MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

MINI PROJECT REPORT - Web Technologies Lab (DSE 3163)

Department of Data Science & Computer Applications

STOCKGENIX

B. Tech Data Science

5th Semester – Batch: B4

Submitted in fulfilment of the requirements of DSE 3163 Web Technologies Lab by

Authored By:

DHRUV GANERIWAL	200968208
DISHITA MIDHA	200968196
CHINMAYI SAHAI	200968144
JANVI JOLLY	200968190
SAATVIK MAHESHWARI	200968142
SIDDHANTH SHIVANAND	200968152

Mentored By:

Mr. Tojo Thomas
Assistant Professor - Senior Scale
DSCA, MIT

Mr. Akshay Bhat
Assistant Professor - Senior Scale
DSCA, MIT

DATE : 04/11/2022

CERTIFICATE

This is to certify that Dhruv Ganeriwal (200968208), Dishita Midha (200968196), Chinmayi Sahai (200968144), Siddhanth Shivanand(200968152), Saatvik Maheshwari(200968250) and Janvi Jolly (200968190) have successfully compiled a mini project titled "STOCKGENIX" rightly bringing forth the competencies and skill sets they gained during the course - Web Technologies Lab (DSE 3163) and thereby resulting in the culmination of this project.

Mr. Akshay Bhat
Assistant Professor - Senior Scale
DSCA, MIT

Mr. Tojo Thomas
Assistant Professor - Senior Scale
DSCA, MIT

CONTENTS

1. Introduction

We all, as a team from the Data Science Branch, are keenly interested in learning about stock trading and finance of the markets outside India. We have created this paper trading website/ stock trading simulator to ease the learning of students about stock trading and finance overall.

1.1 Purpose

The goal of this web app is to allow users to learn about investing in a fun and risk-free manner. Users can access stock data and charts and use the app to buy and sell stocks using their $100k budget to simulate the investing process simplistically.

It consists of a dynamic ReactJS frontend using Material UI, React Routing and Hooks. The backend API uses NodesJS and ExpressJS to perform user authentication and user and stock information retrieval. Furthermore, the data is stored in a MongoDB database.

1.2 Intended Audience

The target audience for this web application are rookie traders who wish to learn how to trade and analyse the fluctuations in stock prices because of market forces. Rookie traders include students who have some background when it comes to finance.

2. Motivation

As undergraduate students, fresh into the professional world, we find it difficult to navigate between classes, projects, internships and other responsibilities that come with being an independent individual. Furthermore, we are often overwhelmed by these numerous responsibilities which we now shoulder and even end up forgetting about some activities until the deadlines have long passed.

As a result, maintaining, balancing and organising our workloads and scheduling our activities for each passing day has become a task of utmost importance.

We would like to express our gratitude and thanks to our advisors Mr. Tojo Thomas and Mr. Akshay Bhat who have been tremendous mentors through valuable project-inputs and motivation to achieve more !

3. Objectives

- To make users improve their knowledge of stock trading.
- To make rookie traders improve their buying and selling techniques.
- To make users learn to read stock charts and learn more on time series data.
- To allow users to start trading stocks in real life and with real money.

4. Role Distribution

Teamwork is one of the most important factors in the success of any project. Success in a team project requires proper management of roles and distribution of work while keeping the strengths of the individual and the overall objectives in mind, regardless of how efficient the members are as individuals.

- **Dhruv Ganeriwal (200968208) -** Back-End Integration, Documentation,API
- **Janvi Jolly (200968190) -** Frontend Design
- **Chinmayi Sahai (200968144) -** Frontend Design.
- **Dishita Midha (200968196) -** Backend Development, API and Database Design
- **Saatvik Maheshwari(200968142)-** Document Design
- **Siddhanth Shivanand(200968152)-** Design (UI/UX)

5. Working Methodology

Given our role distribution, all the members cannot work in parallel at a time. We can however divide the project into parallel pieces of work. The backend and the Integration members can work together while the frontend and the design members can begin working separately. Following this, the Integration and frontend members will work together to connect the backend and front end.

These subunits on the technical end of this project work on a Waterfall Development model. The documentation member works on the documentation **concurrently** while working on the integration while maintaining contact with the frontend and design members.

In the waterfall development model for software development, a particular member is unable to complete all of his work until another part of the project has been completed by the other member who has been assigned the role. The front end member is dependent on the design member to start his work. Similarly, the database design, creation and connection with the backend must be completed prior to API development.

Once the frontend and backend is completed, work must be done towards integrating both these areas of the project.

Content is added to the database once database design and backend is done followed by documenting the entire project.

This is the working methodology used for this project.

6. Software Requirements

Technology stack used:

1. **NodeJs -** NodeJS is a Javascript runtime that allows Javascript to be run outside of the browser. It acts as a foundation for the website's development.It also manages all of the modules that we install in order to use other web frameworks in this project .On the local system, it can be regarded as a stage for running the backend and frontend on separate servers ports using nodemon and npm, respectively. Both npm and nodemon actively monitor code changes and incorporate them as soon as possible files are saved without the need to restart the development servers NodeJS serves as the runtime.

2. **Mongoose -** Mongoose is a JavaScript object-oriented programming library that creates a connection between MongoDB and the Node.js JavaScript runtime environment.

3. **React-Js -** React is a frontend web framework in which UI design is represented by React Components. React components can also be defined by the user. HTML, CSS, and JS can all be used to manipulate React. Meta is in charge of this package. This Aproject will be assembled as a react app, then the backend code and frontend modifications will be added to this created react app.

4. **Express -** Express is a backend framework that helps with API development.We will create routes for POST and GET requests that

must be made in order to get data from the database to the frontend. We will also have routes to write to the database, such as when a user adds an event or adds a comment to any current event.

5. **MongoDB-** MongoDB is an unstructured data database that stores data in the form of documents .Each collection record is saved in the database format to interact. We will use the mongoose node module with an online database, so that data is centrally hosted on the cloud.

6. **MongoDB Atlas -** MongoBD Atlas is an online service that allows data to be stored in the cloud rather than on the local system. This allows all members to access the exact same copy of the database on their local systems while the website is being developed, as well as ensures that changes are reflected. We use Atlas' free tier option to create a cluster for all of this project's databases. This cluster will be hosted on a Google Cloud Platform (GCP) shared instance.

7. **Figma -** Figma is a free UI/UX design platform with separate subscription tiers. It is well-known for having a lower learning curve than many alternatives. All design work was done in Figma.

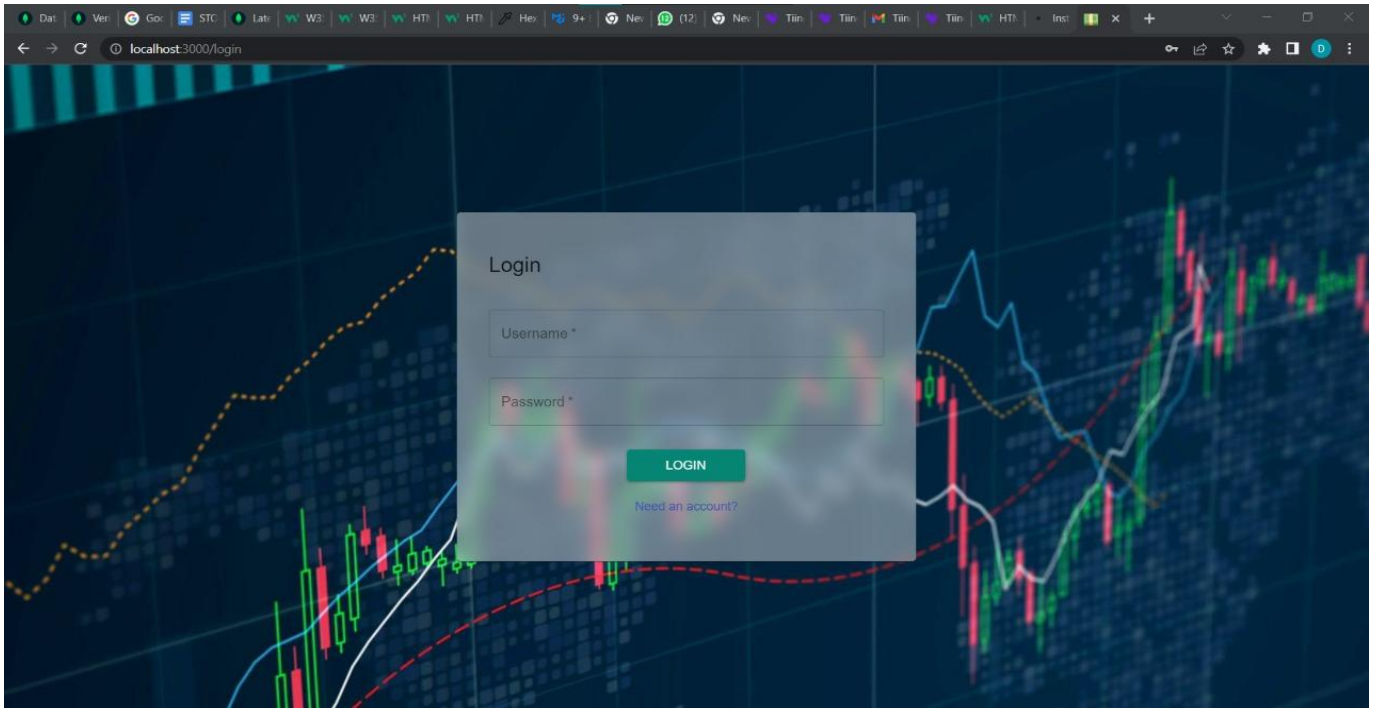
8. **Redux -** Redux is a state container for JavaScript applications. It is a JavaScript open-source library for managing and centralising application state. It is most commonly used for building user interfaces with libraries such as React or Angular. It assists you in developing applications that behave consistently, run in a variety of environments (client, server, and native), and are simple to test. It can be used in

conjunction with React or with any other view library. It is small (2kB including dependencies), but it has a large ecosystem of addons.

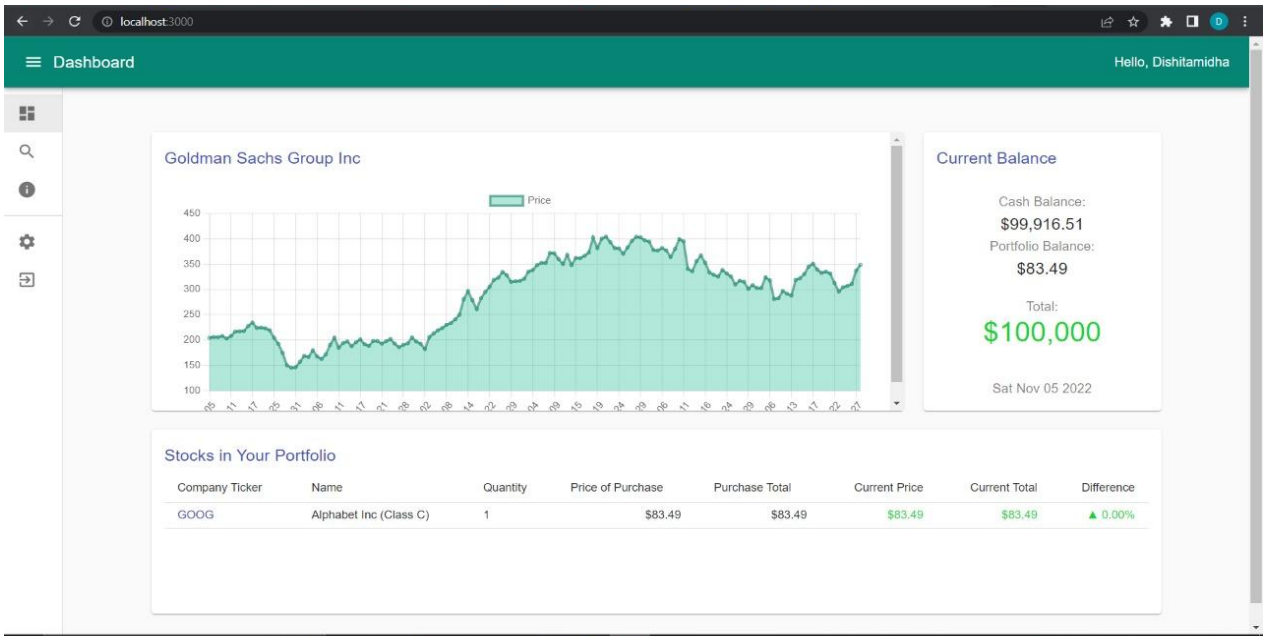
7. UI/UX Design

For the project we have chosen the calmer colours of blue to help the user relax when in front of the scheduler. This is to make sure that the user at no point feels overwhelmed due to the number of events/activities scheduled. It will allow the user to be in a calmer state of mind while making decisions.

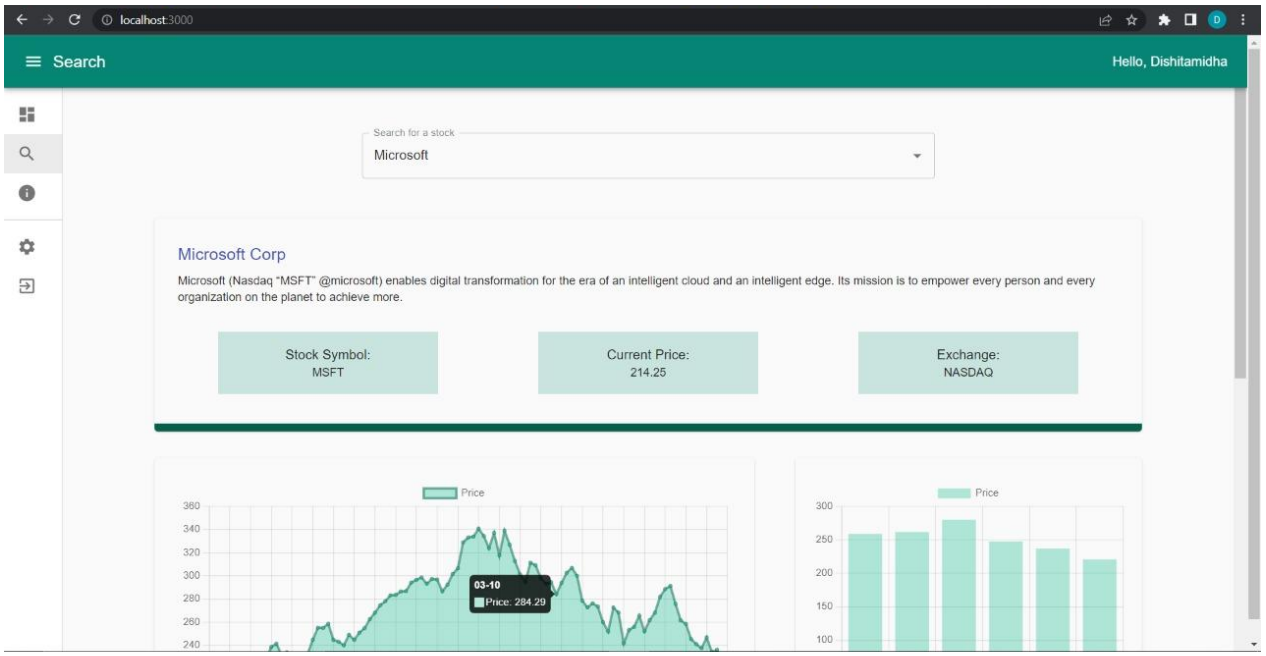
7.1 Landing Page/Login Page:

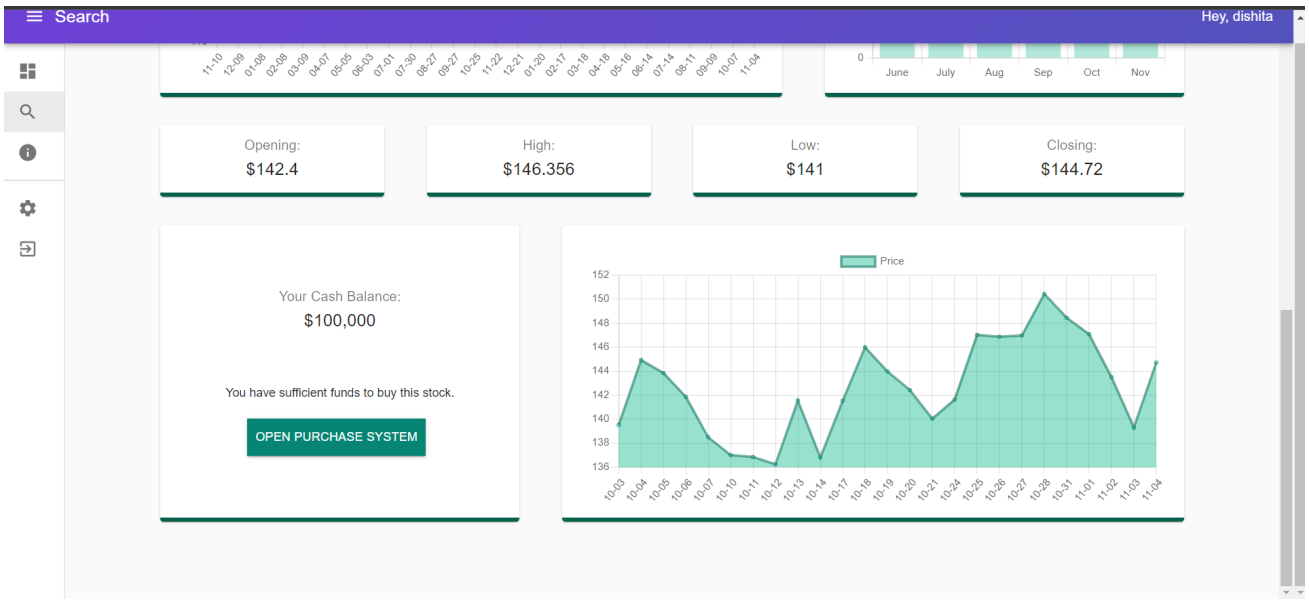


7.2 Dashboard

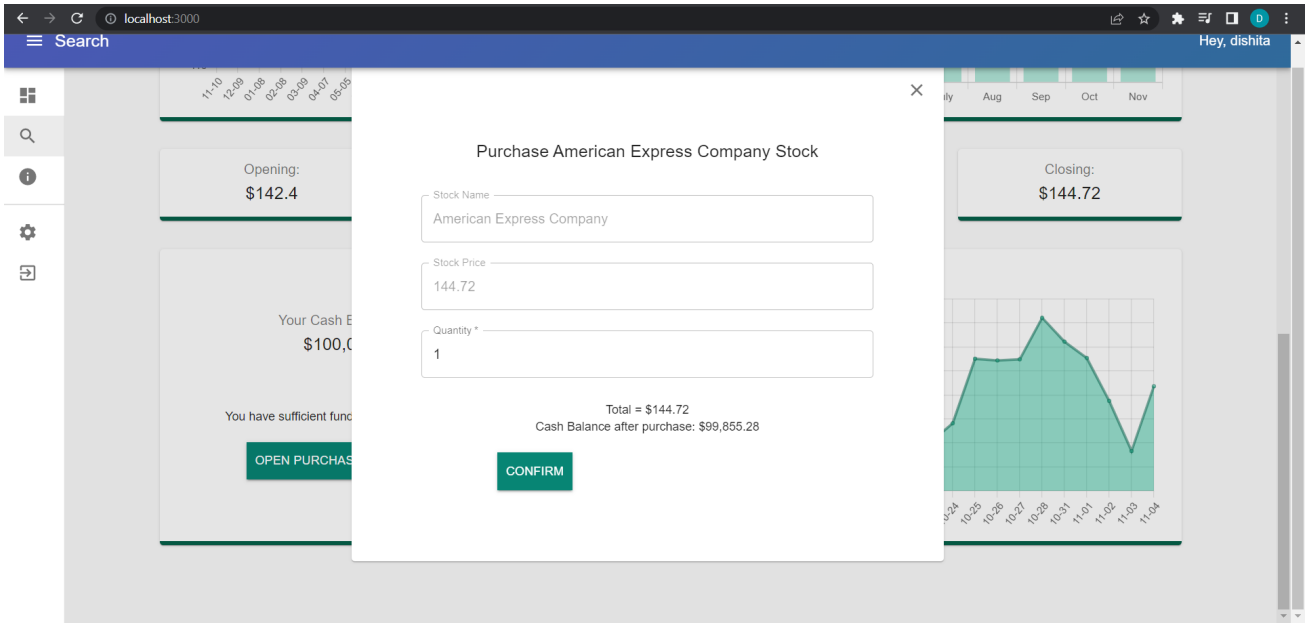


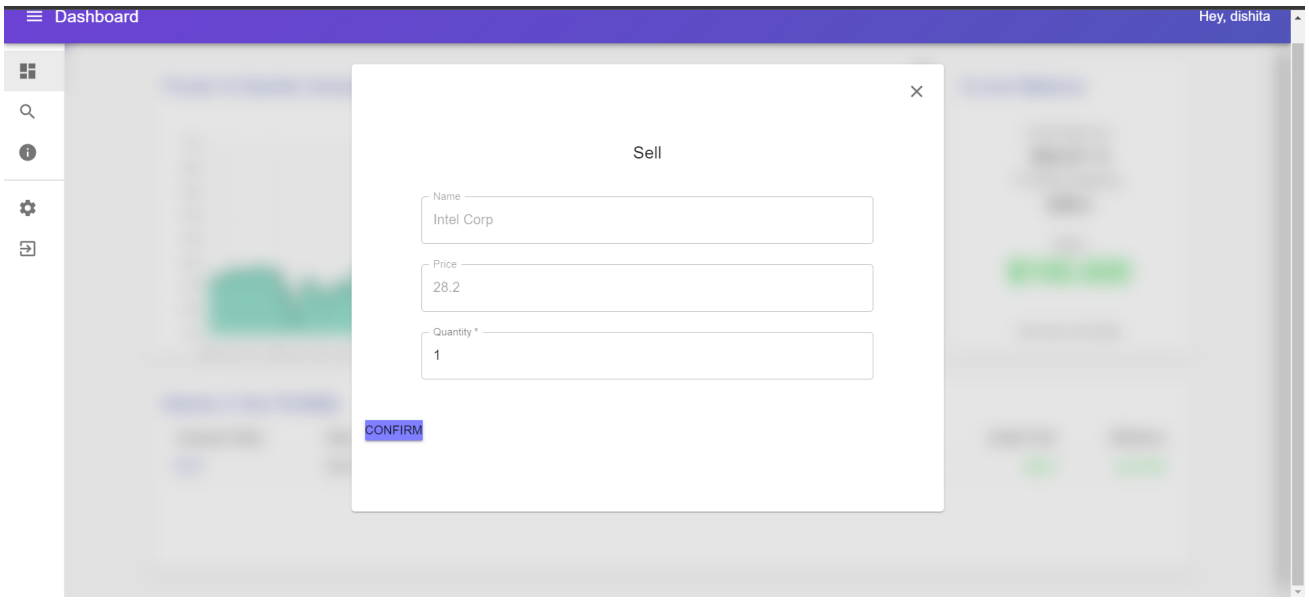
7.3 Stock Information:





7.4 Stock Purchase:





8. Frontend

Frontend is built using **React components** in the client folder.
React is a frontend framework that allows for a great deal of development to be done with JS and modified HTML, which react refers to as JSX. CSS can be used in the same way that it would in any other HTML + CSS page.
A react component is a reusable piece of code that defines a visual component on the front end. React renders these components so that they can be seen by the user.
Given the dynamic nature of fluctuations in stock prices and that they are time-dependent, a template-based mechanism is required for this project. All of these templates can be transformed into React components.
Only native HTML, CSS, and JS can be handled by any browser. All React-related files and components are translated to native HTML, CSS, and JS before being rendered on the client side.Integration is done with the backend using node.js.
We have defined routers based on user privileges. The login page is a public route accessible to everyone while the stock exchange page is attached to a private router. These connect to the back end with the various get and post

methods defined to authenticate entered information and retrieve event information.
When the designs are finished, the frontend work begins. While getting the exact look as in the designs is nearly impossible, an attempt is made to get a very close representation of the design creativity.

9. Backend

The backend mainly consists of the **server folder** built for combining all the backend entities.
We have designed all the routes using the **Express framework**.
The APIs in React all use **HTTP methods based on the REST** (Representational State Transfer) architecture.
We have used APIs based on the **POST**, **GET** and **DELETE** methods in our project.
There are 4 events for which we need routes -

- **Authorisation**
- **Stock Data**
- **News**
- **Stock Information**

Routes for **Authorisation(login)**-

- To check if register details entered are correct for first time users
- To check if login-id and password entered are correct for re-entering users

Routes for **Stock Data** -

- Create new event
- Delete existing event
- Edit existing event details

10. Database

We are using MongoDB Atlas to host and store our database. There are only two collections in our database for storing

- Stock Information
- User information.

MongoDB, being a NOSql, stores data in JSON format(key-value) pairs

```
const stockSchema = new Schema({
  userId: {
    type: String,
    required: true,
  },
  ticker: {
    type: String,
    required: true,
  },
  quantity: {
    type: Number,
    required: true,
  },
  price: {
    type: Number,
    required: true,
  },
  date: {
    type: Date,
    default: Date.now(),
  },
});
```

```
{
    username: {
      type: String,
      required: [true, "Username is required."],
      unique: [true, "An account with this username already exists."],
```

```
    minlength: [4, "Username must be 4-15 characters."],
    maxlength: [15, "Username must be 4-15 characters."],
    lowercase: true,
  },
  password: {
    type: String,
    required: [true, "Password is required."],
  },
  balance:{
    type: Number,
    required: true,
    default: 100000
  }
```

```
_id: ObjectId('63651fe4dc38c3ec5b42f22d')
username: "dishitamidha"
password: "$2a$10$rfEaffZlKoJjOvDwVrOMZOReH9vF5PzgpHRJeFV.ivNFV.8HEH1qe"
balance: 99916.51
createdAt: 2022-11-04T14:21:24.813+00:00
updatedAt: 2022-11-04T18:15:09.707+00:00
__v: 0
```

```
  _id: ObjectId('636556ad089a81eb2614315b')
  userId: "63651fe4dc38c3ec5b42f22d"
  ticker: "GOOG"
  quantity: 1
  price: 83.49
  date: 2022-11-04T14:33:12.210+00:00
  __v: 0
```

All administrators for now, have been given read-write-edit permissions

10.1 Tables

User Table

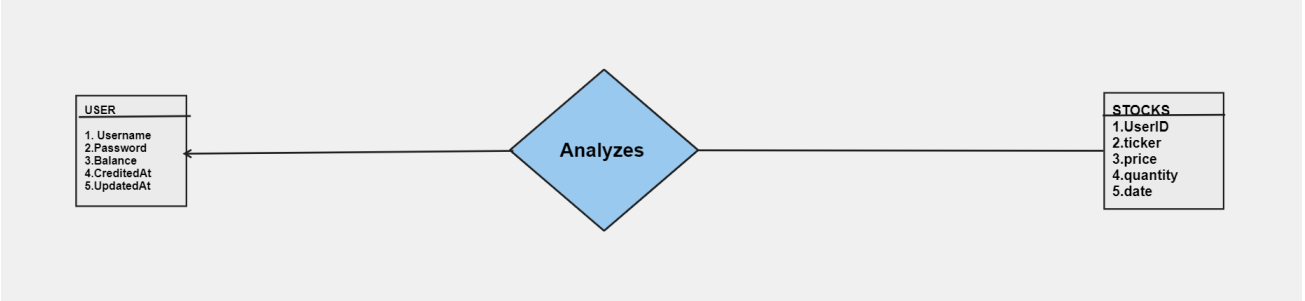
Field Name	Data Type	Format	Description	Example
id	ObjectId	-	Unique object id	-
username	Text	-	Login id	“Dishita Midha”
password	Text	-	Encrypted	“Roar”

			text	“****”
Balance	Float	-	Remaining Amount in wallet	99956.76
Credited At	Date	YYYY-MM-DDThh:mm:ss.sTZD	Time of money credited	“2022-11-01T10: 30:00.173+00:0”
updatedAt	Date	YYYY-MM-DDThh:mm:ss.sTZD	When balance is updated	2022-11-01T10: 30:00.173+00:0

Stocks Table

Field Name	Data Type	Format	Description	Example
id	ObjectId	-	Unique object id	-
userId	Text	-	Identification of each user	17U2ikZPHsGyjKWoqTH
ticker	Text	-	Code word of each stock	“GOOG for google”
Quantity	integer	-	Quantity of stock purchased	-
Price	float	-	Price of each stock	$89.5
Date	date	YYYY-MM-DDThh:mm:ss.sTZD	Date when stock was purchased	2022-11-01T10: 30:00.173+00:0

10.2 Database ER Diagram



11. Conclusion

The project's goal was to create a scheduling stock trading simulator to help a user get a basic understanding of trading without actually taking any risks. The stock is based on real-time value, giving the user the most realistic experience of what it's like to work as a trader. At the end of this project we:

- The website serves as a starting point for anyone who wishes to learn how to invest in the stock market using a straightforward and simplistic model, making it simpler for new traders to develop their purchasing and selling strategies.
- People find it simpler to comprehend equities and make investments in them thanks to the use of charts and statistics, and that’s why our website is very visual and helps us achieve this goal.

12. Scope for Improvements

- Using more number of metrics for better analysis of stocks
- Having a tutorial page to describe day trading
- Including other trading options such as cryptocurrency
- Using another API for fetching stock data because the current one is quite unstable.
- Categorise users and suggest investments on the basis of the category

13. Learning Curve

During the course of this project, we have been introduced to various different website development technologies. At the end of this project, we have learnt,

- Using figma for creating UI design
- Using Lucidchart to create the ER diagrams for a given database.
- Implementing frontend using React components.
- Using Node.js and Express for building the Server / APIs for integration.
- Using Mongodb for database creation and mongodb atlas for hosting the database on cloud.
- Using Redux for state management.

14. Bibliography

- Lucidchart, “Online diagram software & visual solution | Lucidchart,”

  Lucidchart, 2022

  [Last date accessed - 31st October, 2022]

  https://www.lucidchart.com/pages/

- Figma, “Figma: the collaborative interface design tool.,” Figma, 2019.
  https://www.of clashes.
  [Last date accessed - 24th October,2022]

- [Online Video]. Available:
  https://www.youtube.com/watch?v=7CqJlxBYj-M
  [Last date accessed - 27th October,2022]

- Node.js Foundation, “Docs | Node.js,” Node.js, 2019.
  https://nodejs.org/en/docs/
  [Last date accessed - 29th October,2022]

- React, “Getting Started – React,” Reactjs.org, 2019. https://reactjs.org/docs/getting-startefigma.com/ [Last date accessed - 30th October,2022]

- W3Schools, “W3Schools Online Web Tutorials,” W3schools.com, 2019. https://www.w3schools.com/ [Last date accessed - 1st November,2022]

- “DevDocs — Express documentation,” devdocs.io. https://devdocs.io/express/ [Last date accessed - 2nd November,2022]