Summary: assorted trading statistics from the last day/year
- Statistics: overview information and company statistics
- Financials: income statement, balance sheet, and cash flow statement for the company



Stock pages also contain an interactive graph visualisation of the share price. Users can change the time period by selecting a timeframe from the button on the top left (shown as ‘All’ in the image). The graph offers 9 timeframes, from 1 day to all time. Alternatively, users can adjust the slider at the bottom to select any trading period they like. Share price graphs display the ‘closing price’, ‘daily high price’, and ‘daily low price’. The ‘Series 4’ key is used to present the Market Prediction feature, described and exposed in its relevant section.



At the bottom of the page, users can also browse recent news related to the company. News articles are updated in real time, so users can discover new, informative articles about the company that they may make an impact on the user's trading decisions.

Screeners

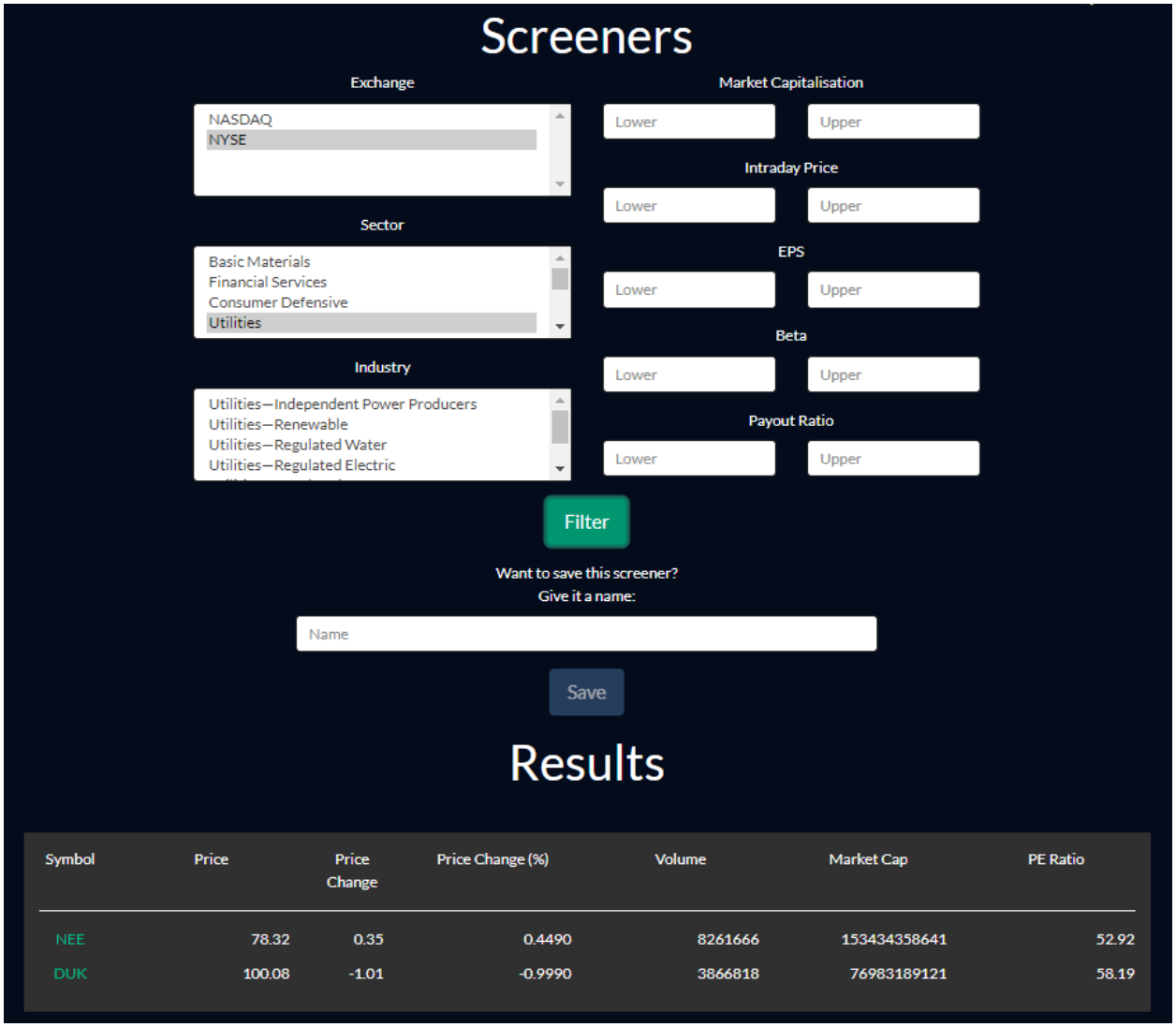
Description

Screeners offer advanced search of stocks on the NYSE and NASDAQ. Users can select parameters they wish to filter by and receive the results of their search, which can then be saved and loaded at a later time.

This feature satisfies requirements pertaining to objective(s):

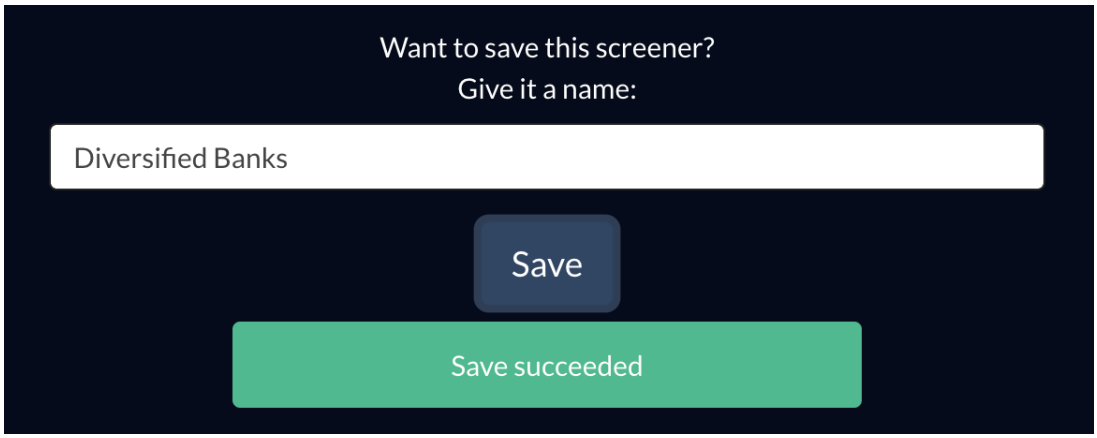
- **D. Stock information**
- **F. Opportunities to find new potential investments**

Exposition

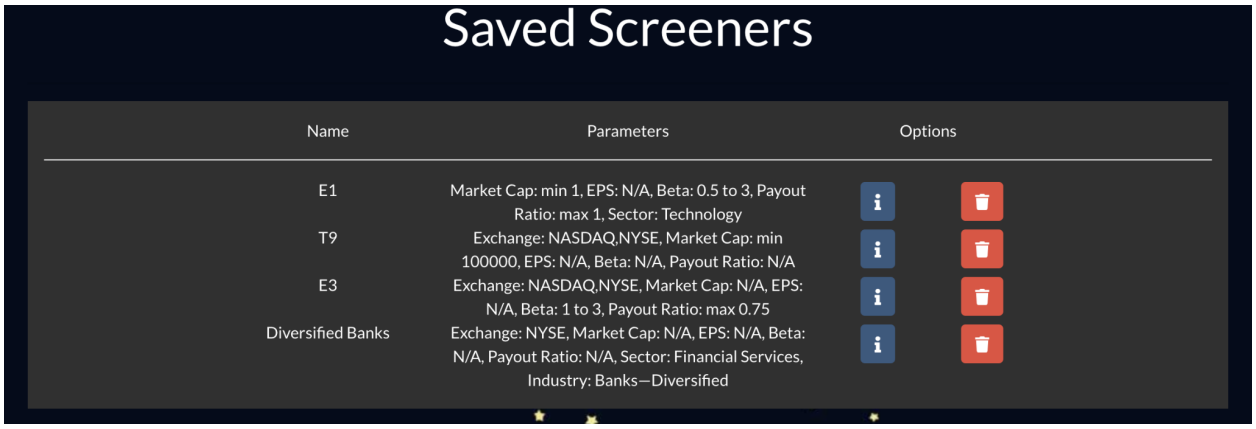


Users can filter through the stocks offered on our platform by clicking on *Screeners* under *More* in the navigation bar. Here, they can select an exchange (currently we offer only NYSE stocks,

but have added exchange screening for scalability), a sector, any of the sector's industries (we added all sector industries for scalability purposes, however we currently do not have a company for each industry), and assorted company statistics (optionally giving a lower and/or upper limit) to filter by. Any stocks that match these given parameters will be displayed in the results section.



To The Moon also allows users to save and load screeners. Saving the current screener is as simple as adding a name to the screener, and clicking 'Save'.



The screener will be immediately added to the 'Saved Screeners' list. To load a screener, users can click the (blue) information icon next to the screener they would like to load in. Deleting a screener is done similarly by clicking the (red) trash icon next to the unwanted screener.

Email Notifications

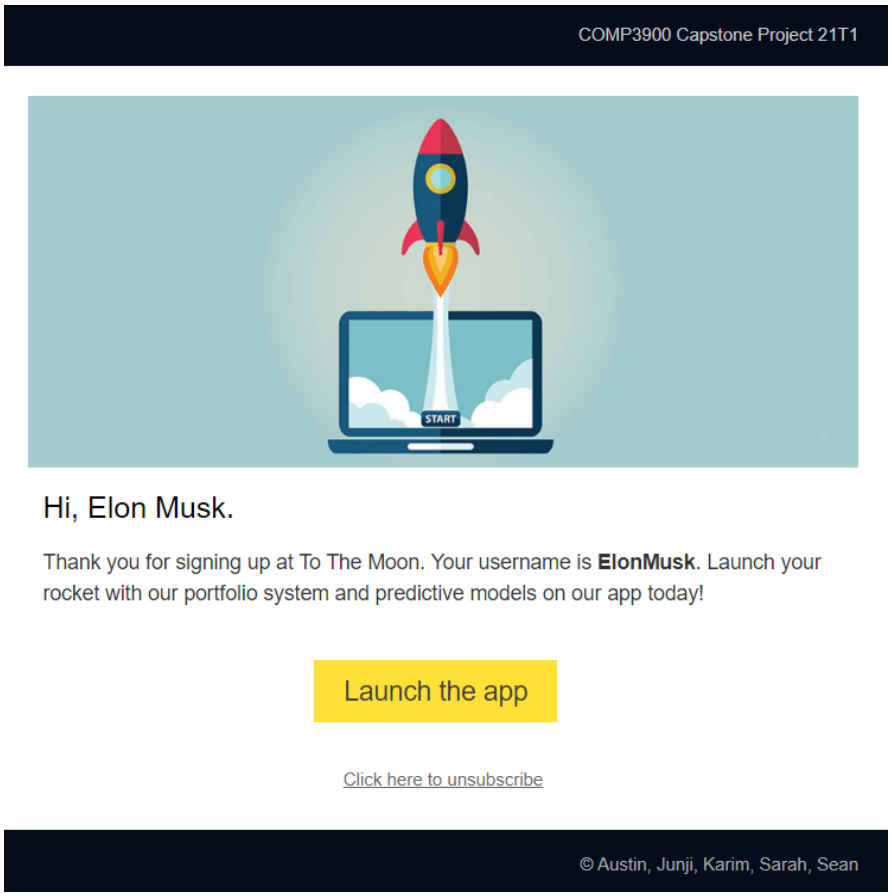
Description

To The Moon offers email notifications to provide users with updates on their portfolios and relevant financial news. Any replies to their comments within forum sections are also notified via email. Notifications are also sent on sign up, and users can opt out at any time.

This feature satisfies requirements pertaining to objective(s):

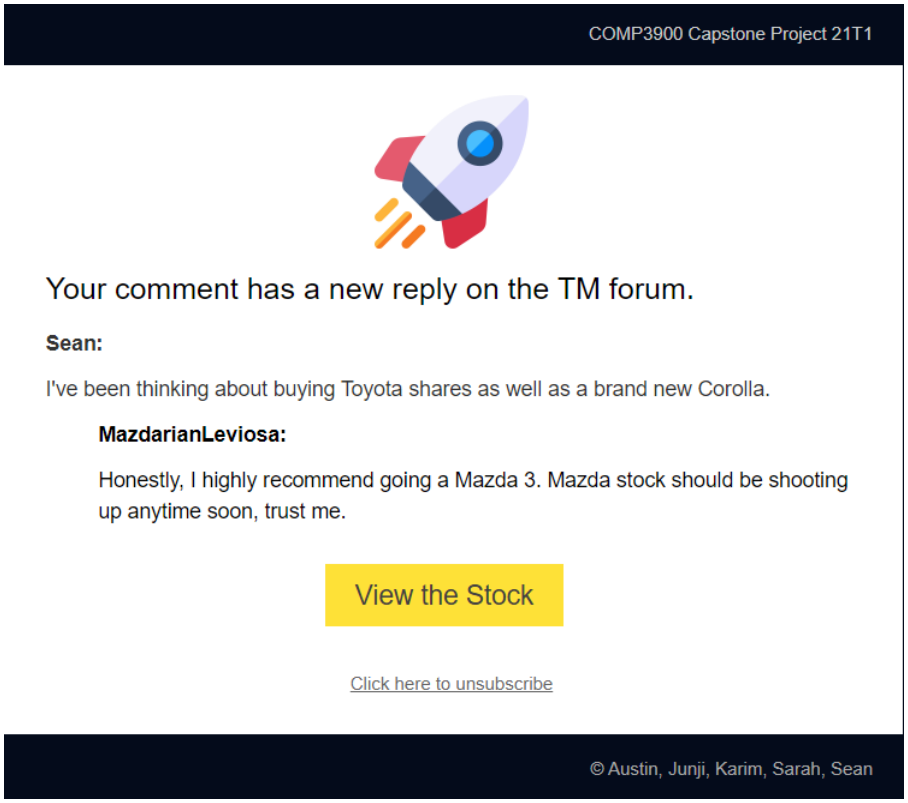
- **B. Portfolio management**
- **E. Investment news and current affairs**
- **H. Social interaction**

Exposition



When a user registers, they will be sent an email notification to signify to the user that their name and username details have been registered to the application. The email also contains a shortcut button back to the application as well as a link offering them a potential unsubscription from all notification services the application offers.

By unsubscribing the user will no longer receive any potential forum notifications or daily portfolio and news performance summaries. When a user clicks the link, it will result in a backend (frontend detached) screen with a potential success or error message.



When a user posts a comment to a stock, the comment opens up the potential for replies. When the parent comment user has notifications on, any reply to their comment allows for the application to send them a notification detailing their original comment and the new reply. The user is also greeted with a link to the specific stock and an unsubscription link if they wish to opt out of notifications.

Your Portfolio Performance

becoming a watchlist soon:

1x **IBM** share(s) with a total change of: **+5.76%**

Total Portfolio Performance Change: +5.76%

Tech:

1x **IBM** share(s) with a total change of: **+2.26%**

1x **ORCL** share(s) with a total change of: **-5.51%**

Total Portfolio Performance Change: -0.57%

Technology and Energy:

1x **ORCL** share(s) with a total change of: **-5.51%**

1x **IBM** share(s) with a total change of: **+2.26%**

1x **NEE** share(s) with a total change of: **-2.74%**

Total Portfolio Performance Change: -1.16%

News articles relevant to your Portfolio

Ford Is The First Legacy Car Manufacturer To Get Its EV Strategy Right

Ford seems to have found the right formula in its EV policy by relying on its iconic brands as a platform. It is a good long-term investment with a decent dividend in place.

MarketWatch

Dow's 175-point jump led by gains for Dow Inc., IBM stocks

Behind positive gains for shares of Dow Inc. and IBM, the Dow Jones Industrial Average is up Wednesday morning. The Dow is trading 177 points higher (0.5%),...

WSJ News Exclusive | Tesla Crashes Raise Safety Questions in Congress About Driver-Assistance Systems

Two Democratic senators want the top automotive-safety regulator to develop recommendations for improving advanced driver-assistance systems such as Tesla's Autopilot.

View your Portfolios

Click here to unsubscribe

If a user has at least one portfolio with current investments saved, the application will send them an overview of their portfolio performance and news articles relating to the stocks in their overall portfolio. This email notification will be delivered on a daily basis so that the user can stay up to date with the application without the express need to visit it. The email notification contains a button linking them to their portfolios and a link for unsubscribing if they no longer wish to receive email notifications.

Novel Features

Notes

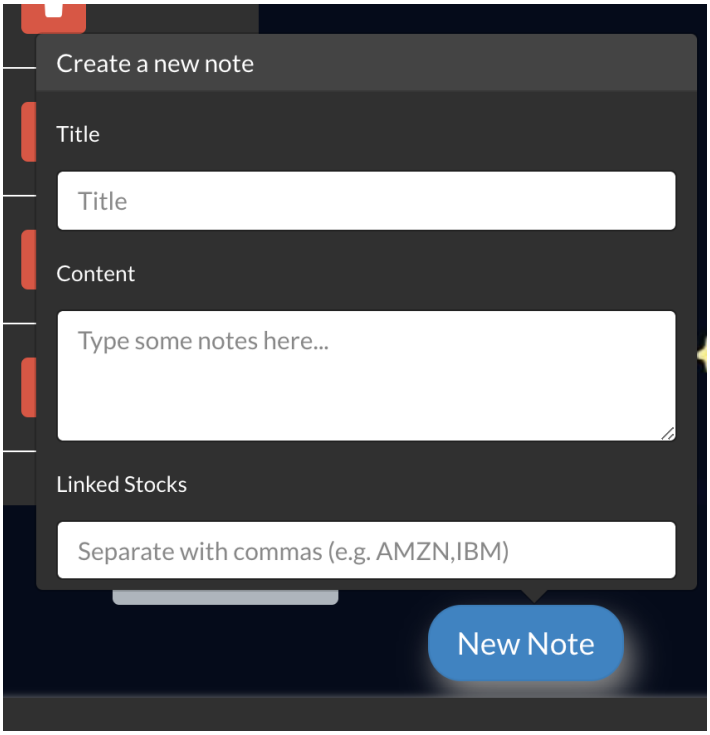
Description

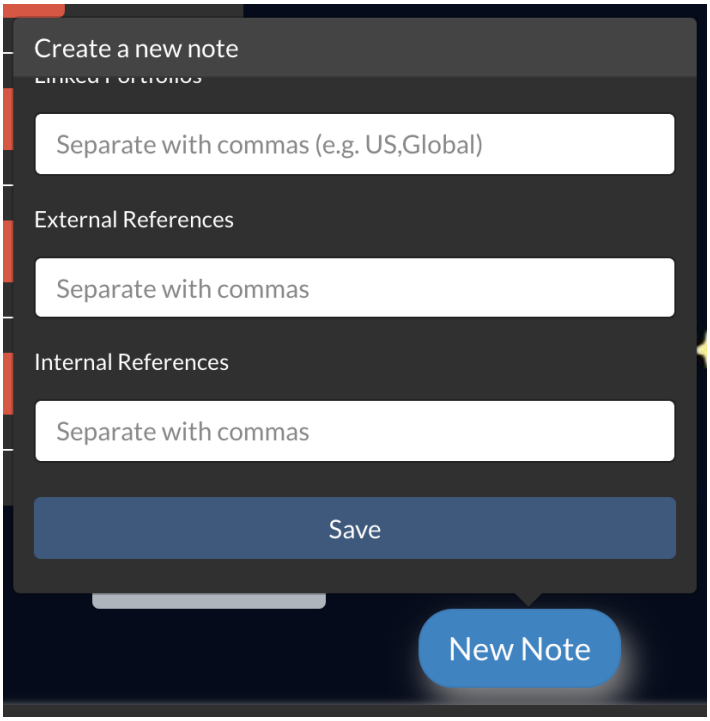
To The Moon allows users to create notes which they can access at any time. Notes can be tagged to any stock page or portfolio on our platform. Notes contain a title, text content, internal references (news articles on our platform), and external references (any articles from outside our platform).

This feature satisfies requirements pertaining to objective(s):

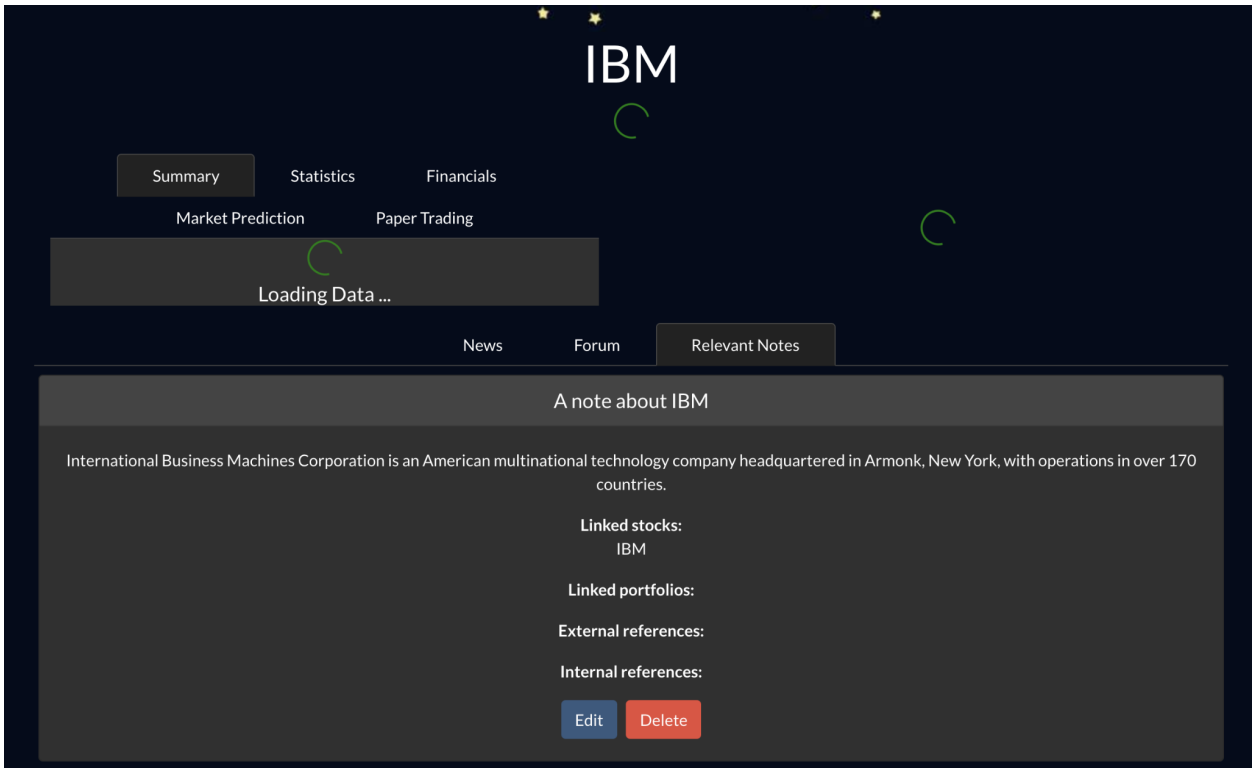
- **F. Opportunities to find new potential investments**

Exposition

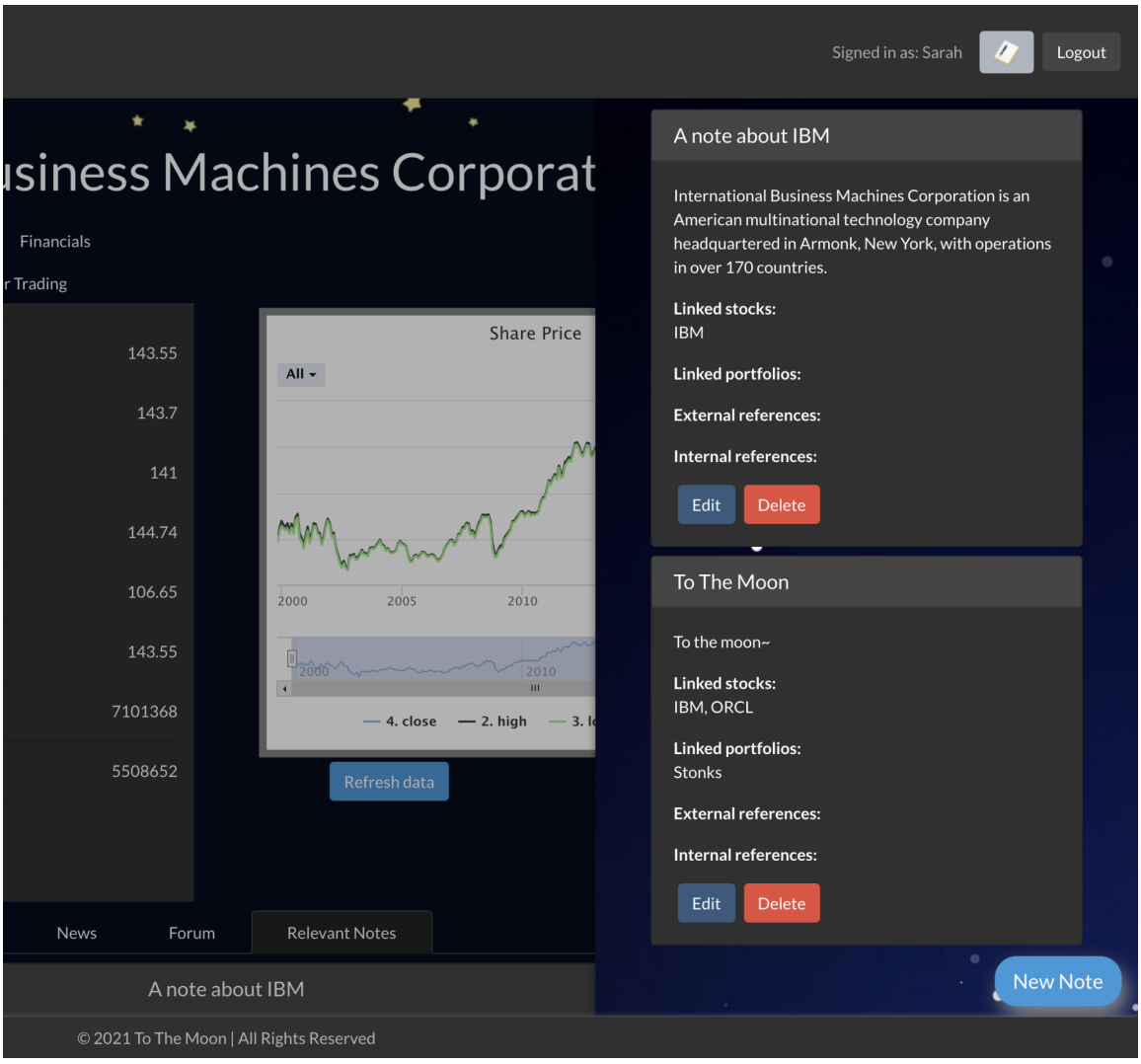


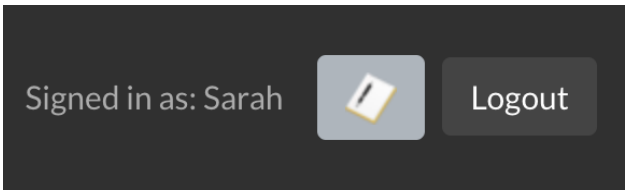


To create a new note, users can click the ‘New Note’ button on the bottom left of any portfolio page or stock page. When clicked, a small popover will appear where the user enters the note’s title, contents, linked stocks and portfolios, and references. Clicking the ‘Save’ button will create the note.



Once a note is created, it will also appear in the ‘Relevant Notes’ sections of any associated stocks and portfolios which have been linked. To edit or delete the note, users can click the ‘Edit’ and ‘Delete’ buttons on each note.





Alternatively, users can use the Notes Pane, accessible from the notepad icon in the navigation bar, besides the ‘Logout’ button available throughout the application. Notes can be viewed, created, edited, and deleted in the pane, and will sync across different contexts.

Social Forum

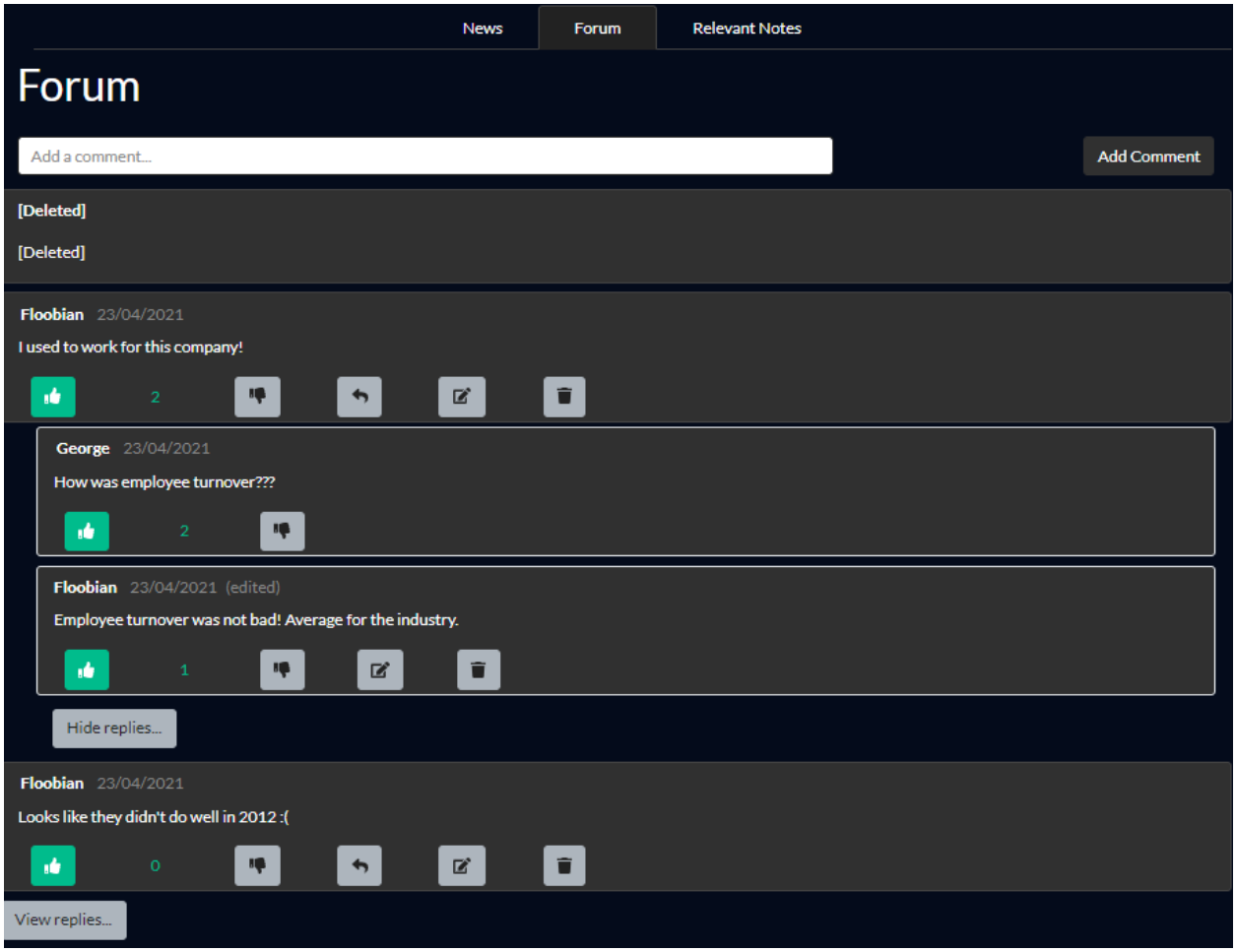
Description

The platform features a forum section within each stock page which allows users to submit comments and reply to other users. To express agreement, users can also ‘upvote’ or ‘downvote’ a comment, to affect its ordering and likability within the section.

This feature satisfies requirements pertaining to objective(s):

- **F. Opportunities to find new potential investments**
- **H. Social interaction**

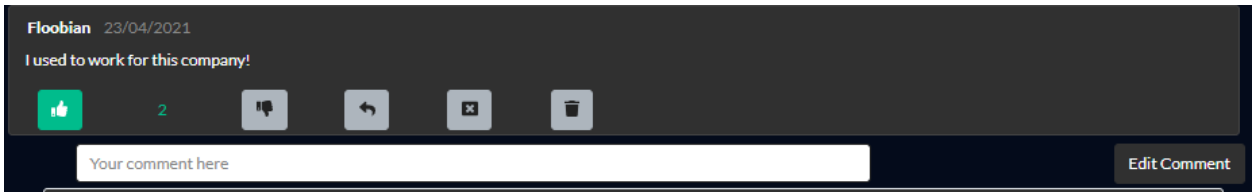
Exposition



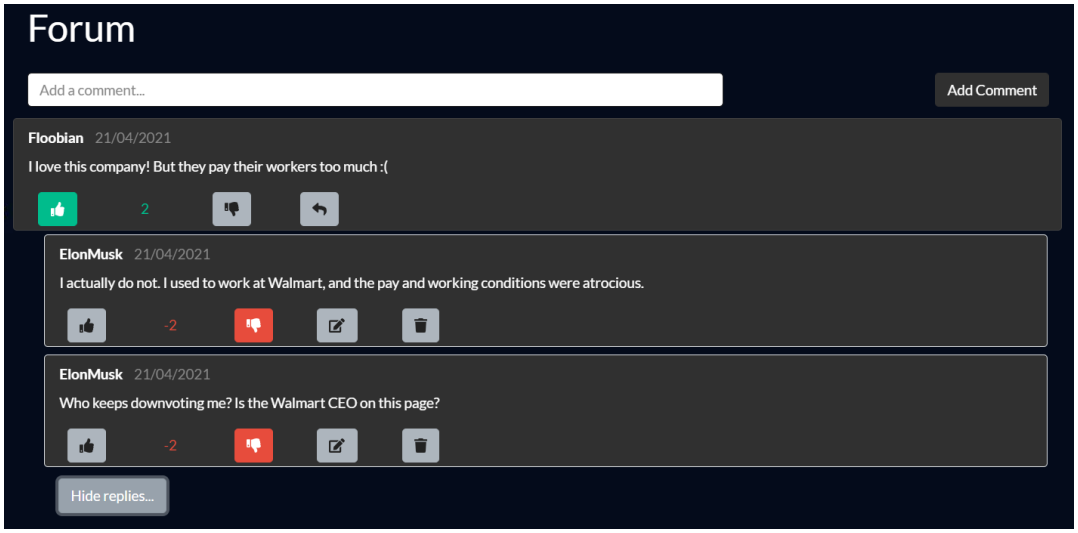
Forum sections are added to all stock pages to provide user discussion. Adding and deleting comments differs for parent and child comments. To add a parent comment users simply fill the add comment section at the top of the forum and click the ‘Add Comment’ button. Users can also delete their own parent comments by pressing the trashcan icon on the parent comment.

Once deleted, parent comment content is no longer available but the comment tab remains so that its children comment(s) can persist.

Adding a child comment is as simple as clicking the reply button on a parent comment, filling in the produced text bar, and pressing the ‘Add Reply’ button. To delete child comments, users press the comments delete button, and the child comment disappears.



Editing comments is performed the same way for parent and child comments. Users press the edit button (the pencil icon button), fill the produced text bar, and press the ‘Edit Comment’ button. Edited comments are given a small ‘(edited)’ tag so that users can see that the commenter has adjusted their comment.



Comment sections have been sorted on the difference between the upvotes and downvotes (comment difference = upvote count - downvote count). With the most positive comments (positive vote difference is displayed in green) being placed at the top of the forum page, and the most negative comments (negative vote difference is displayed in red) being placed at the bottom.

Prediction Models

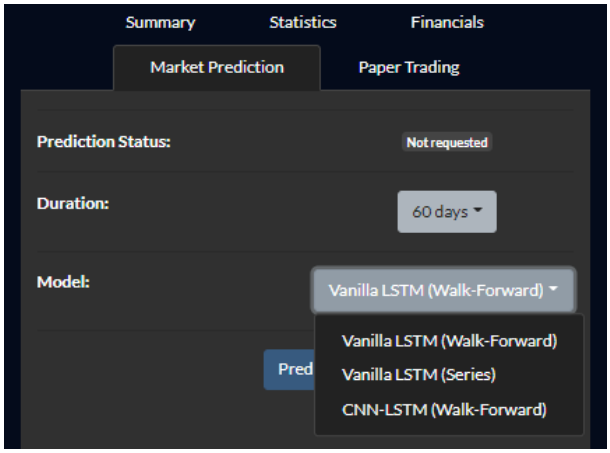
Description

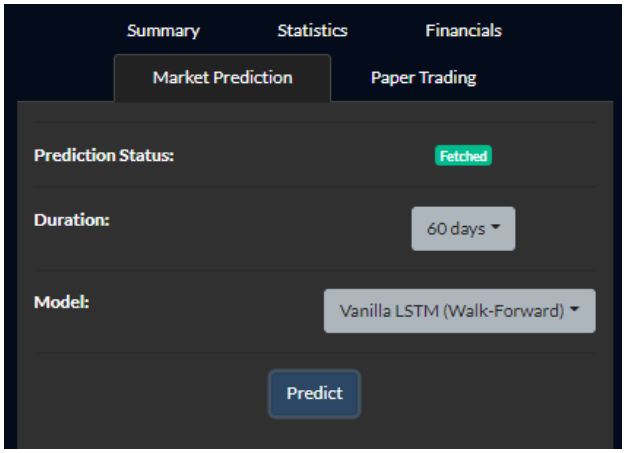
Using machine learning, To The Moon features a prediction system which allows users to receive predictions on the future share price for a selected stock. Users can also select the model to use and the duration of the prediction.

This feature satisfies requirements pertaining to objective(s):

- **C. Visualisation of share information**
- **G. Investment predictions and online trading**

Exposition





Users can generate the machine learning share price predictions for a chosen company by clicking the ‘Market Prediction’ tab in the stock’s page. Initially, ‘Prediction Status’ is ‘Not requested’. Users select a duration (6 durations are available, spanning from as little as 3 days to as much as 60 days), and a machine learning model (models included in snapshot), press the ‘Predict’ button, and a prediction will be generated.

When prediction has been generated, the ‘Prediction Status’ field in the ‘Market Prediction’ tab will change to ‘Fetched’, and the prediction outcome will be plotted on the stock graph for viewing.

Automated Backtesting

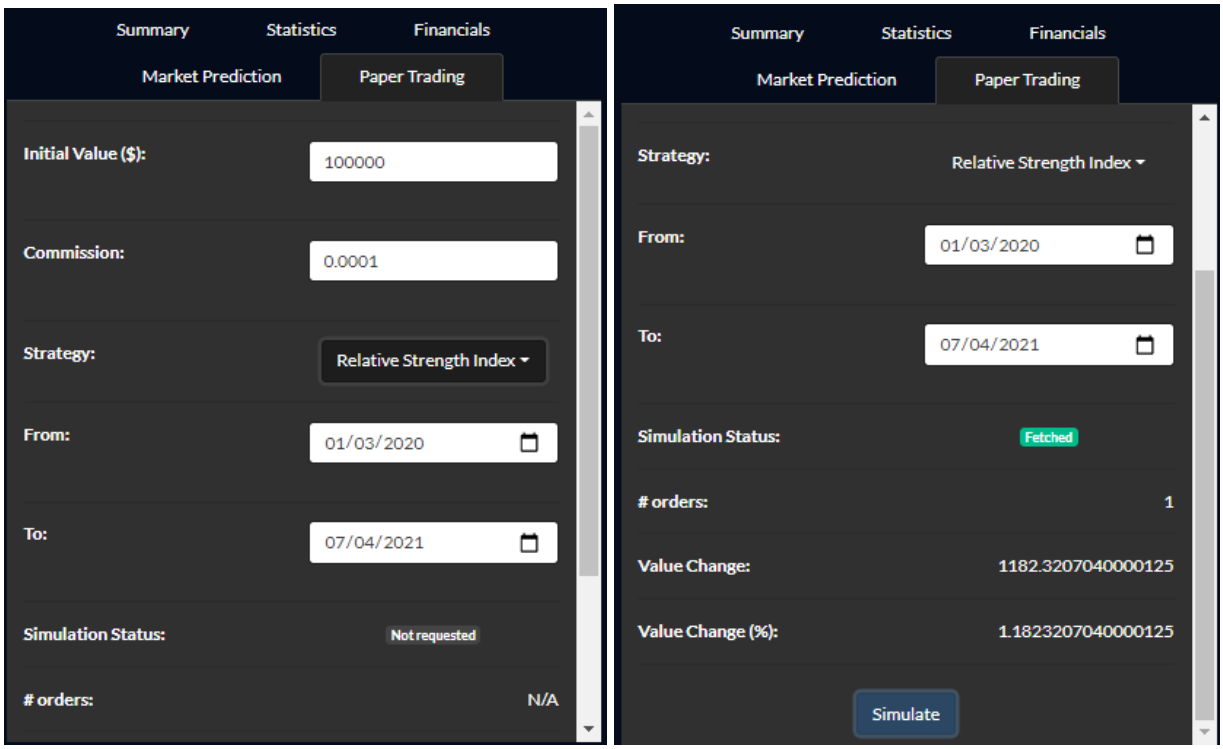
Description

A trading backtest system allows users to test their trading strategies on the share price data of a selected stock. Users can select strategies to use in the simulation, parameters such as initial portfolio value, test dates and commission amount, and request a simulation using the provided settings.

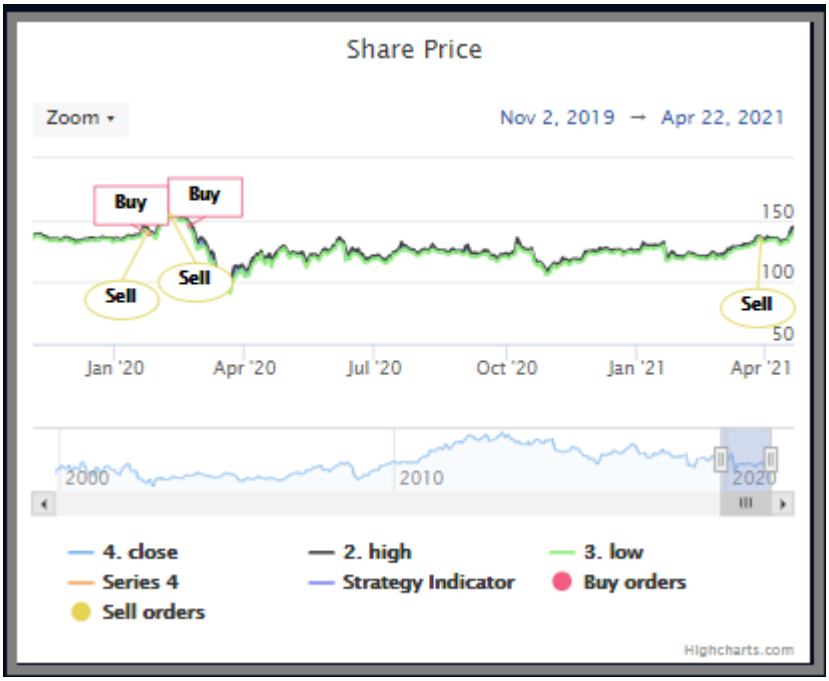
This feature satisfies requirements pertaining to objective(s):

- **G. Investment predictions and online trading**

Exposition



Users can check the outcome of a particular strategy on a particular stock by navigating to the ‘Paper Trading’ tab on a stock’s page. Initially the ‘Simulation Status’ field is set to ‘Not Requested’. After entering their initial investment amount, their brokers commission, a chosen strategy (currently our system offers strategies: Relative Strength Index, Simple Moving Average Crossover, and Simple Moving Average Crossover with Bollinger Bands), and a timeframe, users press the ‘Simulate’ button to receive the strategy’s results. When the simulation is complete, the ‘Simulation Status’ field is set to ‘Fetched’, the number of simulated trades is returned (‘# orders’ field), and the change in their initial investment amount is displayed (given in whole dollars in the ‘Value Change’ field, and as a percentage in the ’Value Change (%)’ field).



After a simulation is complete, the simulated trades (both the buy and sell orders) are displayed on the company's share price graph.

Dashboards

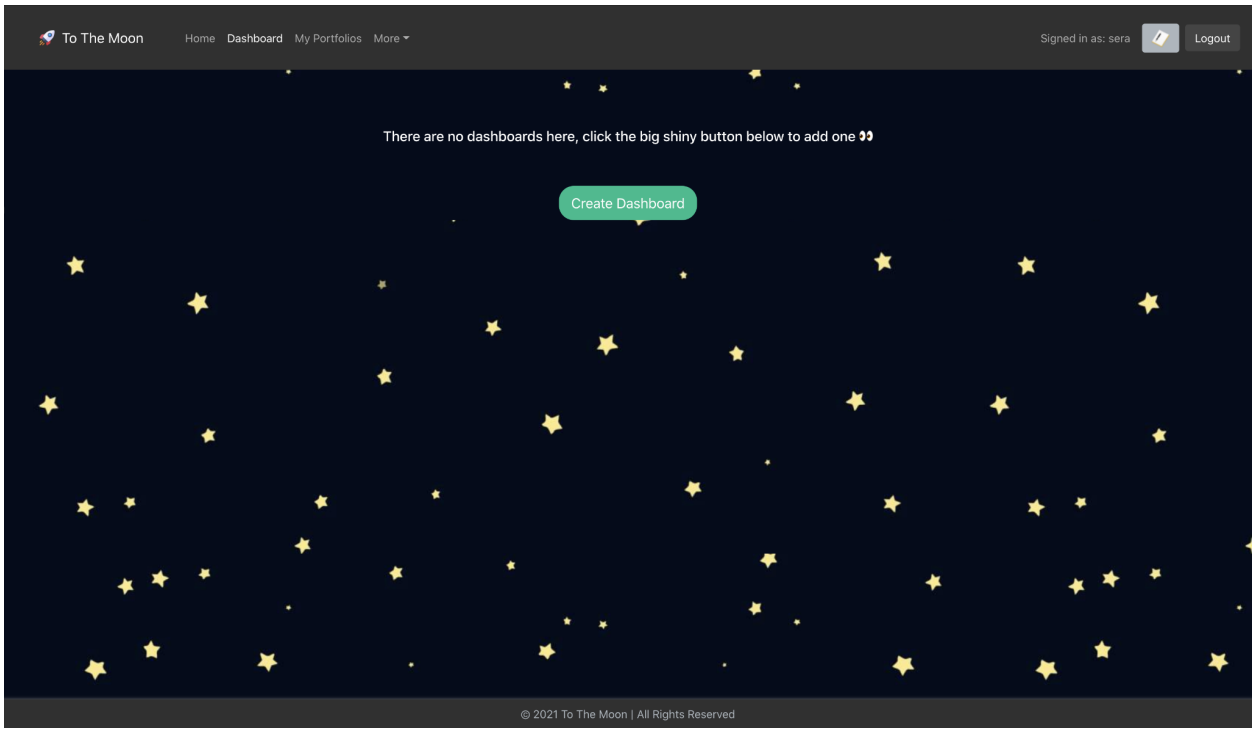
Description

Dashboards allow users to track the content they care about. The 4-panel layout is designed to fill one screen with the content a user is most interested in. Each panel can be customised with either a portfolio (summary or detailed view), news on a chosen stock, or the share price graph of a chosen stock. Dashboard panels can be cleared and refilled with any of the aforementioned options.

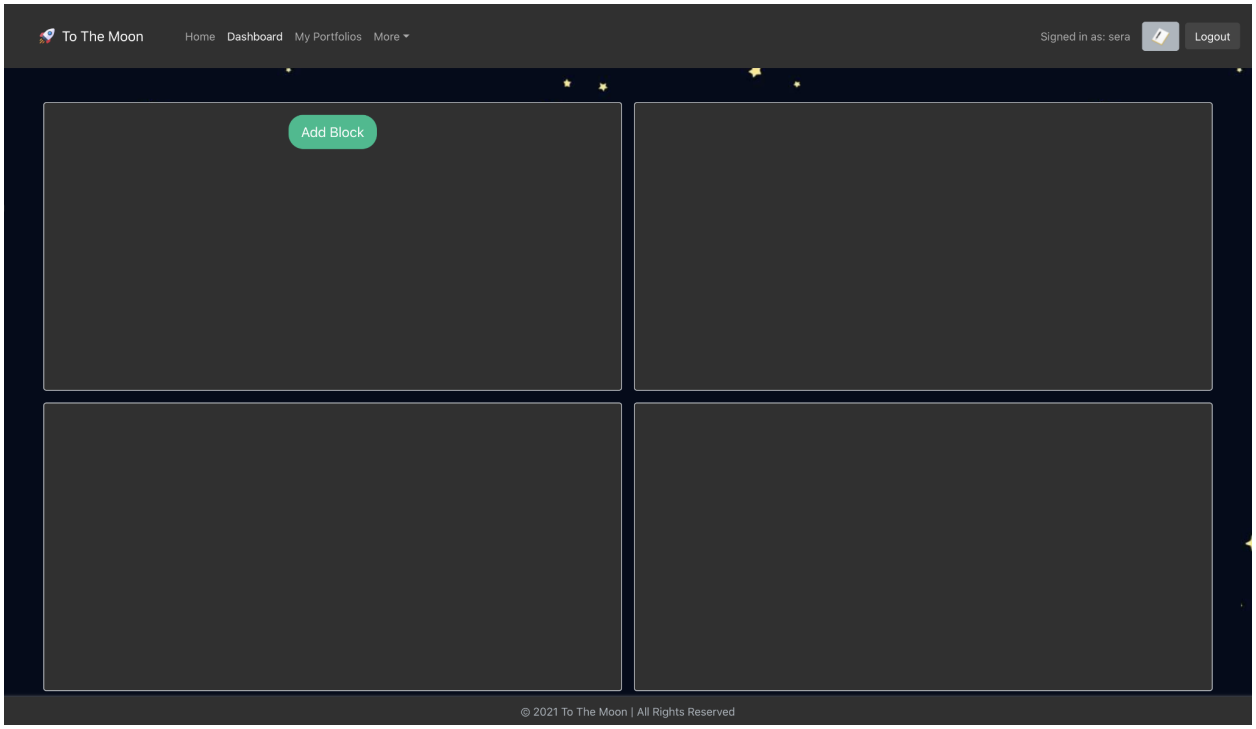
This feature satisfies requirements pertaining to objective(s):

- **B. Portfolio management**
- **C. Visualisation of share information**
- **E. Investment news and current affairs**
- **I. Customisable dashboard**

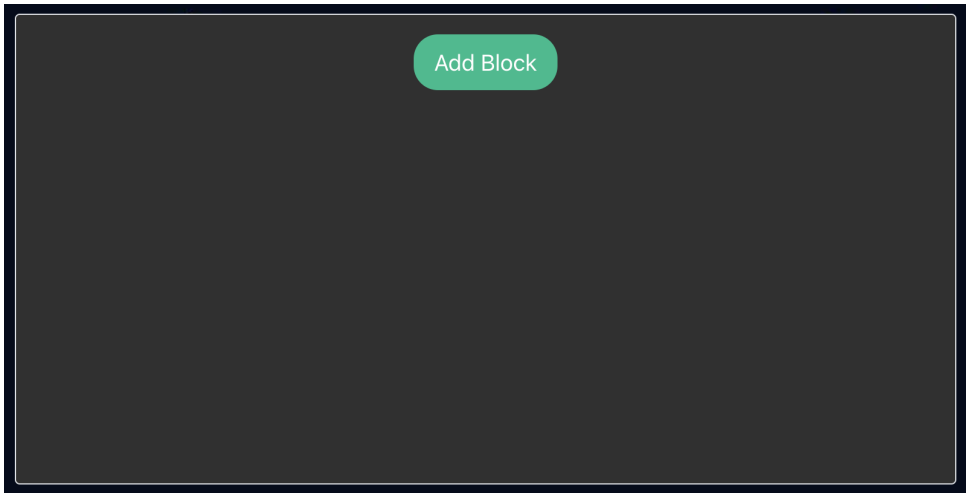
Exposition



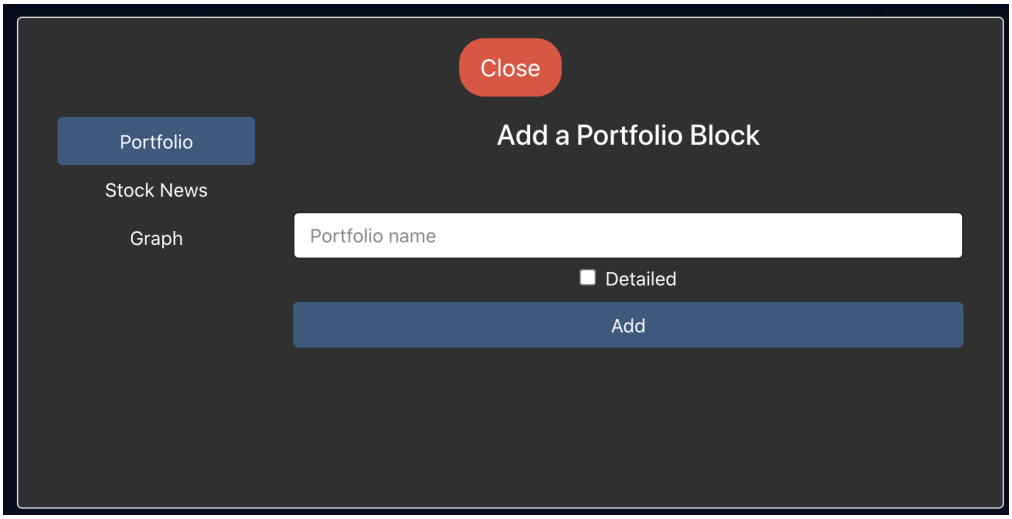
Clicking *Dashboard* in the navigation bar brings users to the dashboard page. For users who have not yet created a dashboard, clicking the ‘Create Dashboard’ button will initialise their dashboard.



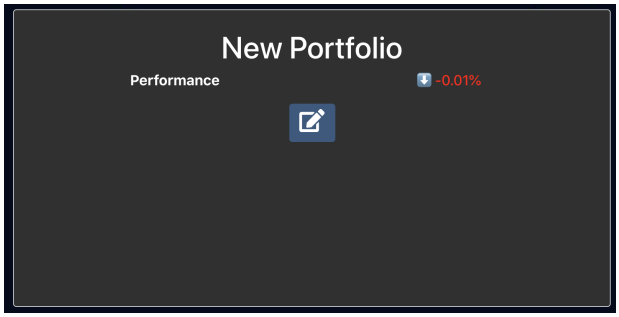
An empty dashboard can be customised by adding content to each of the 4 panels. Each panel's content will be displayed separately.



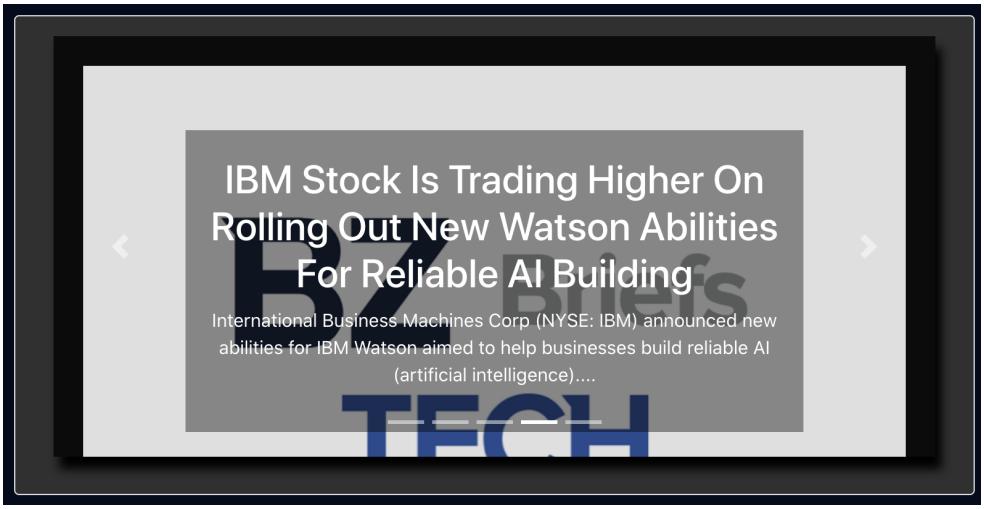
To add content to a panel, click the 'Add Block' button.



Users can choose from 3 options on the left: Portfolio, Stock News or Graph. Each option will display its required parameters in a form on the right. Upon entering the required parameters, the block will be created.



Given a portfolio name, users can create portfolio panels which track the performance of their own portfolios. If ‘Detailed’ is checked in the form to create a portfolio block, the block will also show the details of its investments, similar to the view on a portfolio’s page. Clicking the (blue) edit icon also redirects users to the page for the portfolio, where they have access to all the functionality provided by the Portfolio Management feature.



Given a stock symbol, users can create news panels which track the news for a certain stock. The carousel can be cycled through via the left and right arrows, or left to automatically cycle.

Given a stock symbol, users can create stock graph panels which track the share price for a certain stock. The graph has the same interactivity as in our Stock Information feature, and allows users to zoom and change the time period of the data to view, and also fetch the most recent price data.

A block can be removed from a panel by hovering over the panel and clicking the ‘Remove’ button on the top left. Once a block is removed, the blocks will automatically shuffle to leave empty panels at the latter panels, where new blocks can be added.

Third Party Functionality

Frontend

Library	Description	Use Context
typescript	Programming language built on JavaScript that adds static type definitions.	Typescript was chosen for its static typing to ensure type safety at compile time. The language helps catch errors that can occur at runtime and also serves as typing documentation in a collaborative environment.
npm	Package manager for JavaScript development.	Npm was chosen to manage dependencies within our project.
react	Declarative, component-based UI framework library used to build single-page applications.	React was chosen as a performant and scalable solution to building a dynamic and interactive web application.
react-router-dom	Declarative routing library that keeps UI in sync with the URL.	React-router-dom was chosen for its simple yet powerful features, such as dynamic route matching and location transition handling.
redux (and react-redux)	State container and management library.	Redux allows central management of state between separate pages in a predictable manner. It was chosen for its ease of use, extensive documentation and prevalence in the web development industry.
redux-thunk	Middleware library for Redux which adds interaction with the store via asynchronous logic.	Redux-thunk was chosen for its support for asynchronous action dispatching that vanilla Redux actions lack.
bootstrap (and react-bootstrap)	CSS framework and UI component library for responsive front-end development.	Bootstrap and its react flavour were chosen for its abstraction of component styling, and quick

		development of responsive UI components.
formik	Form library that handles form state (values), validation and errors.	Formik was chosen for its ease of adoption and abstraction of typically boilerplate code associated with form state and validation.
yup	JavaScript schema builder for value parsing and validation.	Yup was chosen for its integration with Formik for form validation and lightweight API.
highcharts	JavaScript charting library for building flexible and interactive charts. The free version of Highcharts is licensed for non-commercial personal use only and a commercial deployment of the application would require a paid license.	Highcharts was chosen for its native support of financial time series graphs, a vital requirement of the project.
react-sliding-pane	React component for a sliding pane which slides out of the window's side.	React-sliding-pane was chosen for its abstraction of a generic sliding pane. The component greatly simplifies the implementation of the Notes pane and ensures reliable CSS functionality.
react-spinners	Collection of React loading spinner components	React-spinners was chosen for its reliable implementation of loading spinners. The library features a larger range of loading spinners which are more customisable than Bootstrap's.
react-toastify	Notification library for toasts	React-toastify was chosen for its ease of setup, high customizability and design.
bootswatch	Collection of themes for Bootstrap	Bootswatch was chosen to enhance the theme and aesthetics of our web application. It was also

		chosen for its ease of adoption and design.

Backend

Library	**Description**	**Use Context**
Flask	Micro web framework for Python.	Flask was used to develop a web server to provide information from various application features to the UI. It was chosen for its lightweight system requirements and ease of use.
Flask CORS	Adds Cross-origin Resource Sharing support for Flask.	The CORS extension allowed application endpoints to automatically resolve domain permissions when requesting or dispatching data.
Swagger	Interface Description Language for RESTful APIs.	Swagger was used to generate API definitions and automatically document endpoints.
Flask RestX	Adds decorators and tools for API definition and documentation to Flask.	RestX was used to define API schemas for use with Swagger.
Flask Mail	The Flask Mail extension allows for an interface to setup SMTP with the overall Flask application by allowing emails to be sent to users.	Flask Mail was used to give users notifications through specific triggers in the app. It was supplemented with Flask's render_template and several HTML templates to generate coherent emails.
Pandas	Data manipulation and analysis library.	Pandas was used to structure, format and manipulate data to dispatch on server endpoints and to perform time series and matrix operations on data processed by the prediction system. It was selected for its

		usability and fast performance.
Numpy	Multidimensional array and matrix manipulation library.	Numpy was used in the prediction system and the application server to support data types required by Pandas, Tensorflow and Scikit Learn. It is also highly optimised for speed.
psycopg2	PostgreSQL database adapter for Python.	Psycopg2 was used to connect to and query the database from the application environment. It has built in object conversion support between Python and Postgres formats and is also a secure medium.
Python Dotenv	Reading and writing tool for .env files.	Dotenv was used as a convenience wrapper for loading configuration information and database passwords from environment variables.
Scikit Learn	Machine learning library for Python.	Scikit Learn was used to develop traditional machine learning models for the prediction system. It was selected for ease of use, ability to quickly construct models and integrated performance metrics for testing.
Tensorflow	Symbolic math and deep learning library.	Tensorflow was used to develop deep learning models for the prediction system. It is highly optimised for speed in training and inference and supports high level model construction via the Keras API.
JSON Web Tokens (via PyJWT)	RFC 7519 encoding, decoding and verification.	JWT was used to ensure security during communication between the

		frontend and backend servers. JWT was chosen for its lightweight system, extensive documentation and it's currently an industry standard.
Backtrader	Trading and market backtesting library, supporting reusable trading strategies and reporting of indicators.	Backtrader was used to run paper trading algorithms for the simulated trades feature using developer-written strategies locally while submitting results to the broker library for reporting.
Docker	OS-level virtualisation framework.	Docker was used to build standalone containers to run the prediction and backtesting services. The containers could be deployed to other team members' machines to run for development and testing integration on other parts of the application.

Database

Library	**Description**	**Use Context**
PostgreSQL	SQL-based Relational Database Management System.	Primary database engine for the data layer.
PL/pgSQL	PL/pgSQL is a procedural programming language alloing more procedural control than standard SQL with more complex operations and computations.	Utilised PL/pgSQL for the voting functions, to allow for proper deletion from arrays, due to multiple states of no vote, current vote or current opposite vote.
uuid-ossp	The uuid-ossp module allows for the generation of universally unique identifiers (UUIDs). A UUID value is a 128-bit quantity generated by an algorithm which makes it universally unique.	Compared to serial values, UUIDs offer more security. They were generated throughout schemas as primary IDs.

Third-Party APIs

Library	Description	Use Context
Alpha Vantage	Financial data API offering company profiles, financial statements, price history and technical indicator data.	Alpha Vantage was used to source historical share price data and company information for display on the application company stock price page.
Alpaca Markets	Algorithmic trading API and brokerage service.	Alpaca was used as a trading broker and account to register paper trades made in the backend and report simulation results for the simulated trading feature.
IEX Cloud	Financial data and services API offering, price history and technical indicator data	IEX Cloud was used to source current share price data for display on the portfolio and dashboard pages. The API has a higher request limit that allows for frequent use independent of the other data APIs to avoid rate limiting.
Finnhub.io Market News API	Financial news and current affairs API.	The Market News API was used to serve general and company-specific financial news on the landing page and stock pages.

Implementation Challenges

Frontend

Challenges

The Redux library was relatively new to many members of the group and presented a learning curve to its mastery. Although valuable, some time was spent learning how to follow its best practices and debugging unfamiliar problems.

Handling user authentication and proper state management of user tokens was also a challenge, as our group had minimal past experience on this.

Recommendations

- Properly handling JWT authentication - the current storage of the token in localStorage is unsafe
- Extract Login into a modal to enhance the user experience and minimise context switches
- Abstract components further - many existing components contain repeated code/functionality that can be extracted into their own components to enable reuse
- SSO authentication for users to be able to sign in using third-party applications such as Google

Backend

Challenges

Stock Price Prediction

The most complex system to implement in the project was the stock price prediction feature, which uses machine learning models to digest stock price time series data and produce another time series with the predicted stock price. Implementation can be split into two phases: training and prediction.

The algorithm to train models can be summarised as:

1. Retrieve the stock price data
2. Split the data into train/test portions
3. Normalise and scale the data
4. Transform the normalised data to the shape or format required by the model in question
5. Feed the data into the model
6. Transform the result back into the original format

We primarily used the adjusted close series for training and testing. Our initial theory was that adding open and high/low series would improve training accuracy, but these additional series did not improve testing results when tried on SVM and ARIMA models. We did not attempt this on the deep learning models as the data is too similar to the adjusted close series and may cause overfitting. However, training with volume data in future could be beneficial in detecting large buy and sell movements.

We also decided that capturing the general trend of the company stock price rather than mirror the exact stock movement day-to-day would be more useful to end-users. Predicting the exact short-term price movement is highly reliant on external factors such as news and company releases, so pursuing additional accuracy with only time series data would be counterproductive.

Improvements to accuracy could be made in the form of using technical indicators beyond the raw stock price such as exponential moving averages, volume-weighted moving averages, Bollinger Bands, indicator crossovers or similar over 20, 50 and 100 day periods (say).

Scaling was also another point of contention - while the unscaled model we used was able to make sound predictions when input data was in the same approximate range as the training data, the model performed poorly outside this range with a discrepancy of at least 20% reported in the SVM model. For unclear reasons the scaled model did not perform significantly better and we needed to apply a manual inverse transform to the scaled output based on the range of the input data to realistically display the scaled predictions.

We tested the following models for use:

- SARIMAX - autoregressive integrated moving average with seasonality and exogenous variables. Parameters: p (regressor) = 2, d (integrator) = 2, q (moving average) = 1, seasonal parameters: p = 4, d = 1, q = 2
- SVC - support vector classifier; parameters: degree = 1, kernel='rbf', c=1000, gamma=0.1
- Regular LSTM (Long Short-Term Memory - a type of recurrent neural network)
    - 1 LSTM Layers (50 units, 20% dropout) + 1 Dense (tanh)
- LSTM with additional hidden layers
    - 4 LSTM Layers (50 units, 20% dropout) + 1 Dense (tanh)
- Convolutional LSTM
    - 2 CNN Layers (64 filters, 3 kernel size, relu, 25% dropout) + Max Pooling + Flatten + LSTM (100 units, 25% dropout) + 1 Dense (100 units) + 1 Dense

We found that the SARIMAX model can capture trends well but its output possesses a periodic wave-shaped oscillation that the regressor cannot account for. We also found a discrepancy between the ranges of the inputs and the prediction in the support vector machine; increasing the cost parameter to reduce misclassification could not control for this. Following these results, we decided to opt exclusively for two LSTM models to use in production, one variant with

several hidden layers and the other with the convolutional layer as these were best able to capture the overall price trend without introducing noise into the prediction. We did find however that the convolutional variant's prediction would always trend down regardless of the price direction of the input data or the number of subdivisions used in each sequence.

We also used the following transforms to reshape one dimensional sequences into a structure understood by Tensorflow:

- Regular variant: transforms shape (n) into (n, n, 1)
    - Shape: N sequences with n points of data in each sequence to produce one feature
    - Example sequence:
      $[d_1, d_2, \ldots, d_{2n}] \rightarrow [[\,[d_1, d_2, \ldots, d_n], [d_2, d_3, \ldots, d_{n+1}], \ldots, [d_n, d_{n+1}, \ldots, d_{2n}]\,]]$
- Convolutional variant: transforms shape (n) into (n, s, p, 1)
    - Shape: N sequences with n points of data divided into s subsequences with p steps per subsequence to output 1 feature
    - Example sequence with two subdivisions:
      $[d_1, d_2, \ldots, d_{2n}] \rightarrow [[\,[[d_1, d_2, \ldots, d_{n/p}], [d_{n/2}, d_{n/2+1}, \ldots, d_n]], \ldots,$
      $[[d_n, d_{n+1}, \ldots, d_{3n/2}], [d_{3n/2}, d_{3n/2+1}, \ldots, d_{2n}]]\,]]$

Possible improvements to the above models would involve other methods of normalising the data that would be dataset- or company-agnostic. Further models to test include random forest regressors, other kinds of recurrent architectures such as gated recurrent units for expedited training and bidirectional variants of the LSTM.

Due to time constraints, we were not able to train models for sentiment analysis of company news and general market trends. However, combining analysis of market news with the time series forecasting problem would be the next logical avenue to explore as these events have a more significant effect on a company's share price and market value than previous performance.

Notifications

The notification system was problematic as Flask does not properly cooperate with Web sockets as the entire application would need to move away from requests, queries and arguments into using web socket emits.

Email was chosen for its ease of implementation with existing technologies as Flask Mail correlates with Flask. Sending emails did carry challenges, as email would need to be an asynchronous request, so the application had to be morphed into allowing threading. A custom Flask class had to be made in order to allow a thread to start with the application as well. Threading was implemented, as emails were then able to be sent asynchronously, with the only problem being accessing private classes relating to the context of the Flask app.

Another challenge involved was implementing email templates, as the HTML in emails has certain restrictions. Using Flask's render_template function, we were able to create dynamic emails, but due to the limitations of email HTML and CSS, concessions had to be made to get the same desired premade UI. Email HTML generators needed to be used to find viable structures and dynamic HTML had to be made through the backend.

Recommendations

These recommendations are extensions of the main application architecture to account for higher user volumes and more distributed performance.

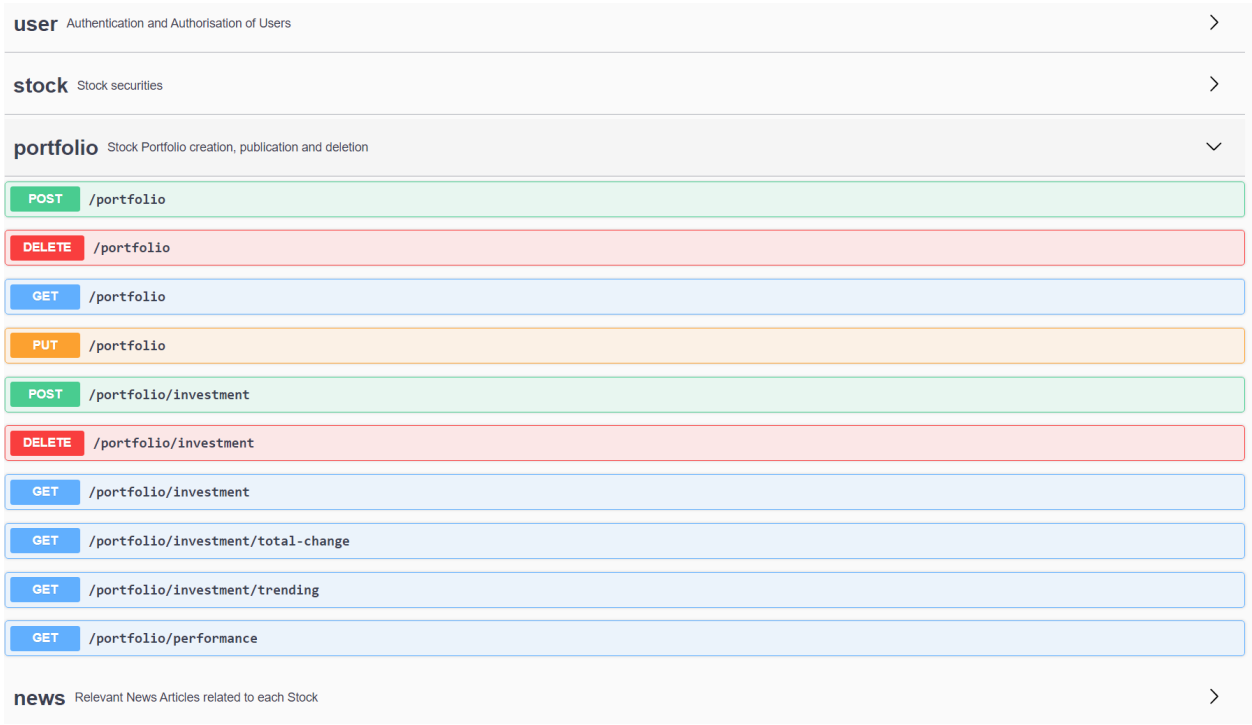
- Further modularisation of core features to run as separate services instead of a monolithic server, for instance, to support more complex forum database queries and handle processing such as weighting comments by registered user status or activity levels.
- Alternatively, implement multithreaded request processing for core functionality which would extend the existing thread creation logic for the notifications feature. This allows requests for disparate features to be fulfilled concurrently without blocking processing for other requests.
- Alternatively, use a higher capacity web server framework with asynchronous IO support to improve application scalability for higher request volumes.
- Redirection of requests to individual services for that service's specific functionality e.g. requests for prediction data made directly to prediction service.

These recommendations relate to the limitations of the technologies and libraries used.

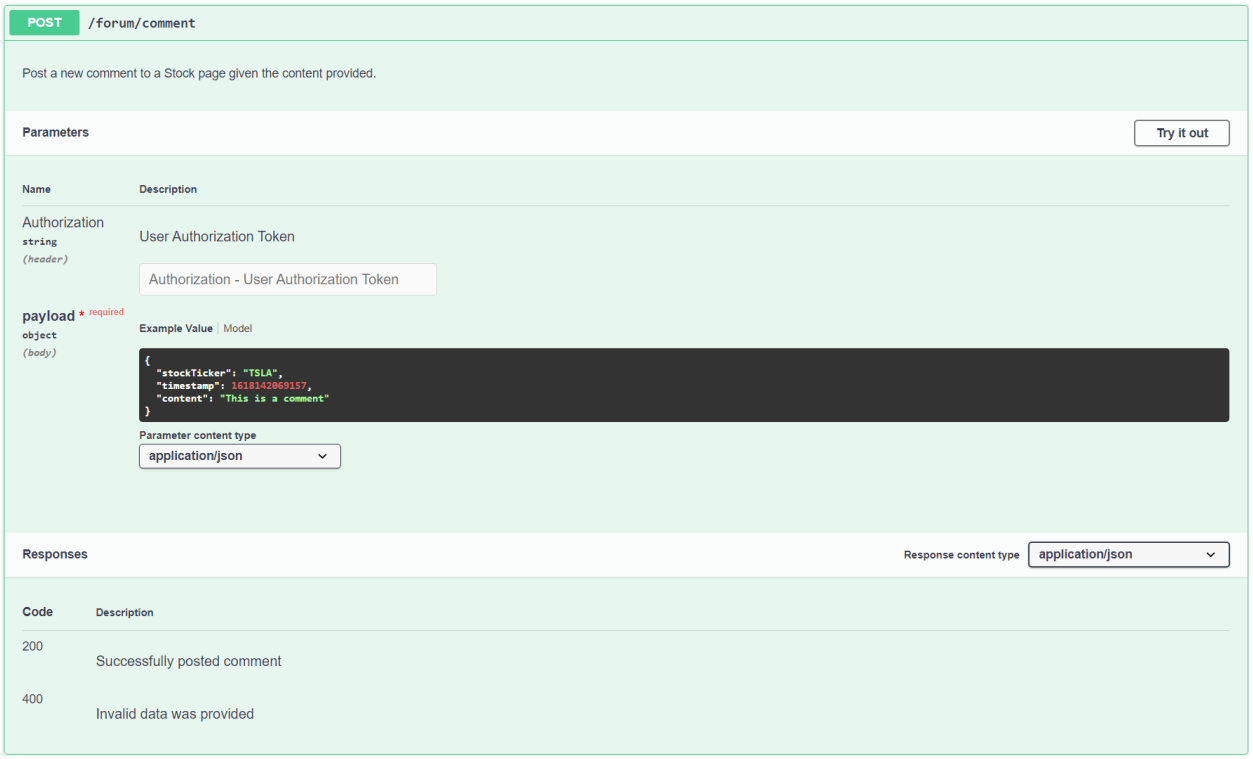
- Utilising web sockets and frameworks that interact with them easily. This opens up web notifications and potential chat bot systems.

Special Architecture

An example of special architecture in our backend involves Flask-RestX. Flask-RestX is an extension for Flask which allows for useful decorators and tools that help describe our API and expose proper documentation using Swagger.



Accessible at the backend address at http://127.0.0.1:5000/ (by default), the routes listed provided extensive documentation on the types of requests the routes receive, the parameters involved and HTTP exception errors.



The implementation allows for extensive testing of dummy data without any built frontend. Parameters are labelled with types and descriptions and examples are also given. This encapsulates a professional test environment and input validation handled by the library.

Database

Challenges

The implementation of the Dashboard database model was a complex challenge faced by our group. The use of table inheritance to abstract parent Dashboard Block properties led to a

severe limitation of using foreign key references to any child tables which inherited from the parent. PostgresQL is unable to ensure that UUID fields across child tables will be unique, and this has not been implemented in an efficient enough manner successfully yet. To work around this, our group had to remove foreign key references to the dashboard_references table, which could result in duplicate blocks, and hence duplicate and corrupt references.

Recommendations

- Add an id field to the portfolios table - the lack of an id field has meant the need to update many tables when the name of a portfolio changes
- More use of foreign keys in tables to enforce accurate representation of the data model
- Draw out ER diagrams in the planning of tables and schema structures. Consistent altering of tables was needed to add in columns that were later needed by other functions. Diagram planning would allow for a more collaborative and visualised approach.

APIs

Challenges

Due to using free/trial plans, the third party APIs we have used to retrieve application data all place rate limits on requests. The specific limit varies from platform to platform (5/min, 500/month for Alpha Vantage and 100/min for IEX finance but only for data from the last 5 years, which could impact machine learning model performance if retraining is needed). However, in either case testing of systems like the portfolio display and stock price charts would cause the rate limits of each API to be exceeded during development and complicate access to data for component testing.

To counteract the rate limit we built caching/storage mechanisms and used several APIs to fetch our data. We also assigned each component to access one particular API best suited for its function to limit the number of requests dispatched by the application.

The algorithm to determine whether data needs to be pulled from the API can be summarised as:

- Get the time of last refresh for the time series data needed
- Localise the time using the company stock exchange's timezone time series metadata
- Check that the current time is within the exchange's trading hours, backdate the time to close of trading (typically 8pm for extended US hours) if the current time is out of hours
- Check that the current date is within the exchange's trading days (typically weekdays), backdate the date to Friday if the current date is on a weekend
- Take the difference between the current date and last refresh date and if the difference is > 15 minutes on intraday data (Alpha Vantage's data refresh rate) or 1 day on daily data, return true

The algorithm is run when a stock market data request is made to the main server in the business layer and updates the state variables in its stock data module accordingly. We also designed logic to write this data to disk locally and retrieve the data from disk should the API be unavailable or the refresh time has not elapsed.
Furthermore, dedicated API endpoints are assigned to the following subsystems to spread request loads and prevent the APIs' respective rate limits from being reached:

- Stock page time series data, company profiles and financials: Alpha Vantage
- Portfolio value indicators and stock views within portfolios: IEX Cloud
- Screener financials values: IEX Cloud
- Stock market page and landing page financial news: Finnhub.io

Recommendations

- With more time and resources involved, different APIs can be used to minimise query times, incorporating batch queries (e.g. TSLA,AAPL,GME instead of querying singular stocks) and opening up higher quality requests (news requests filtering irrelevant articles)

User Documentation/Manual

Setup

Backend and Frontend compile scripts exist in the root directory of our code. The Prediction Model and Backtesting Services can be run **either** using Docker or Flask. If Docker setup/running does not work, please try running the Prediction and Backtesting Services as Web Servers directly.

Backend & Frontend

1. Ensure you have python3 version >= 3.7.4 and pip3. Download the appropriate version, if needed, here: https://www.python.org/downloads/ and https://pip.pypa.io/en/stable/installing/
2. Ensure you have nvm installed. Download it here: https://github.com/nvm-sh/nvm#installing-and-updating
3. Run `nvm install v14.15.4` to install node.
4. Run `nvm use` to ensure you are using the correct version of node.
5. Run `npm install -g install npm@7.5.2` to install npm. If you are on windows, instead follow these instructions: https://github.com/felixrieseberg/npm-windows-upgrade to install the same version.
6. Run `npm i` to install dependencies.
7. Run `npm i -g serve` to install serve.
8. In the root directory, run `npm run build` to build both frontend and backend.

Prediction Model & Backtesting Services - with Docker

1. Ensure you have docker installed. Download it here: https://docs.docker.com/get-docker/ . Open the Docker application once it is installed.
2. Download the docker images for the two services:

- Prediction https://unsw-my.sharepoint.com/:u:/g/personal/z5112826_ad_unsw_edu_au/EQMuLROLmaxKoIzu5-1cLesBD6MuUtwNePDqoFwjneAdBw?e=UarW27
- Backtrading https://unsw-my.sharepoint.com/:u:/g/personal/z5112826_ad_unsw_edu_au/EaSEggHd90JKuGKusyYq80gBoAZIz2HiAzJLmSJJR0qikQ?e=Rk70g2 .

Prediction Model & Backtesting Services - as Web Server

1. Ensure you have python3 version >= 3.7.4 and pip3. Download the appropriate version, if needed, here: https://www.python.org/downloads/ and https://pip.pypa.io/en/stable/installing/
2. `cd` to the service's build directory (this is `<repository-path>/model/lstm` for the prediction server and `<repository-path>/trader` for the backtesting server).

3. Run `virtualenv venv` to create a Python virtual environment. Once created, activate with `source venv/bin/activate` (Linux/Mac) or `.\venv\Scripts\activate` (Windows).
4. Install the prerequisite libraries with `pip3 install -r requirements.txt`.
5. Repeat for both the prediction and backtesting services.

Running the Application

Backend & Frontend

1. In the root directory, run `npm run start` to concurrently run the frontend and backend. Alternatively, you can also run the frontend and backend separately:
    a. Frontend: `serve -s -l 3000 frontend/build`
    b. Backend:
        i. `cd backend/src`
        ii. `FLASK_APP="server.py" FLASK_ENV="production" flask run --port=5000`
2. Visit http://localhost:3000 to start using To The Moon.

Prediction Model & Backtesting Services - with Docker

In separate terminals:

1. Run `docker load < caps_prediction.tar.gz` to load the image. Make sure you are using the relative path to the image.
2. Run `docker run -p 3001:5000 -it caps_prediction_lstm` to start the prediction model.

1. Run `docker load < backtrading.tar.gz` to load the image. Make sure you are using the relative path to the image.
2. Run `docker run -p 3002:5000 -it backtrading` to start the backtrader.

Prediction Model & Backtesting Services - as Web Server

1. If not already in the model/lstm directory, `cd` to that directory and activate the virtual environment.
2. Run `FLASK_APP=run.py FLASK_ENV=production FLASK_RUN_PORT=3001 flask run` (Linux, Windows Git Bash) to start the prediction model.
    On Windows (Powershell), run:
    a. `$env:FLASK_APP="run.py"`
    b. `$env:FLASK_ENV="production"`
    c. `$env:FLASK_RUN_PORT="3001"`
    d. `flask run`

Replace `$env:FLASK_APP=”run.py”` with `set FLASK_APP=”run.py”` and so on for the remaining variables if running in CMD.

3. If not already in the trader directory, `cd` to that directory and activate the virtual environment.
4. In the trader directory run `FLASK_APP=run.py FLASK_ENV=production FLASK_RUN_PORT=3002 flask run` (Linux, Windows Git Bash) to start the backtesting model.

   On Windows (Powershell), run:

   a. `$env:FLASK_APP="run.py"`
   b. `$env:FLASK_ENV="production"`
   c. `$env:FLASK_RUN_PORT="3002"`
   d. `flask run`

   Replace `$env:FLASK_APP=”run.py”` with `set FLASK_APP=”run.py”` and so on for the remaining variables if running in CMD.

Config

To change the port for running frontend, change the number in `serve -s -l <port>` in `package.json` on line 10.

To change the port for running backend, change the number in `flask run --port=<port>` in `package.json` on line 10. Change the corresponding port number in `frontend/src/config.json`.

To change the ports for running prediction and backtrading services, variables for the server URLs and endpoints can be found in `backend/src/.env` with the following default values:
MODELSRVURL="127.0.0.1"
MODELSRVPORT=3001
BACKTRSRVURL="127.0.0.1"
BACKTRSRVPORT=3002

References

(Distractify, 2020)
Yahoo News Suspended Its Comment Section, and People Are Freaking Out - Distractify. Distractify.com. (2020). Retrieved 1 March 2021, from https://www.distractify.com/p/yahoo-comments-gone.

(Highcharts, 2021)
Highcharts Demo. Highcharts.com. (2021). Retrieved 28 February 2021, from https://www.highcharts.com/demo.

(Hristozov, 2019)
Hristozov, K. (2019). MySQL vs PostgreSQL -- Choose the Right Database for Your Project. Okta Developer. Retrieved 1 March 2021, from https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres.

(Investing.com, 2021)
About Us - Investing.com. Investing.com. (2021). Retrieved 2 March 2021, from https://au.investing.com/about-us/

(MetaQuotes, 2021)
MetaTrader 5 Trading Platform - MetaQuotes. MetaQuotes. Retrieved 3 March 2021, from https://www.metaquotes.net/en/metatrader5

(mql5, 2021)
MQL4 and MQL5 Source Code Library - MQL5. mql5.com (2021). Retrieved 3 March 2021 from https://www.mql5.com/en/code

(Raiz, 2021)
How Raiz Works - Raiz. raizinvest.com. (2021). Retrieved 1 March 2021, from https://raizinvest.com.au/blog/how-raiz-works/#:~:text=Raiz%20can%20track%20credit%20cards,and%20invest%20it%20for%20you.

(RapidAPI, 2021)
Alpha Vantage API Documentation. RapidAPI. (2021). Retrieved 3 March 2021, from https://rapidapi.com/alphavantage/api/alpha-vantage.

(React, 2021)
Static Type Checking – React. Reactjs.org. (2021). Retrieved 28 February 2021, from https://reactjs.org/docs/static-type-checking.html.

(Singh, 2021)

Singh, V. (2021). *Flask vs Django in 2021: Which Framework to Choose?*. Hackr.io. Retrieved 3 March 2021, from https://hackr.io/blog/flask-vs-django.

(SitePoint, 2021)
React with TypeScript: Best Practices - SitePoint. Sitepoint.com. (2021). Retrieved 1 March 2021, from https://www.sitepoint.com/react-with-typescript-best-practices/.

Libraries Used

Alpha Vantage - Free Stock APIs in JSON & Excel. Retrieved 15 April 2021, from https://www.alphavantage.co/.

Alpaca Markets. Retrieved 15 April 2021, from https://alpaca.markets/.

Backtrader. Retrieved 21 April 2021, from https://www.backtrader.com/.

Docker. Retrieved 21 April 2021, from https://www.docker.com/.

Finnhub - Real-time Market News API. Retrieved 15 April 2021, from https://finnhub.io/docs/api/market-news.

Flask. Pallets Projects. Retrieved 21 April 2021, from https://flask.palletsprojects.com/en/1.1.x/

Flask RestX. Read the Docs. Retrieved 21 April 2021, from https://flask-restx.readthedocs.io/en/latest/.

Flask CORS. Read the Docs. Retrieved 21 April 2021, from https://flask-cors.readthedocs.io/en/latest/.

Flask Mail. Python Hosted. Retrieved 21 April 2021, from https://pythonhosted.org/Flask-Mail/.

Formik: Build forms in React, without the tears. Retrieved 21 April 2021, from https://formik.org/.

Highcharts: Interactive JavaScript charts for your webpage. Retrieved 21 April 2021, from https://www.highcharts.com/.

IEX Cloud. Retrieved 15 April 2021, from https://iexcloud.io/.

npm. Retrieved 21 April 2021, from https://www.npmjs.com/.

Numpy. Retrieved 21 April 2021, from https://numpy.org/.

Welcome to PyJWT. Read the Docs. Retrieved 21 April 2021, from https://pyjwt.readthedocs.io/en/stable/.

python-dotenv. Pypi. Retrieved 21 April 2021, from https://pypi.org/project/python-dotenv/.

Pandas. Retrieved 21 April 2021, from https://pandas.pydata.org/.

PostgreSQL: The World's Most Advanced Open Source Relational Database. Retrieved 21 April 2021, from https://www.postgresql.org/.

Psycopg2 - Python PostgreSQL Database Adapter. Pypi. Retrieved 21 April 2021, from https://pypi.org/project/psycopg2/

Scikit-learn. Scikit-learn. Retrieved 21 April 2021, from https://scikit-learn.org/stable/.

PL/pgSQL - SQL Procedural Language. Retrieved 21 April 2021, from https://www.postgresql.org/docs/9.6/plpgsql.html.

React – A JavaScript library for building user interfaces. Retrieved 21 April 2021, from https://reactjs.org/.

React Bootstrap. Retrieved 21 April 2021, from https://react-bootstrap.github.io/.

React Router: Declarative Routing for React.js. Retrieved 21 April 2021, from https://reactrouter.com/web/guides/quick-start.

React Sliding Pane. Retrieved 21 April 2021, from https://www.npmjs.com/package/react-sliding-pane.

React Spinners. Retrieved 21 April 2021, from https://www.npmjs.com/package/react-spinners.

React Toastify. Retrieved 21 April 2021, from https://www.npmjs.com/package/react-toastify.

Bootswatch: Free themes for Bootstrap. Retrieved 21 April 2021, from https://bootswatch.com/.

Redux - A Predictable State Container for JS Apps. Retrieved 21 April 2021, from https://redux.js.org/.

Redux Thunk. Retrieved 21 April 2021, from https://github.com/reduxjs/redux-thunk.

Swagger: API Documentation and Design Tools for Teams. Retrieved 21 April 2021, from https://swagger.io/.

Tensorflow. Retrieved 21 April 2021, from https://www.tensorflow.org/.

TypeScript: Typed JavaScript at Any Scale. Retrieved 21 April 2021, from https://www.typescriptlang.org/.

Uuid-ossp. Retrieved 21 April 2021, from https://www.postgresql.org/docs/9.4/uuid-ossp.html.

Yup. Retrieved 21 April 2021, from https://github.com/jquense/yup.