

Voice Controlled Application For Finance (Finansis)

Student ID: 18511510019

name: Mohamed Mohamud Mohamed Nur

Major: Computer science and Technology

College: College of Computer and Artificial Intelligence

MOAHEMD MOAHMUD MOHAMED NUR 18511510019 of 2018 CS

College of Computer and Artificial Intelligence

A thesis submitted in partial satisfaction of the requirements

For The Degree of Bachelor in

Computer Science and Technology

Supervisor WU WINGO

May 2022

I Mohamed NUR with student id 18511510019 hereby to declare that the Work presented in this thesis was original and carried out by myself, supervisor WU WINGO and with the help of people that provide free information on the internet (stack-overflow and YouTube) During the period 11 /2021 - 5/2022. This work presented in this thesis has not been submitted for any other degree.

ABSTRACT

This project aims to develop a voice-controlled application for finance, a voice assistant for finance. this application is the same as Siri, (Siri is a virtual assistant that is part of Apple Inc.'s iOS) but Finansis is more focused on finance more specific the stock market. Finansis developped for traders or normal people that interested in the stock market. The system is a web application that can be accessed through the internet. This system can be used for knowing the stock market news (real-time news), knowing current a stock price (real-time price), knowing stock information, showing a stock chart, asking questions related to the stock market, back-testing a trading strategy, and the most interesting is using Algorithmic Trading to trade stocks. The project have cloud database, Using No-SQL database Mongo DB Server, and all the user interfaces have been designed using React, this project main stack is the MERN stack technologies (MongoDB, Express, React, Node) and plus python.

**Keywords:** Finance, Stock Market, Algorithmic Trading, python, MERN STACK; NO-SQL, Mongo database, scraping, automation.

TABLE OF CONTENTS

[LIST OF TABLES i](#_Toc26007)

[LIST OF FIGURES ii](#_Toc11480)

[1 INTRODUCTION 1](#_Toc19887)

[5.1.1 Future Goals For The System overview 1](#_Toc22873)

[5.1.2 Drawback of Existing system 1](#_Toc14773)

[2 SYSTEM PROBLEMS AND REQUIREMENTS ANALYSIS 2](#_Toc260)

[2.1 Functional requirement 2](#_Toc23201)

[2.2 major problems occurred in this project [2] 5](#_Toc17695)

[2.3 Data flow graph: [3] 8](#_Toc19924)

[2.3.1 The first data flow graph: 8](#_Toc4248)

[2.3.2 The second data flow graph: 8](#_Toc20818)

[2.3.3 The third data flow graph: 9](#_Toc21769)

[3 SYSTEM DESIGN 14](#_Toc5476)

[3.1 System structure diagram 15](#_Toc23108)

[3.2 Module description: 16](#_Toc125)

[3.2.1 Module description division tasks 16](#_Toc23349)

[3.3 Data Analysis 20](#_Toc7961)

[3.4 Database Design 20](#_Toc25272)

[3.4.1 Database Design division tasks 20](#_Toc1959)

[3.5 Algorithmic Design [5] 25](#_Toc12386)

[3.5.1 Algorithmic design division tasks 25](#_Toc31430)

[4 TEST AND ANAYLSIS REPORT 36](#_Toc27724)

[4.1 Test division tasks: 36](#_Toc2646)

[4.1.1 Applicant registration- Recruiter Registration - User login (Done By batis) 36](#_Toc27222)

[4.1.2 User add job – user update job – viewing applications 36](#_Toc9774)

[4.2 Applicant Registration Test [6] 36](#_Toc138)

[4.3 Recruiter Registration Test 37](#_Toc9086)

[4.4 User login Test 38](#_Toc20862)

[4.5 user adding job 39](#_Toc15384)

[4.6 user update job info 40](#_Toc13036)

[4.7 viewing Applications 41](#_Toc11580)

[5 USER MANUAL 42](#_Toc22997)

[5.1 System Operating Environment Requirements. 42](#_Toc6561)

[5.2 System Development Environment Description 42](#_Toc230)

[5.2.1 Programming language: 42](#_Toc12486)

[5.2.2 Frontend: 43](#_Toc8966)

[5.2.3 Backend 43](#_Toc28118)

[5.3 System functions (Recruiter module) 45](#_Toc29479)

[5.3.1 Add jobs 45](#_Toc4771)

[5.3.2 Jobs Applications management 45](#_Toc17365)

[5.3.3 Employees management 47](#_Toc26498)

[5.3.4 Chatting 47](#_Toc25629)

[5.4 Group photo 48](#_Toc20247)

[5.5 Detailed description of division of labor 48](#_Toc25787)

[6 Conclusion 50](#_Toc30021)

[6.1 Need for new System 50](#_Toc4023)

[6.2 Summary of gains and losses 50](#_Toc23266)

[6.2.1 Gains 50](#_Toc28112)

[6.2.2 Loses 50](#_Toc19430)

[REFERENCES 51](#_Toc20590)

LIST OF TABLES

[Table 1 Register 19](#_Toc12057)

[Table 2 Log in 20](#_Toc21054)

[Table 3 forgot (my) password 21](#_Toc12620)

[Table 4 update my information 21](#_Toc3630)

[Table 5 logout 22](#_Toc13082)

[Table 6 logout 22](#_Toc7290)

[Table 7 add {Apple|AAPL|...} stock to (my) watch list 23](#_Toc12115)

[Table 8 delete {Apple|AAPL|...} stock from (my) watch list 24](#_Toc18327)

[Table 9 open (my) watch list 24](#_Toc1600)

[Table 10 Give me the news from {yahoo finance|investing|seekingalpha} 25](#_Toc17717)

[Table 11 Give me the news from {yahoo finance|investing|seekingalpha} 25](#_Toc1848)

[Table 12 what's up with {Apple|China|...} 26](#_Toc11380)

[Table 13 Give me the latest news 26](#_Toc14446)

[Table 14 read the news 26](#_Toc7253)

[Table 15 Give me more news 27](#_Toc20749)

[Table 16 start reading (news) from article {1|5|...} 27](#_Toc9838)

[Table 17 open article (number) {1|5|...} without control 28](#_Toc6535)

[Table 18 give me top stories 28](#_Toc20963)

[Table 19 logout 29](#_Toc9769)

[Table 20 open {Apple|AAPL|...} (symbol) chart 30](#_Toc19089)

[Table 21 give me {Apple|AAPL|...} statistics 30](#_Toc8180)

[Table 22 give me The most {actives|gainers|losers} stocks 31](#_Toc4236)

[Table 23 give me trending stocks 31](#_Toc4008)

[Table 24 open {Apple|AAPL|...} (symbol) chart with your control 32](#_Toc15308)

[Table 25 what is the current price for {Apple|AAPL|...} (symbol) 32](#_Toc24353)

[Table 26 show me {Apple and Tesla} charts 33](#_Toc10275)

[Table 27 go back 34](#_Toc1254)

[Table 28 go to {info|news|...} (page) 34](#_Toc7538)

[Table 29 close the window 34](#_Toc3584)

[Table 30 zoom {in|out} 35](#_Toc18461)

[Table 31 change the chart to {1 minute|1 hour|...} 35](#_Toc31240)

[Table 32 go forward 35](#_Toc5564)

[Table 33 open details page for the answer 36](#_Toc7556)

[Table 34 what is market capitalization 36](#_Toc18964)

[Table 35 back-testing stock trading strategy commands 37](#_Toc32372)

[Table 36 reset back testing (data) 37](#_Toc24645)

[Table 37 force sell 38](#_Toc24359)

[Table 38 sell with profit or without 38](#_Toc8223)

[Table 39 stock (number) {1|5|...} 39](#_Toc12762)

[Table 40 what can you do 39](#_Toc7569)

[Table 41 what is your name 39](#_Toc4788)

[Table 42 (of course) yes 39](#_Toc11532)

[Table 43 No 40](#_Toc32122)

[Table 44 do you have a boyfriend 40](#_Toc2553)

[Table 45 can you hear me 40](#_Toc1648)

[Table 46 stop listening 40](#_Toc21848)

[Table 47 thank you 41](#_Toc5693)

[Table 48 thank you 41](#_Toc31143)

[Table 49 buy stocks 42](#_Toc3312)

[Table 50 sell stocks 42](#_Toc1542)

[Table 51 set stop-loss for stocks 43](#_Toc5696)

[Table 52 find buy signals 43](#_Toc17562)

[Table 53 trade stocks for me 43](#_Toc4888)

[Table 54 show me (a) chart for found buy signals (stocks) 44](#_Toc4143)

[Table 55 show me (a) chart for found buy signals (stocks) 44](#_Toc15443)

[Table 56 show me trading progress 45](#_Toc11553)

[Table 14 User model 47](#_Toc25284)

[Table 15 applicant model 48](#_Toc10620)

[Table 16 Recruiter model 48](#_Toc28793)

[Table 17 Job model 50](#_Toc2146)

[Table 18 Rating model 51](#_Toc3904)

[Table 19 application model 52](#_Toc15533)

[Table 20 Applicant Registration 63](#_Toc6251)

[Table 21 Recruiter Registration 64](#_Toc14038)

[Table 22 User login 65](#_Toc12753)

[Table 23 adding job 66](#_Toc2582)

[Table 24 update job info 66](#_Toc15381)

[Table 25 viewing Applications 67](#_Toc27850)

[Table 26 System Operating Environment Requirements. 68](#_Toc30251)

[Table 27 Function1-division 75](#_Toc6172)

[Table 28 Function2-division 75](#_Toc31255)

[Table 29 Function3-division 75](#_Toc19168)

LIST OF FIGURES

[Figure 1 DFD-1 8](#_Toc5135)

[Figure 2 DFD-2FD-2 8](#_Toc24038)

[Figure 3 DFD (Back-Testing)-3-1 9](#_Toc24034)

[Figure 4 DFD (Questions)-3-2 10](#_Toc13136)

[Figure 8 DFD (Stock Info)-3-3 10](#_Toc31932)

[Figure 6 DFD (User Management)-3-4 11](#_Toc6434)

[Figure 7 DFD (Stock Market News)-3-5 12](#_Toc11120)

[Figure 8 DFD (automating trading)-3-6 13](#_Toc15194)

[Figure 9 System structure diagram 15](#_Toc26002)

[Figure 10 ER-DIAGRAM 20](#_Toc24277)

[Figure 11 Insert Job 26](#_Toc26750)

[Figure 12 Delete Job 28](#_Toc2673)

[Figure 13 Get all jobs 29](#_Toc17820)

[Figure 14 Register 30](#_Toc19311)

[Figure 15 Log in 31](#_Toc27107)

[Figure 16 Update user 32](#_Toc6427)

[Figure 17 Search Job 33](#_Toc16028)

[Figure 18 Apply for Job 33](#_Toc7086)

[Figure 19 Rating a job 35](#_Toc18281)

[Figure 20 add jobs 45](#_Toc23155)

[Figure 21 jobs applications 46](#_Toc24965)

[Figure 22 applications status 46](#_Toc5417)

[Figure 23 applications status 46](#_Toc2125)

[Figure 24 employees management 47](#_Toc19625)

[Figure 25 chatting 47](#_Toc17152)

# INTRODUCTION

The main problem that Finansis application tries to solve is to help people to have financial freedom, to create a passive income for them and lastly to make the developer of this application extremely rich. there are 8 main modules in this voice-controlled application User management, Stock Market News, Stock information, controlling the app with a voice, making Finansis response, asking questions about the stock market, back-testing stock trading strategy, and automating stock trading. there are 59 commands that Finansis understand plus stock related questions (for example 'what is market capitalization’) and Finansis can ask more than 14 yes/no questions and then respond and do something based on a user's answer and lastly, Finansis can ask a user to choose stock from a list of stocks. Finansis has the ability to 'learn' about related Stock market questions and Stocks' symbol, actually Finansis just search in DuckduckGo (search engine like google) and then save the piece of information in the database. two mini-projects help Finansis news API, and Trading API, Trading API is a private project, news API is an API to save Scraped news in the database and let Finansis retrieve news from the database and news API is online (api URL:<https://news-api-lovat.vercel.app/api/v1/news).> there are four python scraping scripts running for 24 hours online, scraping news from yahoo finance, google finance, investing and seeking websites, and all that for free.

(scripts url: https://replit.com/@mHmdnwr/Chromium-Advanced-5ad#main.py)

## 5.1.1 Future Goals For The System overview

* Improving UI design.
* Trading stocks with real money.
* Making the System online.
* Creating our own voice Recognition AI.
* Adding more commands to control the System.
* Adding Smart Contracts

## 5.1.2 Drawback of Existing system

* Existing system is failing in providing high security
* The website design is not good
* Trading stock with fake money.
* Finansis is offline (only run in developer localhost)
* Finansis Can only work in Chrome Browser

# SYSTEM PROBLEMS AND REQUIREMENTS ANALYSIS

## Functional requirement

The modules involved are:

* + - User Management
    - Stock Market News
    - Stock Information
    - Controlling the app with a voice
    - Making Finansis response
    - Asking questions about the stock market
    - Back-testing stock trading strategy
    - Automating stock trading  **(**Trading bot**)**

**User management** module contains all functionality related to users. a user can register to the app, can log in, can update his/her information, can update his/her password, can add and remove stocks from his/her watch list, and if a user forgets his/her password a user can create a new password by receiving a reset token in his/her email. the main function responsible for this module is useUserCommandsHandler in the source code

**The Stock Market News** module contains all functionality related to the news. getting news by source, keyword in the headline, latest or top stories. Get more news. scraping news from yahoo finance, google finance, investing, and seeking websites. saving scraped news in the database. letting Finansis start reading news headlines from the first article, start reading them from specific article's number, or stop reading them. opening article's details page with help of Finansis or without, if a window opened with Finansis’s help, Finansis can scroll the window. the main function is useNewsCommandsHandler.

**Stock information** module contains all functionality related to stock information. opening stock chart using yahoo finance's charts with Finansis's help or without, if the chart is opened with Finansis's help, Finansis can change the chart's typed daily, 1 minute, or whatever valid type, and Finansis can also zoom the chart in or out. opening multiple charts at once only without Finansis's help. Get the stock's current price (real-time price) for free using the following API (URL: https://yahoo-finance-api.vercel.app/AAPL). helping finansis to learn about companies. showing users the most active, gainers, or losers stocks by opening yahoo finance stocks screener. showing users a stock statistics by opening yahoo finance stock statistics section. showing users sold stocks by opening yahoo finance stock chart and then changing the date range to equal date of buying and selling of the sold stock.

**Controlling the app with a voice** module is to control the app by telling Finansis commands. navigating between pages by telling finansis 'go forward', 'go back' or 'go to {news|info|...} page' ({news|info|..} is dynamic values, for example: 'go to news page'). zooming in or out charts, scrolling news page, scrolling details news page. closing any popup window.

**Making Finansis response** module is to make Finansis speak out loud. responding to users, asking users yes/no questions, or asking users to choose a stock from a stock list. stopping Finansis from recognizing when Finansis start speaking out loud.

**Back-testing stock trading strategy** module is to let users test stock trading strategy of this application. users will input start date, end date, initial cash, and account risk per trade. helping developers of this app to come up with a better trading strategy. I will talk about back-testing later in this thesis.

**Asking questions about the stock market** module is to let users ask questions about the stock market and then Finansis answers the question, if the answer of the question not exist in the database, then Finansis will search in the DuckduckGo (search engine like Google) and after that save the answer in the database and lastly answer the question so if someone asked the same question again Finansis just answer it.

**Automating stock trading (Trading Bot)** module is to let Finansis buy, sell and set stop loss for stocks using Algorithmic Trading. the trading flow first set stop loss for bought stocks (I will talk about stop loss in the Algorithmic Trading section), second find sell signals for bought stocks and then sell them if a signal has been found, third scrap S&P 500 stocks (I will also talk about S&P 500 stocks in this thesis), and then save them in python array, fourth loop through all S&P 500 stocks to find buy signals then save them in the database if a signal has been found, lastly, get found buy signal stocks from the database and then buy them. in this project for current version 1.0 all buy and sell stocks in a simulator for the stock market I used Investopedia's simulator (URL: https://www.investopedia.com/simulator).

## major problems occurred in this project [2]

* **Let Finansis recognizes English words:** solved by using react-speech-recognition npm package which uses Web Speech API.
* **Let Finansis speaks out loud:** solved by copying react-speech-kit npm package's 'useSpeechSynthesis' function and then adding more logic to the function. useSpeechSynthesis function uses Web Speech API too.
* **Run scraping news script for 24 hours**: solved by using python's webbot and Beautiful Soup packages to write scraping news scripts and Replit online IDE to run the scrips for 24 hours and also using python's request package to call news API's 'save articles' endpoint.
* **Let Finansis recognizes unknown commands**: solved by creating some logic to check for the user's command against existing commands, you can find the logic in the source code appUtils file under 'code for handling unknown commands' comment.
* **Deploy new's api**: solved by using verce.
* **Open yahoo finance website with different stocks (with dynamic value)**: solved by finding a pattern in yahoo finance' URL which is passing the stock symbol in ‘${dynamic value}’ https://finance.yahoo.com/quote/${dynamic value}?p=${dynamic value}.
* **Get real-time stock prices**: solved by finding free API to provide real-time stock prices (API URL: https://yahoo-finance-api.vercel.app/${symbol}), (youtube https://www.youtube.com/watch?v=JVZXz6awRj4&t=3573s).
* **Let Finansis calls puppeteer scripts (automation scrips)**: solved by creating auto API using node js express which is each endpoint will call specific automation script, and then using Axios to let Finansis calls the API. note: I tried to call puppeteer scripts in react which is Client-side rendering but didn't work because puppeteer scripts only run in Server-side rendering.
* **Let Finansis control the popup window that Finansis opened**: solved by using puppeteer automation functionality and then doing the same as the above problem's solution.
* **Let Finansis calls python scripts (Algorithmic Trading scripts)**: solved by creating API using python Flask each endpoint will call specific python script, and then using Axios to let Finansis calls Flask API.
* **Let the user enter info field by field and finansis keep interacting with the user**: solved by creating 'InputModal' modal which will open when the time to enter information and after a user finishes inputting then entering 'Enter' and then close the modal and save entered data in redux, lastly let finansis response when a data in redux changes.
* **Send emails to users**: solved by creating endpoint in auto API which will call puppeteer automation script to send emails to a user. note: I tried to use nodemailer but it didn't work without a VPN
* **Create back-testing interface**: solved by cloning Investopedia simulator home page and then customizing it.
* **Create back-testing functionality**: solved by creating python Flask buy, sell, get sp500 Data, update stocks price and check is market open or not endpoints and then using them in React.
* **Manage state in big application**: solved by using redux which is react state management
* **get the real stocks in the s&p 500 for a specific year in backTesting functionality**:this problem is has been not solved yet.
* **Let finansis learn about stock market questions:** solved by searching for an answer in DuckduckGo search engine using puppeteer to type the question, select the first website, scrap the answer and then save the answer in the database. actually, if the question was 'what is market capitalization' finansis will search about the question and add 'investopedia' word at the end of the question for example 'what is market capitalization investopedia' so then the first result will be from investopedia website and that because investopedia is a good website for finding stock market answers
* **Let Finansis learn about stocks symbol and their company name**: solved by searching for stock in yahoo finance database using puppeteer to type the stock in yahoo finance search bar, select all suggested stocks, scrap the stocks and then save them in the database.
* **Let Finansis remember an unknown keyword in yahoo finance database for stocks**: solved by creating 'unknownKeywordForCompany' model, if Finansis search for a keyword in yahoo finance search bar, but didn't find any stock with the given keyword so then finansis will save the given keyword in the database, for example searching 'Nur' keyword will return nothing, so next time when searching for 'Nur' keyword it will take short time for Finansis to tell a user 'didn't find a stock with the given keyword', let Finansis remember that 'Nur' keyword returns nothing so no need for searching 'Nur' again.
* **Not having duplicate news article in the database**: solved by making the article's title plus published date plus source a unique value for each article, so when the article has been scrapped, first the script will make get request to the news API by passing the unique value if the article exists then no need to save it again, otherwise send a post request to news API to save the article.
* **not let Finansis recognize her own voice**: solved by stopping Finansis from recognizing when Finansis start to speak out loud then after finishing speaking making Finansis start recognize again. in simple words toggle voice recognition.
* **Use python to build a stock market bot**: solved by searching different Algorithmic stocks trading python scripts, studying them, combining them, and then customizing them by changing conditions and input numbers.

## Data flow graph: [3]

### The first data flow graph:

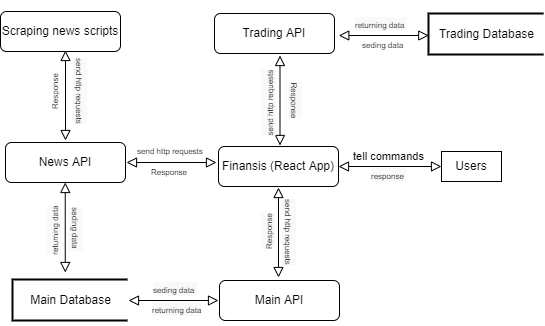


Figure 1 DFD-1

### The second data flow graph:

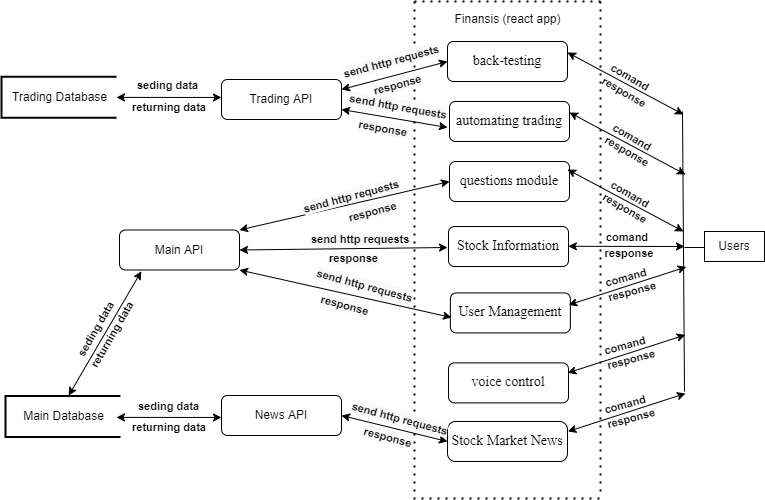


Figure 2 DFD-2FD-2

### The third data flow graph:

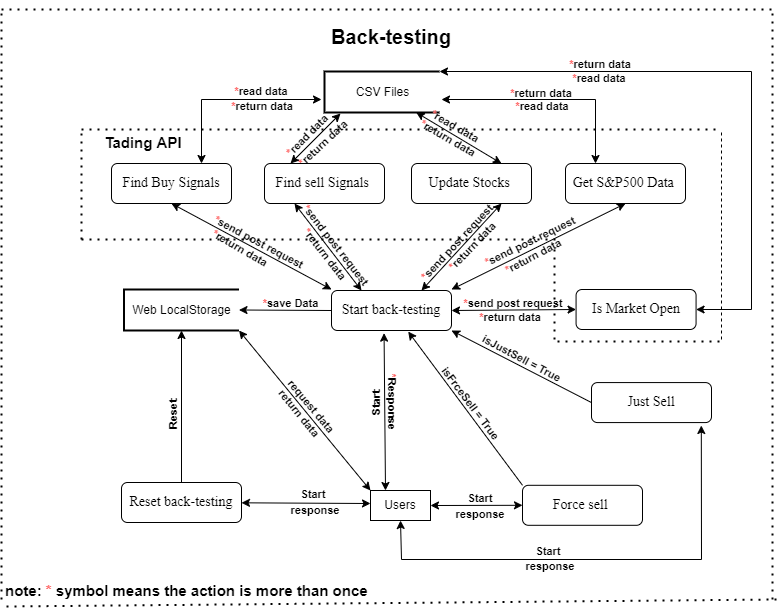


Figure 3 DFD (Back-Testing)-3-1

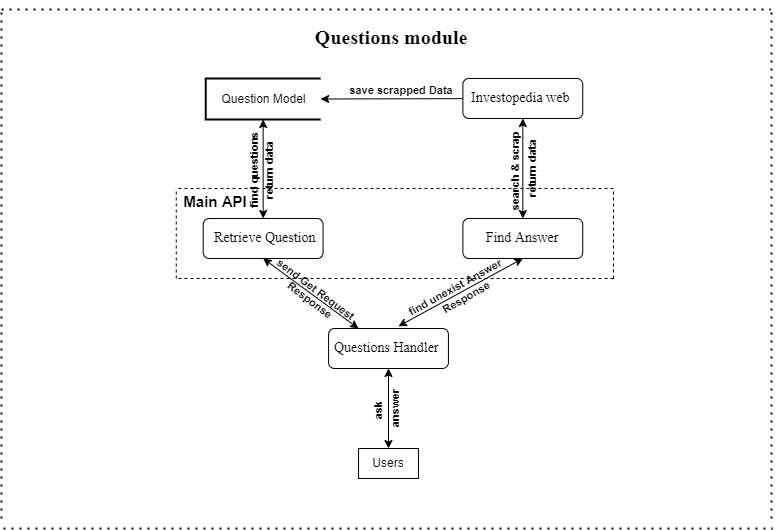


Figure 4 DFD (Questions)-3-2

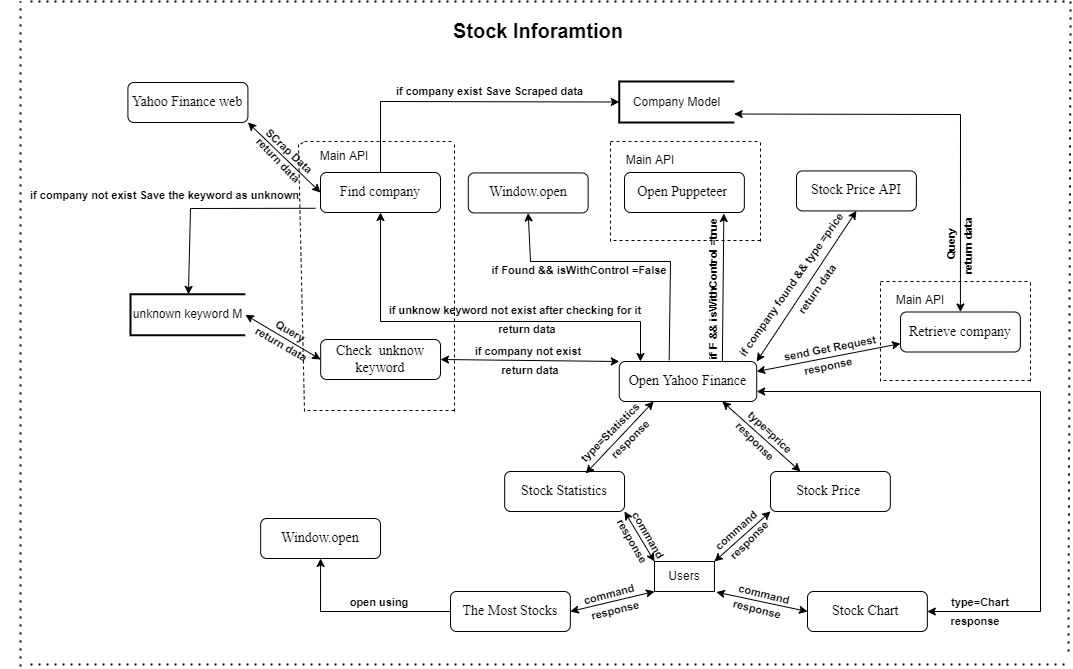


Figure 8 DFD (Stock Info)-3-3

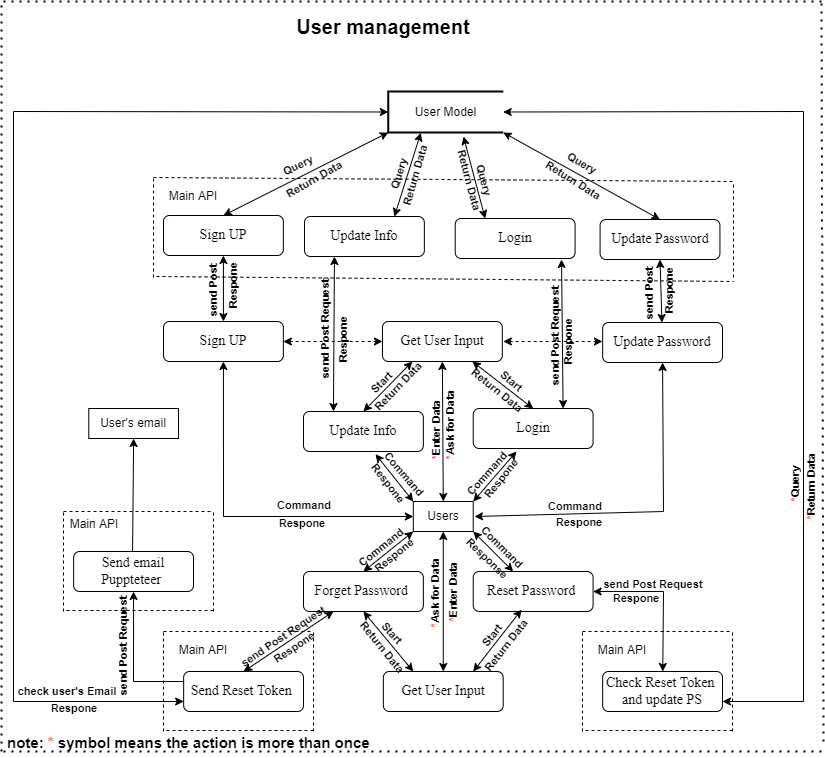


Figure 6 DFD (User Management)-3-4

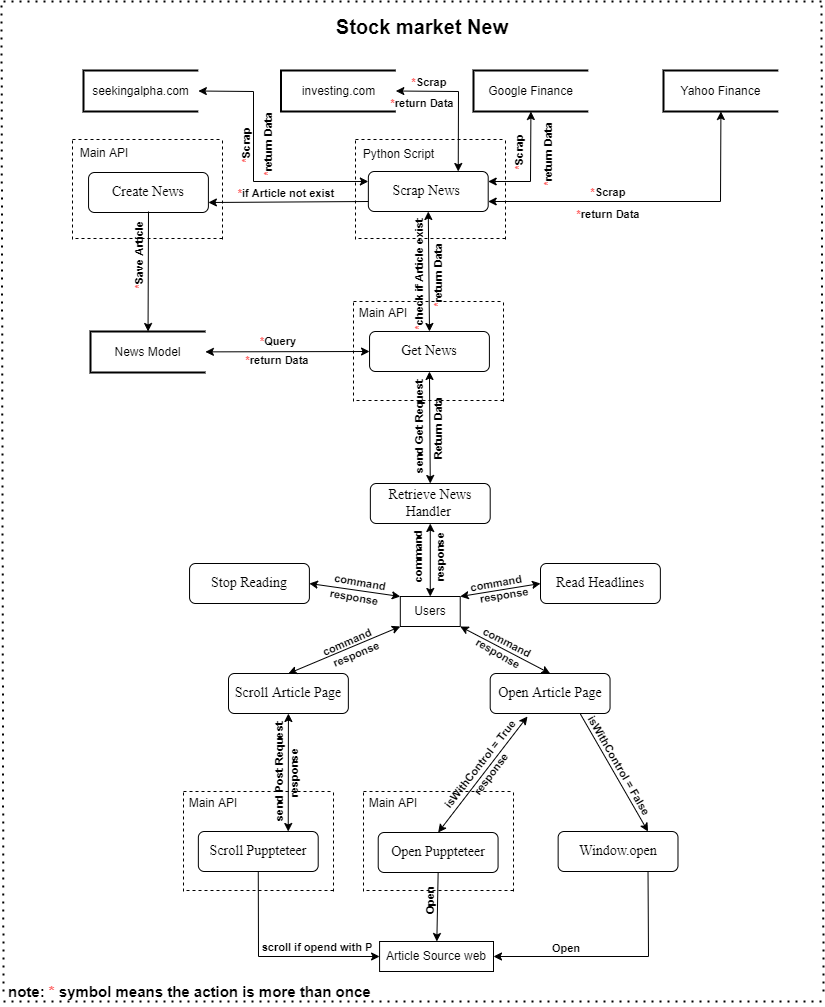


Figure 7 DFD (Stock Market News)-3-5

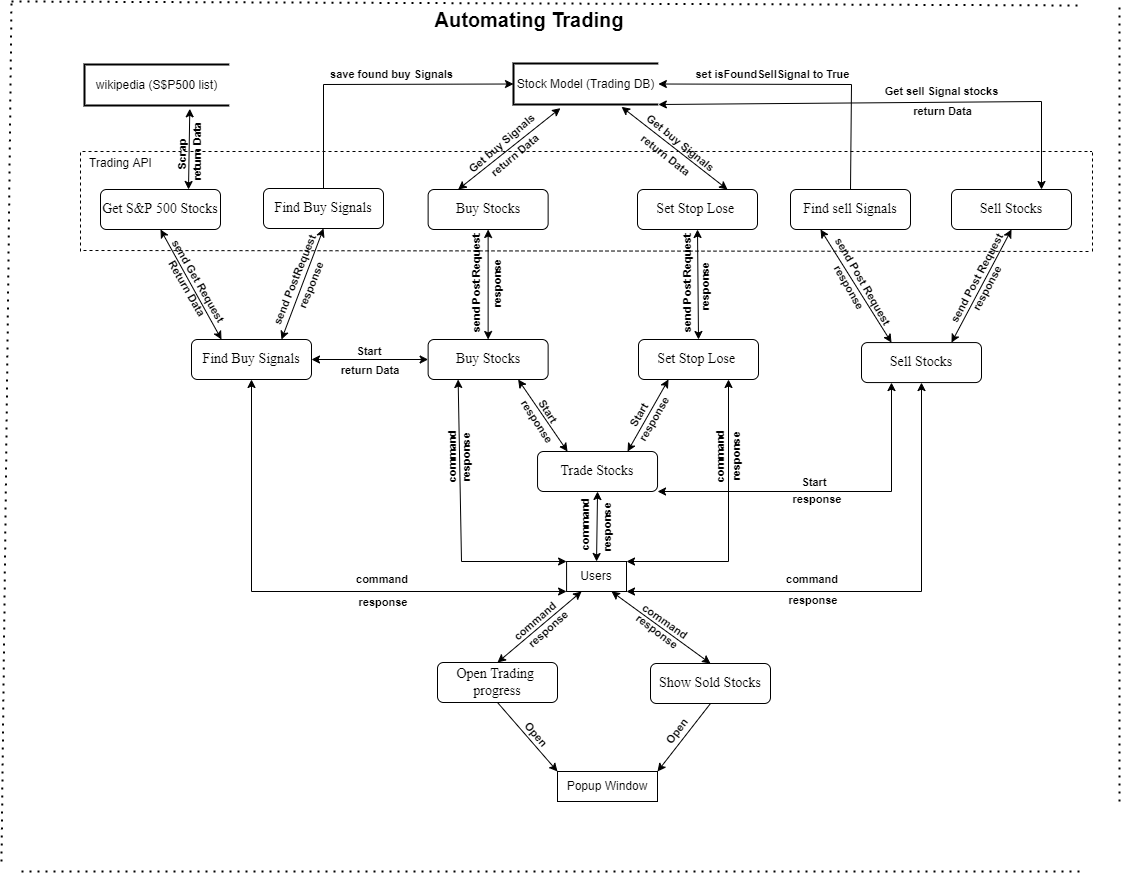


Figure 8 DFD (automating trading)-3-6

# SYSTEM DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analyzed, system design is the first of the three technical activities - design, code and test that is required to build and verify software. [4]

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

## System structure diagram

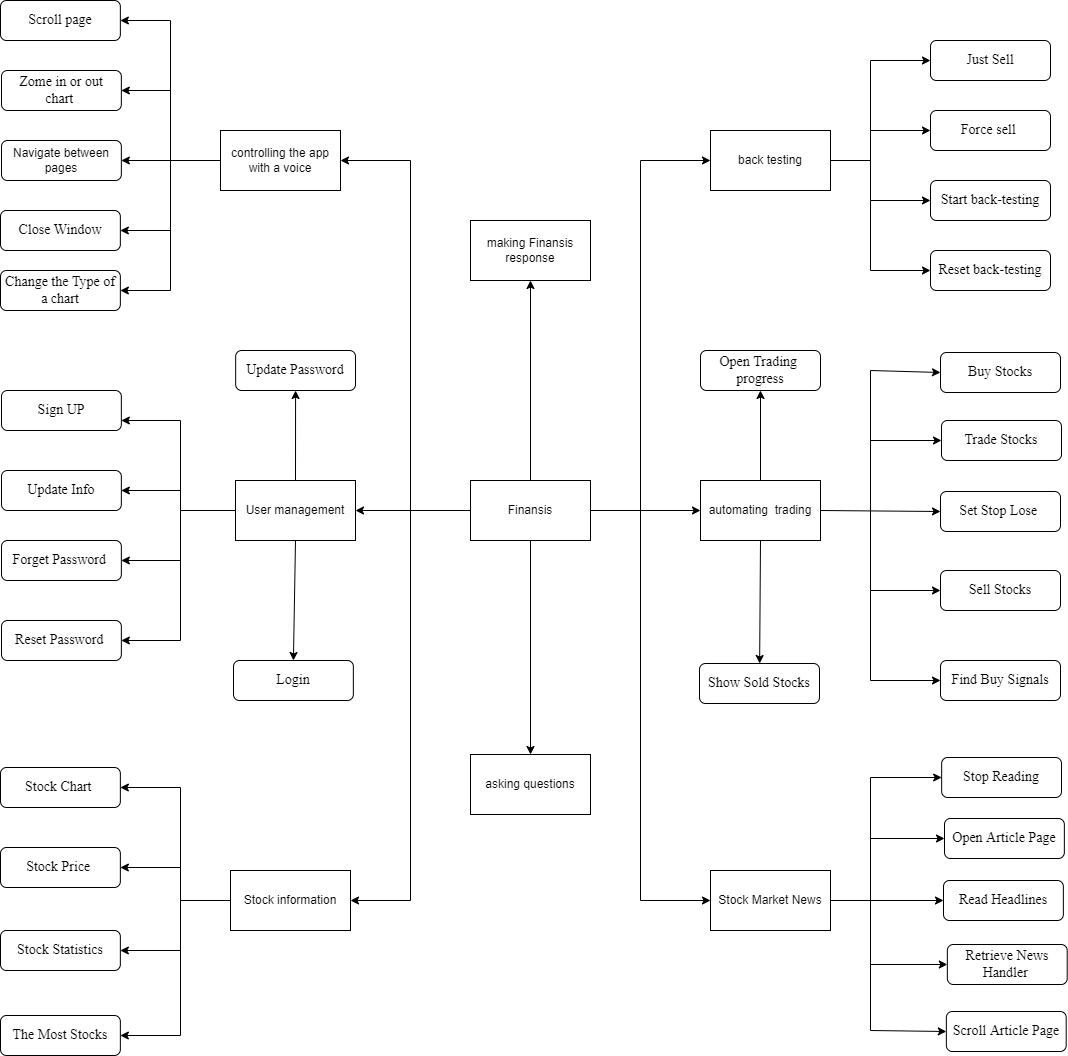


Figure 9 System structure diagram

## Commands description:

note:

- if a word between () that means the word is optional.

- if a word between {} that means the word is dynamic.

- | is separator between the dynamic options for a command.

- ... means that there is more dynamic options then words inside {}.

### User management commands (10):

* sign up
* login
* forgot (my) password
* update my information
* logout
* update (my) password
* add {Apple|AAPL|...} stock to (my) watch list
* delete {Apple|AAPL|...} stock from (my) watch list
* open (my) watch list

### Stock Market News commands (10):

* Give me the news from {yahoo finance|investing|seekingalpha}
* open article (number) {1|5|...}
* what's up with {Apple|China|...}
* Give me the latest news
* read the news
* give me more news
* start reading (news) from article {1|5|...}
* open article (number) {1|5|...} without control
* give me top stories
* give me {Apple|AAPL|...} stock news

### Stock information commands (9):

* open {Apple|AAPL|...} (symbol) chart
* give me {Apple|AAPL|...} statistics
* give me The most {actives|gainers|losers} stocks
* give me trending stocks
* open {Apple|AAPL|...} (symbol) chart with your control
* what is the current price for {Apple|AAPL|...} (symbol)
* show me {Apple and Tesla|AAPL and DD|Apple and Tesla and amazon|...} charts

### controlling the app with a voice commands (5):

* go back
* go to {info|news|...} (page)
* close the window
* zoom {in|out}
* change the chart to {1 minute|1 hour|...}

### asking questions about the stock market commands:

* open details page for the answer
* what is market capitalization
* other questions exist in the database
* and any unknown command start with 'how' or 'what'

### back-testing stock trading strategy commands (4):

* start back testing
* reset back testing (data)
* force sell
* sell with profit or without

### other commands (10):

* stock (number) {1|5|...}
* what can you do
* what is your name
* (of course) yes
* no
* do you have a boyfriend
* can you hear me
* stop listening
* thank you
* what's the date today

### automating stock trading commands (8):

* buy stocks
* sell stocks
* set stop-loss for stocks
* trade stocks for me
* find buy signals
* show me (a) chart for found buy signals (stocks)
* sold stocks chart
* show me trading progress
* User management:

|  |  |
| --- | --- |
| command | sign up |
| input | user commands 'sign up' and then enters name, email, password, confirm password, and gender |
| process | first, check if a user logged in or not, if a user is not logged in then second response 'enter your name' and open inputModal to let a user enter his/her name after a user finishes entering then the third step is to close inputModal and save the user's input, and the same process for email, password, confirm password, and gender fields but with different response messages. and then send the user's data to the sign-up endpoint in the main API, and then in the main API check the user's data if valid or not, if it is valid so create a new user in the database and then log in a user by sending JWT token and then tell a user 'signed up and logged in successfully', otherwise response with an error message. if a user is logged in so then response 'you are logged in and then ask 'do you want to log out and create a new account', if a user response with 'yes' so then do the above sign up process. |
| output | If the information all have been filled correctly, the information will be inserted in the user table. Otherwise, the system will return the error message. |

Table 1 Register

|  |  |
| --- | --- |
| Command | login |
| input | user commands 'login' and then enters email, password. |
| process | first, check if a user logged in or not, if a user is not logged in then second response 'enter your email' and open inputModal to let a user enter his/her email after a user finishes entering then the third step is to close inputModal and save the user's input, and the same process for password and then send the user's data to the login endpoint in the main API, and then in the main API check the user's data if valid or not, if it is valid so log in a user by sending JWT token and then tell a user 'logged in successfully', otherwise response with an error message. if a user is logged in so then response 'you are logged in and then ask 'do you want to logout and login with different account', if a user response with 'yes' so then do the above login process. |
| output | if the user has been already logged in then the system will response 'you are logged in' , if the information is not correct the system will return the error message otherwise log in the user. |

Table 2 Log in

|  |  |
| --- | --- |
| Command | forgot (my) password |
| input | user commands 'forgot (my) password' and then enters email. |
| process | first, check if a user logged in or not, if a user is not logged in then second response 'enter your email' and open inputModal to let a user enter his/her email after a user finishes entering then the third step is to close inputModal and save the user's input, and then send the user's data to the 'forget password' endpoint in the main API, and then in the main API check if the user's data valid or not, if it is valid so send a reset token to a user's email using 'send email' puppeteer script, otherwise response with an error message. if a user is logged in so then response 'you are logged in' and then ask 'do you want to update your password', if a user response with 'yes' so then do the update password process. |
| output | send a reset token to a user's email if a user not logged in and entered a valid email, otherwise response with a error message |

Table 3 forgot (my) password

|  |  |
| --- | --- |
| Command | update my information |
| input | user commands 'update my information' and then enters new name or new gender or both. |
| process | first, check if a user logged in or not, if a user is logged in then second response 'if you want to update your name type your new name, then click enter' and open inputModal with current user's name to let a user update his/her name after a user finishes entering then the third step is to close inputModal and update the user's input, and the same process for gender and then send the user's data to the 'update user info' endpoint in the main API, and then in the main API update user info in the database, and then send update info. if a user is not logged in so then response 'you not logged in' and then ask 'do you want to login', if a user response with 'yes' so then start the login process. |
| output | update user info if a user logged in, otherwise response with a error message |

Table 4 update my information

|  |  |
| --- | --- |
| Command | logout |
| input | user commands 'logout' |
| process | first, check if a user logged in or not, if a user is logged in so then logout the user and response 'log out successfully', otherwise response 'oops, you not logged in to logout'. |
| output | logout a user if a user logged in, otherwise response with a error message |

Table 5 logout

|  |  |
| --- | --- |
| Command | update (my) password |
| input | user commands 'update (my) password' and then enters current password, new password and confirm password. |
| process | first, check if a user logged in or not, if a user is logged in then second response 'enter your current password' and open inputModal to let a user enter his/her current password and the same process for new password and confirm password and then send the user's data to the 'update user password' endpoint in the main API, and then in the main API check if the user's data valid or not, if it is valid update user password and re-login the user, if a user is not logged in so then response 'oops, you not logged in to update your password'. |
| output | update user password if a user logged in and enter valid data, otherwise response with an error message |

Table 6 logout

|  |  |
| --- | --- |
| Command | add {Apple|AAPL|...} stock to (my) watch list |
| input | user commands 'add {Apple} stock to (my) watch list'. a user can use a company name or company stock symbol inside {}. |
| process | first, check if a user said a stock's symbol , a company name or unknown keyword if it is a stock's symbol or company name so then check it in the database and if it exist in the database so then check if the stock not in the user's watch list, if it is not add the stock in the user's watch list. if a stock' symbol or company name not exist in the database so then search the keyword in yahoo finance and then scrap the data if even didn't find any result after scraping so then save the keyword in the database as unknown key word |
| output | save stocks in user's watch list |

Table 7add {Apple|AAPL|...} stock to (my) watch list

|  |  |
| --- | --- |
| Command | delete {Apple|AAPL|...} stock from (my) watch list |
| input | user commands 'delete {Apple} stock from (my) watch list'. a user can use a company name or it's stock symbol inside {} |
| process | first, check if a user said a stock's symbol, a company name, or an unknown keyword if the input is a stock symbol or company name and it is in the database so then check if the stock is in the user's watch list, if it is delete the stock from the user's watch list. if the input is a stock symbol or company name and it is not in the database so then search the keyword in yahoo finance and then scrap the data if even didn't find any result after scraping so then save the keyword in the database as unknown keyword. |
| output | deleting stock from a user's watch list |

Table 8 delete {Apple|AAPL|...} stock from (my) watch list

|  |  |
| --- | --- |
| Command | open (my) watch list |
| input | user commands 'open (my) watch list'. |
| process | opening 'infoModal' with a user's watch list. |
| output | showing a user his/her watch list |

Table 9 open (my) watch list

* Stock Market News:

|  |  |
| --- | --- |
| Command | Give me the news from {yahoo finance|investing|seekingalpha} |
| input | user commands 'Give me the news from {yahoo finance}' a user can also use investing or seeking alpha as valid source inside {} |
| process | first, send Get HTTP request with 'source' query string to News API, and then check if the given source exists in the database, if it exists return 10 documents to the client and then finansis will show the articles to the user, otherwise return 404 code to the client and then finansis will tell the user 'sorry, I didn't find news from {source}'. |
| output | show a user news based on the source, if the source exists in the database otherwise respond with an error message |

Table 10 Give me the news from {yahoo finance|investing|seekingalpha}

|  |  |
| --- | --- |
| Command | open article (number) {1|5|...} |
| input | user commands 'open article (number) {1}'. a user can use another number if the number is in the range of existing articles in the client. if there are ten articles so then a user can choose from 1 to 10. |
| process | first, check if the user logged in, for this command a user must be logged in because this command needs to use the user's executable chrome path to open puppeteer, if the user not logged in stop this process and tell the user "you not logged in, you need to log in for this command". second, check if executableChromePath exist in the user's data, if it not exist so then tell a user "oh no, I don't have your chrome executable path" and then open InputModal lastly tell the user 'paste it here to let me control your browser' then save executableChromePath and stop this process. third, if all check points passed so then send POST HTTP request to auto API (auto API is part of main API) 'open endpoints'. fourth, in auto API open puppeteer if puppeteer opened successfully respond with 200 code (okay), otherwise 500 code (server error). |
| output | open details article page if all check points passed successfully otherwise respond with an error message. |

Table 11 Give me the news from {yahoo finance|investing|seekingalpha}

|  |  |
| --- | --- |
| Command | what's up with {Apple|China|...} |
| input | user commands 'what's up with {China}, a user can use any other keyword inside {}. |
| process | first, send Get HTTP request with 'keywordInTitle' query string to News API, and then search for the given keyword in articles' title, if found result for the search query so then return 10 documents to the client and then Finansis will show the articles to the user otherwise return 404 code to the client and then finansis will tell the user 'sorry, I didn't find news for {keyword} keyword'. |
| output | show a user news based on the given keyword, if searching for the keyword in articles' title returns articles otherwise respond with an error message |

Table 12 what's up with {Apple|China|...}

|  |  |
| --- | --- |
| Command | Give me the latest news |
| input | user commands 'Give me the latest news' |
| process | first, send Get HTTP request with 'sortBy=publishedAt' query string to News API and then return the most recent 10 documents and then finansis will show the articles to the user and if there is something wrong from News API return 500 code (server error) and then finansis will tell the user 'sorry, I didn't find any news'. |
| output | show a user recent news |

Table 13 Give me the latest news

|  |  |
| --- | --- |
| Command | read the news |
| input | user commands 'read the news' |
| process | Finansis will start reading the news' headline from first article to the last article if there are article to read otherwise tell a user 'there is no news to read.'. |
| output | Finansis starts reading the news' headline out loud |

Table 14 read the news

|  |  |
| --- | --- |
| Command | give me more news |
| input | user commands 'give me more news' |
| process | add one to page variable then send Get HTTP request with 'page=2' string query plus mine string query to News API and then return 10 articles if there are more articles and then finansis will show the articles to the user otherwise return 404 code (didn't find results) to the client and then finansis will tell the user 'sorry, I didn't find any news' . |
| output | getting more articles |

Table 15 Give me more news

|  |  |
| --- | --- |
| Command | start reading (news) from article {1|5|...} |
| input | user commands 'start reading (news) from article {5}'. a user can use another number if the number is in the range of existing articles in the client. if there are ten articles so then a user can choose any number from 1 to 10. |
| process | first, check if a number is valid number  and then Finansis will start reading the news' headline from given number article (ex: 5) to the last article if there are article otherwise tell a user 'there is no news to read.'.  if a number is invalid number so tell a user `the article with number {num} not exist so I can't read it.` |
| output | Finansis starts reading the news' headline out loud starting from the given number. |

Table 16 start reading (news) from article {1|5|...}

|  |  |
| --- | --- |
| Command | open article (number) {1|5|...} without control |
| input | user commands 'open article (number) {1} without control'. a user can use another number if the number is in the range of existing articles in the client. if there are ten articles so then a user can choose from 1 to 10. |
| process | first, check if a number is valid number, if it is a valid number open the article |
| output | opening an article page |

Table 17 open article (number) {1|5|...} without control

|  |  |
| --- | --- |
| Command | give me top stories |
| input | user commands 'give me top stories'. |
| process | irst, send Get HTTP request with 'source=google finance' string query to News API and then return top 10 articles because google Finance only shows top stories from all the news websites by default  and then finansis will show the articles to the user and if there is something wrong from News API return 500 code (server error)  and then finansis will tell the user the error message |
| output | showing top news articles |

Table 18 give me top stories

|  |  |
| --- | --- |
| Command | give me {Apple|AAPL|...} stock news |
| input | user commands 'give me {Apple} stock news'. a user can use a company name or it's stock symbol inside {} |
| process | first, check if a user said a stock's symbol, a company name, or an unknown keyword,  if the input is a stock symbol or company name and it is in the database so then open yahoo finances website using `${YAHOO\_FINANCE\_URL}/quote/${symbol}?p=${symbol}` url.  if the input is a stock symbol or company name and it is not in the database so then search the keyword in yahoo finance and then scrap the data if even didn't find any result after scraping so then save the keyword in the database as unknown keyword |
| output | opening yahoo finance website news section with a given stock |

Table 19 logout

* Stock information:

|  |  |
| --- | --- |
| Command | open {Apple|AAPL|...} (symbol) chart |
| input | user commands 'open {Apple} (symbol) chart'. a user can use a company name or it's stock symbol inside {} |
| process | first, check if a user said a stock's symbol, a company name, or an unknown keyword, if the input is a stock symbol or company name and it is in the database so then open yahoo finances website using `${YAHOO\_FINANCE\_URL}/chart/${symbol}` url, otherwise it is not in the database so then search the keyword in yahoo finance and then scrap the data if even didn't find any result after scraping so then save the keyword in the database as unknown keyword |
| output | opening yahoo finance website chart section with a given stock |

Table 20 open {Apple|AAPL|...} (symbol) chart

|  |  |
| --- | --- |
| Command | give me {Apple|AAPL|...} statistics |
| input | user commands 'give me {Apple} statistics'. a user can use a company name or it's stock symbol inside {} |
| process | first, check if a user said a stock's symbol, a company name, or an unknown keyword,  if the input is a stock symbol or company name and it is in the database so then open yahoo finances website using `${YAHOO\_FINANCE\_URL}/quote/${symbol}/key-statistics?p=${symbol}` url.  otherwise it is not in the database so then search the keyword in yahoo finance and then scrap the data if even didn't find any result after scraping so then save the keyword in the database as unknown keyword |
| output | opening yahoo finance website statistics section with a given stock |

Table 21 give me {Apple|AAPL|...} statistics

|  |  |
| --- | --- |
| Command | give me The most {actives|gainers|losers} stocks |
| input | user commands 'give me The most {actives} stocks'. a user can also use gainers or losers inside {} |
| process | first, check if a user said a valid input, if it is open yahoo finances website using `${YAHOO\_FINANCE\_URL}/most-active` url for the most active stocks, using `${YAHOO\_FINANCE\_URL}/gainers` url for the most gainers stocks and using `${YAHOO\_FINANCE\_URL}/losers` url for the most losers stocks. |
| output | opening yahoo finance website the most stocks section with a given keyword |

Table 22 give me The most {actives|gainers|losers} stocks

|  |  |
| --- | --- |
| Command | give me trending stocks |
| input | user commands 'give me trending stocks'. |
| process | open yahoo finances website using `${YAHOO\_FINANCE\_URL}/trending-tickers` url. |
| output | opening yahoo finance website trending stocks section |

Table 23 give me trending stocks

|  |  |
| --- | --- |
| Command | open {Apple|AAPL|...} (symbol) chart with your control |
| input | user commands 'open {Apple} (symbol) chart with your control'. a user can use a company name or it's stock symbol inside {} |
| process | first, check if a user said a valid stock symbol or name, if a user said invalid stock symbol or name and didn't find result after scraping stop this process. second, check if the user logged in, for this command a user must be logged in because this command needs to use the user's executable chrome path to open puppeteer, if the user not logged in stop this process and tell the user "you not logged in, you need to log in for this command". third, check if executableChromePath exist in the user's data, if it not exist so then tell a user "oh no, I don't have your chrome executable path" and then open InputModal lastly tell the user 'paste it here to let me control your browser' then save executableChromePath and stop this process fourth, if all check points passed so then send POST HTTP request to auto API (auto API is part of main API) 'open endpoints'. fifth, in auto API open puppeteer if puppeteer opened successfully respond with 200 code (okay) and open the chart, otherwise 500 code (server error). |
| output | logout a user if a user logged in, otherwise response with a error message |

Table 24 open {Apple|AAPL|...} (symbol) chart with your control

|  |  |
| --- | --- |
| Command | what is the current price for {Apple|AAPL|...} (symbol) |
| input | user commands 'what is the current price for {Apple} (symbol)'. a user can use a company name or it's stock symbol inside {} |
| process | first, check if a user said a valid stock symbol or name, if a user said invalid stock symbol or name and didn't find result after scraping stop this process. otherwise, send http request to Stock Price API then let finansis tell and show it to the user. |
| output | real-time stock price |

Table 25 what is the current price for {Apple|AAPL|...} (symbol)

|  |  |
| --- | --- |
| Command | show me {Apple and Tesla|AAPL and DD|Apple and Tesla and amazon|...} charts |
| input | user commands 'show me {Apple and Tesla} charts'. a user can use any two, three or more stocks split by 'and' word inside {}. for example 'show me {AAPL and Tesla and amazon} charts'. |
| process | first, check if all the stocks that the user said a valid stock symbol or name, if a user said invalid stock symbol or name and didn't find result after scraping stop this process. otherwise, open a window for each stock a user said. |
| output | opening multiple charts |

Table 26 show me {Apple and Tesla} charts

* controlling the app with a voice commands:

|  |  |
| --- | --- |
| Command | go back |
| input | user commands 'go back'. |
| process | first, check if there is element in the custom historyStack, if the condition true so then go back and pop the element from historyStack and then push it to forwardStack. if the condition false so then tell a user 'there is nothing back'. |
| output | navigating back |

Table 27 go back

|  |  |
| --- | --- |
| Command | go to {info|news|...} (page) |
| input | user commands 'go to {info} (page)'. a user can also use back testing, trading , home or news inside {}. |
| process | first, check if a user in the same page, if the condition false tell the user 'you are in {info} page' and then stop this process. second, if the first condition true so then check if a user said a valid page, if the condition false so then tell the user '{nur} page is not exist' and then stop this process. if all conditions true so then navigate to the given page. |
| output | navigating to the given page |

Table 28 go to {info|news|...} (page)

|  |  |
| --- | --- |
| Command | close the window |
| input | user commands 'close the window'. |
| process | fcheck if there is a window opened, if the condition true so then close the window otherwise tell a user 'there is no window open to close it' |
| output | closing popup window |

Table 29 close the window

|  |  |
| --- | --- |
| Command | zoom {in|out} |
| input | user commands 'zoom {in}'. a user can also use 'out' inside {}. |
| process | first, check if a user said a valid zooming keyword, if the condition false so then stop this process and tell a user '{nur} is not valid option for zooming' second, check if a chart window using puppeteer is opened, if the condition false so then stop this process and tell a user 'chart window is close so I can not zoom {in}' third, if all the condition true then send http request to AUTO API to zoom in or out the chart . |
| output | zooming the chart |

Table 30 zoom {in|out}

|  |  |
| --- | --- |
| Command | change the chart to {1 minute|1 hour|...} |
| input | user commands 'change the chart to {1 minute}'. a user can also use '1 hour', '1 day', '1 week' , '1 year' or any other valid chart type inside {} |
| process | first check if a chart window using puppeteer is opened if the condition false so then tell a user 'browser with my control is closed so, I can't change the chart' and stop this process, second, check if a user said a valid chart type, if the condition false so then tell a user '{Nur} is not valid option' and stop this process, third, if all the condition true so then send http request to AUTO API to change the chart type. |
| output | changing chart type |

Table 31 change the chart to {1 minute|1 hour|...}

|  |  |
| --- | --- |
| Command | go forward |
| input | user commands 'go forward'. |
| process | first, check if there is element in the forwardStack, if the condition true so then go forward and pop the element from forwardStack and then push it to historyStack. if the condition false so then tell a user 'there is nothing forward'. |
| output | navigating forward |

Table 32 go forward

* Asking questions about the stock market commands:

|  |  |
| --- | --- |
| Command | open details page for the answer |
| input | user commands 'open details page for the answer'. |
| process | first, check if there is answer to open its detail page, if the condition true so then tell a user 'there is no detail answer page to open' otherwise open a details page for the answer. |
| output | opening answers detail page (source) |

Table 33 open details page for the answer

|  |  |
| --- | --- |
| Command | what is market capitalization |
| input | user asks 'what is market capitalization'. a user can also ask any other question related to the stock market |
| process | first, check if there is answer for the question in the database, if the condition true so then answer the question,  if the condition false and question or the command starts with 'how' or 'what' so then ask a user "do you want me to find answer for 'what is nur'" if a user says yes so then search for an answer in investopedia website and then scrap the answer and save the answer in the database and lastly answer the user  if didn't find an answer after scraping so then tell a user 'didn't find answer for your question' |
| output | answering a users for their questions |

Table 34 what is market capitalization

* back-testing stock trading strategy commands:

|  |  |
| --- | --- |
| Command | start back testing |
| input | user commands 'start back testing' and then enters , initial cash, start date, account risk by trade and end date. |
| process | first, check if a user already has tested data in his local storage if the condition is true so then tell a user 'you already have tested data' and then ask a user 'do you want me to overwrite the old data if a user answers 'no' so then stop this process if a user answers 'yes' so then reset back-testing data that is in the user's local storage, after that tell a user 'enter initial cash then click enter to submit after the reloading' and reload the page after a user enters initial cash tell a user 'enter start date' and after entering let a user enters Account Risk and End Date then start back-testing by first send POST request to 'is market open' endpoint in Trading API and second, send a POST request to 'get 500 s&p index data' endpoint in Trading API third, send a POST request to 'update current stocks price' endpoint in Trading API, fourth, sending POST request to 'sell stocks Back-testing' endpoint in Trading API, fifth, send a POST request to 'buy stocks Back-testing' endpoint in Trading API and keep sending those requests one by one until the start date equals the end date. note if sending the request to 'get 500 s&p stocks' fails so send the request again after 10 seconds and only if the request has been sent successfully and then send a request to 'update current stocks price' endpoint and the same for each request.' |
| output | show a user the result for finansis's stock trading strategy |

Table 35 back-testing stock trading strategy commands

|  |  |
| --- | --- |
| Command | reset back testing (data) |
| input | user commands 'reset back testing (data)'. |
| process | reset back testing by clear the user's localStorage and reloading th page |
| output | reset back testing data |

Table 36 reset back testing (data)

|  |  |
| --- | --- |
| Command | force sell |
| input | user commands 'force sell'. |
| process | first, check if there is more data in the Trading API to test if the condition false so then tell a user 'sorry I don't have more data to test, right now' and then stop this process otherwise check if a user has stocks to sell in holdingStocks backTest array if the condition false so then tell a user 'you don't have any stocks to sell' otherwise start back testing process that mentioned above and add isForce=true flag to it so then send all the requests except 'buy stocks Back-testing' request |
| output | force sell (sell the stock without following sell strategy with any profit) |

Table 37 force sell

|  |  |
| --- | --- |
| Command | sell with profit or without |
| input | user commands 'sell with profit or without'. |
| process | first, check if there is more data in the Trading API to test if the condition false so then tell a user 'sorry I don't have more data to test, right now' and then stop this process otherwise check if a user has stocks to sell in holdingStocks backTest array if the condition false so then tell a user 'you don't have any stocks to sell' otherwise start back testing process that mentioned above and add isJustSell=true flag to it so then send all the requests except 'buy stocks Back-testing' request |
| output | start selling stocks with profit or without |

Table 38 sell with profit or without

* other commands:

|  |  |
| --- | --- |
| Command | stock (number) {1|5|...} |
| input | user respond 'stock (number) {1}', or any other valid number inside {} |
| process | after finansis ask a user 'found the following stocks choose one by saying stock number 3 for example' so then if a user choose valid number so then precede to the main process |
| output | making a user choose a stocks for a list of stocks |

Table 39 stock (number) {1|5|...}

|  |  |
| --- | --- |
| Command | what can you do |
| input | user ask 'what can you do' |
| process | Making Finansis response "I help you with finance, that's why my name is finansis". |
| output | answering the question |

Table 40 what can you do

|  |  |
| --- | --- |
| Command | what is your name |
| input | user ask 'what is your name' |
| process | making Finansis response "my name is finansis" |
| output | answering the question |

Table 41 what is your name

|  |  |
| --- | --- |
| Command | (of course) yes |
| input | user answers '(of course) yes' |
| process | after finansis asks yes no question and then a user answers 'yes' so then precede to the main process |
| output | handling 'yes' answer for yes no questions |

Table 42 (of course) yes

|  |  |
| --- | --- |
| Command | no |
| input | user answers 'no' |
| process | after Finansis asks yes no question and then a user answers 'no' so then stop the main process |
| output | handling 'no' answer for yes no questions |

Table 43 No

|  |  |
| --- | --- |
| Command | do you have a boyfriend (it's a joke) |
| input | user ask 'do you have a boyfriend'. |
| process | Finansis answers 'off course, I do. he is too handsome, too muscular, his name is' and then pause for three seconds and then finances will say the punch line 'Chris Hemsworth'. |
| output | answering the question |

Table 44 do you have a boyfriend

|  |  |
| --- | --- |
| Command | can you hear me |
| input | user ask 'can you hear me'. |
| process | if Finansis is listening so then tell a user 'I can hear you' |
| output | answering the question |

Table 45 can you hear me

|  |  |
| --- | --- |
| Command | stop listening |
| input | user commands 'stop listening' |
| process | making Finansis stop recognizing.. |
| output | stop recognizing |

Table 46 stop listening

|  |  |
| --- | --- |
| Command | thank you (it's a joke) |
| input | user says 'thank you' |
| process | Finansis response 'hate you' and then pause for three seconds  and then finances will say the punch line 'no sorry, I meant I love you'. |
| output | responding |

Table 47 thank you

|  |  |
| --- | --- |
| Command | what's the date today |
| input | user asks 'what's the date today' |
| process | tell a user the date and also open popup window to show it and after 5 seconds close the popup. |
| output | telling and showing a user current date |

Table 48 thank you

* automating stock trading commands:

|  |  |
| --- | --- |
| Command | buy stocks |
| input | user commands 'buy stocks' |
| process | first, start 'finding buy signals' process that I will mention below,  if 'finding buy signals' process finished successfully and found buy signals for stocks so then send HTTP request to 'buy stocks' endpoint in Trading API and then open Investopedia's simulator using puppeteer to automate buying stocks and then save bought stocks in the database otherwise tell a user something went wrong while buying Stocks. |
| output | buying stocks using puppeteer automation functionality |

Table 49 buy stocks

|  |  |
| --- | --- |
| Command | sell stocks |
| input | user commands 'sell stocks' |
| process | first, finding sell signals for bought stocks, if there is an error tell a user 'something went wrong while checking for selling signals' and stop this process otherwise check if found stocks if didn't find any signals so then tell a user 'didn't find any sell signals today' and stop this process otherwise start the second step which is selling stock using puppeteer by sending an HTTP request to Trading API.  note if there were an error while doing this process Finansis will ask the user 'do you want me to try to {sell} again' if the answer is yes try selling again. |
| output | finding sell signals and selling stocks using puppeteer automation functionality |

Table 50 sell stocks

|  |  |
| --- | --- |
| Command | set stop-loss for stocks |
| input | user commands 'set stop-loss for stocks' |
| process | send an HTTP request to trading API's 'set stop loss' endpoint and open Investopedia's simulator using puppeteer to automate setting a stop loss for stocks if there were an error so then tell a user 'something went wrong while buying stop loss' or if didn't find any stocks to set stop loss to them so tell a user "didn't found stocks to set stop loss to them". |
| output | setting stop loss for stocks using puppeteer automation functionality |

Table 51 set stop-loss for stocks

|  |  |
| --- | --- |
| Command | find buy signals |
| input | user commands 'find buy signals' |
| process | first, send an HTTP request to Trading API's 'get S&P 500 stocks' endpoint to scrap s&p 500 stocks and save them, if there were an error so tell a user 'something went wrong while saving S&P 500 stocks' otherwise, send an HTTP request to Trading API's 'find buy signals' endpoint to find buy signals and then save them in the database if there were an error so tell a user 'something went wrong while looking for buy signals' and if didn't find any buy signal so tell a user 'didn't find any buy signal' |
| output | scraping s&p 500 stocks and finding buy signals |

Table 52 find buy signals

|  |  |
| --- | --- |
| Command | trade stocks for me |
| input | user commands 'trade stocks for me' |
| process | first, start the above 'set stop loss' process, second, start the above 'sell stocks' process, lastly, start the above 'buy stocks' process. |
| output | start all trading processes |

Table 53 trade stocks for me

|  |  |
| --- | --- |
| Command | show me (a) chart for found buy signals (stocks) |
| input | user commands 'show me (a) chart for found buy signals (stocks)' |
| process | opening all stocks that found buy signal for them using the above 'opening multiple chart' process |
| output | opening charts for stock that found buy signal for them |

Table 54 show me (a) chart for found buy signals (stocks)

|  |  |
| --- | --- |
| Command | sold stocks chart |
| input | user commands 'sold stocks chart' |
| process | first, check if the user logged in, for this command a user must be logged in because this command needs to use the user's executable chrome path to open puppeteer, if the user is not logged in so then stop this process and tell the user "you not logged in, you need to log in for this command". second, let a user choose one of the sold stocks by telling him 'the following stocks have been sold choose one by saying stock number 3 for example' third, check if executableChromePath exists in the user's data, if it not exist so then tell a user "oh no, I don't have your chrome executable path" and then open InputModal lastly tell the user 'paste it here to let me control your browser' then save executableChromePath and stop this process fourth, if all checkpoints passed so then send a POST HTTP request to auto API (auto API is part of the main API) 'open endpoints'. fifth, in auto API open puppeteer, if puppeteer opened successfully respond with 200 code (okay) and open the chart, otherwise 500 code (server error). lastly, send an HTTP request to Trading API's 'change chart date' endpoint to adjust the start date and end date of the chart. |
| output | opening charts for stock that found buy signal for them |

Table 55 show me (a) chart for found buy signals (stocks)

|  |  |
| --- | --- |
| Command | show me trading progress |
| input | user commands 'show me trading progress' |
| process | tell a user 'here is your trading progress' and then open Progress Modal and then after 10 seconds close the modal |
| output | opening Progress Modal |

Table 56 show me trading progress

## Topics Related To This Project

### The Stock Market

Let’s imagine a new coffee company that decides to launch on the market. First, the company will advertise itself to big investors. If they think the company is a good idea, they get the first crack at investing, and then sponsor the company’s initial public offering, or IPO. This launches the company onto the official public market, where any company or individual who believes the business could be profitable might buy a stock. Buying stocks makes those investors partial owners in the business. Their investment helps the company to grow, and as it becomes more successful, more buyers may see potential and start buying stocks. As demand for those stocks increases, so does their price, increasing the cost for prospective buyers, and raising the value of the company's stocks people already own. For the company, this increased interest helps fund new initiatives, and also boosts its overall market value by showing how many people are willing to invest in their idea. However, if for some reason a company starts to seem less profitable the reverse can also happen. If investors think their stock value is going to decline, they’ll sell their stocks with the hopes of making a profit before the company loses more value. As stocks are sold and demand for the stock goes down, the stock price falls, and with it, the company’s market value. This can leave investors with big losses– unless the company starts to look profitable again. This see-saw of supply and demand is influenced by many factors. Companies are under the unavoidable influence of market forces– such as the fluctuating price of materials, changes in production technology, and the shifting costs of labor. Investors may be worried about changes in leadership, bad publicity, or larger factors like new laws and trade policies. And of course, plenty of investors are simply ready to sell valuable stocks and pursue personal interests. All these variables cause day-to-day noise in the market, which can make companies appear more or less successful. And in the stock market, appearing to lose value often leads to losing investors, and in turn, losing actual value.

Check out the following videos:

<https://clipchamp.com/watch/562lTGNvypO>

<https://clipchamp.com/watch/QggpdDFirxk>

### S&P500

The Standard and Poor's 500, or simply the S&P 500, is a stock market index tracking the performance of 500 large companies listed on stock exchanges in the United States. It is one of the most commonly followed equity indices. As of December 31, 2020, more than $5.4 trillion was invested in assets tied to the performance of the index. a stock market index is a calculation that gives information about a group of stocks.

check out the following videos:

<https://www.youtube.com/watch?v=VJQ6-DDr3jA>

<https://clipchamp.com/watch/IM2k3GuJQQb>

### Algorithmic Trading

Algorithmic trading is a method of executing orders using automated pre-programmed trading instructions accounting for variables such as time, price, and volume.[1] This type of trading attempts to leverage the speed and computational resources of computers relative to human traders. In the twenty-first century, algorithmic trading has been gaining traction with both retail and institutional traders.[2][3] It is widely used by investment banks, pension funds, mutual funds, and hedge funds that may need to spread out the execution of a larger order or perform trades too fast for human traders to react to. A study in 2019 showed that around 92% of trading in the Forex market was performed by trading algorithms rather than humans.

check out the following video:

https://www.youtube.com/watch?v=ezom23o2boM

### How the News Affects Stock Prices

Stock prices tick up and down constantly due to fluctuations in supply and demand. If more people want to buy a stock, its market price will increase. If more people are trying to sell a stock, its price will fall. The relationship between supply and demand is highly sensitive to the news of the moment. Negative news will normally cause people to sell stocks. A bad earnings report, a lapse in corporate governance, big-picture economic and political uncertainty, and unfortunate occurrences all translate to selling pressure and a decrease in the prices of many if not most stocks. Positive news will normally cause individuals to buy stocks. Good earnings reports, an announcement of a new product, a corporate acquisition, and positive economic indicators all translate into buying pressure and an increase in stock prices.

### Backtesting

Backtesting allows a trader to simulate a trading strategy using historical data to generate results and analyze risk and profitability before risking any actual capital.

A well-conducted backtest that yields positive results assures traders that the strategy is fundamentally sound and is likely to yield profits when implemented in reality. In contrast, a well-conducted backtest that yields suboptimal results will prompt traders to alter or reject the strategy.

check out the following video:

<https://www.youtube.com/watch?v=JGxxXSCerXU>

### yahoo finance

Yahoo! Finance is a media property that is part of the Yahoo! network. It provides financial news, data and commentary including stock quotes, press releases, financial reports, and original content. It also offers some online tools for personal finance management. In addition to posting partner content from other web sites, it posts original stories by its team of staff journalists. It is ranked 21st by SimilarWeb on the list of largest news and media websites

### Risk management in stock market

Risk management refers to analysing potential loss from investments that investors carry out and any appropriate action to mitigate the chances of incurring a said loss. Today, risk management forms an essential part of strategies used by investors while trading.

check out the following videos:

<https://videos-fms.jwpsrv.com/0_6278ebb9_0xcc8eff2b2c2dc80fe5f2ca2b394fec1ecb66c2c6/content/conversions/hmUZMORz/videos/Ep8Z0yLr-3480264.mp4>

<https://www.youtube.com/watch?v=RE9tk78ALd0>

### Understanding Position Sizing

Position sizing refers to the size of a position within a particular portfolio, or the dollar amount that an investor is going to trade. Investors use position sizing to help determine how many units of security they can purchase, which helps them to control risk and maximize returns.

### The Stop-Loss Order

A stop-loss order is an order placed with a broker to buy or sell a specific stock once the stock reaches a certain price. A stop-loss is designed to limit an investor's loss on a security position. For example, setting a stop-loss order for 10% below the price at which you bought the stock will limit your loss to 10%. Suppose you just purchased Microsoft (MSFT) at $20 per share. Right after buying the stock, you enter a stop-loss order for $18. If the stock falls below $18, your shares will then be sold at the prevailing market price.

check out the following videos:

<https://videos-fms.jwpsrv.com/0_6278e83a_0x501fe19b6fa687c1a7541a7975361970015b6fc7/content/conversions/hmUZMORz/videos/uR7a5JPa-3480264.mp4>

## Technologies

### programming languages:

- Python

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

- JavaScript

JavaScript, often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. Over 97% of websites use JavaScript on the client side for web page behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard.[14] It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js.

### Runtime system

### - Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[6] unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games). The Node.js distributed development project was previously governed by the Node.js Foundation, and has now merged with the JS Foundation to form the OpenJS Foundation, which is facilitated by the Linux Foundation's Collaborative Projects program.

### Frontend

- React

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library[3] for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

- Html

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[2] A form of HTML, known as HTML5, is used to display video and audio, primarily using the <canvas> element, in collaboration with javascript.

- css

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.[1] CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.[3] This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

- material-ui

Install Material UI, the world's most popular React UI framework.

- styled-components

Styled components are a CSS-in-JS tool that bridges the gap between components and styling, offering numerous features to get you up and running in styling components in a functional and reusable way. It also removes the mapping between components and styles – using components as a low-level styling construct could not be easier!

- Web Speech API (react-speech-recognition)

The Web Speech API makes web apps able to handle voice data. There are two components to this API:

1 - Speech recognition is accessed via the SpeechRecognition interface, which provides the ability to recognize voice context from an audio input (normally via the device's default speech recognition service) and respond appropriately. Generally you'll use the interface's constructor to create a new SpeechRecognition object, which has a number of event handlers available for detecting when speech is input through the device's microphone. The SpeechGrammar interface represents a container for a particular set of grammar that your app should recognize. Grammar is defined using JSpeech Grammar Format (JSGF.)

2 - Speech synthesis is accessed via the SpeechSynthesis interface, a text-to-speech component that allows programs to read out their text content (normally via the device's default speech synthesizer.) Different voice types are represented by 2- SpeechSynthesisVoice objects, and different parts of text that you want to be spoken are represented by SpeechSynthesisUtterance objects. You can get these spoken by passing them to the SpeechSynthesis.speak() method.

- Axios

Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests.

- Redux

Redux is an open-source JavaScript library for managing and centralizing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. Similar to (and inspired by) Facebook's Flux architecture, it was created by Dan Abramov and Andrew Clark. Since mid-2016, the primary maintainers are Mark Erikson and Tim Dorr.

- Create-react-app

The create-react-app is an excellent tool, which allows you to create and run React project very quickly. It does not take any configuration manually. This tool is wrapping all of the required dependencies like Webpack, Babel for React project itself and then you need to focus on writing React code only. This tool sets up the development environment, provides an excellent developer experience, and optimizes the app for production.The create-react-app is an excellent tool for beginners, which allows you to create and run React project very quickly. It does not take any configuration manually. This tool is wrapping all of the required dependencies like Webpack, Babel for React project itself and then you need to focus on writing React code only. This tool sets up the development environment, provides an excellent developer experience, and optimizes the app for production.

### Backend

- axios

Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests.

- bcrypt

A library to help hashing passwords using bcrypt hashing function.

bcrypt is a password-hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher and presented at USENIX in 1999.[1] Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power. There are implementations of bcrypt in C, C++, C#, Embarcadero Delphi, Elixir, Go, Java, JavaScript, Perl, PHP, Python, Ruby, and other languages.

- cors

CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served.

- crypto

JavaScript library for cryptographic hash functions. A cryptographic hash function (CHF) is a mathematical algorithm that maps data of an arbitrary size (often called the "message") to a bit array of a fixed size (the "hash value", "hash", or "message digest"). It is a one-way function, that is, a function for which it is practically infeasible to invert or reverse the computation

- Dotenv

Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env. Storing configuration in the environment separate from code is based on The Twelve-Factor App methodology.

- Express

Express is a popular unopinionated (Not opinionated, can use it in different styles) web framework, written in JavaScript and hosted within the Node.js runtime environment. for creating API. API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.

- Jsonwebtoken

JavaScript library for JSON Web Tokens. JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

- mongoose

Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks.

- morgan

HTTP request logger middleware for node.js

- nodemon

nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

### Package Manager

- pip

pip is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes.

- npm

npm (originally short for Node Package Manager) is a package manager for the JavaScript programming language maintained by npm, Inc. npm is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc.

### IDE

- VS Code

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

### Database

- Mongodb

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc.

- MongoDB Atlas

A fully managed, global cloud database for MongoDB

### version control

- GitHub

GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project.[4] Headquartered in California, it has been a subsidiary of Microsoft since 2018

- Git

Git is a software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems)

### Scraping

- Replit

Replit (rep·lit), formerly Repl.it, is a San Francisco-based start-up and an online IDE (integrated development environment). Replit allows users to write code and build apps and websites using a browser.

- webbot

Web automation library for python for web browser automation and end to end UI testing. webbot provides a much feature rich automation than selenium for all kinds of automation of webpage. Since the major portion of web automation is to perform actions like click and type into webpage elements , webbot automatically handles finding the right elements to perform the actions.

- BeautifulSoup

Beautiful Soup is a Python library for pulling data out of HTML and XML files.

- Cheerio

Cheerio parses markup and provides an API for traversing/manipulating the resulting data structure. It does not interpret the result as a web browser does.

### Automation

- Puppeteer

Puppeteer is a Node library which provides a high-level API to control Chrome or Chromium over the DevTools Protocol.

### Algorithmic Trading

- Flask

it is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. it is used for creating API.

- Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

- requests

Requests is a HTTP library for the Python programming language. The goal of the project is to make HTTP requests simpler and more human-friendly.

- Ta

It is a Technical Analysis library to financial time series datasets (open, close, high, low, volume). You can use it to do feature engineering from financial datasets. It is built on Python Pandas library.

### other technologies

- Google Search engine

- Stack overflow

- Youtube