TrendEase

Discord Trading Bot

Allen Best Jr., Liam Brew, Robert Schaedler III

Problem Statement

Investing is commonly considered to be one of the most complicated stressful activities out there, reserved only for math PhDs and multibillion dollar firms with decades of experience. Afterall, the sheer size, speed and influence of the market makes it one of the most important factors of modern society and life itself. With such huge amounts of information and money being exchanged so rapidly, it is exceedingly easy for a beginner to become lost and be put off. In doing this many lose out on profitable opportunities because they do not have access to the proper tools to help them navigate this vast information.

Bot Description

The TrendEase Discord Trading bot is a stock market analysis and lookup tool that works within the Discord instant messaging platform. When TrendEase is added to a Discord server, it runs passively in the background awaiting a message with the prefix of '*t.*', which activates the bot and orders it to perform the following command. Upon completion of its order the bot returns to its idle state and remains out of the way until it is next needed. TrendEase is equipped with a wide suite of features that contains everything the budding investor may need, including instant stock data queries, scheduling recurring operations, dynamic report generation and custom portfolio monitoring. For a complete list of planned features please refer to this spreadsheet. TrendEase is therefore not just another market API: it supports specific inquiries by giving you the exact data that you need when you need it, keeps track of your real time investments in the face of market changes and events, and, most importantly, interacts directly with you to provide with the customized support you need in today's marketplace.

By incorporating all of this functionality into Discord, users are given all of the tools they need in a convenient, mobile and pre-existing platform. Specialized software or subscription services are not required. By using TrendEase, beginner investors have an efficient and lightweight way to stay on top of their investments and play an immediate role in the market no matter their skill level or preoccupation. Here at TrendEase, we pride ourselves on being **ahead of the trend**.

Use Cases

ID and Name:	01 Generate Stock Report		
Created By:	Liam Brew	**Date Created:**	03.29.2020
Primary Actor:	App User	**Secondary Actors:**	Market API
Description:	The user wishes to generate a report on a certain stock that they have been interested in.		
Trigger:	The user invokes the *t.sreport* command and passes in a stock ticker as a parameter.		
Preconditions:	1. The user is logged into a Discord server that supports TrendEase. 2. The TrendEase and Alpha Vantage API servers are operational.		
Postconditions:	1. TrendEase generates a report on the desired stock.		
Normal Flow:	1. User invokes the *t.sreport* command and passes in the stock ticker as a parameter. 2. TrendEase detects the '*t.*' prefix and identifies the command as *sreport*. 3. TrendEase executes the *sreport* command and queries the Alpha Vantage API for the relevant data. 4. TrendEase passes the API data into the Python graph generation scripts.		

	5. Python parses through the JSON data and uses pandas to generate the required graphs. 6. TrendEase launches the Python report generation script. 7. Python uses the docxtpl tool to populate the .docx report template with the generated stock data. 8. TrendEase sends the report as a message attachment and deletes the generated files server side.
Alternative Flows:	The user can alternatively trigger the command using the alias *srpt*.
Exceptions:	· If the user passes in an invalid stock ticker, TrendEase issues a notification and exits the report generation sequence.
Priority:	High
Frequency of Use:	High
Special Requirements:	None

ID and Name:	02 Cryptocurrency Rating		
Created By:	Liam Brew	**Date Created:**	03.29.2020

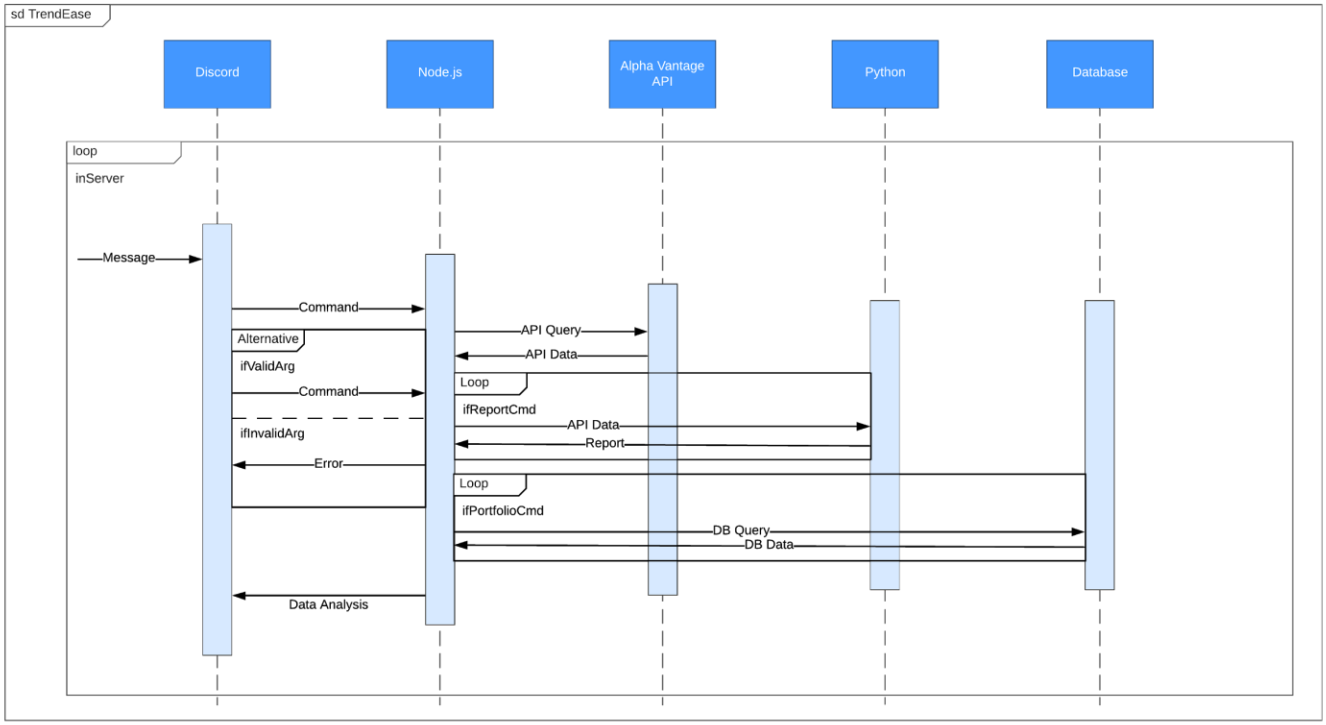
Primary Actor:	App User	**Secondary Actors:**	Market API
Description:	The user wishes to rate a certain cryptocurrency to see if it's worth investing in.		
Trigger:	The user invokes the *t.crate* command and passes in a crypto symbol as its parameter.		
Preconditions:	1. The user is logged into a Discord server that supports TrendEase. 2. The TrendEase and Alpha Vantage API servers are operational.		
Postconditions:	1. TrendEase rates the crypto currency and provides an analysis of the rating.		
Normal Flow:	1. User invokes the *t.crate* command and passes in the crypto symbol as a parameter. 2. TrendEase detects the '*t.*' prefix and identifies the command as *crate*. 3. TrendEase executes the *crate* command and queries the Alpha Vantage API for the relevant data. 4. TrendEase parses the crypto rating from the JSON data and attaches it to a message. 5. TrendEase generates an explanation for this rating and attaches it to a message. 6. TrendEase sends the message and deletes the generated files server side.		

Alternative Flows:	The user can alternatively invoke the command using the alias *irpt*.
Exceptions:	· If the user passes in an invalid crypto symbol, TrendEase issues a notification and exits the function execution sequence.
Priority:	High
Frequency of Use:	High
Special Requirements:	None

Design Sketches

Sequence Diagram

Liam Brew | March 30, 2020

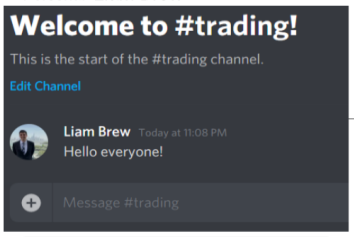


Storyboard

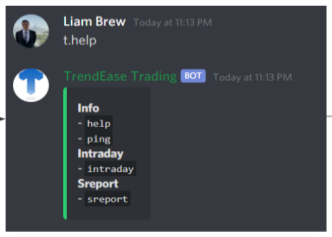
Liam Brew | March 30, 2020

Persona: Liam Brew

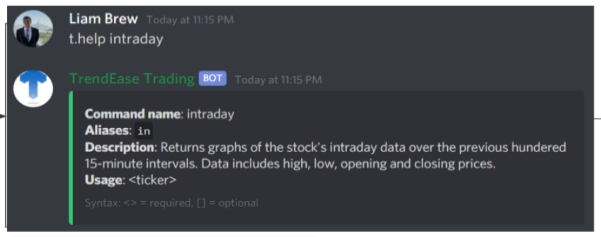
Scenario: First time user performs intraday analysis



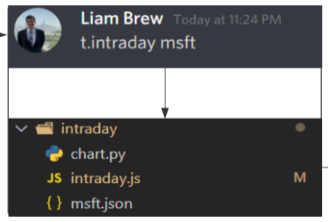
- User logs onto a TrendEase supported Discord server
- Requires no additional installations



- User invokes the help command to discover their options

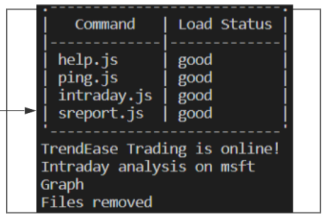


- User again invokes help and uses the command name as a parameter to learn detail information



- User invokes the intraday command on Microsoft's data
- Node.js queries the API and returns the data in msft.json

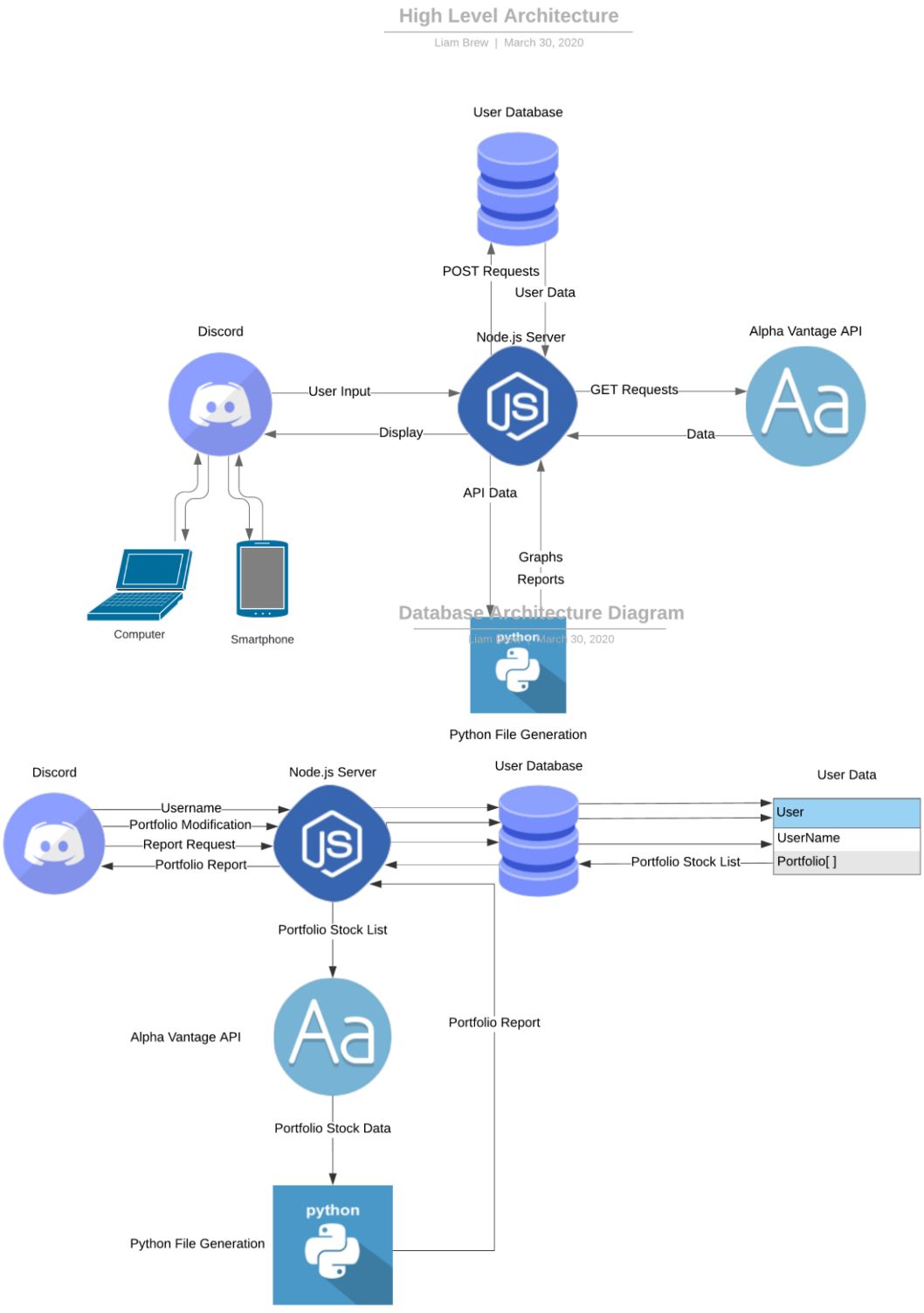
- Chart.py generates the analysis graph on the close price data contained in msft.json
- Node.js attaches this to a Discord message and sends



- The command process finishes and the generated files are removed
- The bot is now ready to accept new commands

Page # 1 Project/Team: TrendEase Trading Date: 3/30/20

Architecture Design



Discord will serve as a front end and interface between the user and the API and data analysis portion. Using Discord's developer API tool suite, discord.js, the team is able to create a

bot that loiters in a channel and passively listens to all messages, executing only the ones that are relevant to it. This in essence serves as a terminal, wherein the user passes in commands and parameters which the bot executes and displays. Once the bot performs the invoked command, it communicates the results as a message in this same channel. This text-based interface provides a lightweight and intuitive method for users to converse directly with the bot, and allocates resources that would have otherwise been spent on building a dedicated UI to improving the functionality of the box.

Node.js forms the cornerstone to this project, connecting all of the individual components together and directing the flow of instructions and data. It is from here that commands are interpreted from the user and executed. Node.js serves as a go-between for the various tasks that the bot performs. Queries to the API are sent with the resultant data being inputted into one of various Python scripts. Database lookup requests are received and processed. Result messages are synthesized from both data and images, creating personalized responses specific to the commanding user. For the development portion of this assignment the team is utilized by a local Node host, but this bot can easily be run off of a dedicated server for full uptime and storage potential. Additionally, Discord's developer services allow this bot to easily be added to other servers in the form of a virtual marketplace. Once this bot is hosted, said servers will need to do nothing more than allow this bot access through the Discord administrative console.

The team is making use of the Alpha Vantage market API, a free tool that includes many of the popular market analyses. The team has added the most popular of these which are found within the previously mentioned command spreadsheet. The team is additionally making use of an API wrapper for Alpha Vantage, which converts the numerous HTTP methods into node functions that synchronize better with the overall project. The results of API queries are returned as JSON files.

The team employs numerous Python scripts to parse through and analyse these files. There are scripts that perform calculations and return numeric values, scripts that generate graphs and charts (via pandas) of the data and scripts that generate and populate word documents to serve as stock reports. The team was deliberately modular when designing and implementing these scripts; each bot command has their own script suite that is responsible only for the execution of said command. This helps to localize issues and reduce the coupling and increase the cohesion of the project.

The user database is a Google FireBase collection of users who make use of the bot as well as their portfolios. The bot determines which user's database to access for a portfolio request by scraping that user's username from the command message, requiring no additional identification methods. The portfolio itself does not store any stock data, just a collection of tickers that a user is interested in. What the portfolio does allow is rapid report generation,

giving the user the option of generating reports for all portfolio contents by the invocation of one command.