MSc in Computer Science - Team Project

CMPU9010

Interim Report

Team 1 - Feel the MERN

Warren Kavanagh	C16463344
Joseph Corcoran	D20127858
Bearach Byrne	C1537916
Rebecca Kelly	C10330971
Daragh Kneeshaw	D20128577
Caolán Power	D21125982

Table of Contents

Table of Figures

List of Tables

1 Introduction

The aim of this project is to build a mobile first web application, FinOptimise, that will provide a beginner friendly platform allowing users to experience virtual investing and to provide a starting point for building a healthy interest in investing.

The primary functionality of FinOptimise will be the presentation of a range of information relating to companies within the S&P500 index (Will Kenton, 2022). This information will take the form of company data such as current share price, share price change and market capitalisation. This information will be accompanied by Environmental, Social and Governance (ESG) ratings. ESG ratings can help potential investors to make informed investment decisions with respect to ESG issues that they may personally find to be important (Deloitte China, 2022). As it is the belief of the team that ESG issues should be central to an investor's decision-making process, it is important that this information is presented in a manner that is intuitive and easy to understand regardless of the level of investing experience the user may have.

Alongside ESG ratings, news articles and Tweets relating to companies will also be presented, with a sentiment analysis carried out on the contents, which will show the user whether the current sentiment is positive, neutral, or negative. The application will also feature a recommender system that will recommend specific companies to the user based on their previous interests.

Users will be able to see what investing may be like by creating a virtual portfolio and competing against their friends, similar to fantasy football (Premier League, 2022). Studies have shown that gamification can increase user engagement in a task and can also increase the "quality and productivity of actions" (Juho Hamari, 2014). This type of player-centred design can be seen in many applications currently available, one such example is Duolingo, where users are encouraged to come back daily to keep up their learning (Phagava, 2021).

This report will provide an overview of the current status of the project as a whole and will detail who this application is aimed at, the technical problem that will be faced and team's solutions to those problems, and how this project will be evaluated.

Figure 1 - Application Logo

2 User Scenario

2.1 Identifying target users

This year Ireland experienced a 38-year high in inflation, reaching 9.1% (Central Statistics Office, 2022) and a global inflation rate of 8.8% (International Monetary Fund, 2022). According to a report by sky news on a new study, this cost-of-living crisis disproportionately affects young people as they tend to have lower income jobs, they are more interested in socialising, fashion and are less likely to own their own home so are vulnerable to increasing rents. The opening line reads “People aged 18 to 30 are more likely to be savers than any other age group yet are still the most financially vulnerable, according to research from Demos think tank.” (Brady, 2021). This shows that young people are vulnerable to increases in living costs but not because they are financially irresponsible. It also shows they are interested in improving their financial futures.

2.1.1 Personas

As described above, the team identifies our target user as young working people who have some disposable income but do not know where to start when it comes to researching and participating in the stock market or investing. Below are two user personas that represent the user group:

User 1

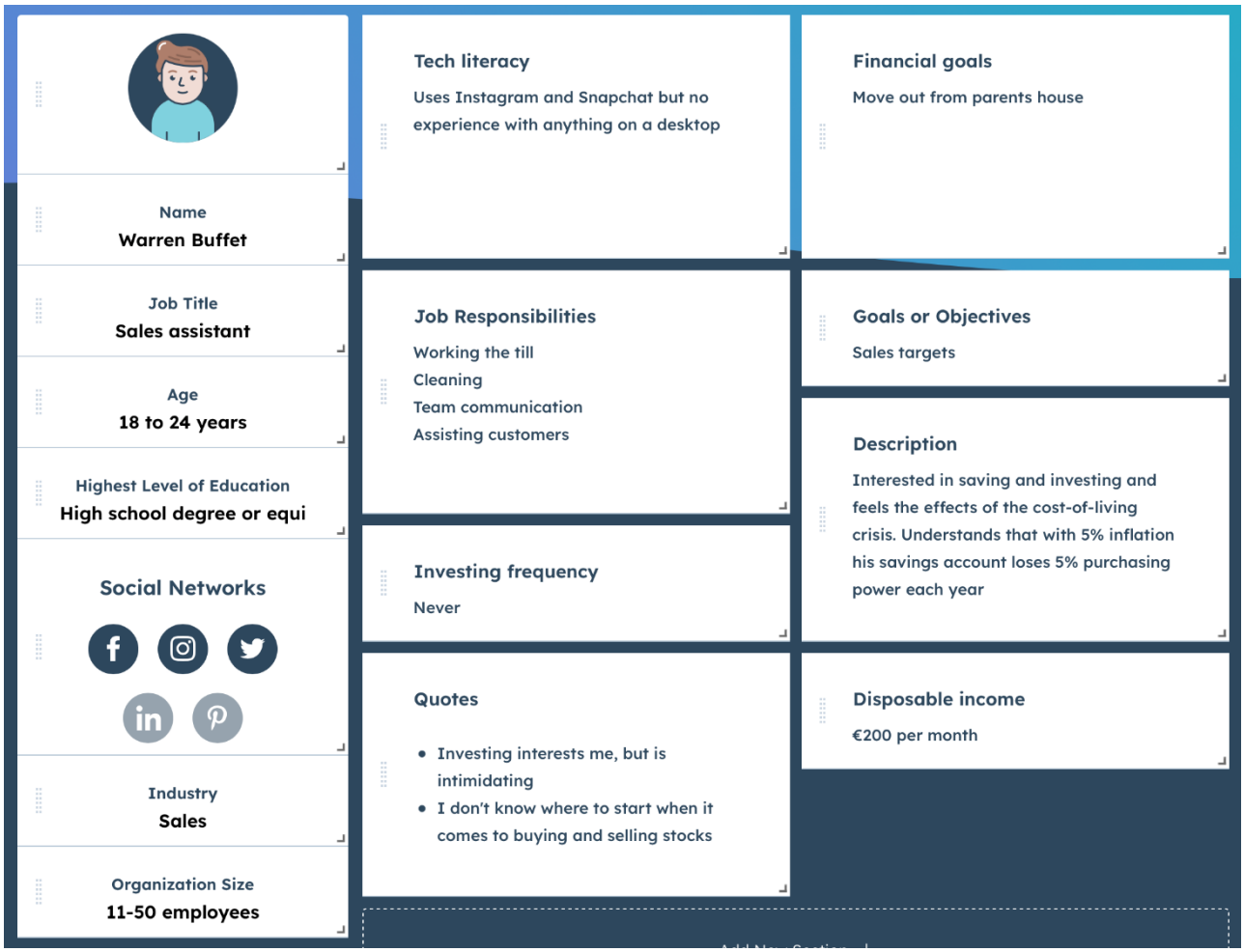


Figure 2 - User Persona 1

User 2

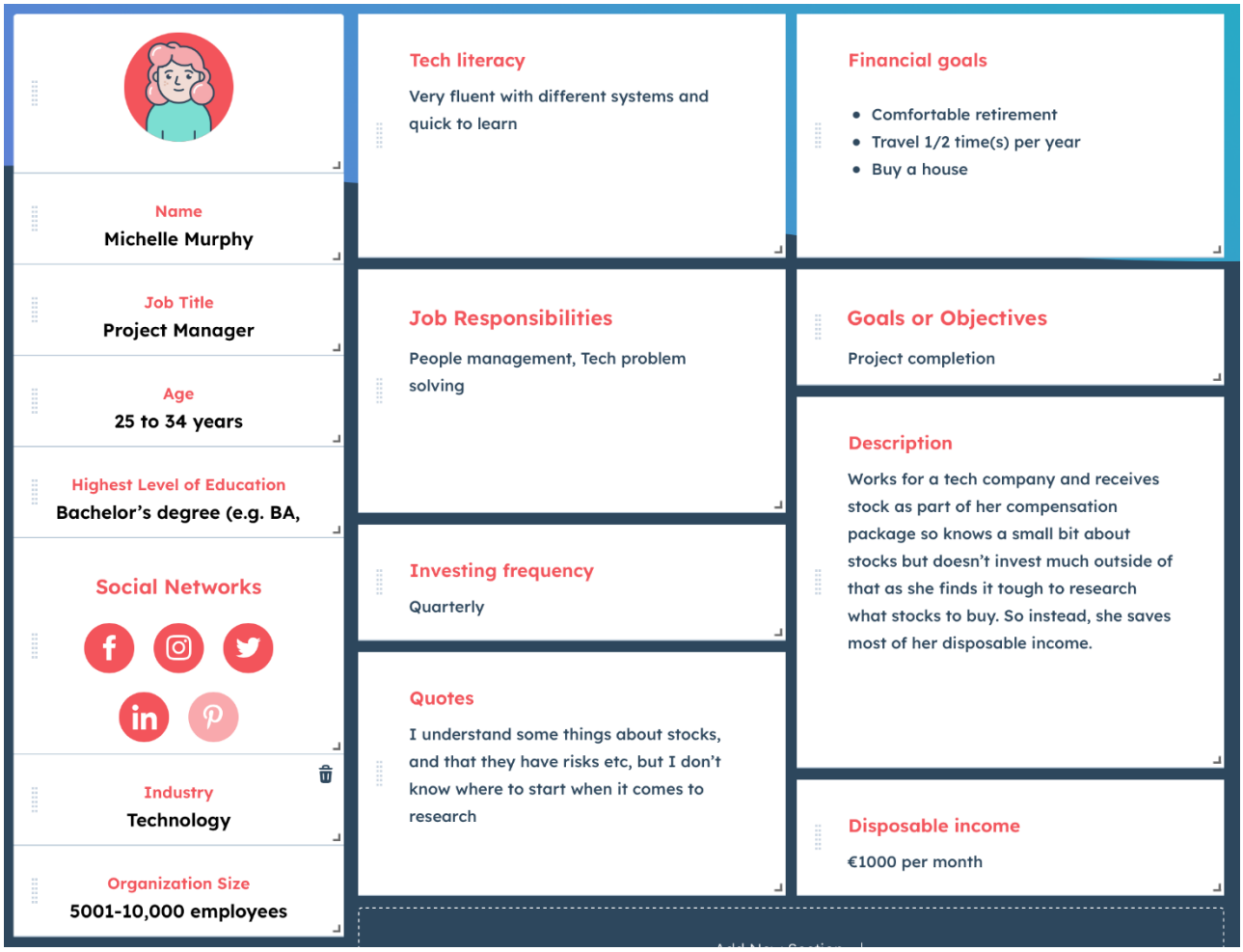


Figure 3 - User Persona 2

These personas will be used to better understand user's needs, experiences, behaviours, and goals. It will help us think of our product from an 'outside' perspective, by viewing it through the lens of these fictitious characters.

2.2 Importance of target users

Identifying the application's target users is essential, as it brings focus to the project development. Prioritising features that solve the biggest issues for the user involves getting user input as much as possible.

2.2.1 Survey

As part of our field research, we conducted a survey to identify whether the results lined up with our other research. The results from 63 respondents are as follows:

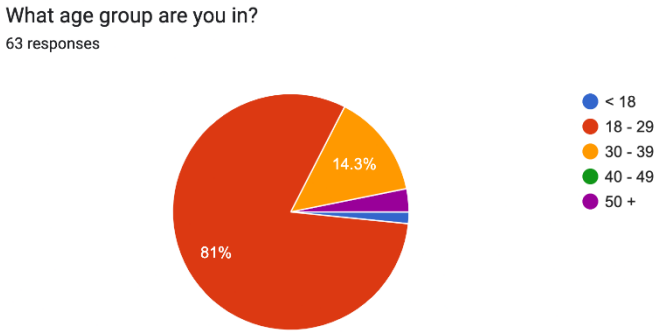


Figure 4 - Survey response - age group

This survey was sent to family and friends and most respondents were in the 18-29 age group.

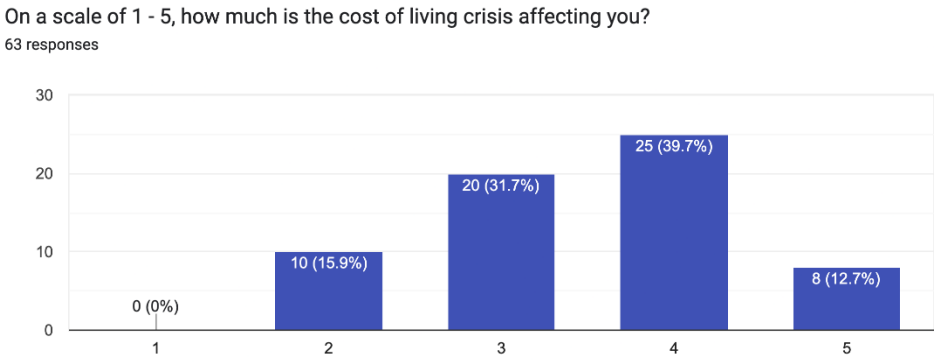


Figure 5 - Survey response - cost of living crisis

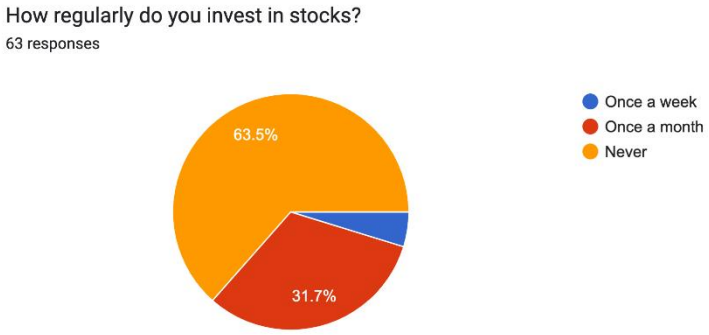


Figure 6 - Survey response - investing frequency

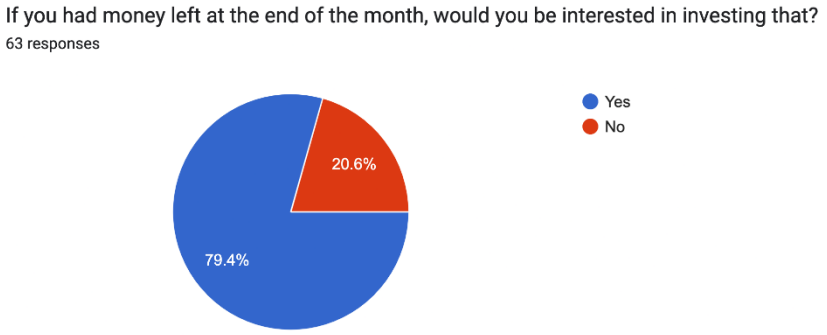


Figure 7 - Survey response - investing interest

Most respondents had an interest in investing but did not currently invest in stocks.

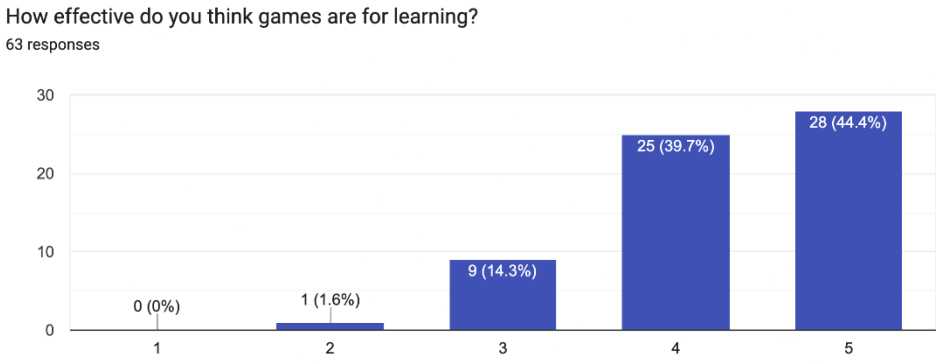


Figure 8 - Survey response - game feature interest

Most survey participants think of games as useful tools to learning new information, so we intend to prioritise this feature in our product.

2.3 Problems faced by the target users

We deduce from our research that the main issues faced by our target users are as follows:

- They are financially responsible and have the ability/interest to invest some money however, investing is not very accessible to them due to lack of knowledge.
- Research on investing can be cumbersome and most users do not know where to start. These will be further elaborated on in the technical problem section of this report.

The main aim is to solve the issue of knowledge accessibility by making it easy for people to research and discover new investments. The process will be made more accessible by creating an investing game that is fun for users and can help them learn as they play, getting hands-on experience with different order types and researching different stocks to better perform in the game, all in a risk free, simulated environment.

3 Technical Problem

3.1 Reasons For Building This Application

The idea for this project went through several incarnations before the current format was finalised. The concept of a virtual stock trading platform, which includes a gamified version of stock trading, was chosen for a variety of reasons.

Through research into several applications that have similar functionality, the group found that a combination of factors made these less accessible to the general public. Within the group, which has a varied level of experience with financial markets (from no experience, having worked in finance and investments), there was a consensus that these platforms tended to overload the user with information, and display graphs on metrics with which the average user would not be familiar. Therefore, a platform that stripped back unnecessary information was deemed an effective way to allow people to start learning how trading works.

As mentioned in section 2.1, the application is aimed at people who are interested in investing and have some disposable income to do so. In line with the research mentioned in this section this is particularly relevant today, amid a global cost of living crisis. Responses to our survey indicated that most respondents do not invest, outside of their pensions, but a vast majority of these confirmed that they would do so if they had the necessary disposable income (see figure 7 in section 2.2.1). In allowing people to invest in companies virtually, without using their own money, this application aims to teach the user about investment through actively doing so.

A large aspect of this project is the gamification of stock trading. Studies have shown that gamification can increase user engagement in a task and can also increase the "quality and productivity of actions" (Juho Hamari, 2014). The initial idea was to have a virtual portfolio with which users could invest in stocks. A few memebrs of the group use the *Fantasy Premier League* platform, which allows players to act as a manager of a football club and buy and sell players for their team. Players compete in leagues, where their performance depends on the performance of their players in real life. It was this paradigm that inspired us to gamify the virtual portfolio feature, and this is what the model is loosely based upon (Premier League, 2022). As well as this, there are several platforms which do similar things to ours that will be evaluated below.

Another aspect that the team wanted to introduce to the application was the concept of investing in ethical companies and having information about the ethics of a company at hand, before investing in them. Therefore, ESG (Environmental, Social, and Governance) ratings are displayed for each company. ESG ratings judge how a company manages its environmental, social and governance risks in comparison to their peers. These ratings allow investors to understand the priorities and risks that company may be liable to face (Plaut, 2022). In our context, this will allow users to see the benefit of good ethics when it comes to business. 89.5% of survey respondents indicated that ESG ratings would have an impact on their interest in a particular company. The game aspect will also have configurable rules, some of which have the option to penalise or reward users, depending on the ESG ratings of their portfolio.

The general and media sentiment towards a company is also missing from most trading platforms. As will be discussed below, this application will show news and twitter feeds relating to each company. Each article and tweet will have a sentiment (positive, neutral, and negative) attached to it. The average sentiment from these will also be displayed on each stock page. This will provide information to the user in relation to how that company is doing in the media, which is often a good indicator of

how the stock will perform (Xinjie Wang, 2022). This will be more legible than lists of figures and graphs on metrics with which beginners are not familiar.

3.2 Core Technical Problems

3.2.1 Overview

This section will outline the core technical problems that are being addressed and will be overcome by developing this application. The solutions to these problems will then be described in the next section. These problems include the issue of how to define and develop a suitable User Interface, choosing a suitable architecture and hosting platform, and the issue of retrieving up-to-date and accurate stock data.

3.2.2 User Interface/User Experience

As per the reasons for developing this application, which are outlined above, it can clearly be seen that the User Interface (UI) and User Experience (UX) of this application are vital components, and integral to the success of the project. The core issue here is that the presentation of data is in an accessible and legible manner, to allow people with no experience in investing to navigate the app effortlessly, while both learning and enjoying themselves in doing so.

The UI primarily follows the guidelines of Flat 2.0 design, which combines elements of skeuomorphic and flat design by adding some depth to mostly flat elements (Ho, 2016). Finding a good balance between saving loading times by implementing minimalist components while also implying information through signifiers (such as clickable elements) is an ongoing process. This is a challenge which can only be overcome by ongoing user evaluation of prototypes and of the application itself, and constant updates to the UI to reflect the feedback of users.

As well as this, the issue of how to present financial information in a format that is intelligible to users of all levels is a central issue. Graphs are used throughout to represent essential information about companies. In general, it is desirable not to overload the user with these, but to present them in a way that makes them more accessible than volumes of text. Text is used sparingly by itself, only when it is necessary, and an effort is made to provide information on complex financial terms using tooltips. Graphs themselves are under constant evaluation and reworking, such as our method of presenting ESG data, which will be further explained in the evaluation section.

3.2.3 Architecture and Hosting Platform

Another core technical problem faced in the development of the application is the deployment and hosting of the live application. There are several issues to think about when researching how to host the application.

One of these issues is cost. As students, the team have limited money to spend on cloud services, therefore research had to be undertaken to find a cloud service provider that provided a free tier which could cover most of our needs. This was one of the reasons for our decision to choose Amazon Web Services (AWS). They provide an extensive free tier and have benefits in owning a student account. At the beginning of the project, it was impossible to predict the exact usages. As such, the usage is continually monitored to ensure the available limits are not exceeded. Recently, it became clear that the database, which was hosted on an AWS EC2 T2 Micro instance, did not have the resources necessary to handle the traffic to it. Therefore, it was necessary to upgrade to a T2 Small instance, with 1GB more RAM. The T2 Small is not included in the free tier, so the team has agreed to pay approximately $3.125 per member per month for the remainder of the project.

Another issue relating to hosting the application was the learning curve involved. Most of the group have not previously had to manage cloud servers, which can be very time consuming and can require a lot of research into how best to structure the architecture to avoid single points of failure. As such, the backend and machine learning components are run on AWS Lambda, which is a serverless computing service that only requires code to be uploaded and some minor configuration to be carried out. This saves the team a lot of time and is also less costly than managing servers, as the service only runs when code is executed (AWS, 2022).

3.2.4 Stock Data Ingress

As the application depends on the availability of stock data related to each of the companies on display, the process of retrieving, storing, and updating this data is one of the key problems associated with development. Accurate, real time, and thorough financial data can be extremely costly, and it can be a taxing effort to retrieve it. As mentioned above, keeping costs to a minimum was a requirement for the team, therefore, several solutions were looked at for the retrieval of accurate stock data.

The application needs both historical and current stock data. Historical data is needed for stock metrics and stock visualisations, such as price graphs. Historical data was retrieved from the Alpha Vantage API. This provides historical data of up to 20 years for some of the companies included in our application. This needs to be run once, to fill the database with the historical data. This API only allows 5 requests per minute and 500 requests per day; therefore, a Python script was written to send 5 requests every minute and populate the database.

It was not ideal that financial data had to be retrieved from various sources due to the constraints outlined above for Alpha Vantage. Therefore, it was not suitable for the retrieval of up-to-date financial information. This data is essential to the application as it is necessary for the near real-time updates of the user's portfolio, which in turn facilitates the running of the game feature. A compromise was arrived at that current financial data should be updated every 20 minutes. With Alpha Vantage ruled out, Yahoo finance was also researched, however, the data would need to be scraped, and this script would need to be run 493 times (one for every stock in the index). This would be resource intensive, and it is unclear if Yahoo's policy would allow a web scraper to be used this frequently. Finally, slickcharts.com was discovered, which allows the scraping of all financial information in one go for free and has sophisticated metrics available.

3.3 Technical Components

The diagram below shows the technical components necessary to complete this project.

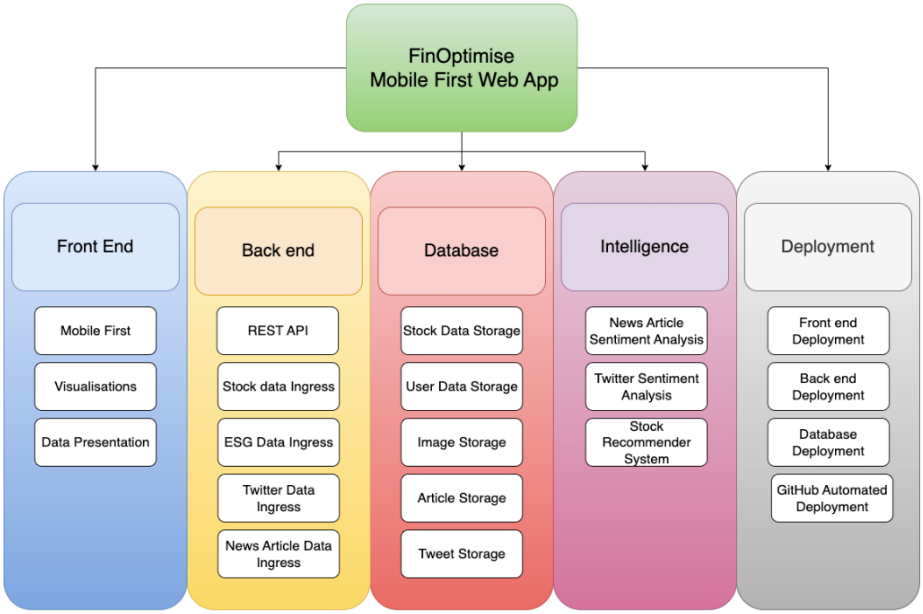


Figure 9 – Technical Component Diagram

3.4 Review of Similar Systems

3.4.1 Wealthbase

3.4.1.1 Overview

Wealthbase is a gamified virtual stock trading web application, found at https://www.wealthbase.com/. It allows users to set up stock trading leagues or enter a league, setting up a portfolio when they do so.

3.4.1.2 Pros

- Clean and legible UI.
- Extremely helpful information on terms and financial jargon, including explanations of rules when setting up a game.

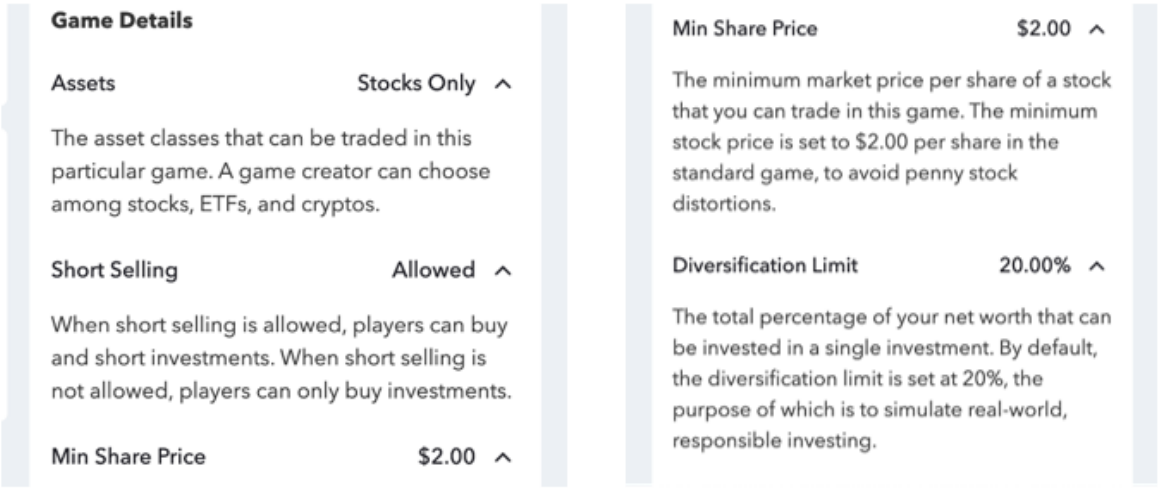


Figure 10 – Wealthbase One

- User feed section - Makes posts from users throughout the application visible. Users can post about the stocks they are trading and outline why.
- Simple overview of portfolio. Clearly laid out and attractive UI. Easy to read breakdown.

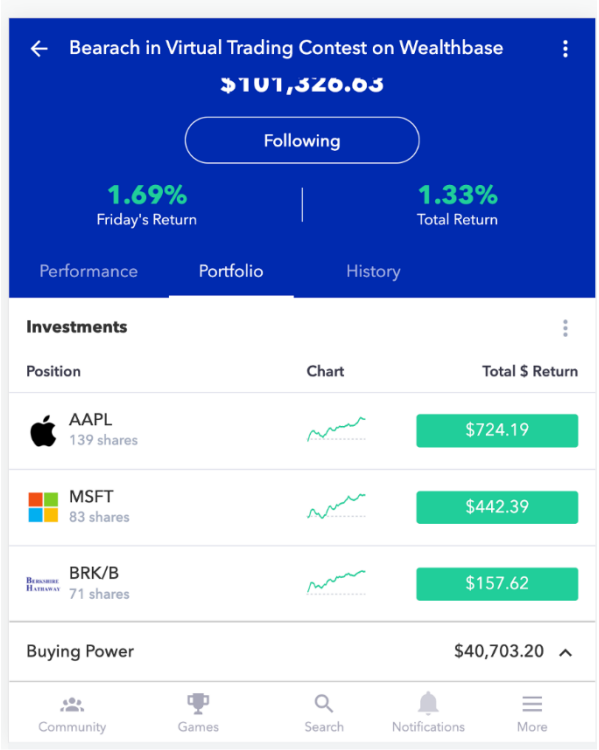


Figure 11 - Wealthbase Two

- Easy to use stock price comparison tool, where users can compare stocks, indexes, or whole portfolios

Figure 12 - Wealthbase Three

3.4.1.3 Cons

- Not a lot of information on the stock card itself, just a small graph to show the movement overview
- No real stock discovery section. Everything is found through trading. It takes a while to find a list of stocks.
- Breakdown of stocks unhelpful. It just shows popular stocks and a very few top movers. No place to see all stocks or see a breakdown by category.

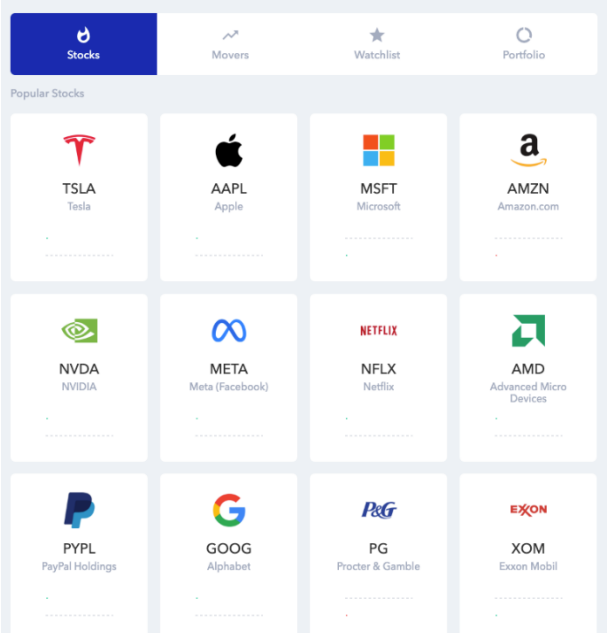


Figure 13 - Wealthbase Four

3.4.2 Investopedia Simulator

3.4.2.1 Overview

Investopedia is a financial information website, which hosts definitions, articles, advice, reviews and more about various financial topics. On this platform they also have the Investopedia Simulator, an investment simulator, like above, which allows users to enter their portfolio into a league or set up their own league. The Investopedia Simulator can be found at https://www.investopedia.com/simulator.

3.4.2.2 Pros

- The site has lots of information, articles and learning resources about investing.
- The simulator also has some relevant articles at the bottom of some screens

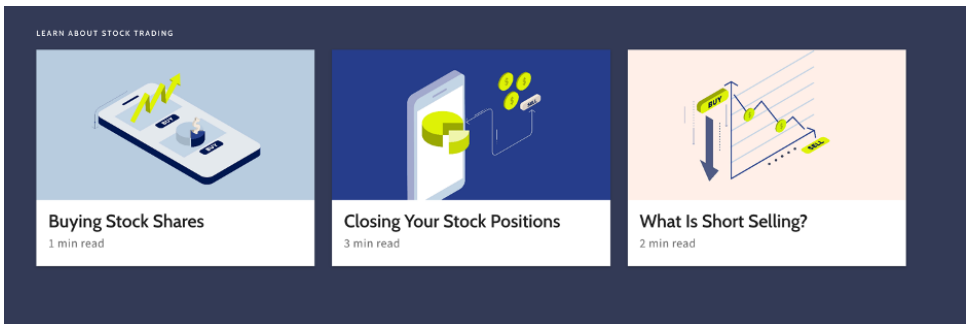


Figure 14 - Investopedia One

- Main stock price graph is quite easy to use, good filtering functionality and user-friendly comparison functionality

Figure 15 - Investopedia Two

- Easy to join public leagues and games. You are added to the overall league by default.
- Lots of game customisable game rules

3.4.2.3 Cons

- UI is dated and unattractive

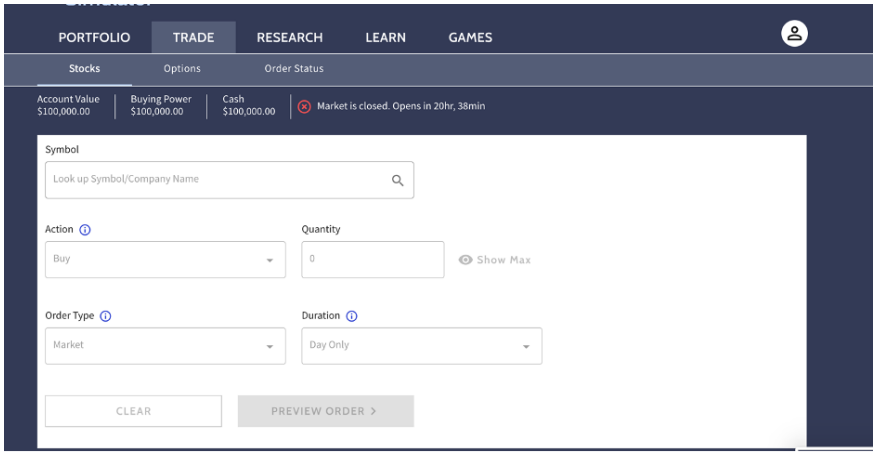


Figure 16 - Investopedia Three

- Breakdown of stocks is quite text heavy and there are no tooltips to explain the terms

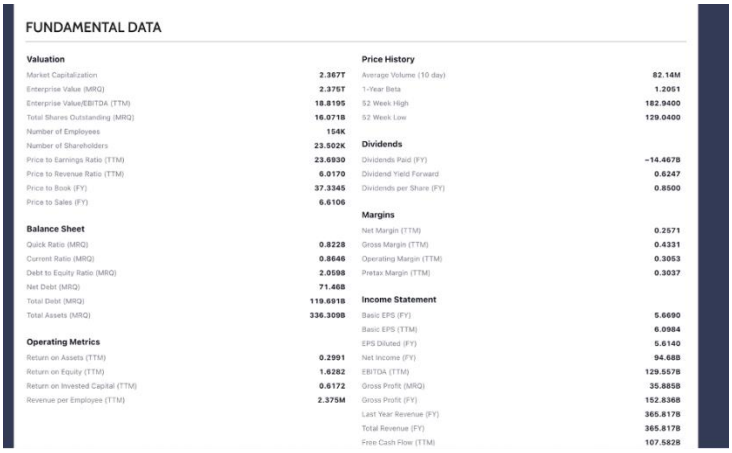


Figure 17 - Investopedia Four

- The stock price graph is quite busy with technical jargon
- To search for stocks, the user will need to know the name of the stock or company as there is no stock discovery feature. There is a list of all stocks in the “Research” section, as well as a list of several most traded stocks. There is no indication over what period these are the “most traded”

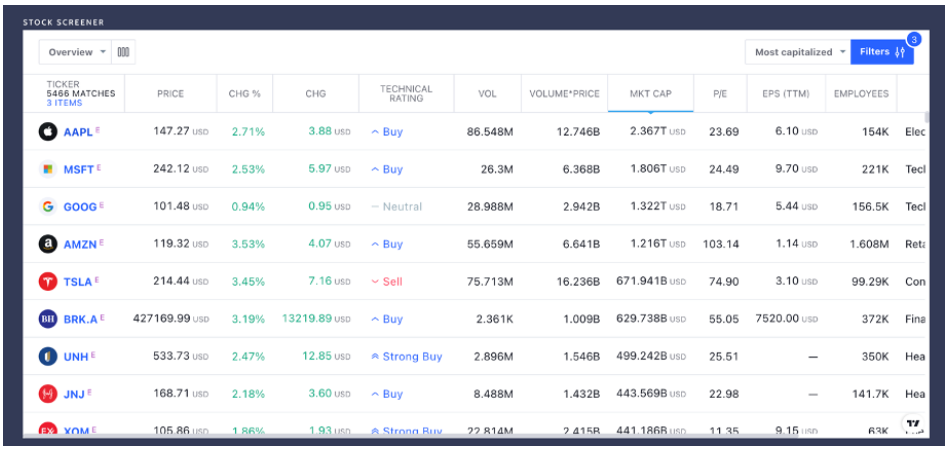


Figure 18 - Investopedia Five

3.5 Analysis of Comparisons

Both systems reviewed above possess aspects which our application will incorporate. The Wealthbase simulator is more like our application. In both cases, there is less of an emphasis on stock discovery and more of an assumption that the user knows for what they are searching. While both are suitable for beginners, there is potential for FinOptimise to have a more accessible UI, which does not alienate them.

ESG information, a stock recommendation system and overall media sentiment are among the features unavailable on these sites. While these are not cons as such, they are aspects which elevate our application beyond a simple stock trading simulator.

4 Technical Solution

This section explains how the technical problem described in *section 3* will be solved. The subsection *4.1* explains what the system does, section *4.1.5* explains how the components of the system work and the architecture of the system, finally section *4.3* gives an overview of data resources and data collection in the project.

4.1 System Overview

The FinOptimise application will be a full stack, mobile-first web application which allows the user to learn about stock market trading through a stock trading multiplayer game. Additional sources of data will also be available to the user in the application to inform their choices on buying/selling stocks.

4.1.1 Stock Leagues

A central component of this application will be the stock leagues available to the user to gamify the experience. A user will be able to join an existing league or create a new league to which they can invite other players to participate. Leagues will have custom rules defined by the user upon creation of the league. Each member of a league has a portfolio associated with that league. The portfolio has a balance which the user can spend buying stocks. The balance of the portfolio will then go up or down depending upon stock performance, with each user's portfolio balance being ranked on a league leader board.

4.1.2 Stock Discovery

Users can browse stocks through the stock discovery section. This section breaks down stocks into given categories to make it easier for the user to browse stocks. There is also the option for a user to search for specific stocks within a given category, for example company name or current price.

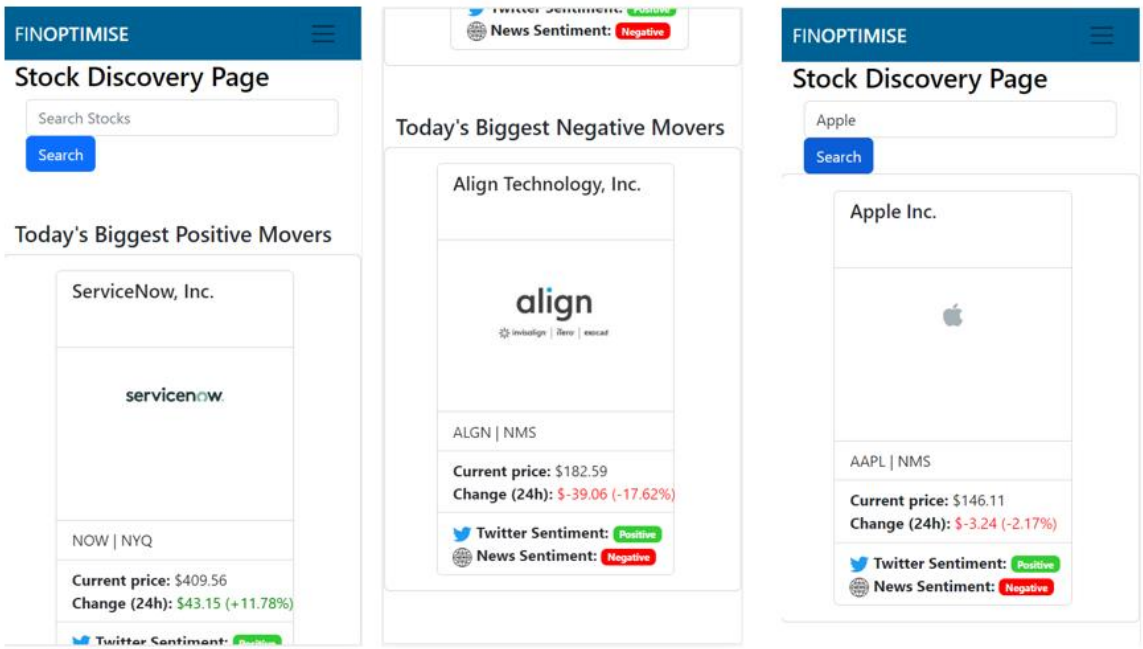


Figure 19 - Stock Discovery Mobile View

4.1.3 Stock Information

Once the user has discovered a stock through the stock discovery page, they can then learn more about the stock by visiting the individual stock page. The individual stock page is designed to give the user as much information about the stock without overwhelming them so they can decide to buy shares in this stock or not. The information provided is the stock price, environmental, social and governance (ESG) ratings, related news sentiment and related twitter sentiment. Additional information on certain financial terms should be available to the user through info icons

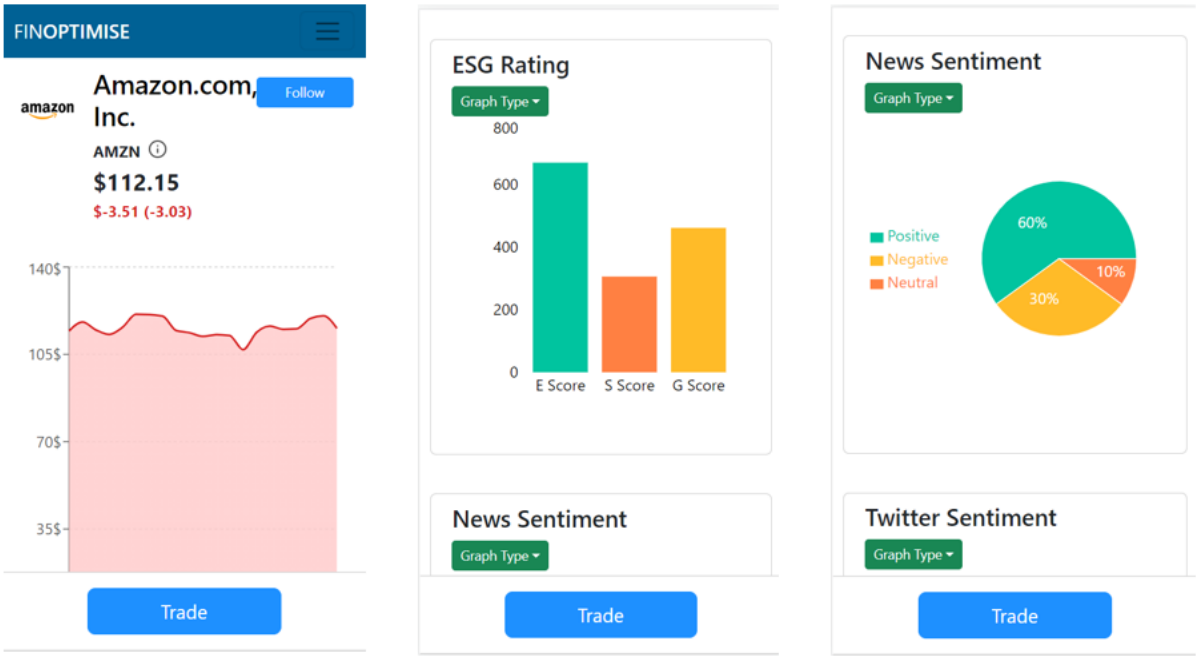


Figure 20- Individual Stock Page Mobile View Part One

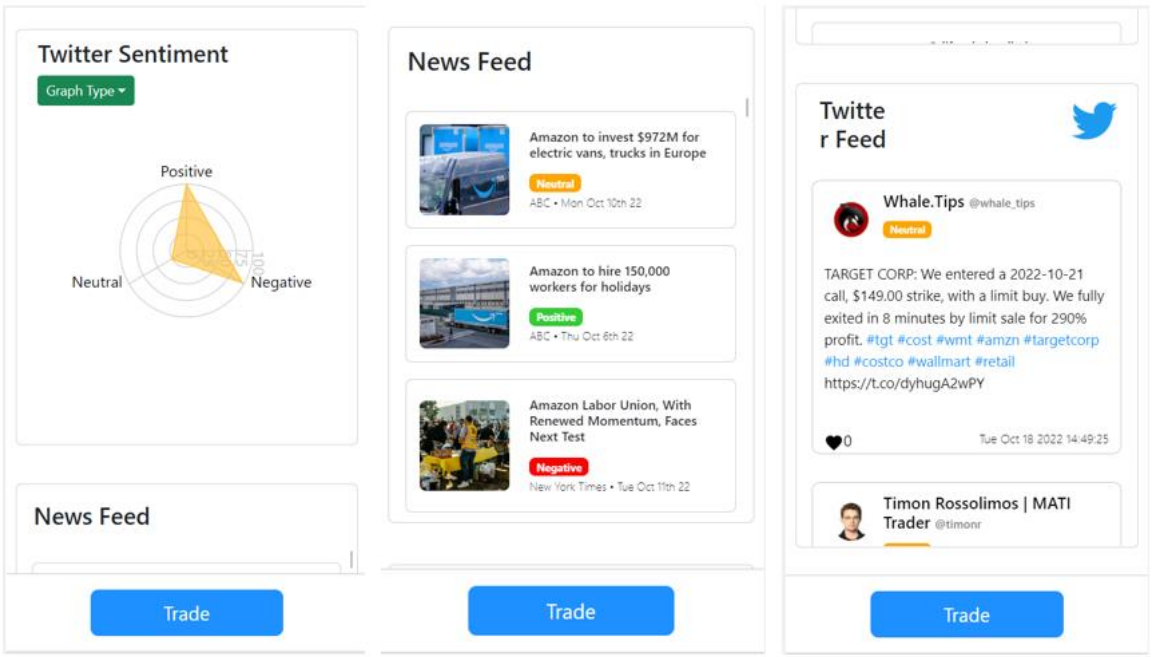


Figure 21 - Individual Stock Page Mobile View Part Two

4.1.4 Stock Purchase/Sale

A user can then decide if they want to purchase the stock or, if they already hold shares of the stock in their portfolio, to sell these shares. The order types available to the user will be a market buy/sell or a limit buy/sell order.

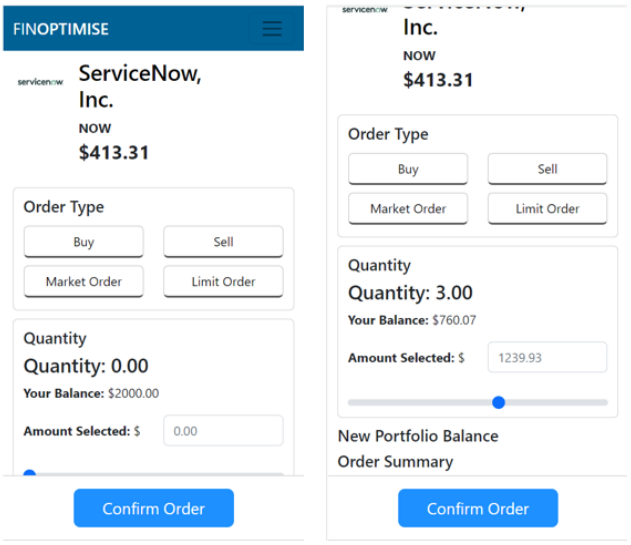


Figure 22 - Confirm order Mobile View

4.1.5 Stock Recommendations

Based upon how the user interacts with the system, the stock recommendation model will recommend stocks the user may be interested in. The stock recommendation model will not suggest stocks which will be profitable for the user, only stocks which the user may be interested in to customise the experience. The recommendation system will recommend stocks to the user based upon the stocks in which they are currently invested, similar stocks will be recommended to the user.

4.2 System Workings and Architecture

The user will interact with the application through a web browser. They first must register an account within the application, then they can login and use the application. If a user wants to begin trading, they can then choose to join an existing default league created by the application admins or create a league of their own. If the user opts to create a league, they will configure the league rules and timeline for the league.

Once the user has either created or joined a league, a portfolio will be created for them with a balance to begin trading stocks within that league. They can then browse stocks through the stock discovery screen and begin to buy/sell stocks.

4.2.1 Basic Architecture

Starting with the simple overview of the architecture, web application is a MongoDB, Express JS, React and NodeJS (MERN) stack with Python being the technology for machine learning components and data ingress. The react frontend sends HTTP requests to the Express JS REST API backend which then queries the Mongo Database for data which is sent back to the React frontend. The Python data ingress and machine learning components update the data within the database.

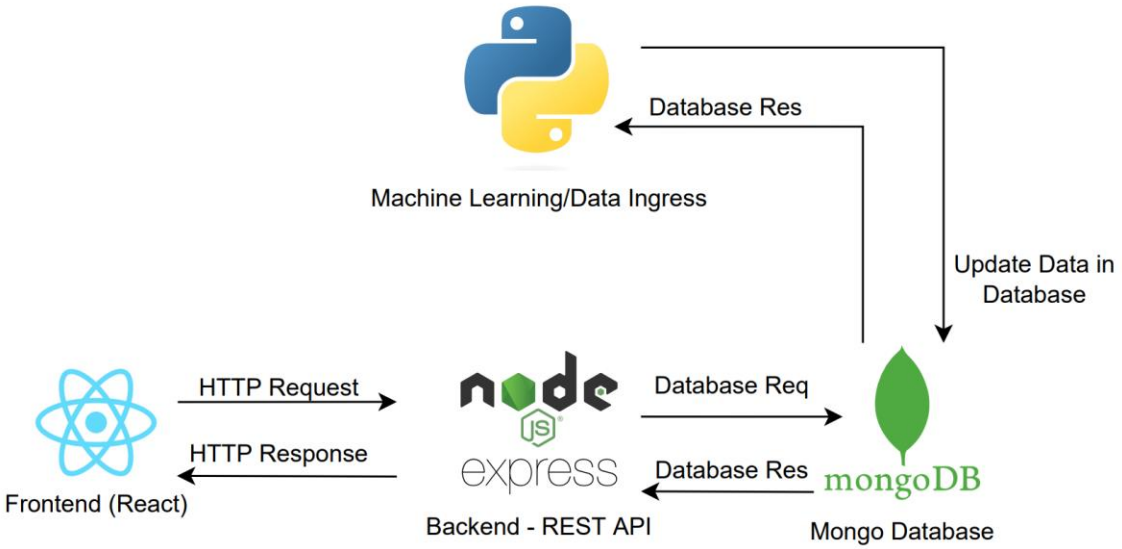


Figure 23 - Simple System Architecture

Once the basic architecture was finalised, the cloud architecture was then worked on. From research between the team and the prerequisite learning list supplied by the lecturers it was decided to go with the service AWS Amplify (Amazon Web Services, Inc., 2022) to host the web application. The basic architecture designed was then tailored to the services an AWS Amplify project provides.

AWS Amplify was chosen as it enables simple GitHub integration. As soon as a pull request is approved and merged to the master branch on GitHub it is deployed to AWS Amplify cloud. This means deployment is continually monitored and is not an afterthought towards the end of the project. Small deployments are continually pushed to the master branch and if a deployment fails it can be dealt with immediately. Another reason AWS Amplify was chosen is due to its serverless architecture. The serverless architecture means no management of servers and load balancers for hosting the frontend, REST API and machine learning/data ingress components of the system. An alternative service researched for hosting was AWS Elastic Beanstalk. This service is not serverless and includes the management of servers and load balancers which take away from time spent writing code. It also has a cost benefit; for the serverless AWS Lambda functions code execution is the only charge meaning when the team are not executing code on the cloud there is no cost unlike keeping a server running. For the database, a server was required to run it, as we opted for MongoDB. The alternative considered was DynamoDB which is commonly used with AWS Amplify, it is only a simple key-value database and for this reason it was not chosen. An AWS S3 bucket is used to store image files which are used for things such as company stock logos within the application.

The cloud architecture incorporating the services AWS has to offer is shown below:

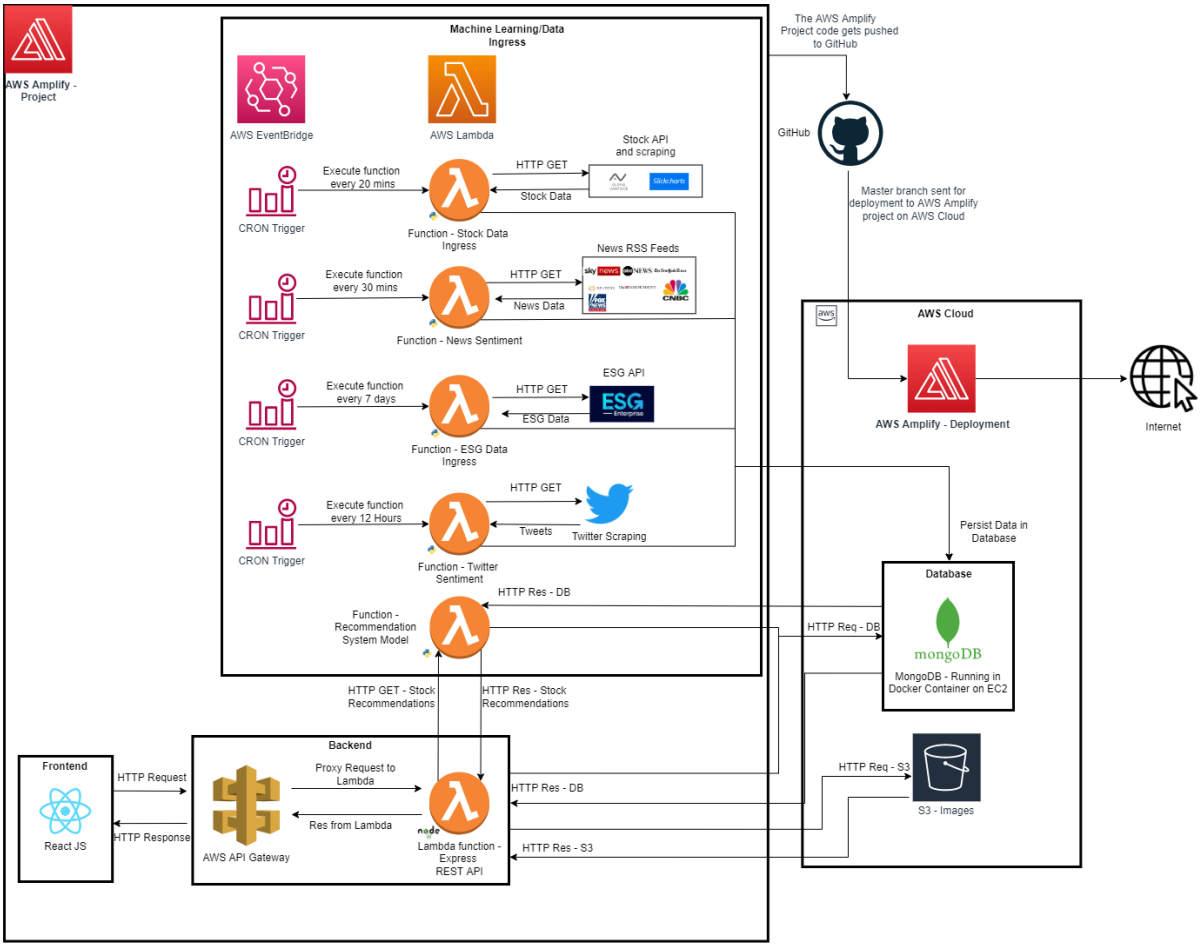


Figure 24 - Cloud Architecture

The system can be broken down further into three main components.

1. Frontend
2. Backend
3. Data/Machine Learning

Justification for the technologies used in each component of the system are explained in the next sections.

4.2.2 Frontend Technologies

The frontend technologies used can be broken down into three key areas, ReactJS, Redux and styling components.

4.2.2.1 React JS

ReactJS is a frontend library which was chosen for building the system.

4.2.2.1.1 Familiarity

Each member of the team has experience with React. Given the project's short time frame, the learning curve of using another library was factored into the decision. Other libraries, such as Vue.js, were investigated but they did not offer any clear advantage over ReactJS. The syntax of React is also intuitive and each member of the team has experience with JSX.

4.2.2.1.2 Component Based Architecture

React has a component-based architecture where the user interface can be broken down into reusable chunks of code. This has the advantage of making code reusable throughout the project. Each individual page has a file within the "Screen" directory. These screens act as containers and hold multiple individual components which are the reusable components. This architecture also makes the division of work between the team easier as it can be broken down per component.

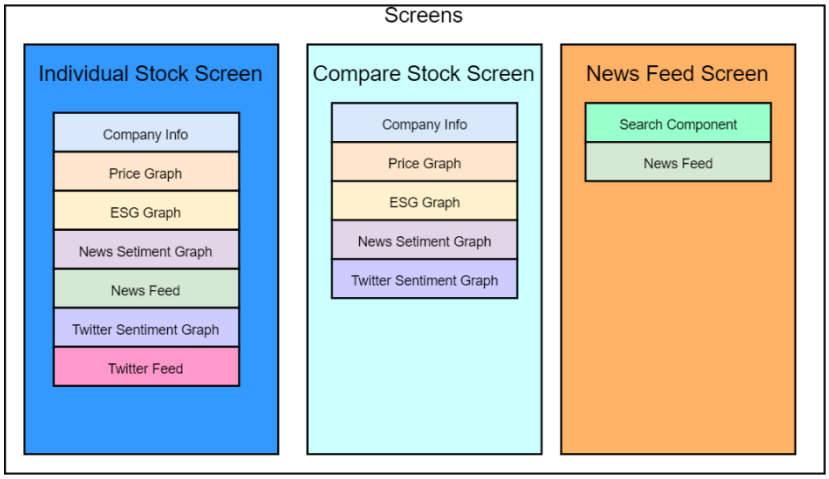


Figure 25 – ReactJS Reusability between screens

4.2.2.1.3 Improved Performance

Within traditional multipage applications, when a user wants to navigate to a new page a request is sent to the server and a new HTML page is rendered to the document object model (DOM). In React, single page applications (SPA) can be developed where there is a single HTML document which is conditionally updated. The benefit of a single HTML page is that there are no full page reloads and server requests do not have to be sent to render new HTML pages. Server requests are only sent for data used in the view. React makes use of a concept called the virtual DOM (Codecademy, 2022) to update the real DOM. Updating the real DOM is expensive, so React saves a copy of it which is the virtual DOM. Once a component gets updated React updates the virtual DOM first which is not an expensive task and compares it to the real DOM, this is called diffing. Only the components in the real DOM which have changed get updated. Not every component in the DOM gets updated which results in faster rendering for a better user experience.

4.2.2.2 Redux

Redux (Abramov, Redux, 2022) is the package used for global state management within the frontend React app. It was decided to use Redux as it solved the problem of multiple components requiring the same piece of state, for example a user details, in the application and passing this state down through individual component props (Abramov, When should I use Redux?, 2022). Alternatives looked at were using Reacts component level state, however, this would be difficult to manage as the application grows. Another alternative looked at was Reacts Context API (Meta Platforms, Inc., 2022), the team were already familiar with Redux which solves the same problem. There also seems to be a greater number of online resources for Redux compared to the React Context API.

4.2.2.3 Styling - Mobile First, Bootstrap and Sass

A mobile first design has been used to develop the application. For the components within the application, React Bootstrap (Bootstrap team, 2022) is being used for general components such as cards, lists and buttons. Each page is laid out using the grid system (Bootstrap team, 2022) in React

Bootstrap which uses flexbox to make the user interface responsive. Breakpoints are set depending upon the size of the display for the number of columns within a particular row. Particular attention is being put into extra-small screen breakpoints of <576px. React Bootstrap does a lot of the heavy lifting, however, custom styling is still required, for this Syntactically Awesome StyleSheets (Sass) is used. Sass was chosen over basic CSS as it offers all the same functionality as CSS but adds additional features which are useful, in particular *variables*. Within the Sass files, variables are used for values such as colours and font-sizes. Updating a single variable which is used throughout the stylesheets makes the managing of stylesheets more maintainable as they grow. Relative units are also being used within the stylesheets for custom styling to support a responsive design.

4.2.2.4 Screen Mock-ups

Screen mock-ups have been created using FigmaAppendix A - Front-end Prototypes. The table below lists the prototypes created and their locations in the appendix.

Prototype	Section
Evilometer	8.1.1 ESG Ratings – "Evilometer" – Static Image Prototype
Register Form	8.1.2 Registration Form – Static Image Prototype
Stock Discovery Page	8.1.3 Stock Discovery Page – Figma Prototype
Portfolio Page	8.1.4 Portfolio Page – Figma Prototype
Individual Stock Page	8.1.5 Individual Stock Page – Figma Prototype
Confirm Order Page	8.1.6 Confirm Order Page – Figma Prototype
Stock League	8.1.7 Stock Trading Leagues – Figma Prototype

Table 1 - List of screen mockups

Figure 26 - Desktop View Stock Page Part One

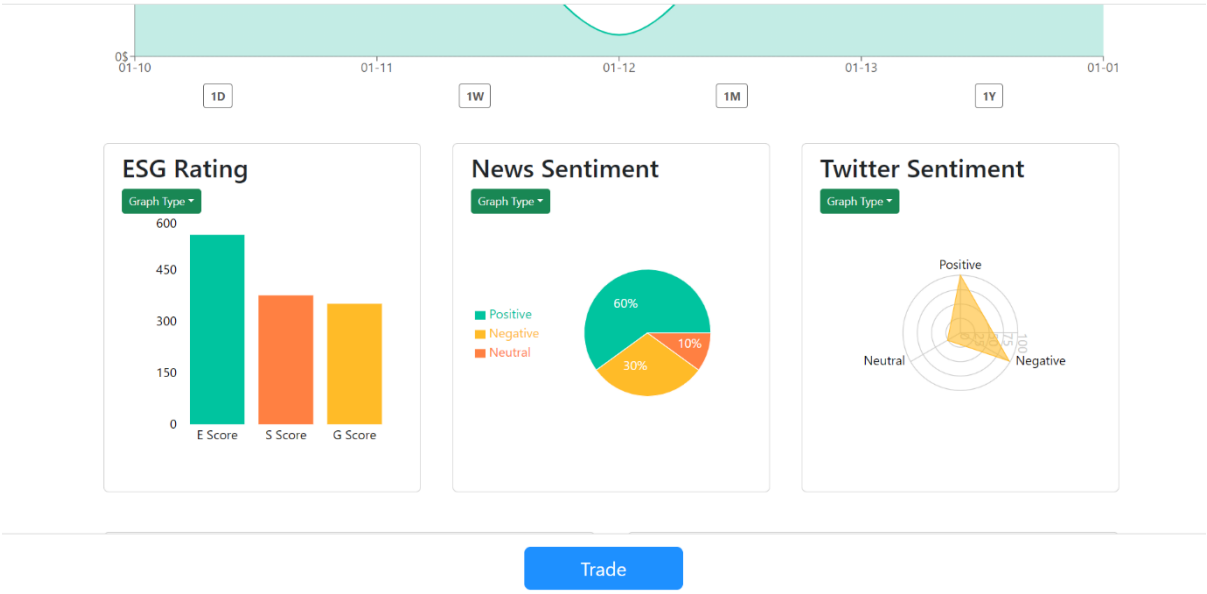


Figure 27 - Desktop View Stock Page Part Two

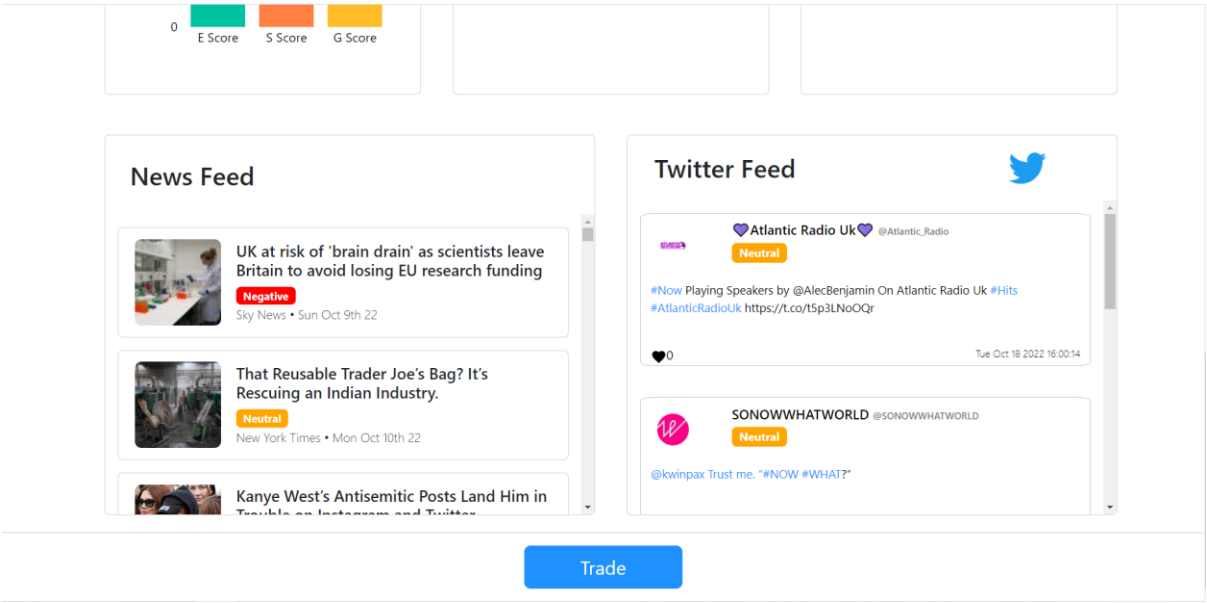


Figure 28 - Desktop View Stock Page Part Three

4.2.3 Backend Technologies

The backend technologies can be broken down into AWS Lambda, NodeJS, ExpressJS and API Gateway, Python, AWS EventBridge, MongoDB, and AWS S3.

4.2.3.1 AWS Lambda

A large part of the backend relies on the service AWS Lambda (Amazon Web Services, Inc, 2022). AWS Lambda is a serverless, cloud computing service which enables code to be run in the cloud without managing servers. Within AWS Lambda every piece of executable code is divided into a function. Functions are isolated execution environments with language support for Java, Go, PowerShell, Node.js, C#, Python and Ruby. Functions are configurable with their own environmental variables, memory allocation, security settings and more. Lambda functions are event based and have triggers associated with them. The triggers used within this project are API calls for the ExpressJS Rest API running in a lambda function and CRON-based timing triggers for data ingress.

By design, using AWS Lambda results in a microservice's architecture with the division of executable code into functions and was one of the reasons it was chosen. The alternative investigated was to run a machine learning/data ingress Python server and a server for the NodeJS server for the Express and React App. Managing of full servers is time consuming, but it can also represent single points of failure in an architecture. If the machine learning/data components were fully deployed on one server, if it went down all the machine learning/data ingress components would be offline. In the current architecture, if the *Stock Data Ingress Function* encounters an error there is no concern that the *News Sentiment* function or any other function will go offline as they are isolated environments. Debugging a single isolated function is also less time consuming then SSH'ing into a server to try and debug code. Functions can also be scaled independently, for running of the machine learning models these functions can be configured to have increased CPU power and memory individually instead of having to scale a full server instance for the sake of one component. Lambda costs less than a traditional server as charges are not incurred when the function is not executing.

4.2.3.2 Amplify CLI

The Amplify command line interface (CLI) is used extensively throughout the development within this project. When a new resource needs to be added such as a new Lambda function it is done so through the CLI. The CLI then generates *cloudformation-templates* which are configuration files. These are committed to GitHub and used in the deployment to tell the AWS Amplify project hosted on the AWS cloud what resources to deploy for the project. The benefit of using the CLI to provision resources instead of a browser interface is that the *cloudformation-templates* are kept track of in GitHub. Previous deployments of the system can be reverted to if a deployment fails.

4.2.3.3 NodeJS, Express JS and API Gateway

ExpressJS (OpenJS Foundation, 2022) was chosen for the API framework. The API uses the Representational state transfer (REST) architecture style to communicate with the frontend React application sending information in JSON (JavaScript Object Notation) format. Each team member is familiar with Express and the REST architecture contributing to why it was chosen. Another reason why it was chosen is because of NodeJS's default asynchronous architecture. When performing blocking tasks like database calls they can be awaited upon, so the system does not hang. Alternatives investigated were Pythons Django. However, keeping the same language between the frontend and backend was opted for as the same packages if required could be used through Node Package Manager (npm).

The Express application is hosted in AWS Lambda. As mentioned in the previous section, each Lambda function has a trigger associated with it. For the Express API, the trigger is AWS API Gateway. AWS API Gateway is a service in its own with many features but the only function it is used for within this project is to proxy requests to the Lambda function hosting the Express App. AWS API Gateway acts as the Express apps gateway to the internet. Within the AWS Lambda function the express application is wrapped in serverless express (Vendia, 2022) wrapper to enable it to run inside a Lambda function. Express handles all the routing of requests using the Express router and custom middleware is defined for authentication and error handling. An Express API can be initialised within an Amplify Project through the Amplify cli.

```
PS C:\Users\warre\OneDrive\Documents\College\Msc\FinalYearProject\AWS\Amplify\stockApp> amplify add api
? Select from one of the below mentioned services: REST
√ Would you like to add a new path to an existing REST API: (y/N) · no
√ Provide a friendly name for your resource to be used as a label for this category in the project: · ExpressAPI
√ Provide a path (e.g., /book/{isbn}): · /api
√ Choose a Lambda source · Create a new Lambda function
? Provide an AWS Lambda function name: ExpressFunction
? Choose the runtime that you want to use: NodeJS
? Choose the function template that you want to use: Serverless ExpressJS function (Integration with API Gateway)

Available advanced settings:
- Resource access permissions
- Scheduled recurring invocation
- Lambda layers configuration
- Environment variables configuration
- Secret values configuration

? Do you want to configure advanced settings? No
? Do you want to edit the local lambda function now? No
Successfully added resource ExpressFunction locally.
```

Figure 29 - Creating an Express API through AWS Amplify CLI

4.2.3.4 Python

Python was chosen for any data and machine learning scripts within the backend. This includes the scraping of data and training machine learning models. The reason for this is that data manipulation is easier with modules such as Pandas (NumFOCUS, Inc. , 2022) and NumPy (NumPy, 2022). For developing machine learning models it also offers the natural language toolkit (NLTK) module (NLTK Project, 2022) used for Twitter and News Sentiment analysis along with the Scikit-learn module (Cournapeau, 2022) used in model training. Alternatives looked at were NodeJS however the modules for data driven work did not have as much documentation/weren't as well maintained as the Python modules.

4.2.3.5 AWS EventBridge

AWS EventBridge is a service which allows for the creation of events for event driven applications in the cloud through its rule system. An option for rules is running them on a recurring schedule which is used within this project, it can be specified using a CRON syntax. Associated with a rule is a target, in this project the targets are AWS Lambda functions. There are four rules within this project:

1. **Stock Data Rule** – Runs the stock data scraping lambda function every 20 minutes Monday to Friday.
2. **ESG Data Rule** – Runs the ESG data scraping lambda function every 7 days.
3. **News Article Rule** – Runs the News scraping and sentiment classification Lambda function every 30 minutes.
4. **Twitter Rule** – Runs the Twitter scraping and sentiment classification Lambda function every 12 hours.

```
PS C:\Users\warre\OneDrive\Documents\College\Msc\FinalYearProject\AWS\Amplify\stockApp> amplify add function
? Select which capability you want to add: Lambda function (serverless function)
? Provide an AWS Lambda function name: NewsSentimentfunction
? Choose the runtime that you want to use: Python
Only one template found - using Hello World by default.

Available advanced settings:
- Resource access permissions
- Scheduled recurring invocation
- Lambda layers configuration
- Environment variables configuration
- Secret values configuration

? Do you want to configure advanced settings? Yes
? Do you want to access other resources in this project from your Lambda function? No
? Do you want to invoke this function on a recurring schedule? Yes
? At which interval should the function be invoked: Minutes
? Enter the rate in minutes: 30
? Do you want to enable Lambda layers for this function? No
? Do you want to configure environment variables for this function? No
? Do you want to configure secret values this function can access? No
? Do you want to edit the local lambda function now? No
Successfully added resource NewsSentimentfunction locally.
```

Figure 30 - Creating Lambda through Amplify Cli

4.2.3.6 Database – MongoDB

The database used is the NoSQL document database MongoDB. One of the main reasons MongoDB was chosen was due to its non-strict schema. As this project moves along the schema will inevitably change as the application evolves and certain aspects change. Not having to worry about updating entire tables like in a SQL database when the schema changes in the project is a big advantage. The documents within MongoDB are by default stored in a JSON format which ties in well with manipulating data in the REST API backend using NodeJS and JavaScript in the frontend as they can be stored as objects. MongoDB supports horizontal scaling meaning data can be partitioned across multiple servers providing a solution for scaling. Most production databases either SQL or NoSQL have a solution for scaling. The main reason MongoDB was chosen is due to how it fits the use case of the project. One disadvantage of MongoDB is the lack of relationships between data and JOIN queries which are present in SQL databases. However, a workaround for this is using references (MongoDB, Inc, 2022) within Mongoose schemas (Mongoose, 2022) which link document id's together to create relationships. A single MongoDB instance is deployed on an EC2 instance (Amazon Web Services, Inc., 2022) on AWS running in a docker container. The database credentials are securely stored using the service AWS Systems Manger Parameter Store (Amazon Web Services, Inc., 2022).

4.2.3.7 S3 – Images

AWS Simple Storage Service (S3) is a service which allows for objects to be stored in the cloud. It can be accessed through the AWS console and object files uploaded to S3 buckets. It was decided to store image files used within an S3 bucket due to ease of use for uploading images to it. These images include company logos for stocks, default news source logos and profile pictures. In the MongoDB documents a reference to the image URL is stored which is the location of the image file in the S3 bucket. An alternative to this is storing images directly in the MongoDB database however there are a lot of image files used in this project. It was decided to store the images in S3 to not use storage space in MongoDB and to reduce the size of data returned from database calls in the backend.

4.3 Data

4.3.1 Data Sources

The data used within this project can be broken down into the categories listed below. The exact data sources for these categories can be found in *Table 9 - Data Sources in Appendix B – Table of Data Sources.*

1. **Stock Data**
    a. Real time – Current up-to-date stock market prices.

b. Historical – Historical stock market prices of company and company information.

2. **ESG Data**
    a. Real Time – Current Environmental, Social and Governance scores of companies.
3. **News Data**
    a. Real time – This is current up-to-date financial news from Really Simple Syndication (RSS) feeds. Some RSS feeds updated more frequently than others.
    b. Training Data – For training the news sentiment classifier.
4. **Twitter Data**
    a. Real Time – This is current up-to-date financial tweets scraped from Twitter.
    b. Training Data – For training the Twitter sentiment classifier.
5. **Stock Recommendation System Data**
    a. Real Time – Stock recommendations are based upon how a user interacts with the system and what similar stocks they have in their portfolio.

4.3.2 Data Collection and Storage

The data collection is run through Python scripts apart from the stock recommendation system which is collected by users using the system.

For real time data, the scripts for data collection are run using AWS Lambda functions. These AWS Lambda functions are triggered using the AWS EventBridge rules on a recurring schedule explained in the previous section. The functions are all written in Python 3.9 and use the PyMongo (MongoDB, Inc., 2022) library to communicate with the MongoDB database.

4.3.2.1 Stock Data

The historical stock data is scraped once using a Python script to populate the database with historical stock values. This data contains daily market close prices for companies in the S&P500. It also contains information about the company including name, industry, market cap, etc. The amount of historical price data depends upon the company, newer companies have less data available. The real time stock prices are scraped every 20 minutes from the Slickcharts website (Slickcharts, 2022) using a Python script running in an AWS Lambda function. The stock data is stored in the MongoDB document Stocks.

4.3.2.2 ESG Data

The ESG data is collected every 7 days from the ESG Enterprise API. ESG data is collected less frequently as it does not change very often. The data consists of a score for environmental, social and governance rating of the company along with a total score. This data is stored within the Stocks document as it is related to a particular stock.

4.3.2.3 News Data

The training data for training the news sentiment classifier is collected through a web browser by downloading csv files and storing them in the GitHub repository. The real time data for news is scraped from RSS feeds which news companies make freely available to use. RSS feeds can be displayed on websites if they link back to the original news source. The attributes stored on each news article are a headline, source, URL, category, description, image link, date, and sentiment. News articles are scraped from 32 RSS feeds every 30 minutes. The script uses the Python library *feedparser* (McKee & Pilgrim, 2020).

4.3.2.4 Twitter Data

The training data for the Twitter sentiment classifier is stored in GitHub. The real time data for Twitter is scraped from Twitter using the Python library snscrape (JustAnotherArchivist, 2022). The tweets are currently filtered by searching for the stock symbol within the tweet. The attributes stored on a tweet

are *tweet id, isVerified, displayName, imageUrl, stock, date, username, content, sentiment* and *like count*. This data is stored in the Tweets document within MongoDB. Twitter data is scraped every 12 hours with 3 tweets being scraped for each of the 500 companies in the database.

4.3.2.5 Stock Recommendation System Data

This data will mainly be based upon the Stocks within the users Portfolio document in MongoDB. The recommendation system will be a content-based recommendation system (Google Developers, 2022) which recommends stocks to the user with similar attributes to stocks they have in their portfolio. The features that will be used for scoring similarity are historical price, exchange, sector, industry, current price, market cap, revenue growth, ESG ratings, number of employees and weight in S&P 500.

5 Evaluation

The evaluation of the project is broken into four components: overall project evaluation, user evaluation, testing, and intelligent aspects of the application.

5.1 Overall Project Evaluation

Project evaluation is an extremely important aspect of project management that is frequently overlooked, despite the importance and need for it (Will, 2016). In his article on how to assess agile projects, Brian Will offers a relatively straightforward method of doing this. By breaking the agile project down into seven areas, each area then having between three and ten specific assessment items. These items must then be given a rating between 1-5 by the team with respect to:

Rating	*Description*
1	Never followed / not adhered to / not implemented
2	Rarely followed
3	Inconsistently followed or done incorrectly; role, artifact, process, or best practice is inconsistently implemented or followed, or used incorrectly (for example, calling regular meeting "Daily Scrums")
4	More of less consistently followed; mostly implemented role, artifact, process, or best practice
5	Consistently followed in the spirit of Agile / Scrum Development; fully implemented role, artifact, process, or best practice the way it is intended in Agile / Scrum

Figure 31 – Rating system description

The below image, taken from Will's blog post, provides a view of what the final spreadsheet might look like:

#	Area	#	Item	Target Value	Current Value
1	Roles & Responsibilities	1	Product Owner	5	1
		2	Development Team	5	2
		3	Scrum Master	5	1
2	Planning & Estimation	4	User Stories Defined	5	2
		5	Estimation Poker or T-Shirt Sizing	5	3
		6	Velocity Tracking	5	2
3	Artifacts	7	Product Backlog	5	1
		8	Sprint Backlog	5	3
		9	Product Increment(s)	5	3
		10	Product Vision	5	2
		11	Product Roadmap	5	2
		12	Release Plan	5	1
4	Process - Scrum	13	Sprint(s)	5	2
		14	Sprint Planning	5	1
		15	Daily Scrums	5	3
		16	Sprint Review(s)	5	1
		17	Sprint Retrospective(s)	5	2
5	Process - Engineering Best Practices	18	Good Design	5	3
		19	Good Technical Architecture	5	2
		20	Common Programming Practices	5	1
		21	Appropriate Level of Documentation	5	2
		22	Automated Unit Tests	5	3
		23	Automated Functional Tests	5	2
		24	Continuous Integration	5	1
		25	Automated Test Status	5	2
		26	Automated Regression Testing	5	3
6	Delivery Effectiveness	27	Working Software after each Sprint	5	2
		28	Working Software Releases	5	3
		29	User Acceptance Testing	5	2
7	Organizational Support	30	Marketing	5	1
		31	Sales	5	2
		32	IT	5	3
		33	Stakeholder Involvement	5	2
		34	Agile Coach Participation	5	1

Figure 32 – Sample finished evaluation matrix (Will, 2016)

This was updated by the team, with irrelevant items (such as marketing and sales) removed. A second evaluation column was added for week 9 using sample values, this will tell us whether our ratings increase or decrease compared to the previous evaluation.

#	Area	#	Item	Target Value	Evaluation 1 - Interim (Week 6)	Evaluation 2 - (Week 9)
1	Roles & Responsibility	1	Product Owner	5	4	4
		2	Development Team	5	4	4
		3	Scrum Master	5	4	4
2	Planning & Estimation	4	User Stories Defined	5	2.5	3.5
		5	Estimation	5	2	2
		6	Velocity Tracking	5	2.5	4
3	Artifacts	7	Product Backlog	5	3	3
		8	Sprint Backlog	5	3	3
		9	Product Increment	5	4	4
		10	Product Vision	5	3.5	3.5
		11	Product Roadmap	5	2.5	2.5
		12	Release Plan	5	3	3
4	Process - Scrum	13	Sprints	5	4	4
		14	Sprint Planning	5	4	4
		15	Daily Scrums	5	5	2
		16	Sprint Reviews	5	4	4
		17	Sprint Retros	5	5	5
5	Process - Engineering Best Practices	18	Design	5	3	3
		19	Technical Architecture	5	3	3
		20	Programming Practices	5	3.5	3.5
		21	Documentation	5	3.5	3.5
		22	Automated Unit Tests	5	0	3
		23	Automated Functional Tests	5	0	3
		24	User Evaluation	5	5	3
6	Delivery Effectiveness	25	Working Software After Sprint	5	3	3
7	Organisational Support	26	Presentations	5	4	4
		27	Deployment	5	3.5	4
		28	Stakeholder Involvement	5	5	5

Figure 33 - Interim Evaluation for the project

From this, the team can generate a radar chart showing how well they have been performing when it comes to the different project areas. The various evaluations will be added to this chart and the team will be able to visually compare performance between evaluations.

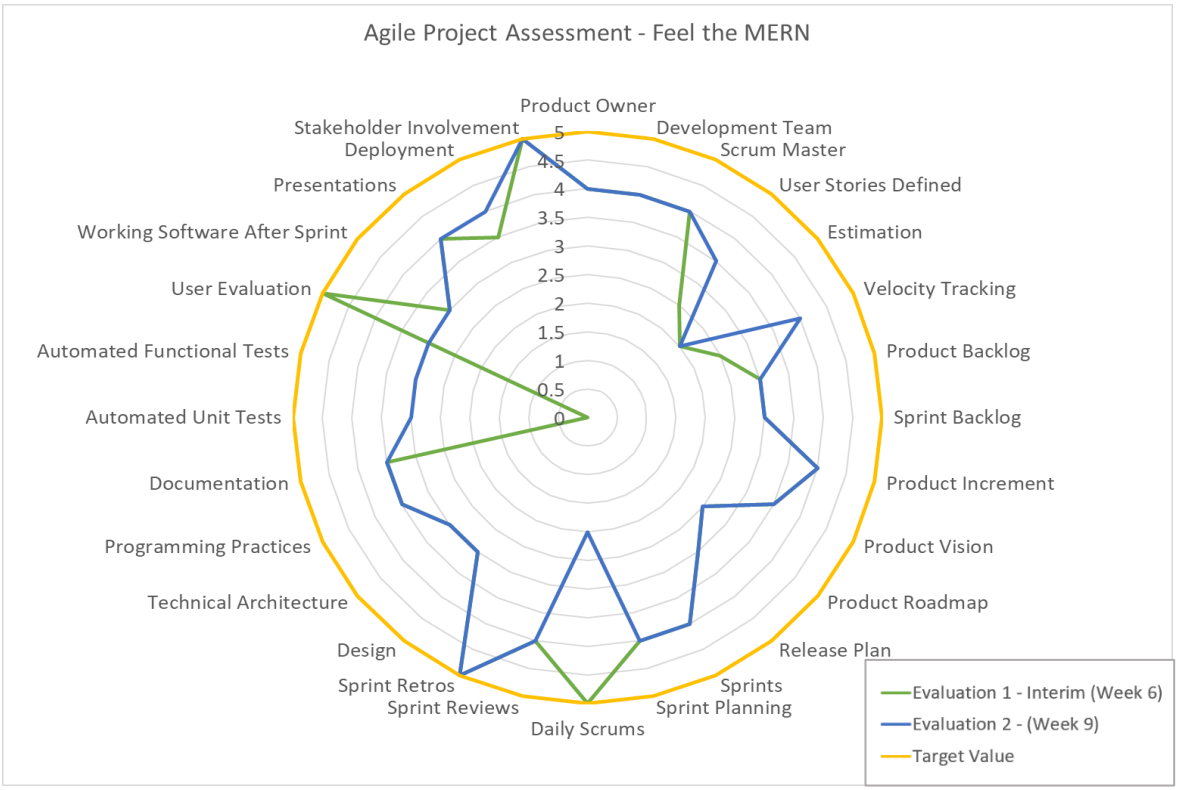


Figure 34 - Radar chart from the team's first evaluation

This method of overall project evaluation will be carried out towards the end of the project and will be included in the final report along with a discussion/analysis about the team's findings.

5.2 User Evaluation – Prototypes

This section will look at some of the user evaluation that has been carried out to date on some of the frontend UI/UX prototype element. This will demonstrate how the team has used user evaluation over the first 6 weeks of the project to shape and change how the development of the project is carried out. This will give a clear view of how the team plans to continue using this process for the duration of the project.

The primary method of user/usability evaluation that was used was that of Cognitive Walkthroughs (LaVond, 2021). This process contains several steps, the below is taken in a modified form from Nielsen Norman Group's guide on "How to Conduct a Cognitive Walkthrough Workshop" (Salazar, 2022):

1. Determine who will participate.
2. Define the inputs.
3. Establish the roles & rules.
4. Gather materials:
    a. The prototype.
    b. The formal action sequences.
    c. The personas that will be used.
    d. Materials for recording outcomes.
5. Conducting the cognitive walkthrough:
    a. Brief the participant(s).
    b. Review the customer persona.
    c. Introduce a task.
    d. Walk through that task.
6. Determine success/failure:
    a. Will users try to achieve the right result?
    b. Will users notice that the correct action is available?
    c. Will users associate the correct action with the result they are trying to achieve?
    d. After the action is performed, will users see that progress is made toward the goal?

While the team has attempted to closely follow this process for every user evaluation, in certain cases some steps have been ignored where it does not impact the overall result of the evaluation/cognitive walkthrough.

This process has been followed for the below evaluations and will continue to be used for user evaluations going forward into other system components, and eventually for the finished system.

5.2.1 ESG Ratings – "Evilometer" – For Prototypes See Appendix A – Section 8.1.1

Evaluation Number	1
Link to Figma	Done using diagrams.net
Experience in investing	None
Pros	• "[the new Evilometer prototype is] Much easier for to understand at a glance. I have no idea what the other one is showing me" • The addition of the limit images aided the user in understanding ESG ratings.
Cons	• "It reminds me of a pain scoring sheet in a hospital" • The lack of values on the Evilometer is confusing. No reference for scale.
Comments	Any other comments
Changes to be made	The decision was made to carry on with development of the Evilometer prototype

Table 2 - Evaluation for ESG Ratings presentation prototype

5.2.2 Registration Form Page – For Prototypes See Appendix A – Section 8.1.2

Evaluation Number	1
Link to Figma	Done using diagrams.net
Experience in investing	None
Pros	• Newer layout is simpler. • More likely to sign up on the new website due to there being less fields.
Cons	• Question of whether username is needed.
Comments	Any other comments
Changes to be made	Registration page will be redesigned using the new prototype.

Table 3 - Evaluation for registration form page

5.2.3 Stock Discovery Page – For Prototypes See Appendix A – Section 8.1.3

Evaluation Number	1
Link to Figma	https://www.figma.com/proto/gp0aEVcyMb3X1cfhielZJX/Discovery-Page-Prototype-(v1)?node-id=1%3A704&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A704
Experience in investing	None
Pros	• Colour scheme
Cons	• They felt the single column scrolling could be done better.
Comments	They said that they wouldn't like as much vertical scrolling as is needed to "explore" the stocks. They suggested that something along the lines of how Netflix handles suggestions might work better.
Changes to be made	The Figma prototype will be updated based on the Netflix style scrolling suggestion.

Table 4 - Evaluation for stock discovery page

Evaluation Number	2
Link to Figma	https://www.figma.com/proto/MVmRifShILiXcGGaY4V3A8/Discovery-Page-Prototype-(v2)?node-id=1%3A3&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A335
Experience in investing	None
Pros	• Really liked the new method of presenting stocks, i.e., splitting them up by category and horizontally scrolling across those categories.
Cons	• "Cards" were still a bit too big.
Comments	"Overall, a big improvement on the first iteration"
Changes to be made	No changes suggested by the user.

Table 5 - Evaluation number 2 for stock discovery page

5.2.4 Portfolio Page – For Prototypes See Appendix A – Section 8.1.4

Evaluation Number	1
Link to Figma	https://www.figma.com/proto/h7dqeBQfEuMBGLRucKgVv7/Portfolio-Page?page-id=28%3A6&node-id=28%3A7&viewport=942%2C570%2C0.17&scaling=scale-down&starting-point-node-id=28%3A7&show-proto-sidebar=1
Experience in investing	None
Pros	• Clear to see main balance up top of page. • Liked the green for if there was a profit made. • Drop down menu for portfolio and buttons for graph they liked. This was first thing they clicked. • When asked to buy Amazon stock they found the button easily.
Cons	• Writing on the graph is too small and should not be grey. • For the "My Balances" make the euro one standout a bit more so you know that's what you have left to spend. • Maybe make text bit bigger on transaction history but not too much bigger.
Comments	• Liked the overall page few small things to be fixed. • First things they tried to click where the portfolio dropdown and the graph buttons.
Changes to be made	• Make the text on the graph bigger for the axis or make graph more interactive where it displays price when hovering. • Make the euro amount stand out a bit more and make it clear that this is how much you have available spend.

Table 6 - Evaluation for portfolio page

5.2.5 Individual Stock Page – For Prototypes See Appendix A – Section 8.1.5

Evaluation Number	1
Link to Figma	https://www.figma.com/proto/h7dqeBQfEuMBGLRucKgVv7/Portfolio-Page?page-id=28%3A6&node-id=28%3A349&viewport=176%2C437%2C0.38&scaling=scale-down&starting-point-node-id=28%3A349&show-proto-sidebar=1
Experience in investing	None
Pros	• Liked the look of visuals for ESG, News Sentiment and Twitter. • Liked that they change based upon dropdown menu.

	• Liked that there were not too many stock metrics, so it did not feel too overwhelming. • Indications of green for positive, orange for neutral and red for negative was easy to understand. • Trade button there as they scrolled was nice to have and easy to find. • Compare tool and slider easy to use and conveys enough information, liked that clicking on it would go into more detail.
Cons	• On the price graph the writing is too small and should not be grey. • For the stock metrics you need to explain these. • The user was a bit confused on the stock metrics, as they first thought this was about their portfolio as you start with the "My Position" box. It should be clearly stated that the stock metrics are about Amazon in title. • Unsure what ESG means, and a bit confused by it. • What is the ESG score out of? How do I know if this is a good ESG score or a bad ESG score?
Comments	• Liked the general look. • The cons where more about the information on the page and not understanding it.
Changes to be made	• Increase text size on graph axis and do not use grey font colour. • Stock metrics keep them simple but explain them well. • For stock metrics the user liked that were not too many, but they need to be explained. • Mainly was ESG ratings that they were confused by need to say what each one means, what score is it out of and how do I know if this is a good score/bad score.

Table 7 - Evaluation for individual stock page

5.2.6 Confirm Order Page – For Prototypes See Appendix A – Section 8.1.6

Evaluation Number	1
Link to Figma	https://www.figma.com/proto/h7dqeBQfEuMBGLRucKgVv7/Portfolio-Page?page-id=28%3A6&node-id=32%3A1021&viewport=176%2C437%2C0.38&scaling=scale-down&starting-point-node-id=28%3A7&show-proto-sidebar=1
Experience in investing	None
Pros	• The slider for setting the stock amount was good feature. • The visual that displayed leftover account was a good indication, liked colour change. • The order summary they liked.
Cons	• The Buy/Sell buttons up the top, this was confusing as they taught when they clicked this it executed the order. • The confirm order at bottom of page should be there always as they scroll like the "Trade" button in the Individual stock page prototype. • Need to explain what a market order and limit order is.

Comments	• Straightforward design is nice, the user was just a little confused by some elements especially with Buy/Sell/Confirm order.
Changes to be made	• Need to make clear that the “Buy” and “Sell” button at the top is the order type not the actual action of buying and selling. • The confirm order at the bottom of the screen should be there always and then when you click it, it takes you to the “Order Summary” screen/section.

Table 8 - Evaluation for confirm order page

5.3 Testing

It was originally decided by the team for the Project Plan that the team would use Test Driven Development (TDD) (Hamilton, 2022) for the duration of the project, however within the first 2 weeks of the project it became evident that this would not be practical. This was confirmed to the team during feedback from the lecturers with respect to the Project Plan Presentation, with the lecturers stating their concerns about a TDD approach to the project. The main concerns were that the team had little experience with TDD and that it would be too time consuming to apply it to the project which has a timeline of just 13 weeks. It was decided by the team that it would result in a worse application overall.

With this in mind, the team decided to progress with a system of Test-After Development (TAD, sometimes called Test Later Development, or TLD), where code is written and manually tested, with code tests written after the fact (Hexacta Architecture Team, 2009). This development method, while not being the first choice of the team, should help to provide a better application overall due to some of the inherent benefits provided (Hu, 2022):

- Better edge case tests – As the testing is not done until after the code is written, the team will have a better understanding of edge cases relating to the features and as such will be able to write more comprehensive tests.
- Fewer trivial tests – Tests will not be written with the sole goal of passing beforehand, meaning that less time will be wasted writing these trivial/inconsequential cases.
- Shorter timeline – This is the greatest advantage of TAD over TDD, as writing tests after writing the code focuses the development team on the code that is being written without the distraction of switching between writing code and tests constantly.

5.3.1 Accessibility Testing

Accessibility testing will be carried out using WebAIM Wave (Utah State University, 2001).

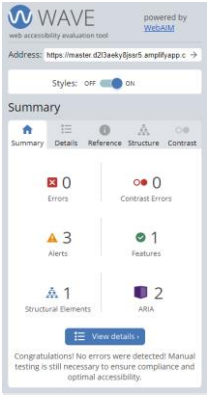


Figure 35 - Sample WebAIM Wave report on an early page from the team's live site

In addition to the above, the guide provided by W3C.org "Easy Checks – A First Review of Web Accessibility" will be used to ensure that the application is developed in an accessible manner (Web Accessibility Initiative, 2017).

5.4 Machine Learning Evaluation

There will be three machine learning components to this project, News Sentiment system, Twitter Sentiment system and the Recommender System. These will all be evaluated in an equivalent manner, using the evaluation phase of the Cross Industry Process for Data Mining (CRISP-DM) process (John Kelleher, 2015). This process consists of six phases (Hotz, 2022):

1. Business understanding – What does the business need?
2. Data understanding – What data do we have / need? Is it clean?
3. Data preparation – How do we organize the data for modelling?
4. Modelling – What modelling techniques should we apply?
5. Evaluation – Which model best meets the business objectives?
6. Deployment – How do stakeholders access the results?

Within the evaluation phase there are three tasks (Hotz, 2022):

1. Evaluate Results
2. Review Process
3. Determine Next Steps

The evaluation for the recommender system will largely take the form of A|B user testing due to the presence of the cold start problem for recommender systems (Milankovich, 2015).

The evaluation of both sentiment analysis systems will be done using a number of metrics, accuracy, precision, and recall. For each of these models, the dataset created will be split into train and test sets in a ratio of 70% / 30%, i.e., 70% of the data will be used for training and 30% will be used for testing.

A Confusion Matrix allows for the model results to be analysed using the above-mentioned metrics, by splitting the results into four categories, True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN). (Sarang Narkhede, 2018)

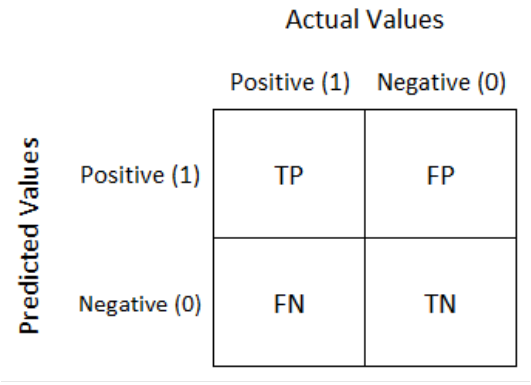


Figure 36 - A confusion matrix (Sarang Narkhede, 2018)

Accuracy - is defined as the percentage of correct predictions for test data, calculated as follows:

$$Accuracy = \frac{\#\ correct\ predictions}{\#\ predictions} = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision – is defined as from all classes that we have predicted as being positive, how many are actually positive.

$$Precision = \frac{TP}{(TP + FP)}$$

Recall – is defined as the percentage of correct predictions based on all positive classes.

$$Recall = \frac{TP}{(TP + FN)}$$

Precision and recall can be condensed into a single measure, known as **F_1 Measure**, this will also be calculated.

$$F_1 Measure = 2\ x\ \frac{(Precision\ x\ Recall)}{(Precision + Recall)}$$

(John Kelleher, 2015)

6 Conclusion

6.1 Project Management Strategy

6.1.1 Methodology

The scrum methodology (Valpadasu Hema, 2020) has been in place for the project so far, with this continuing for the remainder of the project. As the team is entirely composed of Software Developers with no Data Science specialists, members have been working on both the development side as well as the data side to fully understand the project. There is a breakdown of team roles provided in the table below.

Name		Student No.	Software Role	Data Role	Project Management Roles *	Testing/Evaluation Roles
Bearach	Byrne	C15379616	Backend Dev.	Recommender Sys	Scrum Master/Project Manager	Tester
Joseph	Corcoran	D20127858	Frontend Dev.	Recommender Sys	Product Owner	Evaluator
Warren	Kavanagh	C16463344	Frontend Dev.	Sentiment Analysis	Development Team	Tester
Rebecca	Kelly	C10330971	Frontend Dev.	Data Visualisation	Development Team	Evaluator
Daragh	Kneeshaw	D20128577	Backend Dev.	Recommender Sys	Development Team	Tester
Caolan	Power	D21125982	Backend Dev.	Sentiment Analysis	Development Team	Evaluator
					* Roles will rotate every sprint	

Figure 37 - Team member roles

A sprint cycle lasts a week and we the team conducts daily stand-up meetings to track progress. Zenhub and GitHub are used to track and manage the project as well as a spreadsheet that is reviewed at a minimum once a week, and on an ad-hoc basis. GitHub is used as the project repository and is used to track Pull Requests and Commits. Zenhub is linked to GitHub. Itis used to create Projects, Epics, Issues and Stories, which are the tasks that are assigned and completed during sprint cycles. Zenhub manages deadlines and milestones. Each week, epics and issues are assigned to team members so that they have a clear picture of what is needed from them for the upcoming week.

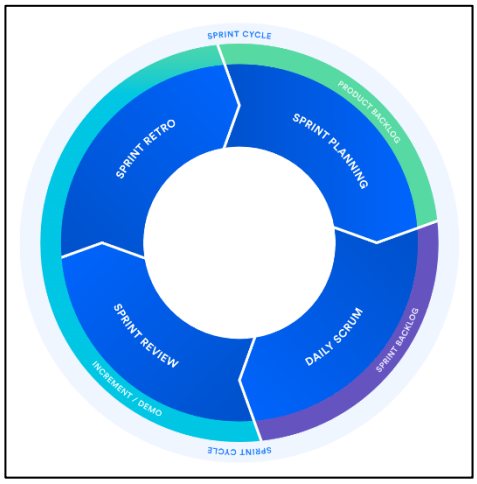


Figure 38 - The Scrum sprint cycle (Rehkopf, 2019)

The development of the project is on schedule to be completed by the end of week 11 (30/11/2022) to allow for evaluation, testing, improvements, and creation of the final report and presentation.

Project success would be the on-time completion of all Epics and Stories set out in the Zenhub board, which would result in the creation of the entire web app.

Week No.	Week	Features to be included in release	Comments
2	26 Sept – 30 Sept	Login, Register, View All Stocks, Company Information Metrics	Some elements such as SSO login were not implemented
3	3 Oct – 7 Oct	Profile Page, Ticker Cards, Individual Stock Page, Data Ingress	Profile Page postponed as it is not a priority. Up-to-Date Stock price ingress incomplete due to issues with Lambda.
4	10 Oct – 14 Oct	Search function for stock discovery. Ribbon for ticker cards. Categories for stock discovery. News and Twitter sentiment analysis.	Delays in Twitter due to front end Twitter component issues. Search function not finished.
5	17 Oct – 21 Oct	Portfolio schema and routes, buy/sell stock	Portfolio items delayed due to delays in previous weeks
6	24 Oct – 28 Oct	Interim Report, portfolio page, order page	
7	31 Oct – 4 Nov	User Dashboard, Profile Page, Password Reset, Game	Game depends on the portfolio being finished
8	7 Nov – 11 Nov	Follow Friends	Expected that recommender system for stocks will start on this week
9	14 Nov – 18 Nov	Recommender System, Stock Comparison	
10	21 Nov – 25 Nov	Testing & Evaluation	
11	28 Nov – 2 Dec	Final Report Finalising	

6.1.2 Team Meetings

The primary basis used in the creation of the meeting schedule is given in the Atlassian Agile Manifesto (West, 2019). The team meets for a daily stand-up on Monday, Tuesday, Wednesday, and Thursday. Each member must answer three questions:

- What did you do yesterday?
- What are you going to do today?
- Do you have any blockers affecting your tasks?

During these meetings, people can raise any other issues or ideas that that they need to with the team. In addition to the daily stand-ups, a 2+ hour face to face meeting on Friday organising the next sprint cycle is held. For the first few weeks, additional meetings on Monday and Thursday were held, however these have since been dropped. The meeting schedule is subject to change as needed throughout the project, as we want to avoid holding redundant meetings. All team members must attend these meetings; if they are unable, they must have a valid reason that should be accepted by a simple majority of the team.

Weekly Sprint Meeting Schedule - Group 1							
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
09:00							
10:00							
11:00							
12:00							
13:00							
14:00							
15:00							
16:00							
17:00							
18:00							

Figure 39 - All Team Meeting Schedule

Colour	Title	Duration	Location
	Sprint Planning	Up to 2 hours	MS Teams
	Meeting With Mentor	30-60 mins	MS Teams
	Daily Standup	Up to 30 mins	MS Teams
	Weekly Lecture	2 hours	Face-to-Face
	Sprint Retrospective	Up to 30 mins	Face-to-Face
	Sprint Review	Up to 60 mins	Face-to-Face

Figure 40 - Meeting Schedule Legend

More specific development meetings are organised on an ad-hoc basis by the relevant team members. If a team member is not involved in a specific meeting they may sit in if they wish to, to inform themselves on a specific aspect of the project. It is up to a team member to ask if they are needed at a meeting.

Meetings are held online through MS Teams for the scheduled meetings mentioned above, and through Discord for the ad-hoc development meetings, along with the face-to-face meeting on Friday. There is also a room available on campus that may be used for any ad-hoc meeting if the members wish to have the meeting face-to-face.

For decision making, most of the time decisions are decided by a simple majority however in the instance where unanimous and expertise decision making is needed, this will be decided on a case-by-case basis by the team by simple majority.

The role of scrum master, along with all project management roles will be rotated weekly.

6.2 Biggest Challenges

Machine Learning/ intelligence

As the entire team is from the software development stream, our biggest challenge is implementing machine learning into the project.

For the news sentiment analysis, the first problem occurred in trying to source labelled financial news datasets. Most sentiment analysis datasets are related to Movie/Google reviews and use a

different type of language compared to financial news headlines. Three labelled datasets were found, the sentiment within them is heavily skewed towards neutral which may impact the outcome of the final trained model. The packages used for sentiment analysis being NLTK and Scikit-Learn also presented a steep learning curve. The packages themselves are easy to use but understanding how each classifier model trained differs from each other in terms of the algorithms used takes time to research into.

Each model also has parameters which can be tuned associated with it to give better results, the values used for tuning these parameters and the resulting effect it has on the models output takes time to understand. Metrics for scoring models such as accuracy, f-scores and recall are all new to the team. A lot of time was also spent researching into feature engineering for sentiment analysis and how to extract features from text.

Currently the Twitter and News Sentiment models are being trained off seven features, but it is not easy to say how many features are enough. Deploying these models has also presented a bit of a challenge, especially using the NLTK library. The NLTK library works by downloading packages using its own downloader not the default Python PIP package manger. Packages in the deployment must be bundled in ZIP files and cannot be specified through a requirements.txt or Pipfile.

The content recommender system has proven to be a problem for the team, as it provides a different challenge to the other machine learning components. Due to the Cold Start Problem (LENDAVE, 2021) present with recommender systems without a corpus of user data, evaluation of the system is going to be a challenge as the team does not have a dataset to draw from. Evaluation of this system is discussed in section 5.4. Development work on this component is in very early stages and as such very little development work has been carried out to date. When development proper begins the team will have a better understanding of challenges that will need to be faced in the creation of this system.

UI/UX challenges

Presenting the information to the end user in a legible manner is a huge challenge in this project. To aid the user in understanding the information at hand, care was taken to design the application by following known mental models in other apps. Whenever information can be visualised (be it in a graph or some other form of visual aid), it is, along with making these visualisations interactable. Trying to show the information in a creative manner as opposed to listing numbers in a table is the biggest challenge here. Some of the information requires custom UI (for example the Evilometer) which provides its own unique challenge beyond learning a graphing package.

As most users will access the application on phone, the application is designed mobile-first, which presents its own challenges due to the limited screen size of a phone device. The mobile screen forces the designer to pick the most important information to present to the user and anything that can be inferred must be left out. The application follows the general guidelines of Flat 2.0 design, to reap the benefits of minimalism and reducing the cognitive load on the users, without rendering it unusable by the end user by signifying clickable events. The application (and prototypes) is being constantly evaluated by team members and users to ensure usability.

Tech Stack Learning Challenges

To mitigate the potential issues in deploying the application (seen in other teams in previous reports over the years), the system was designed with deployment first in mind. This meant before the team started pushing out features and programming, they deployed a '*hello world*' style application first.

As mentioned previously, the application was deployed through AWS Amplify. The main challenge to overcome was the steep learning curve compared to other deployment methods. To tackle this challenge, every member of the team completed a tutorial on how to set up an Amplify application, allowing each team member to understand how it worked.

The MongoDB Docker image also required learning, with it being noted that the documentation for the docker being quite difficult to process, particularly the setting up of the environment for the docker using docker-compose.

While the team had experience with MongoDB and querying the database, the creation of views would also be needed due to the number of schemas in the application. Views, similar to 'joins' in relational databases, provide the ability to combine data from two different collections within a database. In this case, it would provide the ability for the user's portfolio value to be calculated with up-to-date stock prices from the stock prices collection in the same database.

As discussed above in section 3.2.3, Lambda functions would be used throughout the application. Once again, this was a new technology for the team and the team followed tutorials to learn how to implement these. A cohort of members encountered issues when following this tutorial, with *pipenv* being the main issue for the users. The team overcame this issue by having a group call to go through each person's issue one by one.

Application Performance Challenges

CRON and Database

A few challenges arose in the first number of weeks of development as data ingress occurred. Due to the amount of data that needs to be updated regularly, there are currently four CRON jobs in place. These are up-to-date stock data, ESG rating updates, news article ingress and Twitter data ingress. While the ESG rating update CRON job is only run once a week due to the infrequency of those ratings, the others are run regularly during the week. The stock data web scraping is done every twenty minutes during the week, while the news articles are updated every thirty minutes. As well as this, the Twitter data is updated twice a day, with the frequency expected to increase to get more up-to-date data for users. When more that one of these are running at the same time, the database may feel a bit of a strain.

As a result of the completion of the stock data ingress which was scraping a lot of data and sending it to the database every twenty minutes, the database crashed multiple times. This was due to a lack of RAM on the AWS EC2 free tier. The free tier, a T2 micro instance, provided 1GB of RAM along with one virtual CPU core. The upgraded solution, a T2 small install, provides 2GB of RAM, with still only one virtual CPU core, and a small cost between the team members. It is hoped that this will suffice, however the performance of the system will be monitored in the long term with potentially more upgrades to follow.

Code Inefficiencies

When creating an application for users, it is important that all elements of a page load quickly. Originally, while all the data was not available, there were three separate requests being sent to the database for the individual stock page. This slowed down the page considerably due to the daily price data being request three times.

The frontend designers noted this slowness and requested that the backend re-examine the code to potentially increase the performance of the page. It was noted that this could be reduced to one request for the page. This was updated and the speed the page loads has now increased

significantly. It is important to continuously look for code inefficiencies and bad smells.

6.3 Timeline (How will you use the time remaining to achieve a successful outcome?)

After working together for 6 weeks, the group has a clearer idea of the time involved to implement various features.

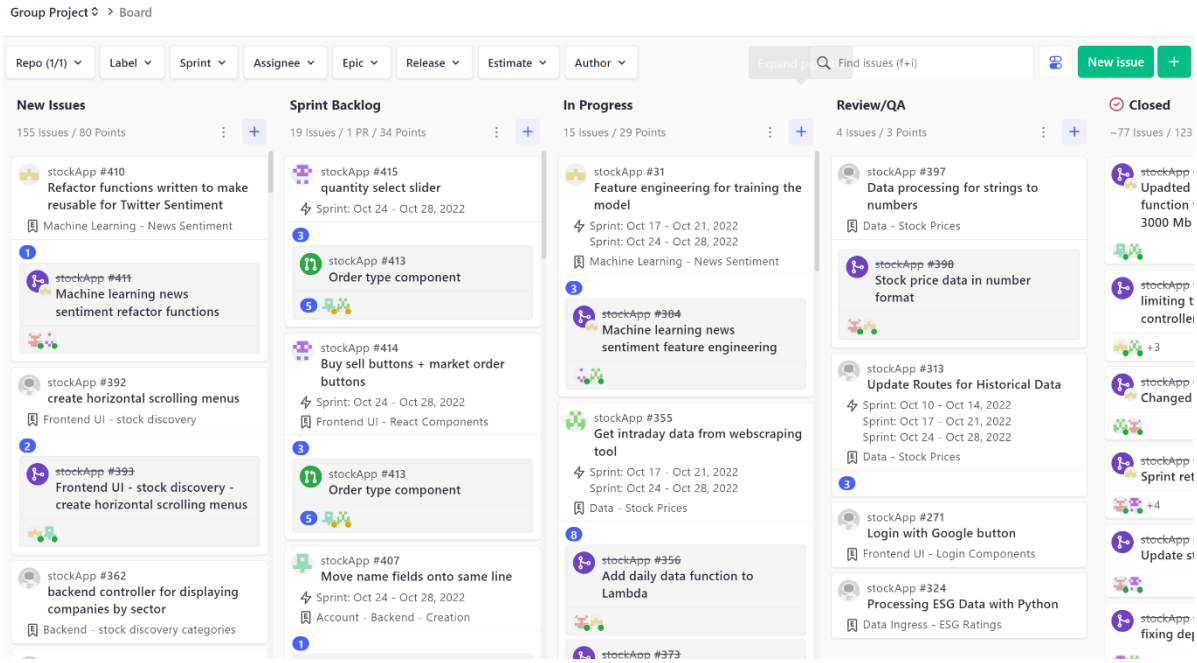


Figure 41 - Zenhub Board

The following Gantt chart identifies our timeline for the remaining weeks. For the next two weeks we will focus on finishing the stock discovery feature, individual stock pages, portfolio feature and the twitter and news sentiment analysis. We will also be starting the game feature and recommender system next week along with the first user evaluation of the system. This will lead into building the comparison tool and social functionality on week eight and nine, and modifying the application based on user evaluation. All things going well, we will implement the Message Boards feature and Direct Message on the app, along with a second user evaluation and updates to the application based on this. For the final two weeks we will be implementing Pro Trader Mode and the Educational Section. The last week is reserved for writing the report and the demo.

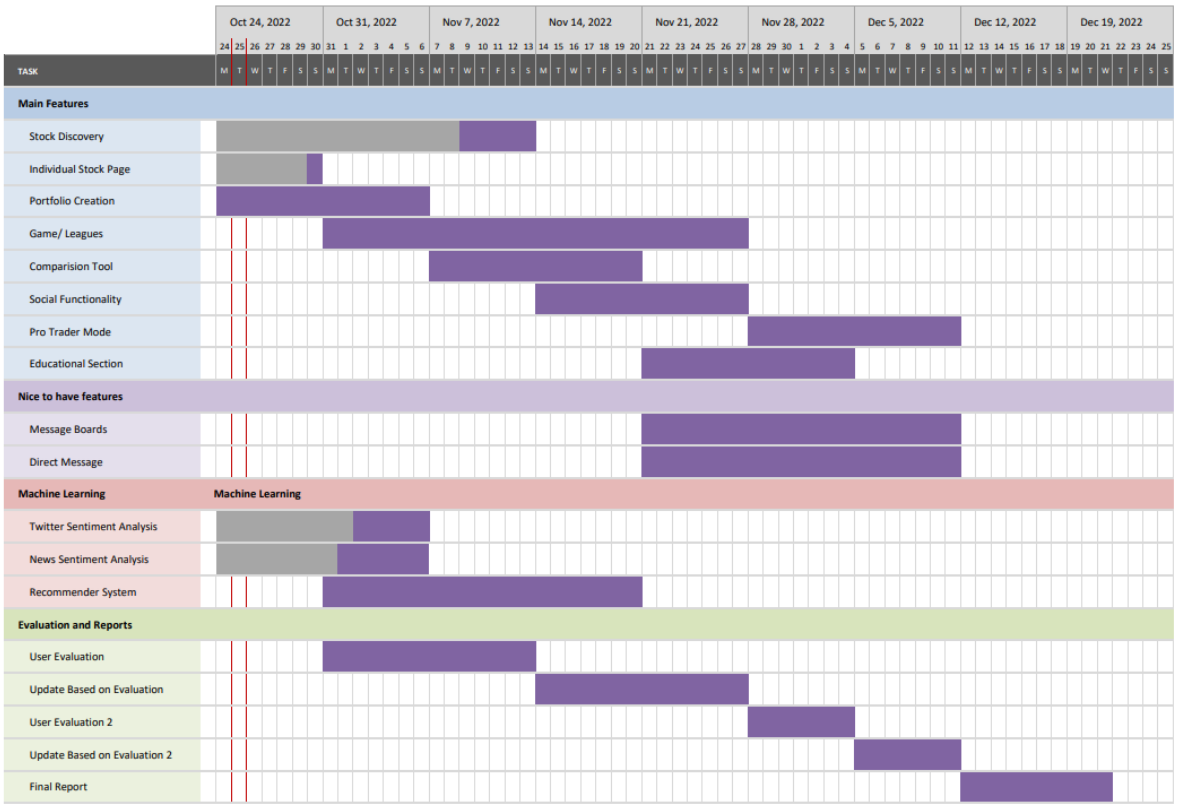


Figure 42 - Gantt chart showing timeline

7 References

Amazon Web Services, Inc. (2022). *Amplify Documentation*. Retrieved from docs.amplify.aws: https://docs.amplify.aws/

Abramov, D. (2022). *Redux*. Retrieved from redux.js: https://redux.js.org/

Abramov, D. (2022). *When should I use Redux?* Retrieved from redux.js: https://redux.js.org/faq/general#when-should-i-use-redux

Amazon Web Services, Inc. (2022). *AWS Lambda*. Retrieved from aws.amazon.com: https://aws.amazon.com/lambda/

Amazon Web Services, Inc. (2022). *Amazon EC2*. Retrieved from aws.amazon.com: https://aws.amazon.com/ec2/

Amazon Web Services, Inc. (2022). *AWS Systems Manager Parameter Store*. Retrieved from aws.amazon.com: https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html

AWS. (2022, October 25). *AWS Lambda.* Retrieved from AWS.Amazon.com: https://aws.amazon.com/lambda/

Bootstrap team. (2022). *Grid system*. Retrieved from react-bootstrap: https://react-bootstrap.github.io/layout/grid/

Bootstrap team. (2022). *React Bootstrap*. Retrieved from react-bootstrap: https://react-bootstrap.github.io/

Brady, E. (2021, October 13). *Young people most affected by cost of living crisis, study finds.* Retrieved from SkyNews: https://news.sky.com/story/young-people-most-affected-by-cost-of-living-crisis-study-finds-12432524

Central Statistics Office. (2022, October 25). *Consumer Price Index.* Retrieved from Central Statistics Office: https://www.cso.ie/en/statistics/prices/consumerpriceindex/

Codecademy. (2022). *React: The Virtual DOM*. Retrieved from codecademy: https://www.codecademy.com/article/react-virtual-dom

Cournapeau, D. (2022). *Scikit-learn*. Retrieved from scikit-learn.org: https://scikit-learn.org/stable/

Data Protection Commission. (2019, October 1). *Quick Guide to the Principles of Data Protection.* Retrieved from dataprotection.ie: https://www.dataprotection.ie/sites/default/files/uploads/2019-11/Guidance%20on%20the%20Principles%20of%20Data%20Protection_Oct19.pdf

Deloitte China. (2022, October 1). *What is ESG rating?* Retrieved from Deloitte.com: https://www2.deloitte.com/cn/en/pages/risk/articles/what-is-esg-rating.html

Google Developers. (2022, July 18). *Content-based Filtering*. Retrieved from developers.google.com: https://developers.google.com/machine-learning/recommendation/content-based/basics#:~:text=To%20do%20so%2C%20you%20must,any%20information%20about%20other%20users.

Hamilton, T. (2022, September 03). *What is Test Driven Development (TDD)? Tutorial with Example.* Retrieved from guru99.com: https://www.guru99.com/test-driven-development.html

Hexacta Architecture Team. (2009, June 16). *TEST-DRIVEN DEVELOPMENT VS TEST-AFTER DEVELOPER.* Retrieved from Hexacta.com: https://www.hexacta.com/test-driven-development-vs-test-after-testing/#:~:text=Someone%20who%20practices%20Test%2DAfter,their%20design%20decisions%20up%20front.

Ho, H. (2016, September 22). *Flat 2.0: Why Fully Flat Design is Outdated.* Retrieved from Medium.com: https://medium.com/@ux_hai/flat-2-0-why-fully-flat-design-is-outdated-95fb5609f729

Hotz, N. (2022, August 08). *What is CRISP DM?* Retrieved from Data Science Process Alliance: https://www.datascience-pm.com/crisp-dm-2/

Hu, T. (2022, September 19). *Test-driven vs. test-later development: when should you write your tests?* Retrieved from Codecov.io: https://about.codecov.io/blog/test-driven-vs-test-later-development-when-should-you-write-your-tests/

International Monetary Fund. (2022, October 1). *Inflation and uncertainty.* Retrieved from International Monetary Fund: https://www.imf.org/en/Publications/WEO/Issues/2022/10/11/world-economic-outlook-october-2022#:~:text=Global%20inflation%20is%20forecast%20to,to%204.1%20percent%20by%202024

Jest. (2022, October 24). *Jest*. Retrieved from Jest.io: https://jestjs.io/

John Kelleher, B. M. (2015). *Fundamentals of Machine Learning for Predictive Data Analytics.* Cambridge Massachusetts: The MIT Press.

Juho Hamari, J. K. (2014). Does Gamification Work? — A Literature Review of Empirical Studies on Gamification. *Hawaii International Conference on System Science* (pp. 3025-3034). Hawaii: IEEE.

JustAnotherArchivist. (2022). *snscrape*. Retrieved from github.com: https://github.com/JustAnotherArchivist/snscrape

LaVond, W. (2021, October 1). *How to Conduct a Cognitive Walkthrough.* Retrieved from Interaction-design.org: https://www.interaction-design.org/literature/article/how-to-conduct-a-cognitive-walkthrough

LENDAVE, V. (2021, September 26). *Cold-Start Problem in Recommender Systems and its Mitigation Techniques.* Retrieved from Analytics India Mag: https://analyticsindiamag.com/cold-start-problem-in-recommender-systems-and-its-mitigation-techniques/

McKee, K., & Pilgrim, M. (2020). *Feedparser Documentation*. Retrieved from feedparser.readthedocs.io: https://feedparser.readthedocs.io/en/latest/index.html

Meta Platforms, Inc. (2022). *Context*. Retrieved from Reactjs: https://reactjs.org/docs/context.html

Milankovich, M. (2015, July 15). *The Cold Start Problem for Recommender Systems.* Retrieved from Medium.com: https://medium.com/@markmilankovich/the-cold-start-problem-for-recommender-systems-

89a76505a7#:~:text=With%20recommendation%20engines%2C%20the%20%E2%80%9Ccold,start%20and%20user%20cold%20starts.

MongoDB, Inc. (2022). *Database References*. Retrieved from mongodb: https://www.mongodb.com/docs/v5.3/reference/database-references/

MongoDB, Inc. (2022). *PyMongo 4.3.2 Documentation*. Retrieved from pymongo.readthedocs.io: https://pymongo.readthedocs.io/en/stable/

Mongoose. (2022). *Populate*. Retrieved from mongoosejs: https://mongoosejs.com/docs/populate.html

NLTK Project. (2022). *Documentation*. Retrieved from nltk.org: https://www.nltk.org/index.html

NumFOCUS, Inc. . (2022). *Pandas*. Retrieved from pandas.pydata.org: https://pandas.pydata.org/

NumPy. (2022). *Numpy*. Retrieved from numpy.org: https://numpy.org/

OpenJS Foundation. (2022). *Express*. Retrieved from expressjs: https://expressjs.com/

Phagava, K. (2021, March 25). *Analyzing gamification in Duolingo — learning platform.* Retrieved from UX Design: https://bootcamp.uxdesign.cc/analyzing-gamification-in-duolingo-learning-platform-7b5200d144c3

Plaut, A. (2022, October 24). *What Is an ESG Rating?* Retrieved from fool.com: https://www.fool.com/investing/stock-market/types-of-stocks/esg-investing/esg-rating/

Premier League. (2022, October 25). *Fantasy Rules*. Retrieved from PremierLeague.com: https://fantasy.premierleague.com/help/rules

Rehkopf, M. (2019, January 04). *What are sprints?* Retrieved from Atlassian.com: https://www.atlassian.com/agile/scrum/sprints

Salazar, K. (2022, April 10). *How to Conduct a Cognitive Walkthrough Workshop.* Retrieved from nngroup.com: https://www.nngroup.com/articles/cognitive-walkthrough-workshop/

Sarang Narkhede. (2018, May 09). Retrieved from towardsdatascience.com: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Slickcharts. (2022). *Slickcharts*. Retrieved from slickcharts.com: https://www.slickcharts.com/

Utah State University. (2001, January 01). *WAVE Web Accessibility Evaluation Tool*. Retrieved from Wave.Webaim.org: https://wave.webaim.org/about

Valpadasu Hema, S. T. (2020). Scrum: An Effective Software Development Agile Tool. *International Conference on Recent Advancements in Engineering and Management (ICRAEM-2020)* (pp. 1-10). Warangal, India: IOP Publishing Ltd.

Vendia. (2022). *Serverless Express*. Retrieved from GitHub: https://github.com/vendia/serverless-express

Web Accessibility Initiative. (2017, December 22). *Easy Checks – A First Review of Web Accessibility.* Retrieved from w3.org: https://www.w3.org/WAI/test-evaluate/preliminary/

West, D. (2019, January 04). *Sprint Planning.* Retrieved from Atlassian.com: https://www.atlassian.com/agile/scrum/sprint-planning

Will Kenton, M. J. (2022, February 15). *S&P 500 Index: What It's for and Why It's Important in Investing.* Retrieved from Investopedia: https://www.investopedia.com/terms/s/sp500.asp

Will, B. (2016, December 13). *Agile Assessments – How to Assess and Evaluate Agile / Scrum Projects*. Retrieved from LinkedIn.com: https://www.linkedin.com/pulse/agile-assessments-how-assess-evaluate-scrum-projects-brian-will

Xinjie Wang, Z. X. (2022). The causal relationship between social media sentiment and stock return: Experimental evidence from an online message forum. *Economics Letters*, 1-6.

8 Appendix

8.1 Appendix A - Front-end Prototypes

8.1.1 ESG Ratings – "Evilometer" – Static Image Prototype

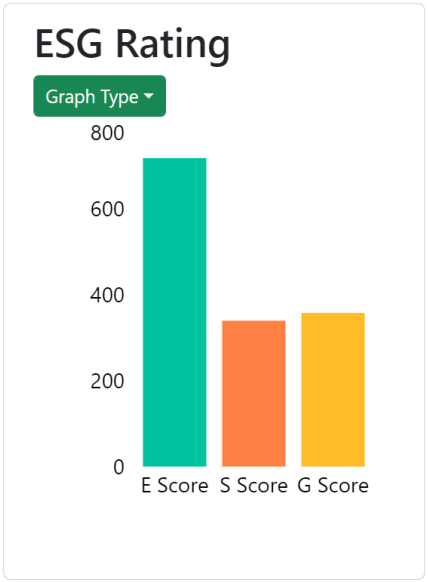


Figure 43 – Initial Design - Bar chart showing ESG ratings

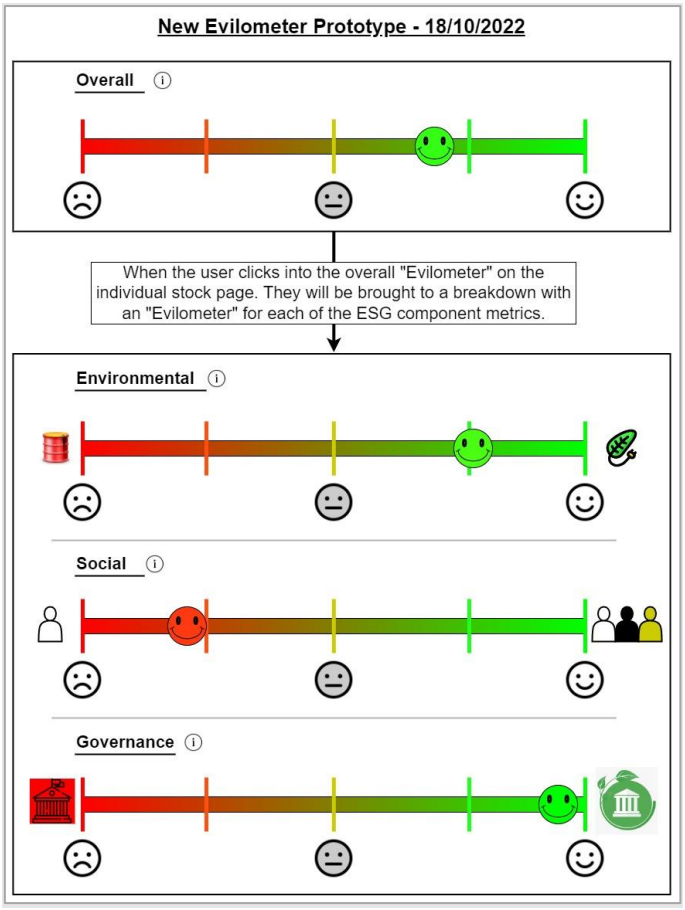


Figure 44 - Evilometer prototype

8.1.2 Registration Form – Static Image Prototype

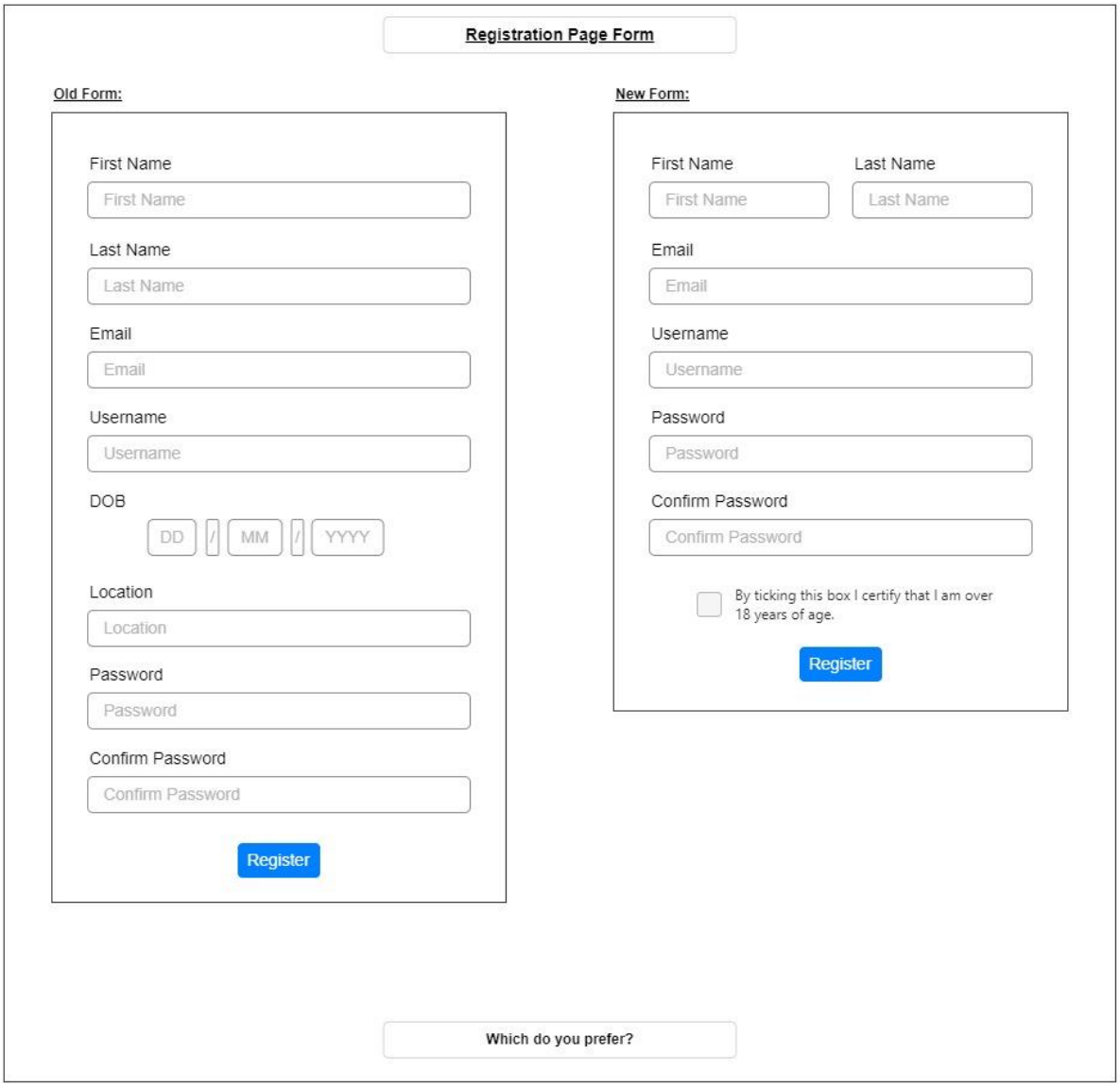


Figure 45 - Registration form prototypes

8.1.3 Stock Discovery Page – Figma Prototype

Link to prototype version 1: https://www.figma.com/proto/gp0aEVcyMb3X1cfhielZJX/Discovery-Page-Prototype-(v1)?node-id=1%3A704&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A704

Link to prototype version 2: https://www.figma.com/proto/MVmRifShILiXcGGaY4V3A8/Discovery-Page-Prototype-(v2)?node-id=1%3A3&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A335

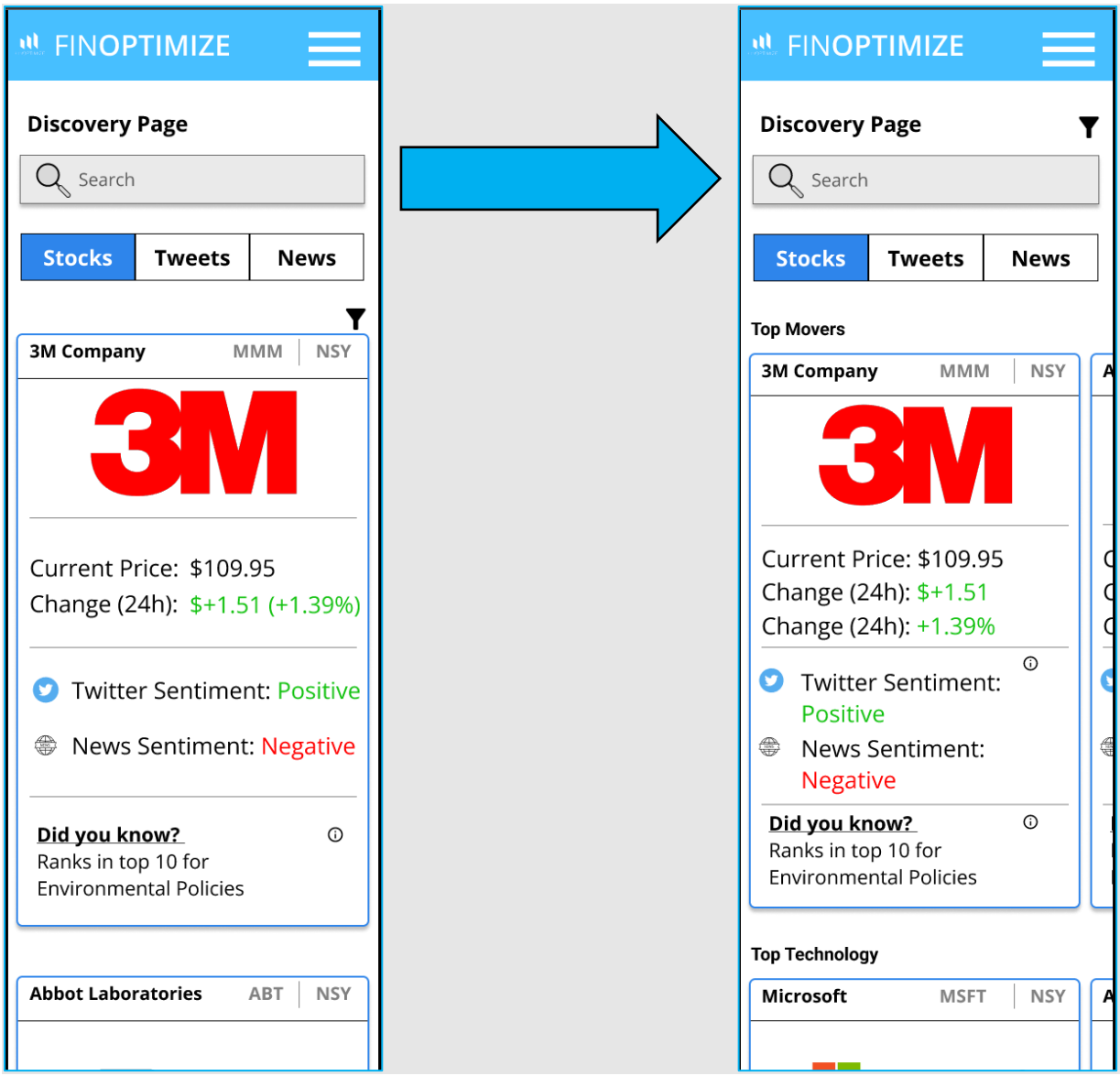


Figure 46 - Discovery Page prototype version 1 to version 2

8.1.4 Portfolio Page – Figma Prototype

Link to Figma prototype https://www.figma.com/proto/h7dqeBQfEuMBGLRucKgVv7/Portfolio-Page?page-id=28%3A6&node-id=28%3A7&viewport=642%2C1156%2C0.85&scaling=scale-down&starting-point-node-id=28%3A7&show-proto-sidebar=1

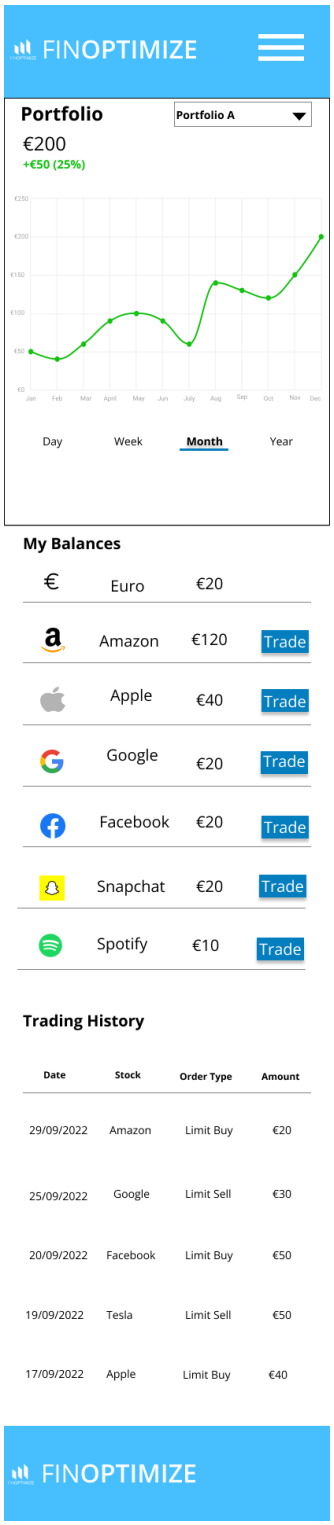


Figure 47- Portfolio Figma

8.1.5 Individual Stock Page – Figma Prototype

Link to Figma prototype https://www.figma.com/proto/h7dqeBQfEuMBGLRucKgVv7/Portfolio-Page?page-id=28%3A6&node-id=28%3A349&viewport=642%2C1156%2C0.85&scaling=scale-down&starting-point-node-id=28%3A7&show-proto-sidebar=1

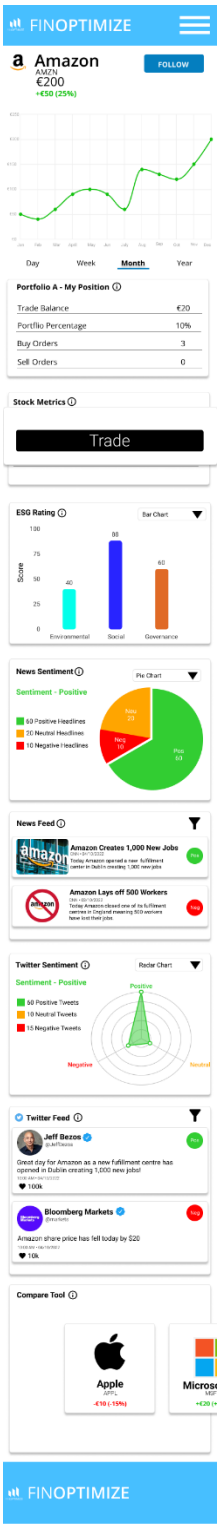
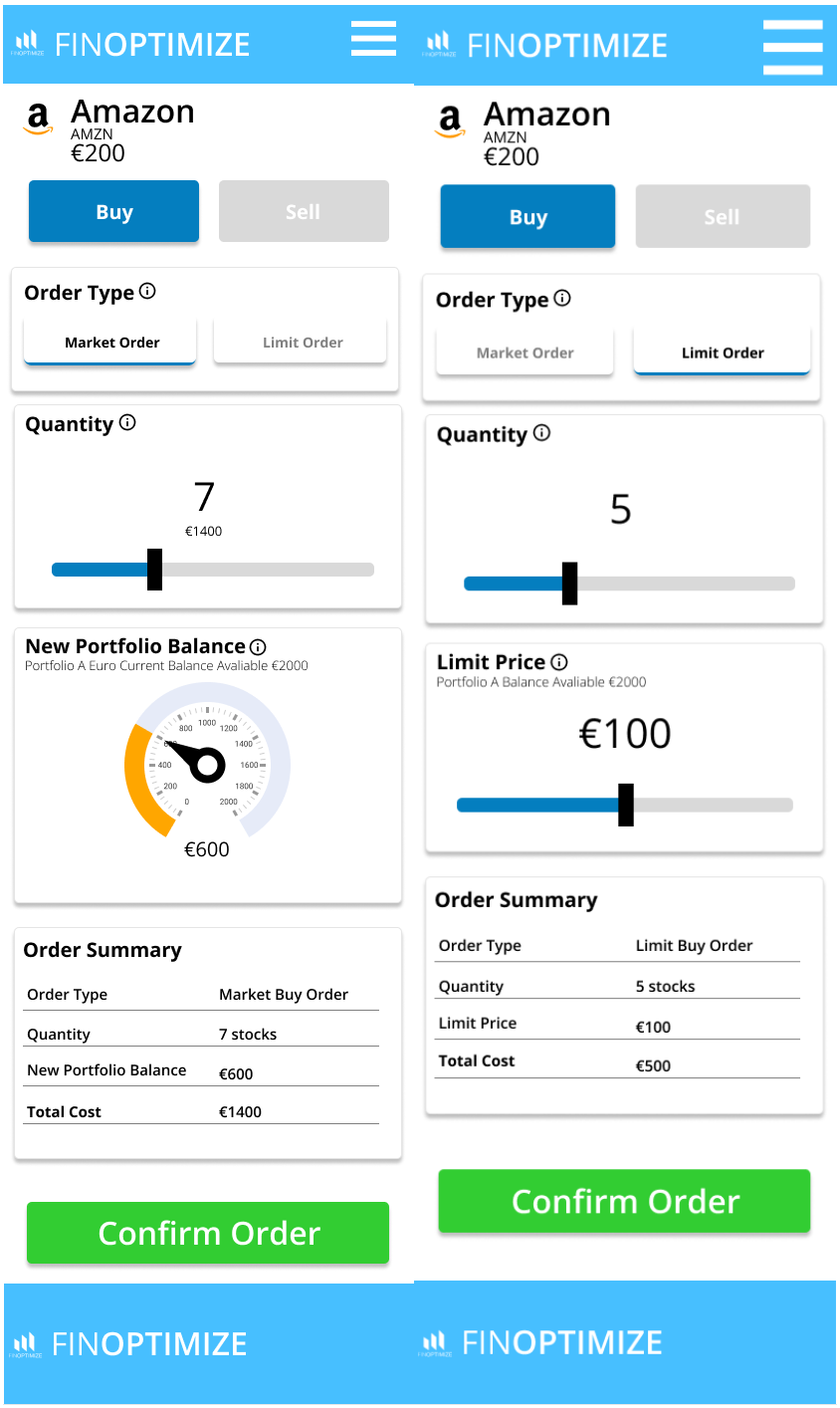


Figure 48 - Stock Page Figma

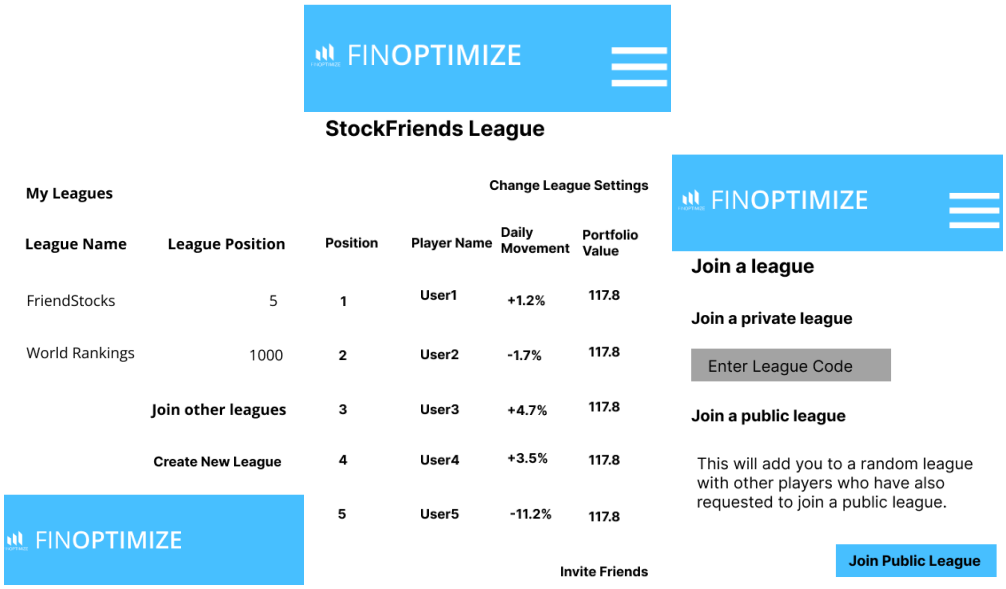
8.1.6 Confirm Order Page – Figma Prototype

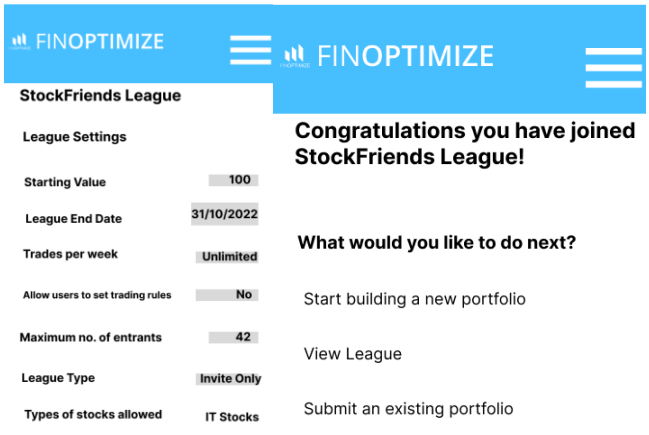
Link to Figma prototype https://www.figma.com/proto/h7dqeBQfEuMBGLRucKgVv7/Portfolio-Page?page-id=28%3A6&node-id=64%3A1102&viewport=642%2C1156%2C0.85&scaling=scale-down&starting-point-node-id=28%3A7&show-proto-sidebar=1

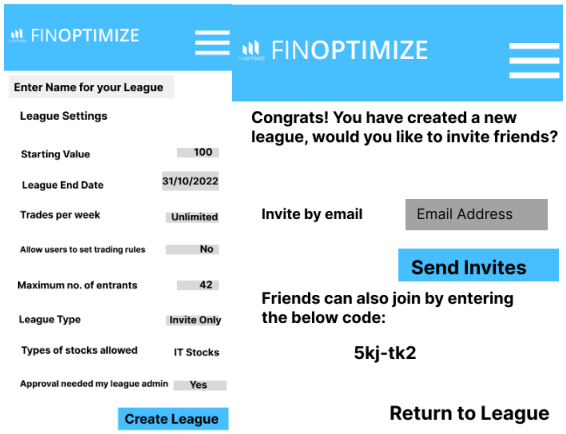


8.1.7 Stock Trading Leagues – Figma Prototype

Link to Figma prototype https://www.figma.com/proto/vRc8QVdG6nDr6yGtVzJU3S/Untitled?node-id=9%3A846&scaling=scale-down&page-id=0%3A1&starting-point-node-id=9%3A846&show-proto-sidebar=1







8.2 Appendix B – Table of Data Sources

Category	Subcategory	Method of Collecting	Source
Stock Data	Real Time	Scraping	https://www.slickcharts.com/
Stock Data	Historical	API	https://www.alphavantage.co/
ESG Data	Real Time	API	https://rapidapi.com/esg.enterprise.app/api/esg-environmental-social-governance-data/
News Data	Real Time	Scraping	https://feeds.skynews.com/feeds/rss/business.xml
News Data	Real Time	Scraping	https://feeds.skynews.com/feeds/rss/technology.xml
News Data	Real Time	Scraping	https://abcnews.go.com/abcnews/moneyheadlines
News Data	Real Time	Scraping	https://abcnews.go.com/abcnews/technologyheadlines
News Data	Real Time	Scraping	https://rss.nytimes.com/services/xml/rss/nyt/Business.xml
News Data	Real Time	Scraping	https://rss.nytimes.com/services/xml/rss/nyt/Economy.xml
News Data	Real Time	Scraping	https://rss.nytimes.com/services/xml/rss/nyt/EnergyEnvironment.xml
News Data	Real Time	Scraping	https://rss.nytimes.com/services/xml/rss/nyt/Technology.xml
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=business-finance&post_type=best
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=deals&post_type=best
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=political-general&post_type=best
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=environment&post_type=best
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=tech&post_type=best
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=health&post_type=best
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=sports&post_type=best

Category	Subcategory	Method of Collecting	Source
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=lifestyle-entertainment&post_type=best
News Data	Real Time	Scraping	https://www.reutersagency.com/feed/?best-topics=human-interest&post_type=best
News Data	Real Time	Scraping	https://www.independent.ie/business/technology/rss/
News Data	Real Time	Scraping	https://www.independent.ie/business/world/rss/
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=15839135
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=10000664
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=10001147
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=100003114
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=100727362
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=19836768
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=20910258
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=10000108
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=10000116
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=10000101
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=10000110
News Data	Real Time	Scraping	https://search.cnbc.com/rs/search/combinedcms/view.xml?partnerId=wrss01&id=10000113
News Data	Real Time	Scraping	https://moxie.foxnews.com/google-publisher/tech.xml

Category	Subcategory	Method of Collecting	Source
News Data	Training Data	CSV Download	https://www.kaggle.com/datasets/ankurzing/aspect-based-sentiment-analysis-for-financial-news
News Data	Training Data	CSV Download	https://huggingface.co/datasets/financial_phrasebank
News Data	Training Data	CSV Download	https://zenodo.org/record/5211931#.Y1aNLHbMKiO
Twitter Data	Real Data	Scraping	https://github.com/JustAnotherArchivist/snscrape https://twitter.com/
Twitter Data	Training Data	CSV Download	https://www.kaggle.com/datasets/kazanova/sentiment140
Twitter Data	Training Data	CSV Download	https://www.kaggle.com/code/tommyupton/twitter-stock-market-sentiment-analysis/data
Twitter Data	Training Data	CSV Download	https://sraf.nd.edu/loughranmcdonald-master-dictionary/
Stock Recommendation	Real Time	Generated	Content based filtering recommender system. Stocks are recommended based upon stocks user has in their current portfolio. Features of stocks used are: • Historical Price • Exchange • Sector • Industry • Current price • Market cap • Revenue growth • ESG Ratings • Number of Employees • Weight in S&P 500

Table 9 - Data Sources