**1. Objectives and Aims.**

**1.1 Problem identification**

In the current stock trading world, manual application of trading strategies can be very exhausting and time consuming to professional traders. To clarify, day traders spend endless days and nights perfecting their trading plans, only to get on their computer at market open and stare at a candlestick chart for hours waiting for the perfect opportunity to arise and apply their strategy. Moreover, traders find themselves mentally exhausted as they usually spend two to three hours staring at a screen to perform trades which last a few minutes, sometimes seconds. In addition, another issue with the current trading proposal is the difference in time zones. Thus, users outside the United States of America are often unable to trade due to the stock market opening late at night which makes it difficult and unhealthy for them being awake staring at charts at 2am.

**1.2 Aims**

The primary aim of my application is to automate the aforementioned, by setting up bots to execute trading orders when the opportunity occurs, through a user-friendly experience. Further aims of the project are implementation of real money trading through APIs such as the one Binance provides, adoption of the software by hedge fund companies and automate their employees’ tasks.

**1.3 Objectives**

The application will be called Strategify and will be developed using the quasar framework. The main objectives are the following.

* Create a user-friendly front end using quasar’s utilities and Vue js.
* Quasar uses node js to its core which will be used to create the back end which connects the app with the database
* Google’s firebase will be used to store all the application’s data
* Binance API will be implemented to fetch live candlestick data

**1.4 Brief description of Strategify**

I aim to develop a web application platform where users can automate their own strategies for cryptocurrency trading and let a bot trade for them while they are away or sleeping. The application will also provide already existing strategies made by professionals for the users to copy. The platform will also provide fictional U.S. dollars for the users to practise and test their strategies with. Moreover, each user can have active and idle strategies which they can create through the application’s UI. Active strategies will place market orders when the requirements specified are met. Whenever a user creates a strategy, they must place BUY and SELL requirements based on indicators and price point. They also must choose a ticker pair from the ones provided by the application (e.g., BTCUSDT as in bitcoin against the US dollar stable coin Tether) as well as the amount of money they want to trade. A strategy the user could create is for instance a BUY order of 100$ to be placed when the RSI indicator hits the value 30 and a SELL order when it hits 70 on ETHUSDT at the four-hour candlestick chart.

**2. Literature Review**

**2.1 Introduction**

Researching on the development of such a system can be very challenging because it requires the intercommunication of many different technologies. Moreover, most of the technical research was taken on mastering the Quasar framework along with Vue js and understanding web sockets and data fetching from external APIs. Furthermore, a lot of research was undertaken on understanding the fundamentals of investing and trading such as technical analysis and grasp understanding of complex candlestick charts components.

**2.2 Quasar Framework**

Quasar is a javascript framework which aims to provide developers with the ease of having a cross platform app for web mobile and desktop application, all in one codebase [2]. Additionally, Quasar uses Vue js for front end development and provides several components on their website for easy UI creation. The reason for me choosing this framework is the responsive element it provides, its ease of understanding and use but also secures the scalability potential of the application as it can easily be further developed in other platforms other than web, with just a few weeks in the codebase. The Quasar Framework was first released its 1.0 version in 2019 which makes it a newly released framework [1]. Thus, knowledge on its implementation was not taught to me in school or university. With decent understanding of basic vanilla javascript you could get around the framework, but to further expand my knowledge on it, I have undertaken an online course which goes over all the fundamentals of the framework and explains well on how to scale to a larger-scale application [3].

**2.3 Fundamentals of Trading.**

The world of finance and decentralised finance specifically has caught my interest in the past few years which made investing in assets and researching start-ups a hobby for me. Although, I have an adequate understanding of different blockchain technologies, I also need to get a good understanding on trading specifically to understand my target audience and correctly develop such a platform. For obtaining such knowledge, I have undertaken another online course which has its primary focus on day trading basics [4].

**2.4 APIs and web sockets.**

The first time I had ever used APIs was at university which made me comfortable using them also helped me understand their functionality. Moreover, to make sure I completely understand the APIs needed for the app I had created a mini trading bot with hardcoded strategies in order to make sure I grasp the knowledge needed to create the application’s back end. The bot was created by following a tutorial video [5]. Although I have successfully created a bot using the desired APIs, it was made using python. In addition, I aim to learn how to implement the same APIs with javascript for my project’s needs.

**3.1 Software architecture**

Diagram

Description automatically generated

*Figure 01. Software architecture diagram*

In figure 01 we can see a diagram visualisation of the system’s underlying architecture. The strategify component represents the web application which is the files required for the web application to run. It contains the required files for a web application to function, such as html and javascript files.

Firebase hosting service is a hosting service provided by google which will be used to host my application [8]. The reason for choosing such a solution is because I am not familiar with web hosting, and thus, firebase will take care of the hosting for me. Moreover, firebase Realtime database will be my application’s database for storing all the users’ data. Firebase is an ideal solution for my system for many reasons. First and foremost, Firebase is a NoSQL database that stores data in json format, making it easier to communicate with the user interface [8] because working with objects in javascript is not an issue. Furthermore, firebase provides several scaling plans which will be handled by google [8], meaning all further scaling solutions and database issues will be handled by google at a charge.

The application will also be communicating with external API’s as seen in figure 01 which are Quasar’s API and web sockets for data fetching. Additionally, Quasar’s API will be used to provide UI components such as dialogs and text fields, similar to other more popular javascript frameworks such as Bootstrap. Implementation of Binance’s live data fetching from web sockets will take place in the application. The data fetched will be in json format and it contains all necessary information for the application to function such as open and close prices and metrics such as trading volume.

The users will be using the application through its user interface which will be created using Vue js and Quasar. Node js will be the bridge between the application’s front end and back end. Moreover, when the user makes requests through the user interface, data is collected or altered to the database using Node js.

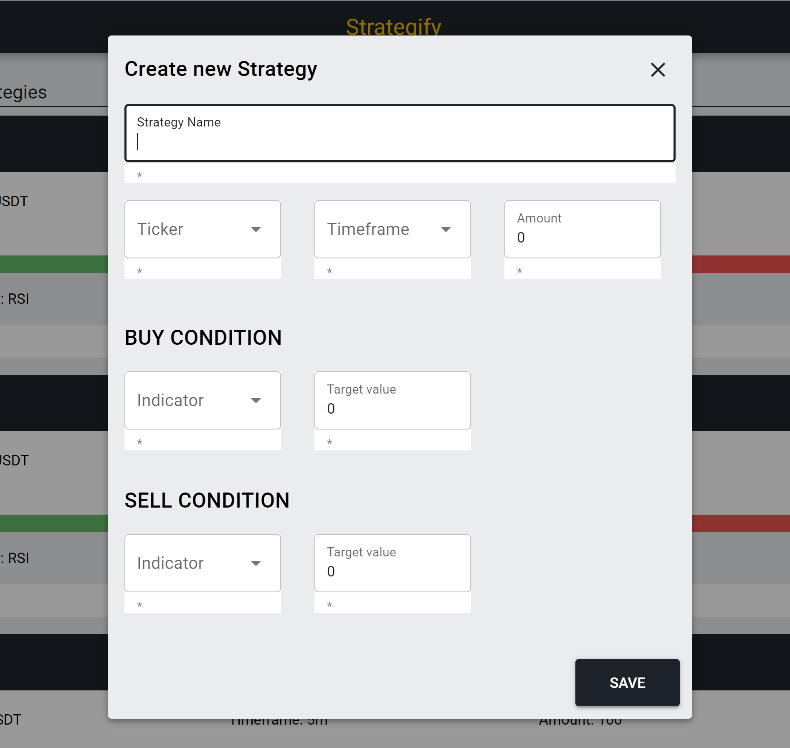
**3.2. Prototype description**

In this section, screenshots of the developed prototype will be shown and be discussed on their functionality. In addition, with each screenshot I will be discussing the features I have implemented so far.



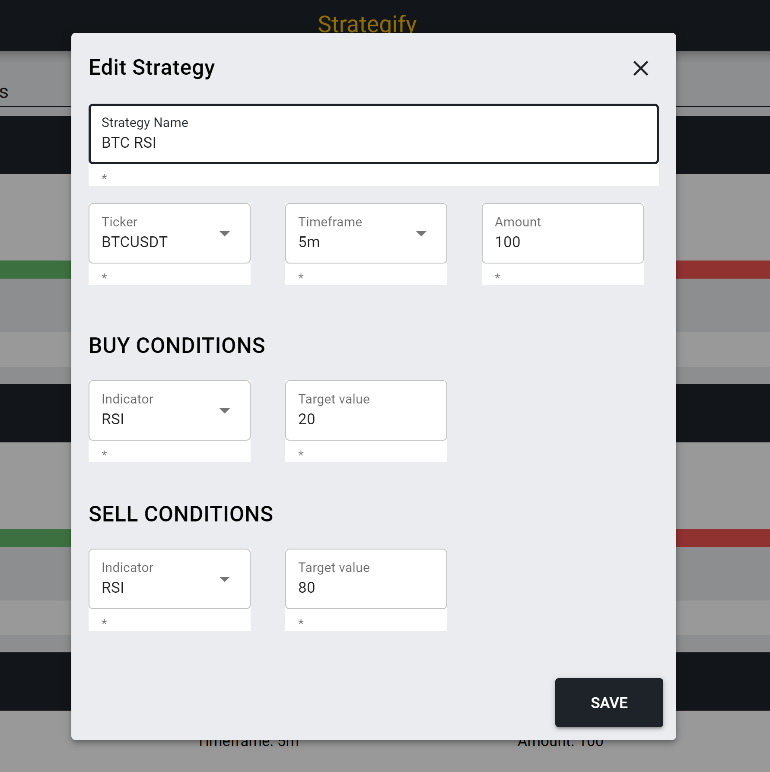
*Figure 02. Strategify’s dashboard*

The main section of the dashboard is a list of all the user’s created strategies. Each strategy is shown with its name, ticker, timeframe, amount, active status and finally buy and sell conditions. On the left, there is a drawer which shows the navigation list, the user can click on any page they wish to redirect from there. The only pages existing right now at these early development stages is the My Strategies page which is the one shown in Figure 02 and a settings page which is empty and will be developed in the future. On the top right there is a log out button which the user can click if they wish to log out from their account.



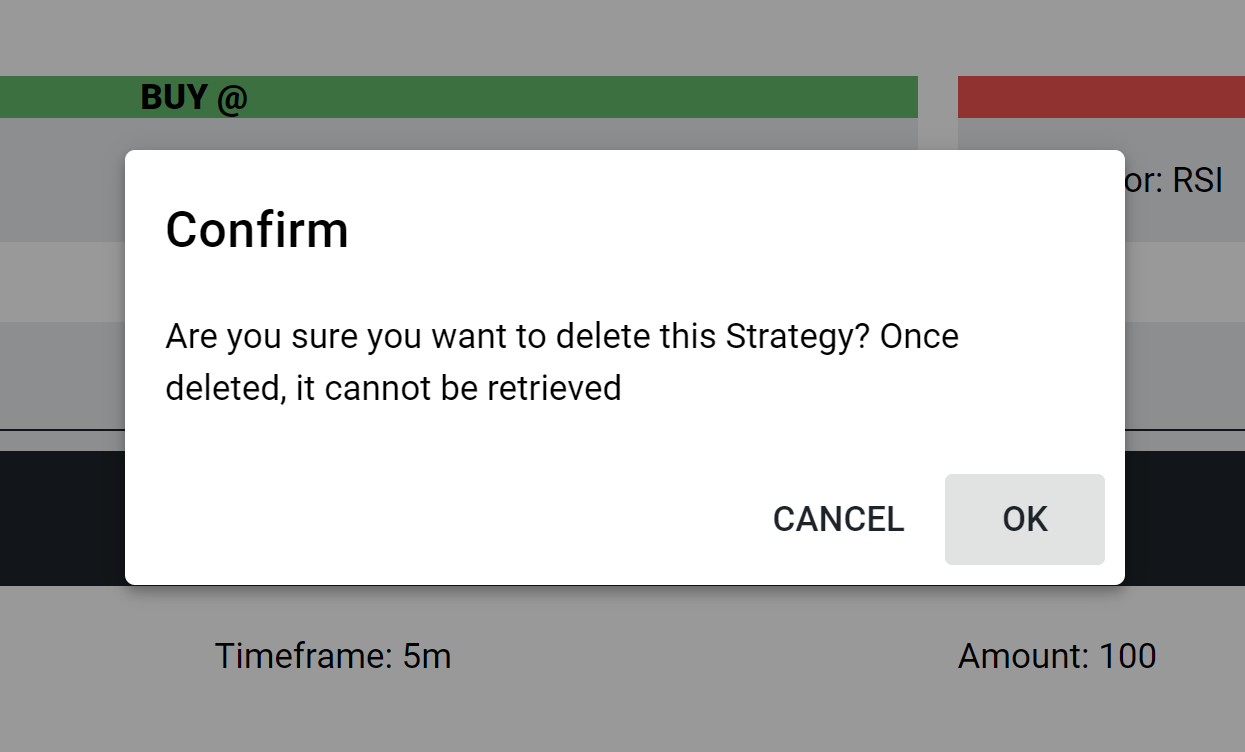
*Figure 03. Create a new strategy*

In figure 02, there is a button at the bottom showing a plus icon which can be used to create a new strategy. After pressing the button, a dialog appears as shown in figure 03. The user then can fill all the fields and then click on save to create a new strategy.



*Figure 04. Edit strategy*

If the user wishes to edit a strategy, they click on the edit icon on the top right of each strategy. Moreover, a dialog appears with the strategies data loaded, then the user changes the fields they wish to edit and click on the save button.



*Figure 05. Delete a strategy*

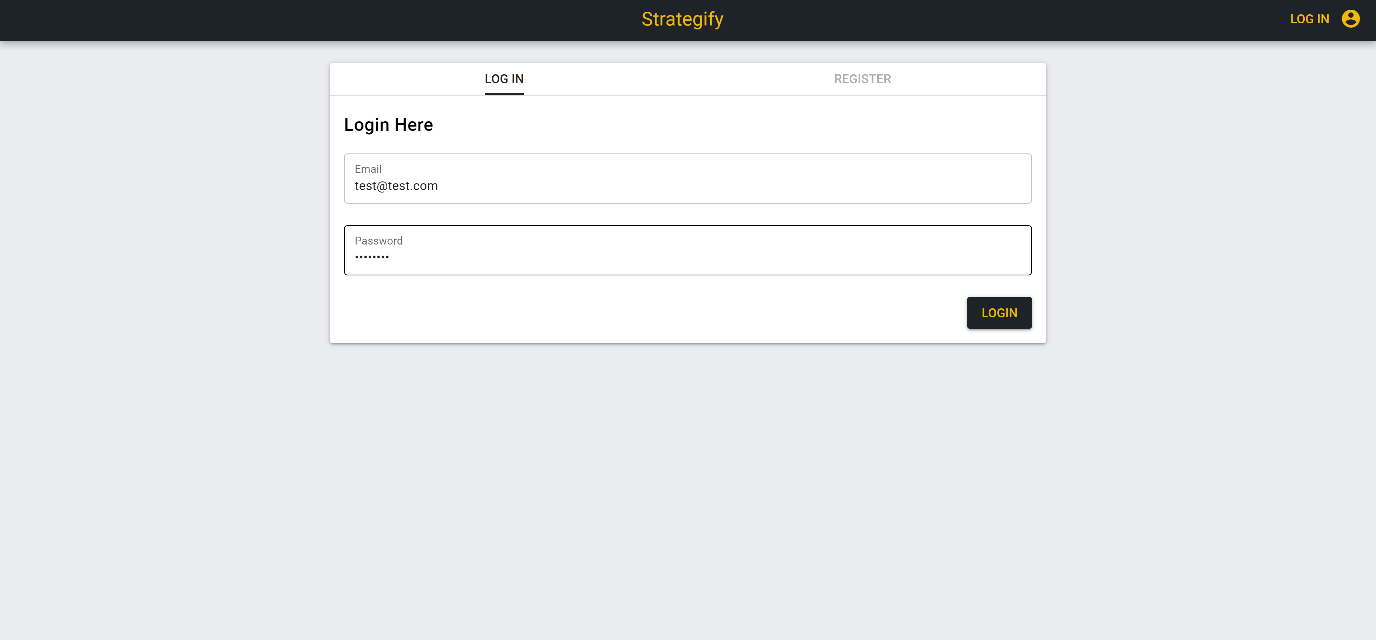
For the user to achieve the removal of one of the strategies, the delete button on the top right of each strategy needs to be clicked. After clicking the button, a dialog appears asking the user for permission to delete the strategy as shown in Figure 05. If the OK button is pressed, the program proceeds with deleting the strategy.

A screenshot of a computer

Description automatically generated

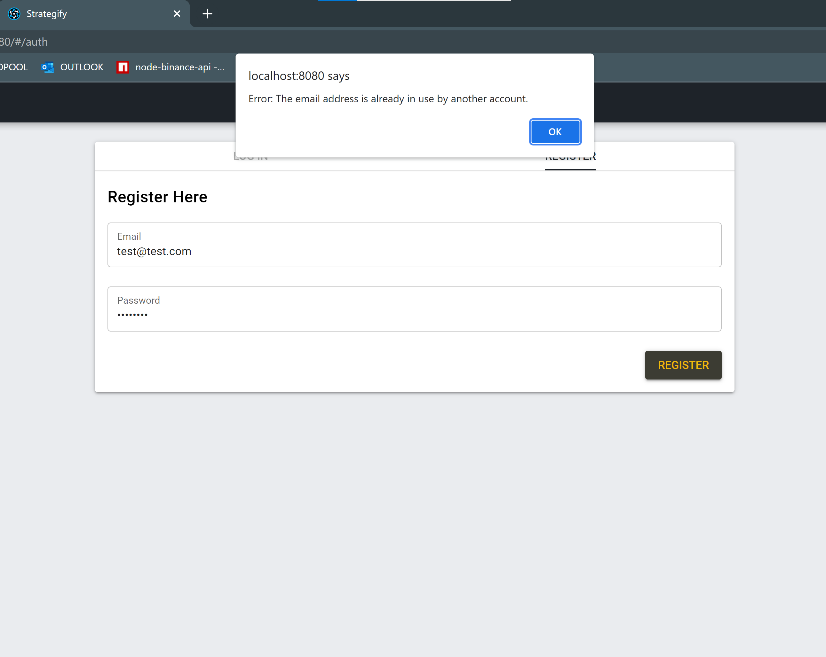
*Figure 06. Separating active and inactive strategies*

To activate a strategy, the user have to click on the active checkbox shown on each strategy’s card. The user interface then separates the strategies into two lists the active and inactive strategies. This way helps the user to notice which strategies are active more effectively instead of going through the list.



*Figure 07. Login page*

The user can logout of their account anytime by clicking the top right LOGOUT button as shown in figure 01. The user is then redirected to the login page shown in figure 07. Moreover, the user here is presented with a panel to choose between logging in or registering to the application. The application keeps users’ authentication data in email and password format thus, the user must fill an email and password field to complete the process. The form uses firebase’s authentication API which gives an error if the user tries to create an account using a pre-existing email as shown in figure 08.



*Figure 08. Register with an existing email error.*

**3.3 Features to implement**

* Data fetching on cryptocurrency which will be displayed to users.
* Adding fictional coins which the users can use to trade.
* Implement algorithms to visualise trades for users using their coins and the data from the web sockets.

**3.4 Data structure**

As mentioned before, the application uses firebase for storing the users’ data. Furthermore, firebase is a NoSQL database and makes use of objects to store data. The data stored so far can be seen in figure 09. To explain, an object called “strats” contains objects each named as each user’s ID, each of these objects contains the user’s strategies.

Text

Description automatically generated

*Figure 09. Data structure*

**4.Planning and timetables**

**4.1 Introduction**

The application will be developed based on the agile methodology. Agile is a project developing management methodology in which the developers establish good communications with the product owner. In agile the developers work in short development cycles called sprints in which each sprint they present a further developed version of the product instead of presenting the product after a long period of working on it by themselves [6]. There are several reasons to which the choice of going with agile over waterfall was a better fit. Firstly, I’m a new developer with not much experience on full stack software developing and thus, with frequent sprints where I meet with the client, which is my project mentor, I get advised by him and change the software accordingly. In addition, one of agile’s main principles is “working software over comprehensive documentation” [7] to which I strongly believe in. Personally, I would rather spend most of my time, energy, and resources into creating a working software rather than documenting it.

Moreover, I have decided for each sprint to last two weeks although this is a subject to change as it first needs to be discussed with my project mentor. Nevertheless, I have created a plan with timescales on technical achievements, the plan is presented below.

**4.2 Timescale Plan**

**Completed:**

**15 Jul – 15 Aug 2021:** Familiarize myself with finance and trading concepts

**15 Aug – 15 Sep 2021:** Learn Vue js and Node js through online research

**15 Sep – 30 Sep 2021:** Create a concept of the application on paper (data structures, interfaces)

**1 Oct – 15 Oct 2021:** Develop the user interface with hard coded data (dashboard, log in page),

Start working on the interim report

**15 Oct – 30 Oct 2021:** Complete further improvements into the UI with the options to add, edit and delete strategies

**1 Nov – 15 Nov 2021:** Generate a database; implement authentication on the log in page and store user’s data with hard coded data.

**Started:**

**15 Nov – 30 Nov 2021:** Implement firebase’s API so users can alter and generate data on the system’s database, complete and submit interim report

**To be done:**

**01 Dec – 15 Dec 2021:** Create the coins component of the application for the users

**15 Dec – 30 Dec 2021:** Set up a web socket in the application and display the data to the users through the UI

**01 Jan – 30 Jan 2022:** Research and implementation on algorithms that could work on calculating the profit and loss of fictional trades based on the data from the web socket, start working on dissertation

**01 Feb – 15 Feb 2022:** Perform additional testing of all developed features and make sure everything works smoothly, prepare for presentation

**15 Feb – 30 Feb 2022:** Attend presentation, revalue secondary aims

**01 March – 04 May 2022:** Work on secondary features and work on plan changes that occurred during development

**04 May 2022:** Submit dissertation and working product

**05 May – 20 May 2022:** Prepare for mini viva and presentation

**4.3 Summary**

To summarize, my plan is working on developing the user interface during the first semester and develop the applications back end mostly during the second semester and the final weeks of the first semester. I decided to take such approach to keep my code organized and avoid the confusion that occurs when trying to build everything simultaneously.

**References**

[1] <https://dreamonkey.com/en/blog/quasar-conf-2020-all-the-latest-news/>

[2] <https://quasar.dev/introduction-to-quasar>

[3] <https://www.udemy.com/course/quasarframework/>

[4] <https://www.youtube.com/playlist?list=PLic3jeMsmNn3bHRQw_RaSaTBF5ryOj3lQ>

[5] <https://www.youtube.com/watch?v=GdlFhF6gjKo&t=3361s>

[6] <https://www.cio.com/article/3156998/agile-project-management-a-beginners-guide.html>

[7] <https://searchcio.techtarget.com/definition/Agile-Manifesto>

[8] <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>