Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

B.Tech. CE Semester – VI

Subject: SDP

Project Title:

Stock Market Look-up App

Submitted By:

Khunt Brijesh CE063 18CEUOS048

Meet Dobariya CE035 18CEUOG031

Guided By:

Prof. Prashant M. Jadav

Asso. Professor, CE Dept.

Dharmsinh Desai University

DHARMSINH DESAI UNIVERSITY, NADIAD

FACULTY OF TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the project entitled "Stock Market Look-Up System" is a bonafide report of the work carried out by Brijesh Khunt, Student ID 18CEUOS048 and Meet Dobariya, Student ID 18CEUOG031 of Department of Computer Engineering, semester VI, under the guidance and supervision for the subject Software Development Project. They were involved in Project training during the academic year 2020- 2021.

Prof. Prashant M. Jadav

Department of Computer Engineering,

Faculty of Technology,

Dharmsinh Desai University,

Nadiad.

Prof. C. K. Bhensdadia

Head of department.

Table of Content

Table of Content **3**

Abstract **4**

Introduction **5**

Purpose : **5**

Scope : **5**

Technologies : **6**

Tools : **6**

Software Requirement Specification **7**

Functional Requirements **7**

Design **9**

Use Case Diagram : 9

Class Diagram : **10**

Activity Diagram : 11

Sequence Diagram : 12

Data Dictionary 15

Implementation Details **18**

Screen-shots of the System **19**

Conclusion **26**

Limitations and Future Enhancement **27**

Reference / Bibliography **28**

Abstract

Stock Market systems contain real time live prices of the stocks and indexes.Using our system users can make an account and also get the historical prices and charts. so that users can make decisions when to buy and how much. Users can also make triggers on the stocks.User can also make watch-lists as he wants.Our System provides reliable price charts and also information about the company. Our system provides information about basic fundamental data about the company. Users can also add indicators in the chart.

Introduction

Stock Market look-up system has 2 users which is End User and Admin.

In our system, we can provide a search option which is used for searching companies and add them to the watchlist. Users also can look into the chart and take decisions on them. Users also can get OHLC (Open,High,Low,Close) price of that day.User can Produce Trigger so that when price reaches to that level then system notified the user. Users can also view the company details (like current price,symbol,sectors,description,etc..).

Purpose :

This document consists of all the functional and nonfunctional requirements of the Stock Market Look-up system and all the other requirement related details. This SRS document will be used as a reference and a guidance for the designer and development process ahead.

Scope :

This System can be useful to everyone who wants to see the real-time price of the US market stock and for he/she needs to register through our portal.This system is designed to help traders and investors to do research on the market at one platform. Scope of the application is global and open for all users. End Users have to provide his details (name,email,phone_no.) and other information which is used to notify users about triggering so that they can use this platform.

Technologies :

- HTML 5, CSS3
- Material UI
- React JS
- Express JS
- Node.JS
- Highcharts
- MongoDB
- Alpaca API for getting market data
- Polygon.io for getting fundamental data

Tools :

- Visual Studio Code with Extension for typescripting(ESLint)
- Postman for testing Web Request
- MongoDB compass (GUI of MongoDB)

Software Requirement Specification

Types of User

- End User
- Admin

Functional Requirements

R.1: User

R.1.1: Login

IP: Email and Password

OP: login success and redirected to Home page

R.1.2: Register

IP: Name, Email, Password, Phone number

OP: Registration Successful

R.1.3: View the company_details

IP: Company Symbol

OP: Name,Current Price,(OHLC),Sector,Description,

R.1.4: Add the Stocks on Watch-List

IP: Company Symbol

OP: Stocks price and name shown to the dashboard

R.1.5: Logout

IP: click on logout

OP: Successfully Logged out

R.1.6: View Chart (Historical and Live)

IP: click on selected stock in watchlist

OP: View chart using Highchart library

R.2: Triggering

R.2.1: Put the Trigger

IP: Trigger Price of the particular stock

OP: success/failure message

R.2.2: Update the Trigger

IP: Updated Trigger Price

OP: success/failure message

R.2.3: Delete the Trigger

IP: Triggered Stock

OP: success/failure message

R.3: Admin

R.3.1: Login

IP: email and password

OP: success login

R.3.2: View user

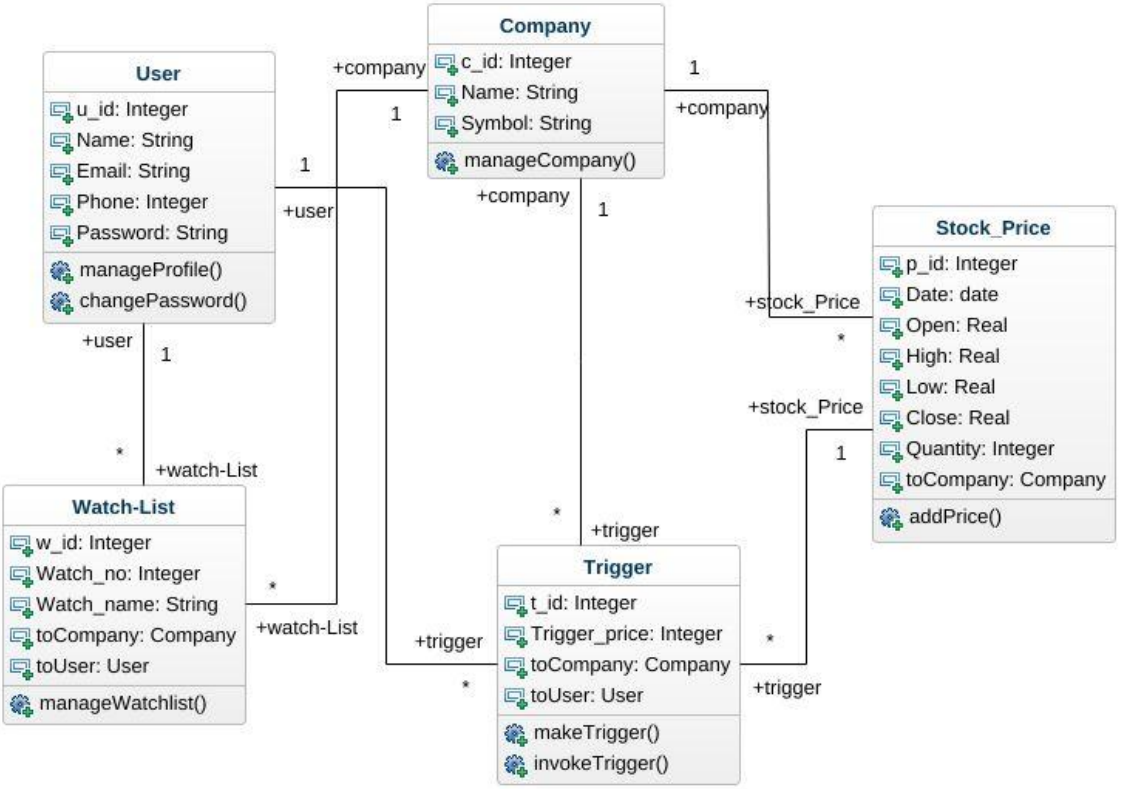
IP: UserName

OP: details of End User

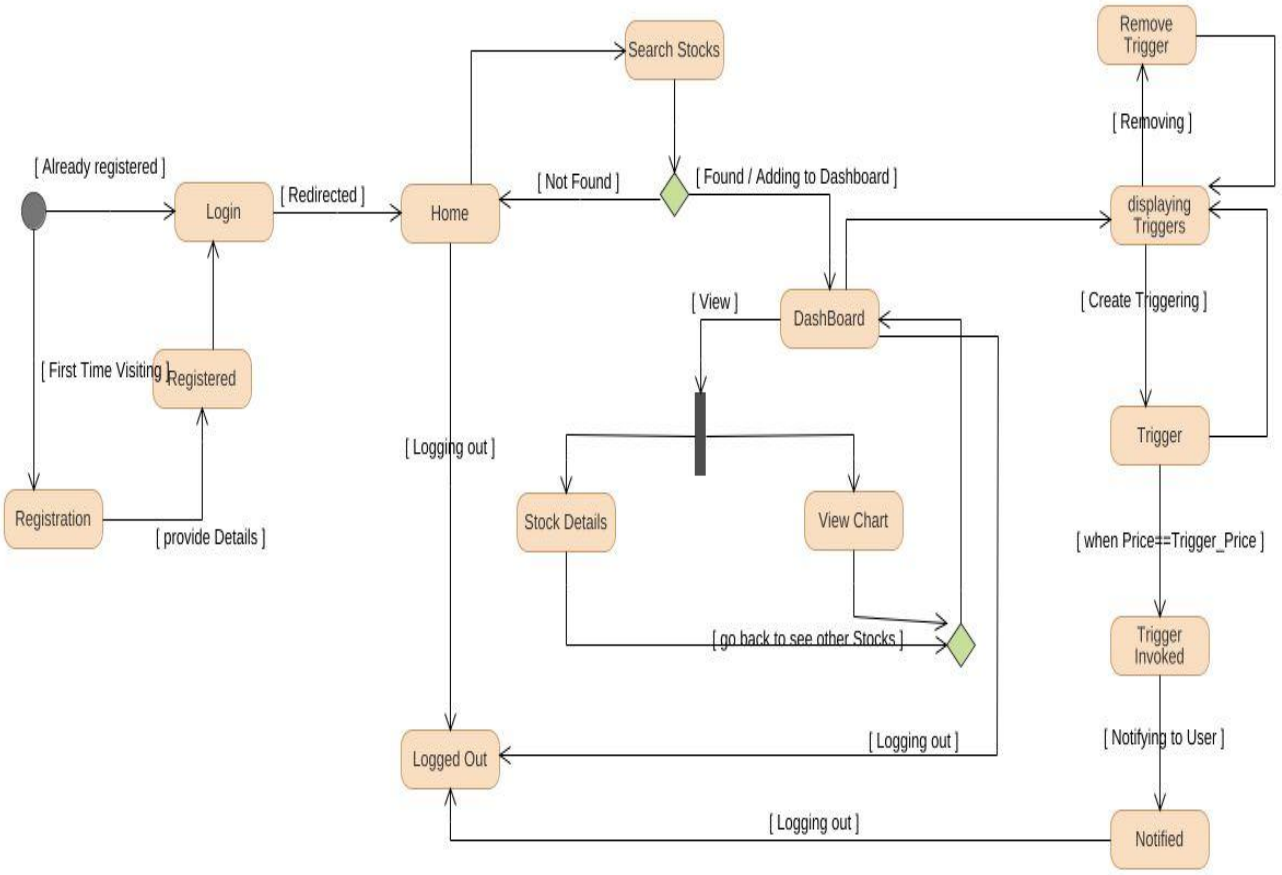
Design

Use Case Diagram :

Class Diagram :

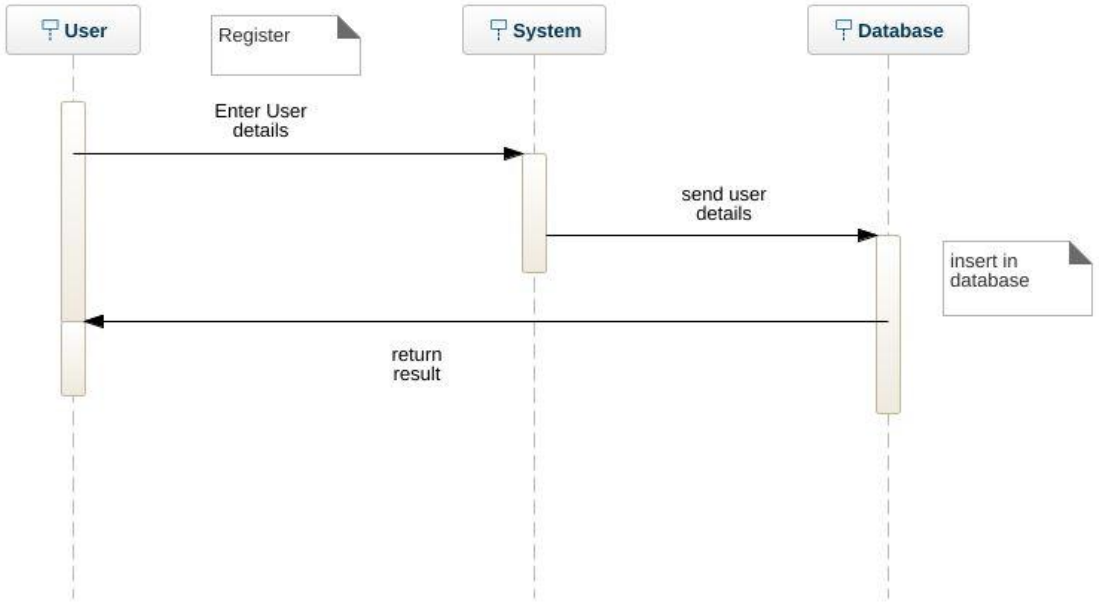


Activity Diagram :

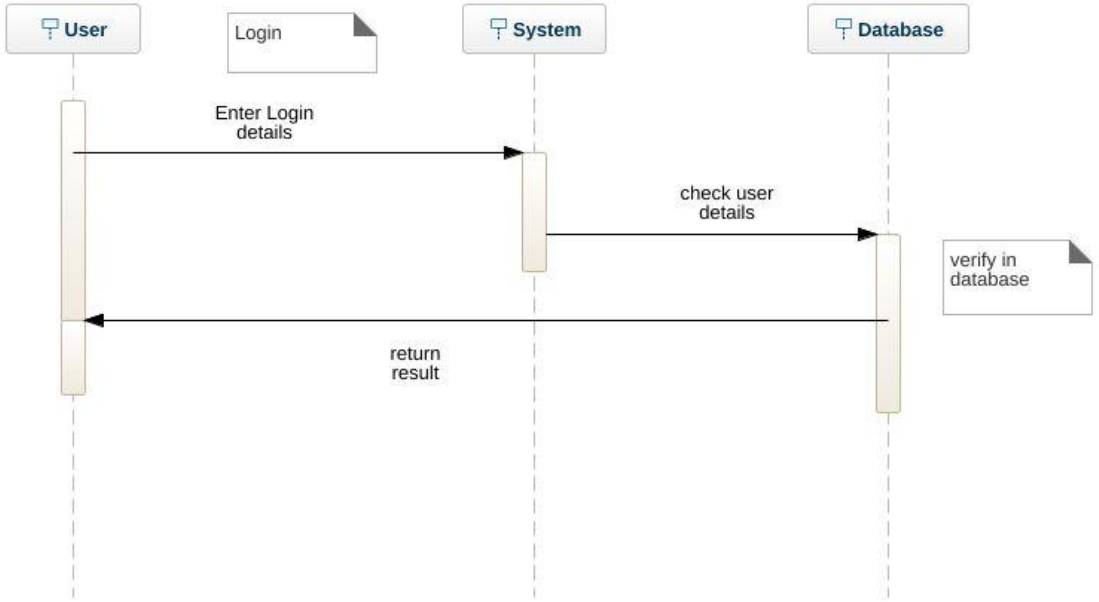


Sequence Diagram :

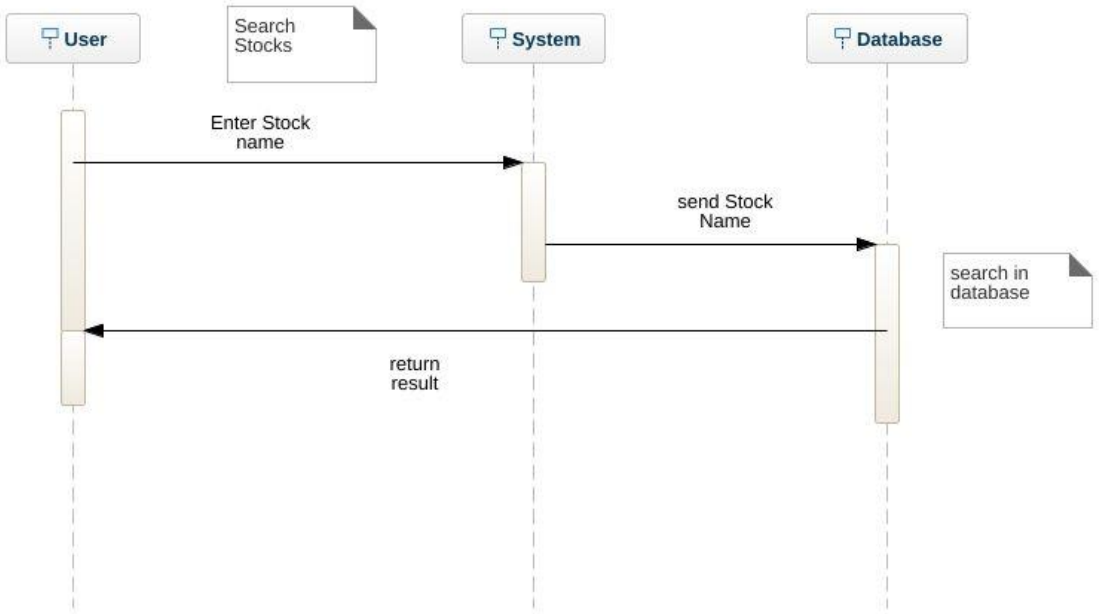
→ Register :



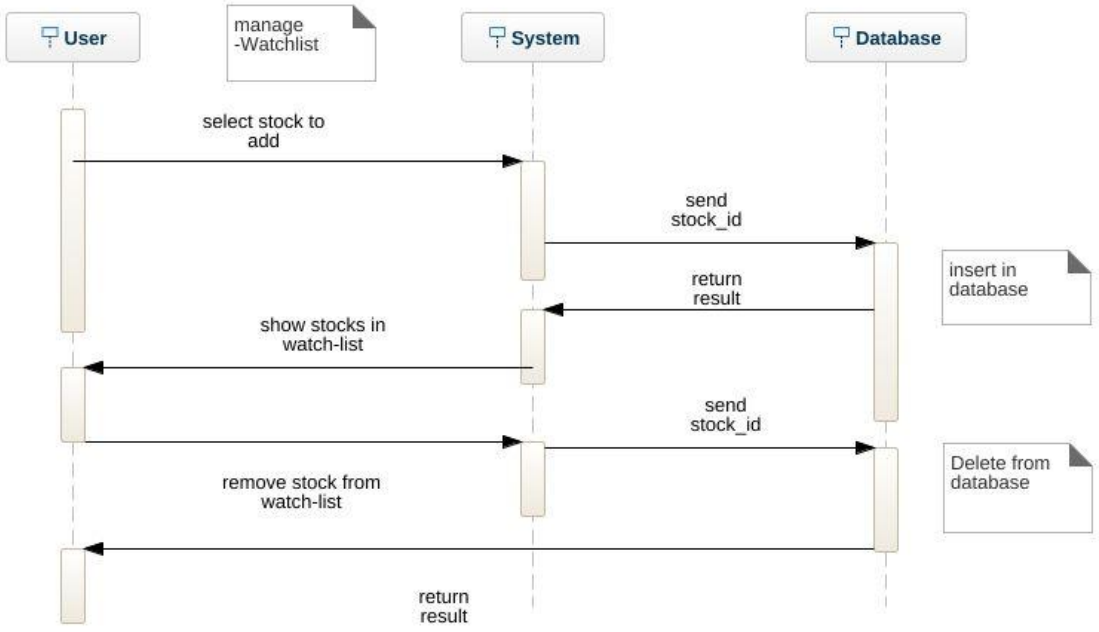
→ Login :



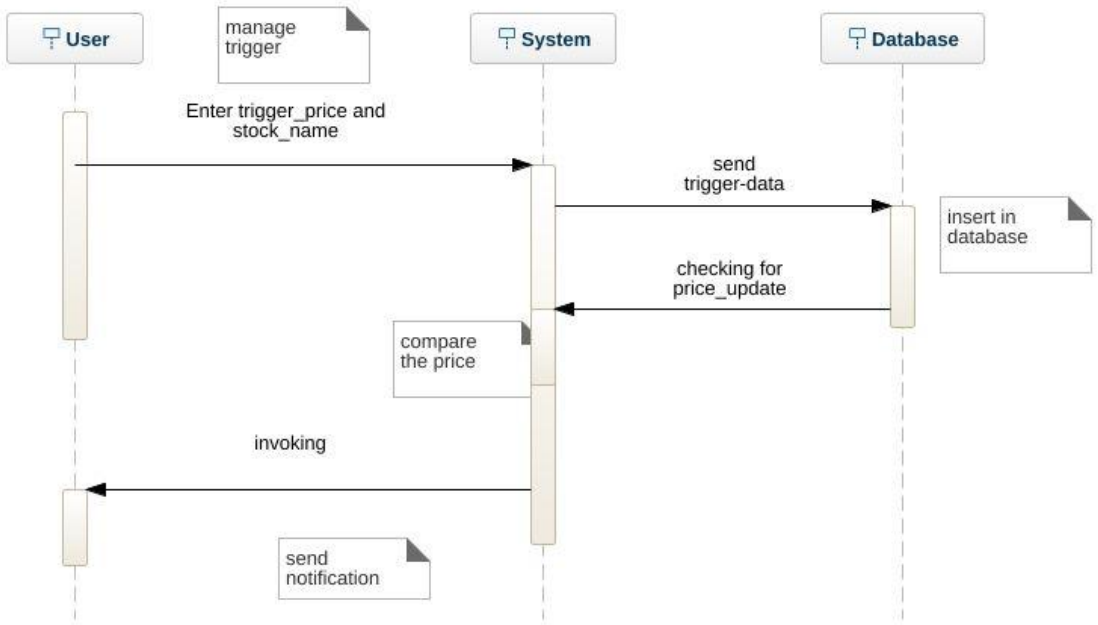
→ Search Stocks :



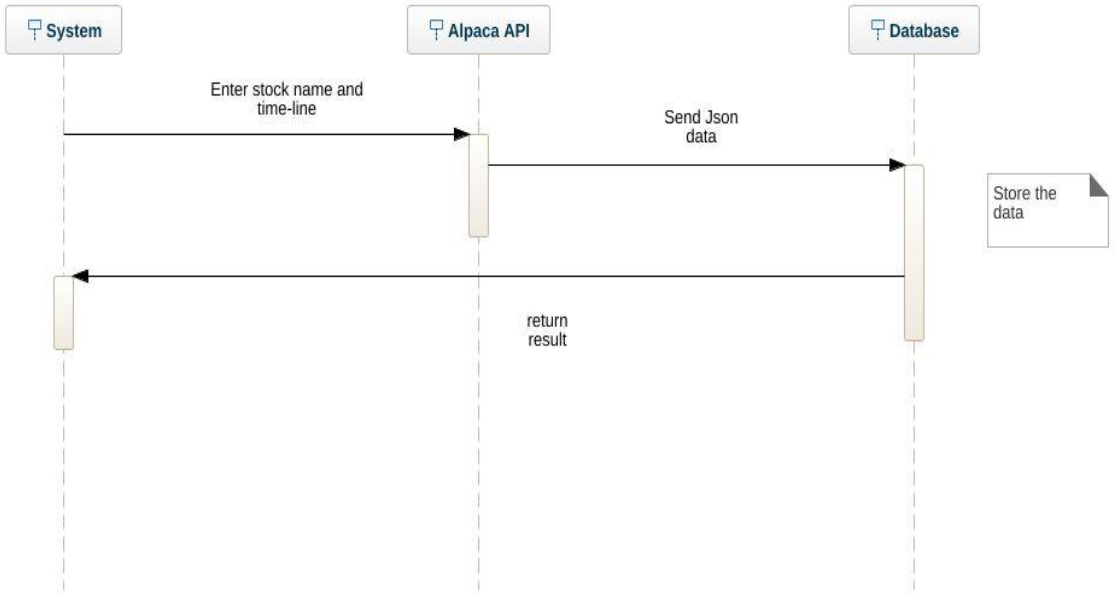
→ Manage Watchlist :



→ Manage Triggers :



→ Manage Stock-Prices :



Data Dictionary :

User

Sr No.	Field Name	Data Types	Width	Required	Unique	Pk/Fk	Referenced Table	Description
1	u_id	number	10		True	Pk		Automatically entered
2	Name	Varchar2	20	True	False			
3	Email	Varchar2	20	True	True			
4	Password	Varchar2	25	True	False			
5	date	Date	-	True	False			Default to current time

Company

Sr No.	Field Name	Data Types	Width	Required	Unique	Pk /Fk	Referenced Table	Description
1	c_id	number	10		True	Pk		Automatically entered
2	Name	Varchar2	20	True	False			
3	Symbol	Varchar2	15	True	True			
4	Lastprice	Varchar2	15	True	False			

Stock_price

Sr No.	Field Name	Data Types	Width	Required	Unique	Pk/Fk	Referenced Table	Description
1	p_id	number	10		True	Pk		Automatically entered
2	Date	Date		True	False			
3	Open	Number		True	False			
4	High	Number		True	False			
5	Low	Number		True	False			
6	Close	Number		True	False			
7	Volume	Number		True	False			
8	Symbol	Varchar2	10	True	True			

Watch-List

Sr No.	Field Name	Data Types	Width	Required	Unique	Pk/Fk	Referenced Table	Description
1	w_id	Number	10		True	Pk		Automatically entered
2	Symbols	[Varchar2]	20	True	False			Default is []
3	u_id	varchar2	10	True	True	Fk	User	

Trigger

Sr No.	Field Name	Data Types	Width	Required	Unique	Pk/Fk	Referenced Table	Description
1	t_id	Number	10		True	Pk		Automatically entered
2	Symbol	Varchar2	20	True	False			
3	Name	Varchar2	20	False	False			
4	strikeupperprice	Number	10	True	False			
5	strikelowerprice	Number	10	True	False			
6	email	Varchar2	20	True	False			
7	date	Date	-	True	False			Default to current time
8	u_id	varchar2	10	True	True	Fk	User	

StockInfo

Sr No.	Field Name	Data Types	Width	Required	Unique	Pk/Fk	Referenced Table	Description
1	si_id	Number	10		True	Pk		Automatically entered
2	logo	Varchar2	20	True	False			
3	listdate	Varchar2	20	True	False			
4	country	Varchar2	10	True	False			
5	industry	Varchar2	10	True	False			
6	sector	Varchar2	20	True	False			
7	market cap	Number	10	True	False			
8	employees	Number	10	True	Fase			
9	phone	Number	15	True	False			
10	ceo	Varchar2	20	True	False			
11	url	Varchar2	50	True	False			

11	description	Varchar2	100	True	False			
12	exchange	Varchar2	10	True	False			
13	name	Varchar2	15	True	False			
14	symbol	Varchar2	10	True	True			
15	hq_address	Varchar2	50	True	False			
16	hq_state	Varchar2	15	True	False			
17	tags	[Varchar2]	-	True	False			
18	similar	[Varchar2]	-	True	False			

Implementation Details

→ we have implemented a system in the MERN stack. We have used a highchart library for showing data. used Material-UI library for designing.

→ we used Alpaca api for fetching all price data and Polygon.io api for fetching Fundamental company data.

→ we have used JWT authentication for registering and login users. in that we send email & password details to the backend. so it generates the token and that token sends it to the frontend and saves it to localstorage of the browser. when we do another request then we put that token into the header of every request which is sent to the backend. so, the backend verifies it and receives the userid from that token.

→ we have a select-option bar in which users can select the stocks and it to the watchlist (limit is maximum 5 stocks).

→ in the watchlist we implemented the live price of the stocks that are fetched using api in every 3 seconds.

→ stock charts show OHLC data from 1 jan,2018 to up-to date.

→ here we have used US stock market data. so the market opening time of the US market in India is 7:00 pm to 1:30 am. so, at that time we can see a live chart (Candlestick bar chart) which is updated every 1 minute.

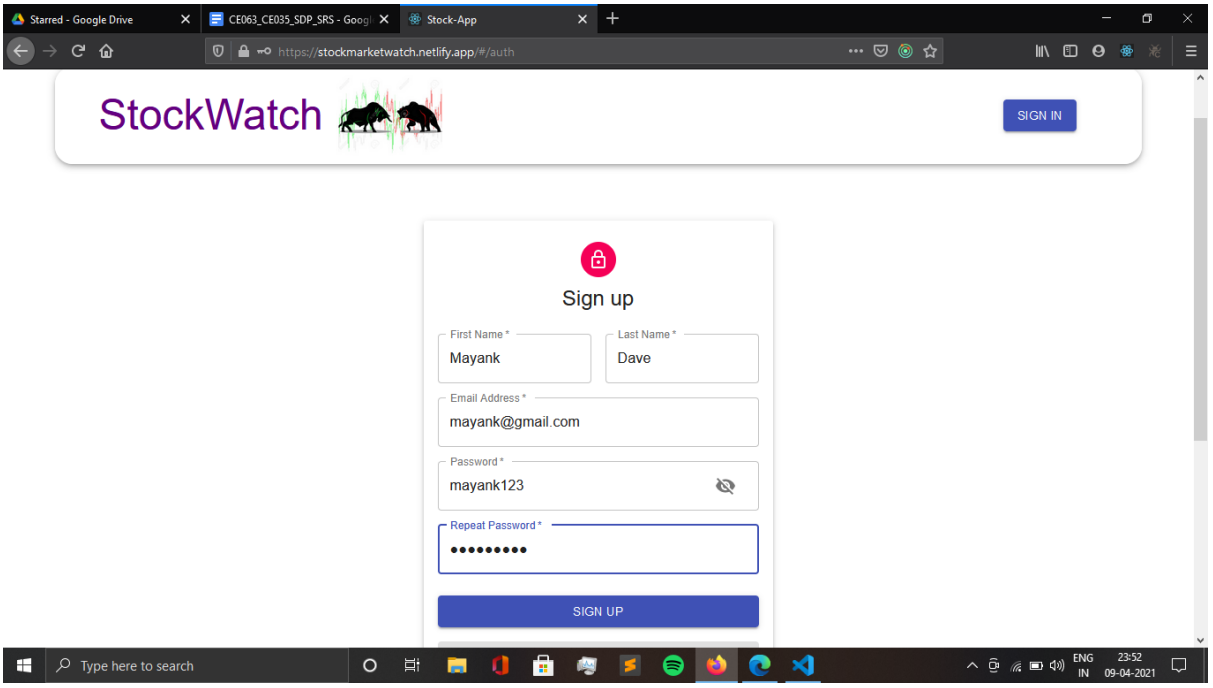
→ we use a highcharts library which provides a wide range of charting tools. like, we can view full-screen, export chart data as image file and also .csv file, also (line,candlestick,ohlc) chart, we can set the flag & labels, draw the resistance and support line, many indicators we can apply and do technical analysis. also select the (range) 1m,3m,1y,3y data for analysis.

→ we can add the trigger (providing upper and lower limit). in that when current live price is going outside of this range then we mail the user about the trigger executed. no limit in adding no. of triggers.

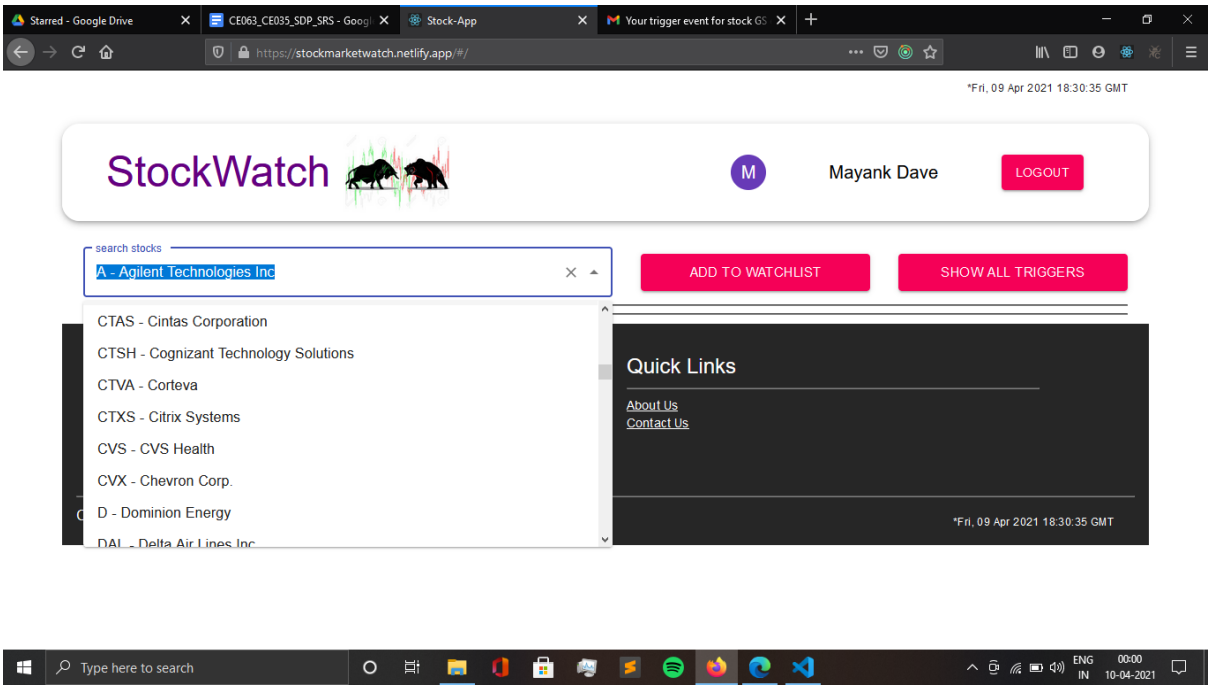
→ we have provided only 500 stock details which are in the list of S&P 500 index. fetched data is combined from many exchanges. Clock is present in our web-app showing UTC (Universal Time) time format.

Screen-shots of the System

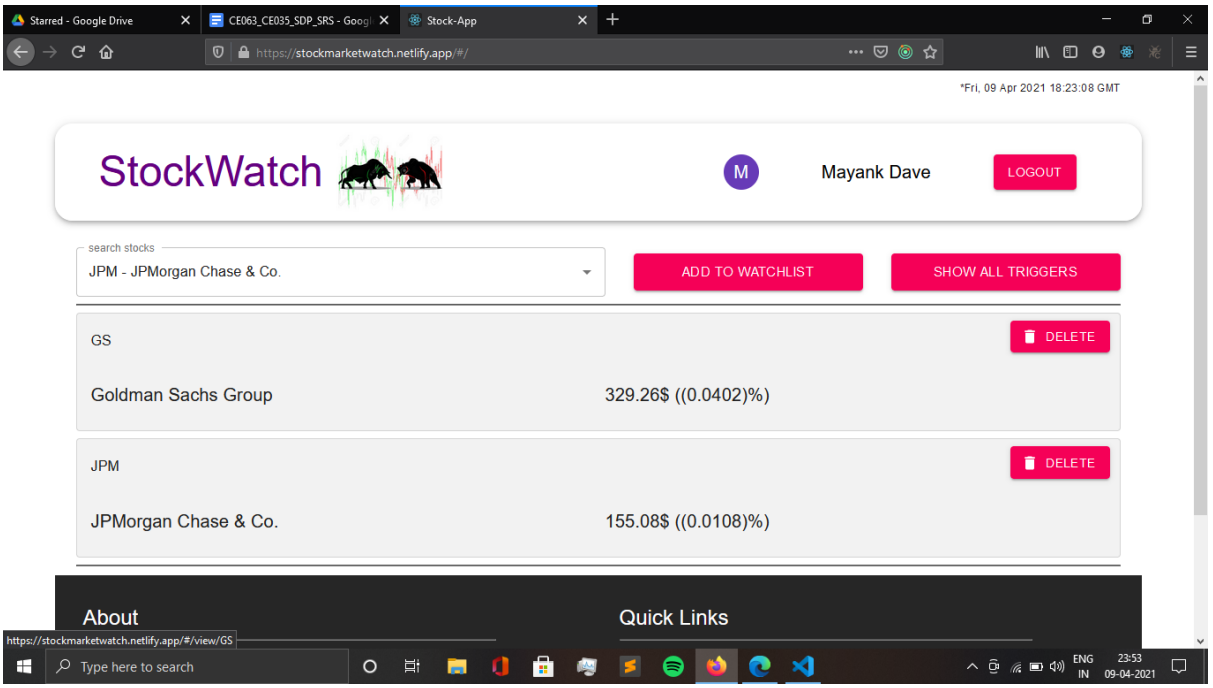
→ SignUP page



→ home page and browse stock list



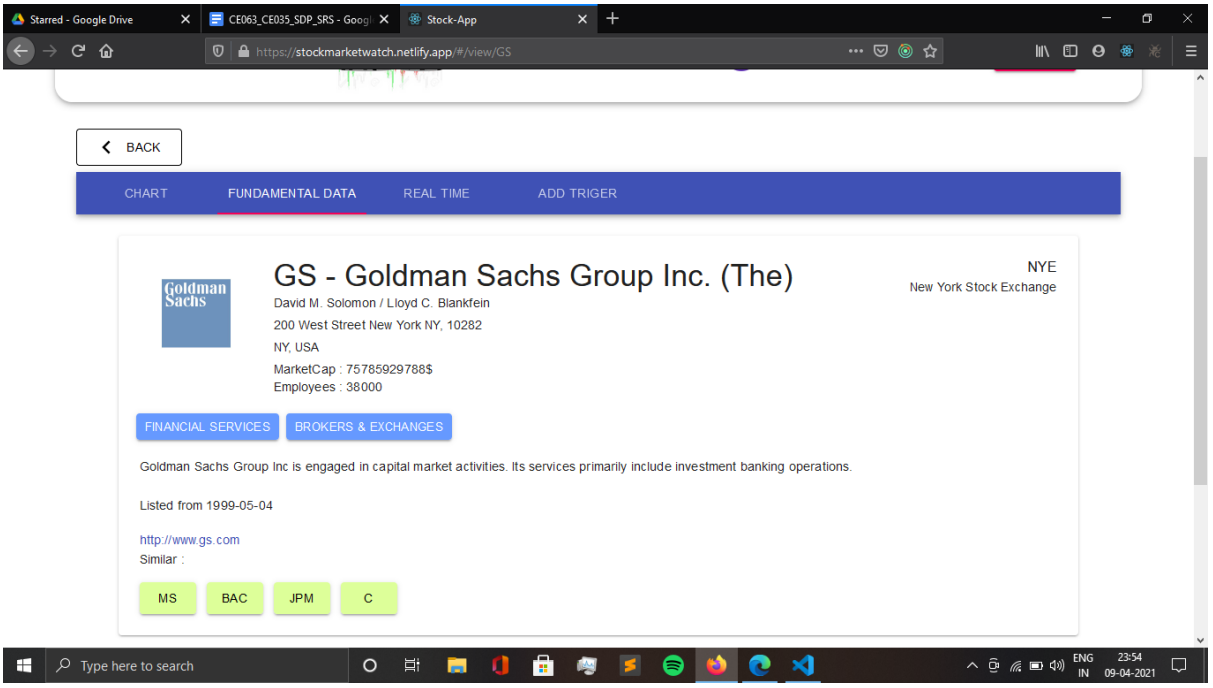
→ 2 stock are added to watchlist



→ historical chart with stock-tools

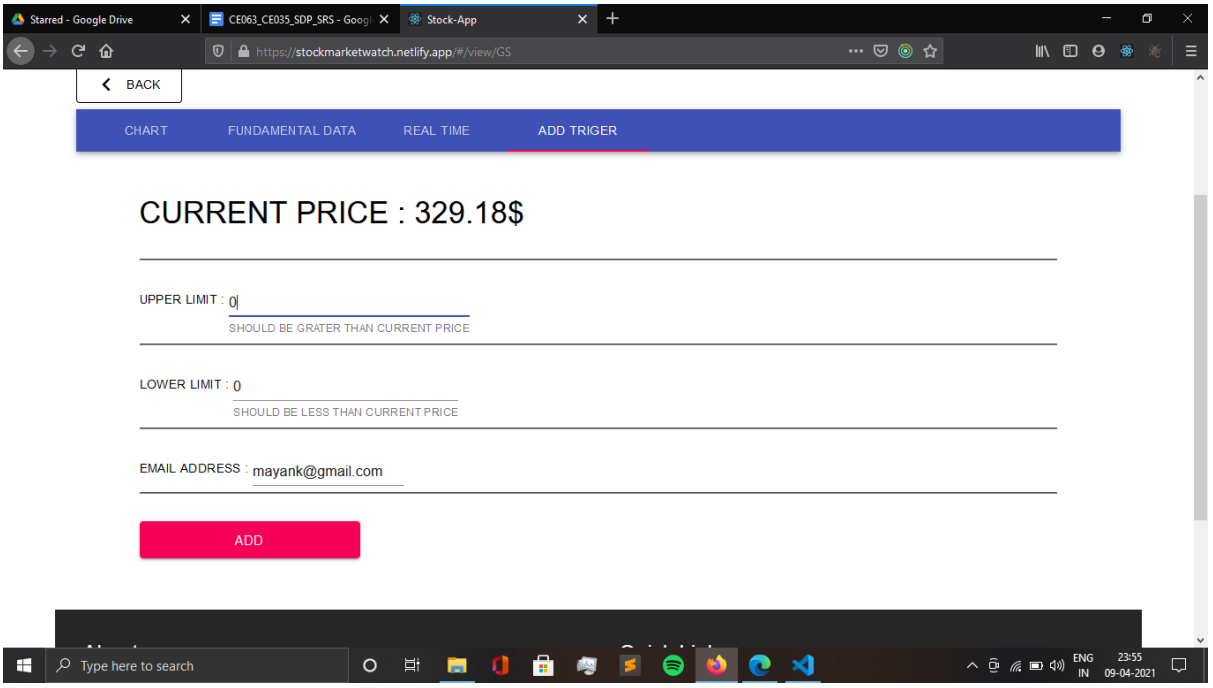
→ Line chart (can change it to OHLC and candlestick chart in tools)

→ Fundamental data

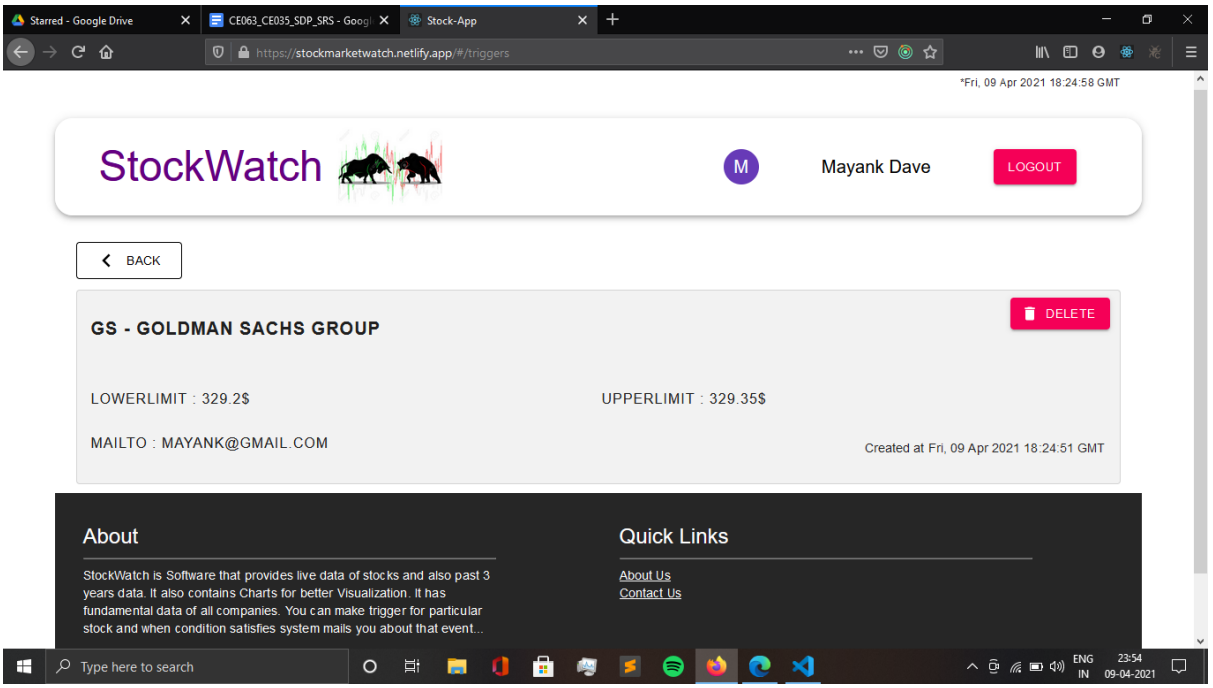


→ Live price chart of stock

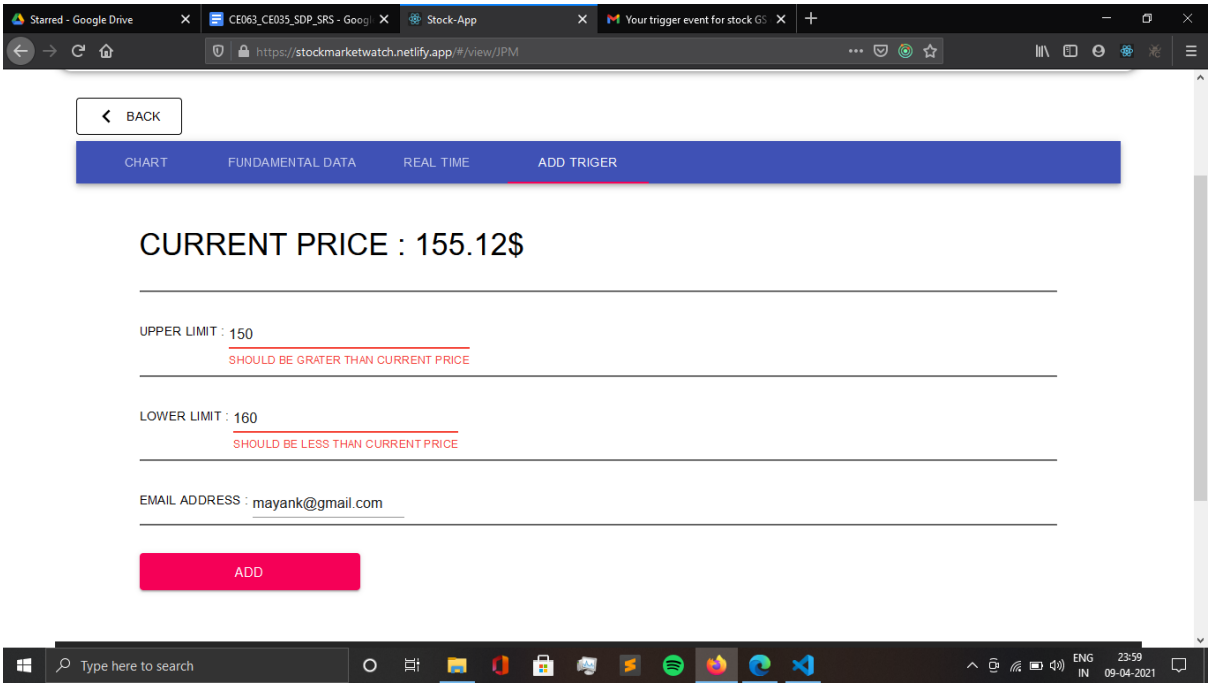
→ create Trigger page :



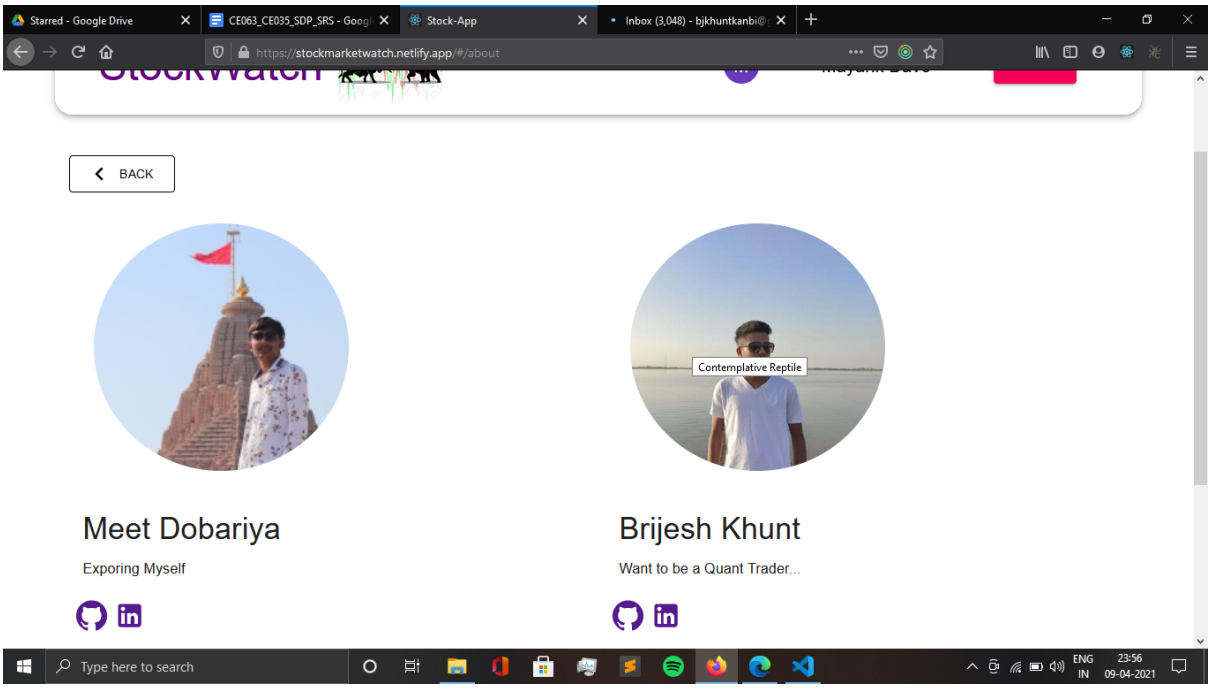
→ All triggers we can see it here



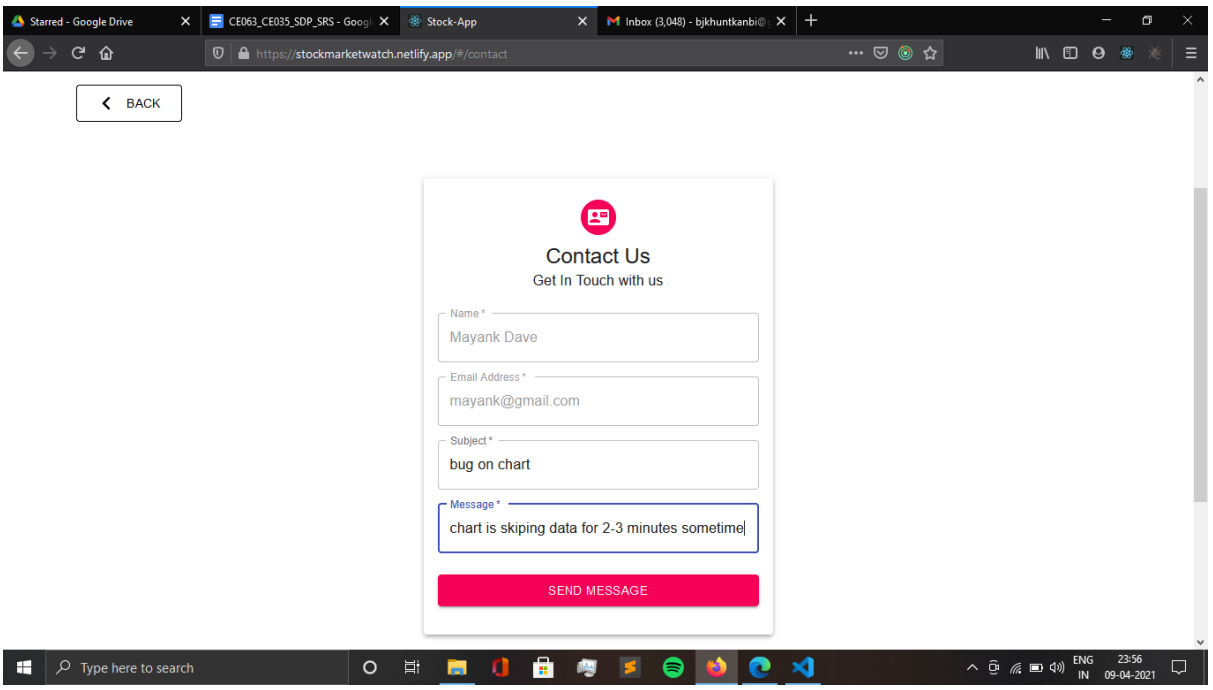
→ while putting we can see errors if we unfollow the rules



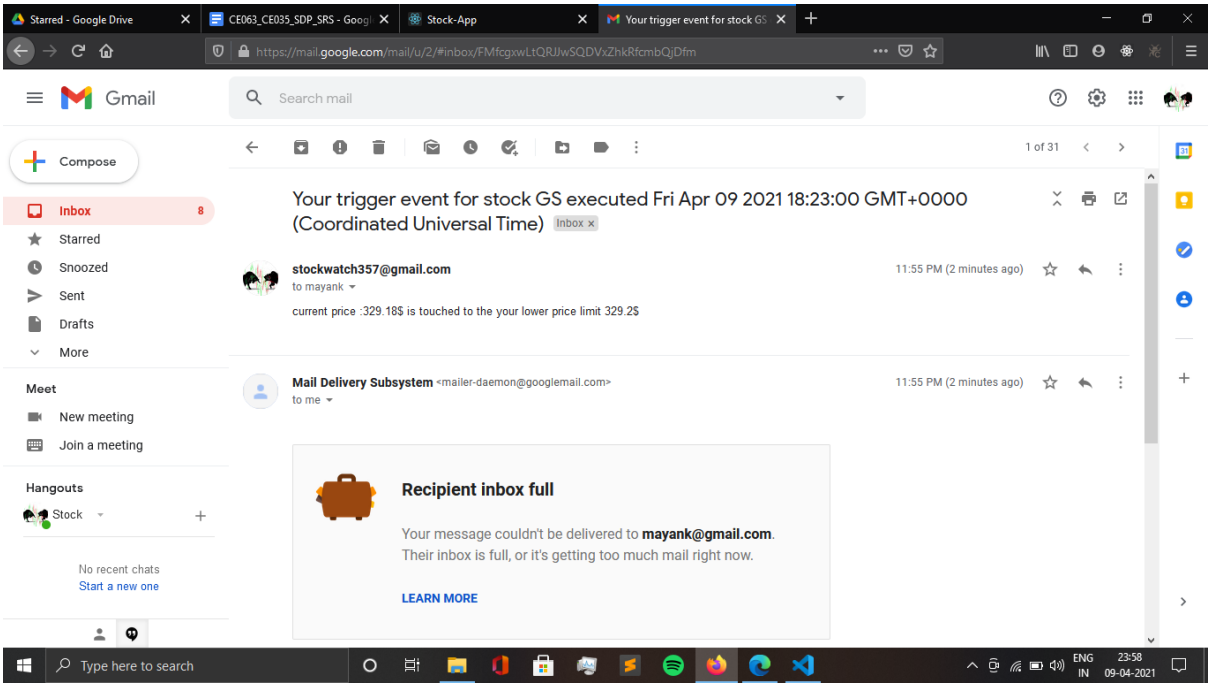
→ About page (Admin Site Section)



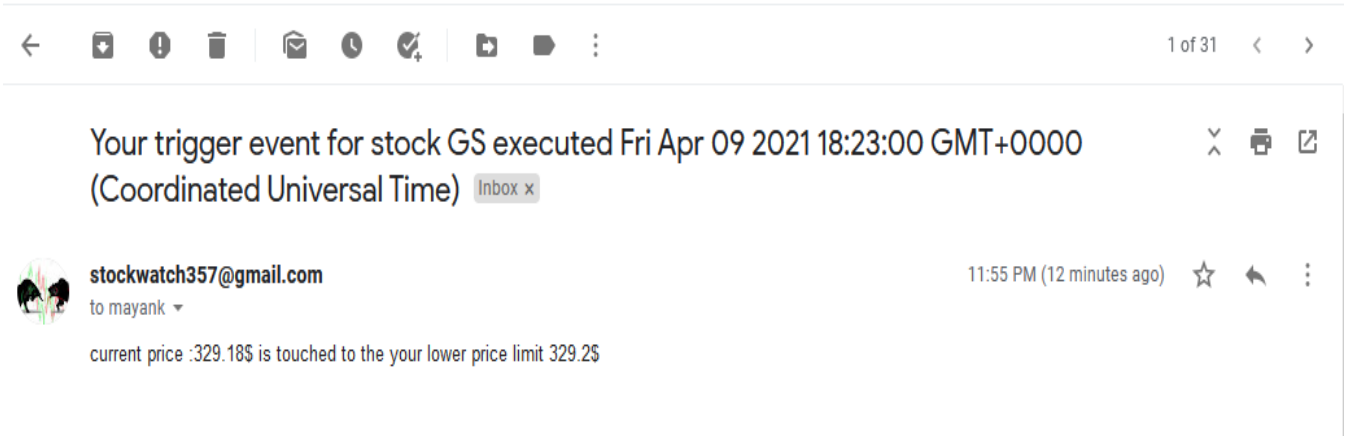
→ Contact page (query or bug related to site) (contact admin)



→ Mail which we sent to end user about triigger



→ Format of the mail



Conclusion

→ All the main functionality of the system we have completely implemented.

→ we learn about the user registration and login and more about how JWT works and use it in Authentication and Authorization.

→ learn about how to get data from api and store it in a database and convert json objects to a good readable or maintainable format for our mongoose models. even if we have a rate limit in api calls for fetching data and manages that using javascript setInterval method.

→ learn about highchart library and its options for showing data in charts.

→ learn about cron-job which is used for cleaning and many more…

→ so, we make 4-5 jobs. Which is a script related to deleting the liveprice table everyday before market time starts. one job for fetching prices (for 500 stocks) every minute of trading sessions. one job for checking every minute if triggers is reached or executed for every entry in the triggers table. one job for fetching 1D ohlc data for every stock at the end of day and it to the historical price table.

→Our Apps are very useful for trading quants who want to make an instant move to change their position in the market when the trigger is executed. and also users can also apply many indicators and apply technical analysis using our advanced chart and stock tools.

→ we have deployed the backend in heroku and it is up and running. frontend is deployed in netlify.

→ our Web-app url :

https://stockmarketwatch.netlify.app

Limitations and Future Enhancement

→ apps are limited to only a few stocks (500). we can make it adding more stocks in this web-app.

→ we can also make comparison charts between 2 to 3 stocks.

→ We can extend our market to indian exchanges also (NSE,BSE).

→ we can add the latest news and info about the company. so our users can also catch it and react to the market with that environment.

→ we can improve our triggering system by increasing computational power.

→ we can improve the notification system which is currently only using mail.

→ also adding upcoming IPO details and company related details (balance sheet, DRHP) and for ipo subscription status…

→In the future, we can also update the clock based on the user's local region timing.

Reference / Bibliography

https://www.youtube.com/

https://www.google.com/

https://stackoverflow.com/

https://alpaca.markets/docs/

https://polygon.io/docs/getting-started

https://www.nasdaq.com/

https://docs.mongodb.com/

https://reactjs.org/tutorial/tutorial.html

https://material-ui.com/

https://console.cron-job.org