

CS6019 Capstone Project

STOCK TRADING SIMULATOR



Chun-Hao Hsu

MSD 2023

Table of Contents

1. Introduction	2
Problem	2
2. Background.....	3
Design Choices.....	3
3. Solution Description.....	4
Structure.....	4
Process flow.....	5
Front end	6
Back end (Server)	10
Database.....	11
4. Result.....	12
Successes.....	12
Limitation.....	12
5. Conclusions.....	13
Design Choices.....	13
Future Work.....	13

1. Introduction

Nowadays, we see the impact of inflation on the economy. The government is trying hard to control the inflation with diverse strategies. For ordinary people, inflation gradually erodes the purchasing power of money. Therefore, investing is crucial to protecting and growing our wealth over time.

It is always great to start to learn as early as possible. The fundamental way to accumulate our wealth is by saving. However, the high inflation now is eroding our savings. Investing in stocks becomes another choice since, from the historical data, it is a hedge against inflation. With the advent of online trading platforms, investing in stocks has become more accessible to individual investors, allowing them to participate in the stock market easily.

Problem

Although we know it is better to start earlier, young people seldom have extra money to spend on the investment. The average federal student loan debt is \$37,338 per borrower. A total of 45.3 million borrowers have student loan debt. Therefore, it will be good to have a tool to practice buying and selling stocks without spending real money.

2. Background

For novice investors, using a simulator is a great way to learn about investing. They can learn about basic investment concepts, get used to reading stock tables, understand the impact of market volatility, test trading strategies, and much more.

Design Choices

I decided to build the simulator on the web browser so it is easy to develop with HTML and JavaScript.

For the back-end, in my initial research into which back-end to choose, I prioritized using the same language to cover the full-stack web development project.

Therefore, I use JavaScript with express.js to build up my server and operate in the Node.js environment. Furthermore, it can communicate with MySQL, which I use to store users' transaction data.

Finally, to track everyone's portfolio and let users access the web application and the future management of the website, I decided to deploy the entire web application to the AWS cloud server. I utilized AWS Cognito for user authentication, S3 for setting up my web page, EC3 for running the back-end server, and RDS for storing the database.

3. Solution Description

Structure



Figure 1 Web Structure (locally)

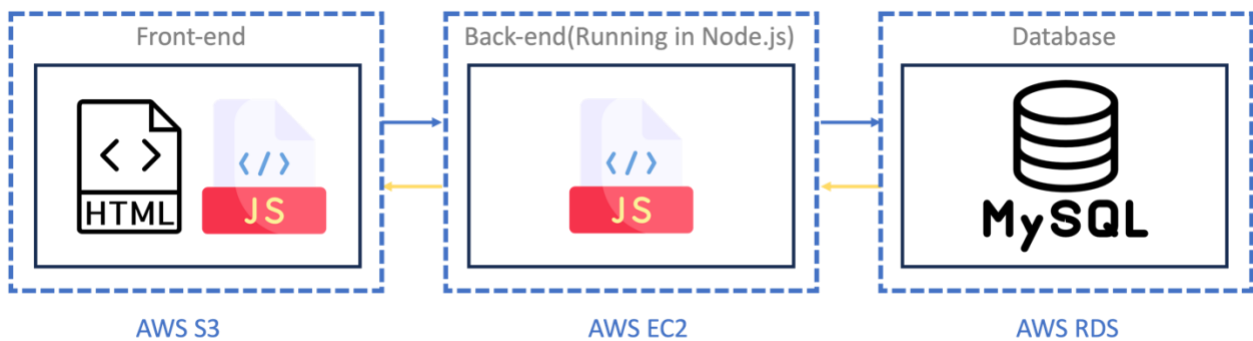


Figure 2 Deployed Structure on AWS

Process flow

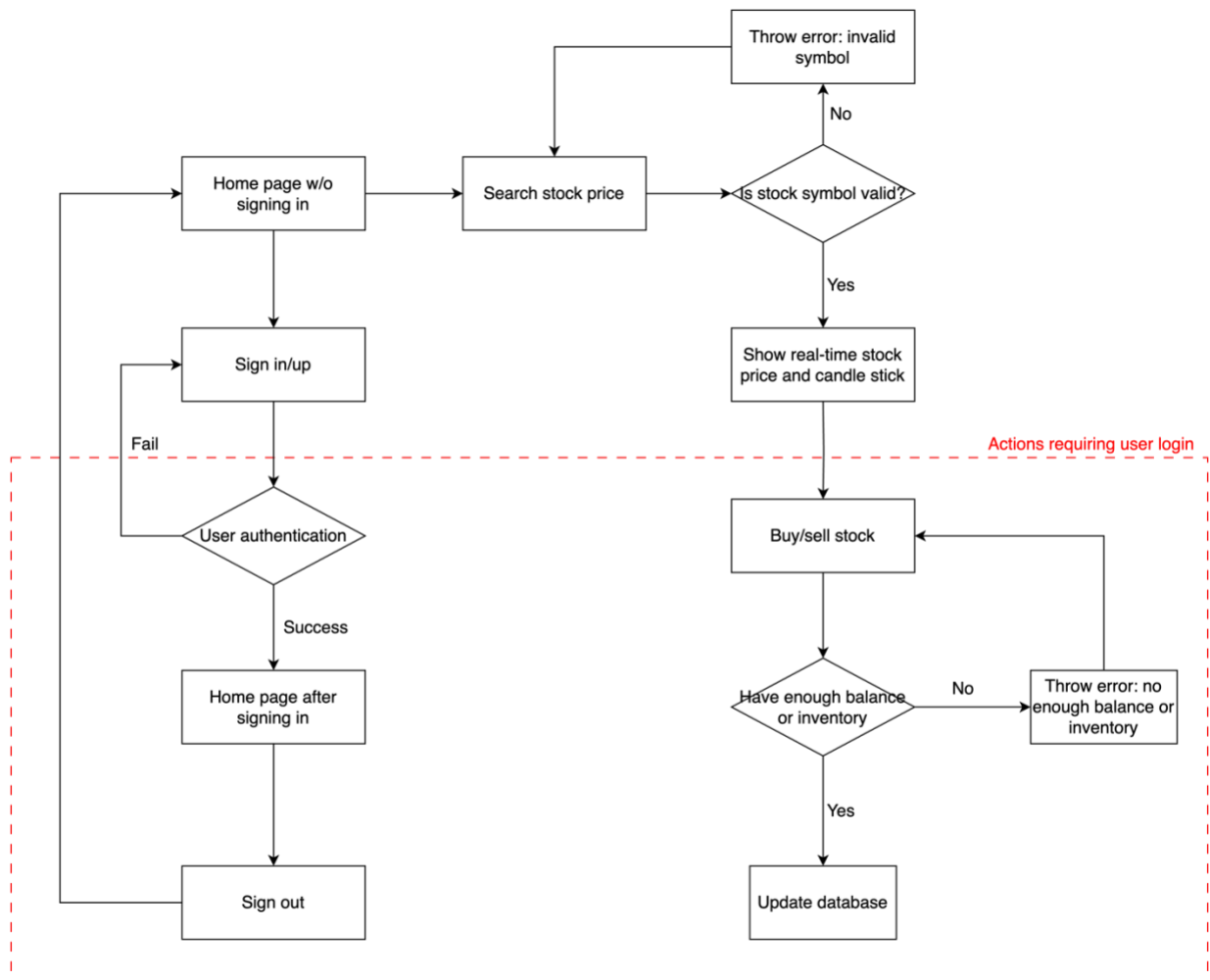


Figure 3 Process flow

Front end

On the home page, before the user signs in, it can only be used to look up the real-time stock price. It will display the current price and the change after the last closing price.

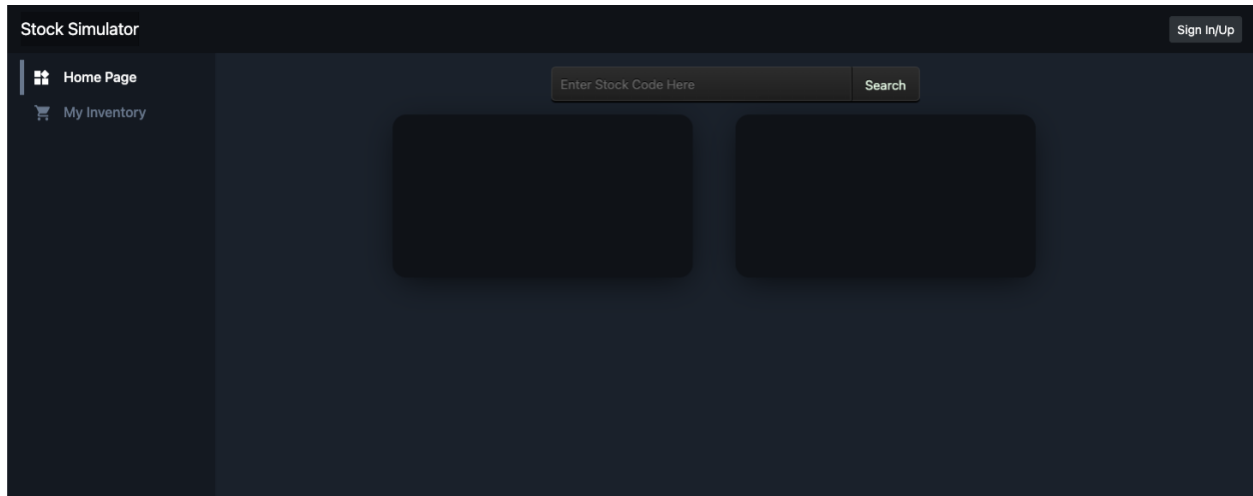


Figure 4 Home Page (without the user logging in)

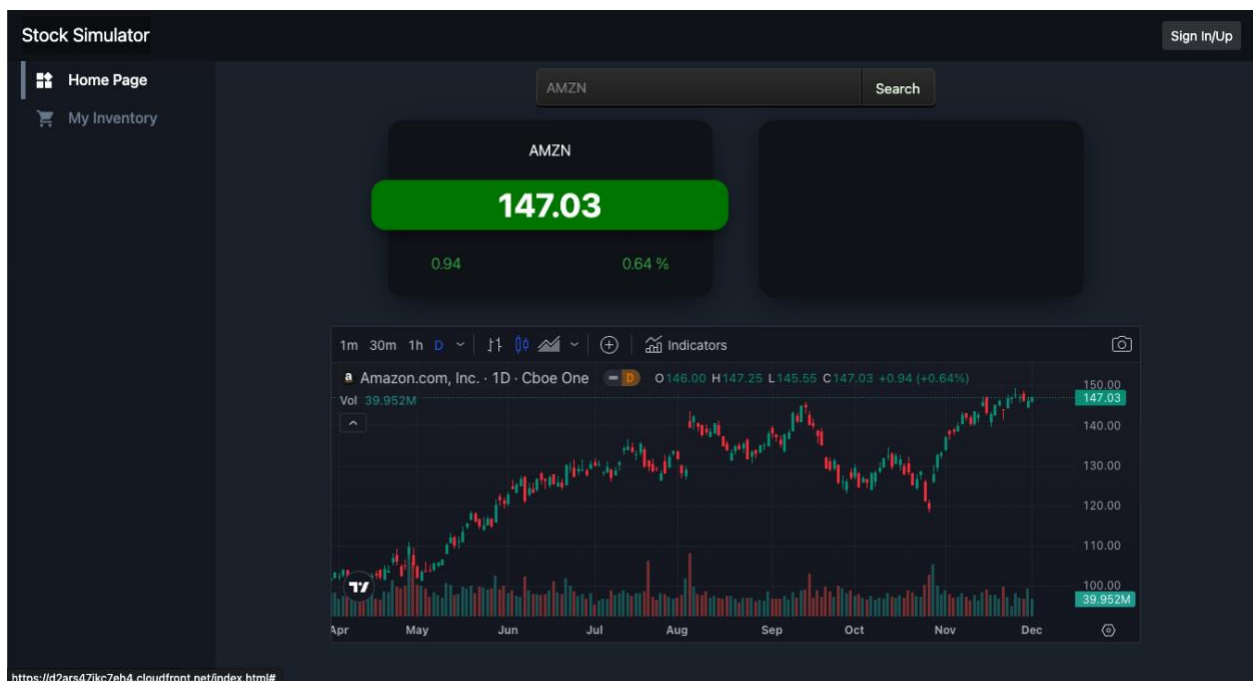
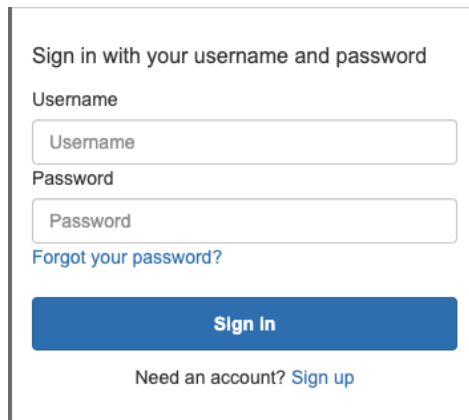


Figure 5 Search current stock price

Once the user logs in, it will show the user's available balance and the buy/sell buttons. Users can buy or sell the select symbol by querying the stock information. Users can buy with enough available balance and sell with enough inventory.



Sign in with your username and password

Username

Password

[Forgot your password?](#)

Sign In

Need an account? [Sign up](#)

Figure 6 Sign-in/up UI (AWS Cognito default UI)

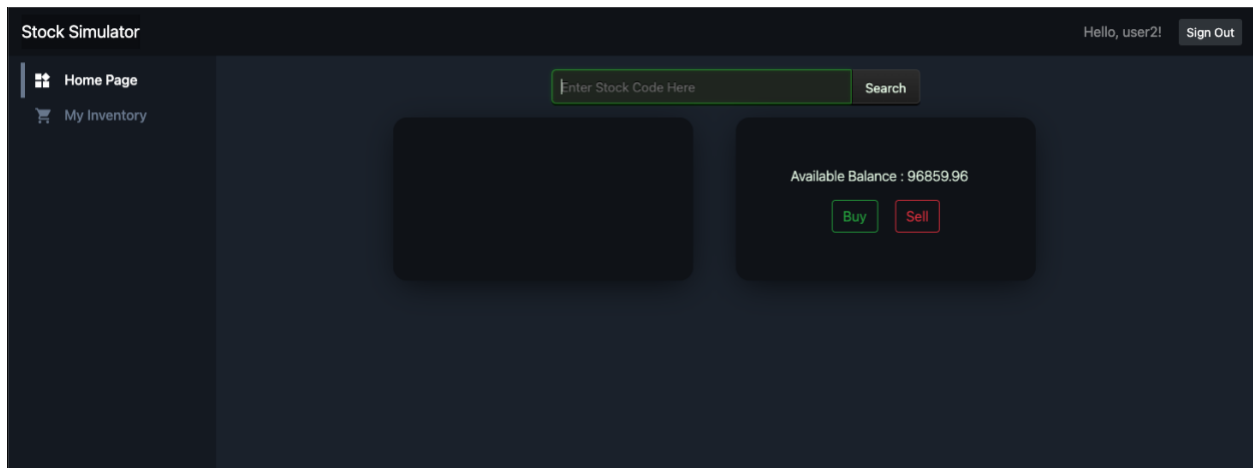
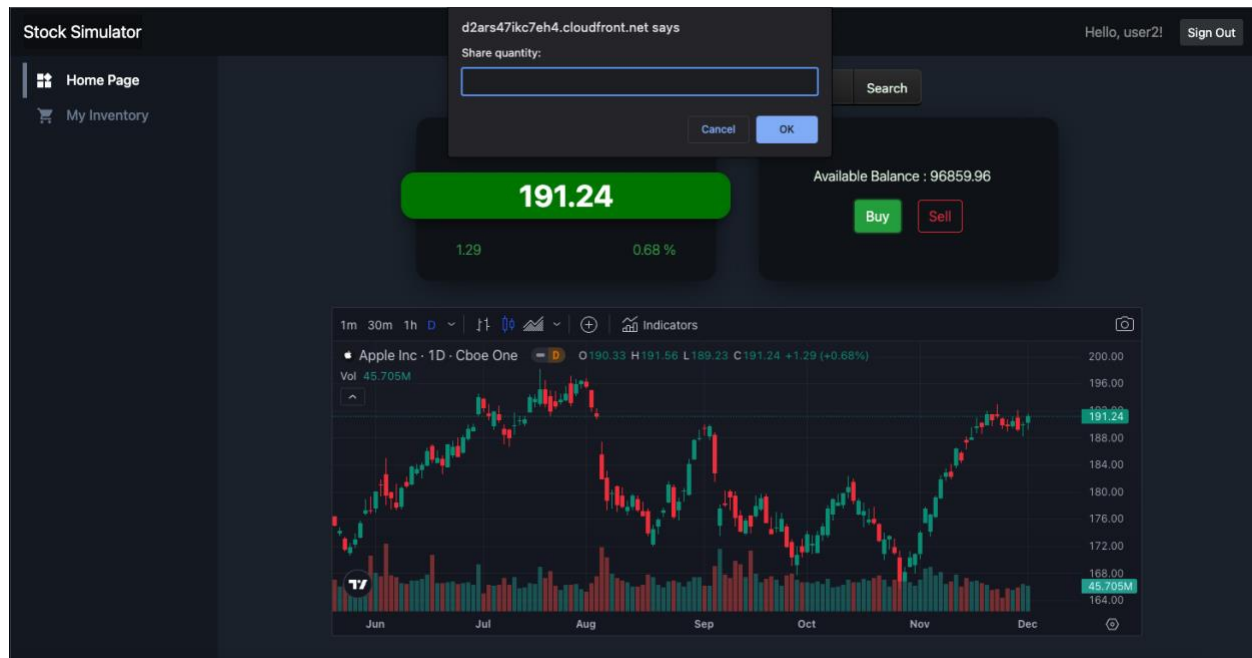


Figure 7 Home page (after logging in)

Once the user makes a new transaction, the browser will send the request to the back-end server, and the server will communicate with the database to update the transaction table.



tID	uID	companyName	dateTime	stockPrice	shareQuantity	action	changes
1	e74d9fa7-7212-4cd5-86b6-2d679c51000c	AAPL	2023-11-29 11:40:08	190.4	5	Buy	-952
2	e74d9fa7-7212-4cd5-86b6-2d679c51000c	SBUX	2023-11-29 11:40:28	101.18	10	Buy	-1011.8
3	e74d9fa7-7212-4cd5-86b6-2d679c51000c	AMZN	2023-11-29 11:40:37	147.03	8	Buy	-1176.24
4	aa0305ac-804d-492b-95a6-8475ce2679a9	META	2023-11-29 11:41:05	338.99	5	Buy	-1694.95
5	aa0305ac-804d-492b-95a6-8475ce2679a9	UPST	2023-11-29 11:41:14	26.17	20	Buy	-523.4
6	aa0305ac-804d-492b-95a6-8475ce2679a9	NVDA	2023-11-29 11:41:27	478.21	3	Buy	-1434.63

Figure 8 Buy/Sell

On the inventory page, the browser will request the server to ask for the inventory data with the user ID. Before displaying it on the web page, it will also calculate the reward rate based on the current stock price.

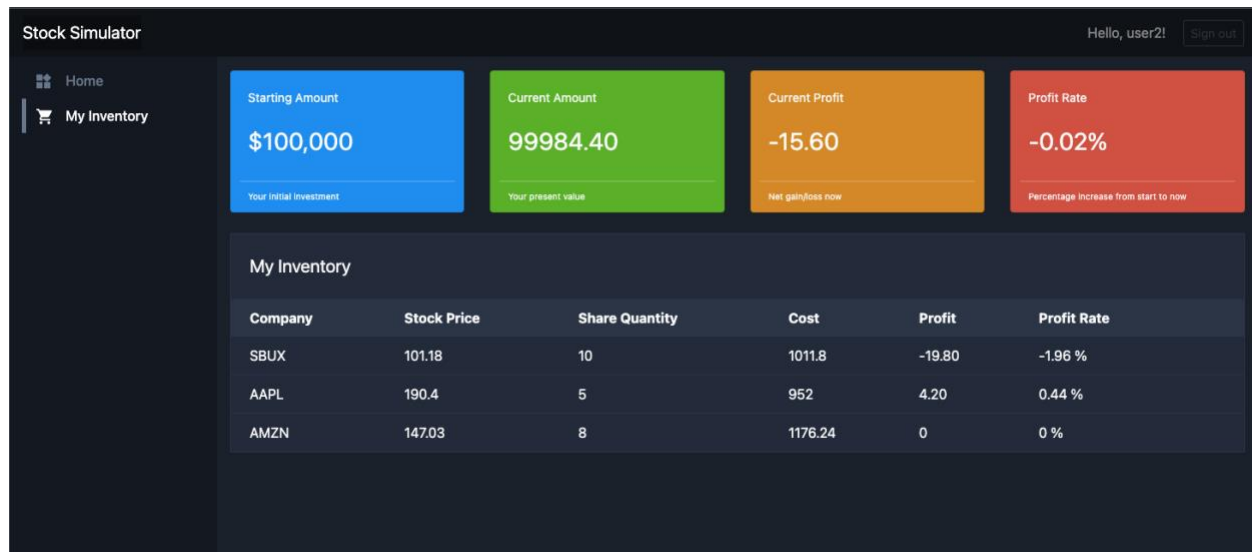


Figure 9 Inventory Page

Back end (Server)

On the back-end side, I used express.js to build up my server to handle client requests, including GET and POST, and route to the correct actions of querying the database.

```
app.get('/inventory', function(req, res) {
  const { uID } = req.query;
  console.log("inventory: " + uID)
  const query = `SELECT * FROM Inventory WHERE uID = ?`;
  db.query(query, uID, (error, results) => {
    if (error) {
      console.error('Query failed: (property) error: string');
      res.status(500).json({ error: 'Failed to retrieve data' });
      return;
    }
    console.log('Query result(inventory):', results);
    res.json(results); //
  });
});

app.get('/balance', (req, res) => {
  const { uID, userName } = req.query;
  console.log("balance: " + uID)
  const query = 'SELECT * FROM Transactions WHERE uID = ?';

  db.query(query, uID, (error, result) => {
    if (error) {
      console.error('Query failed:', error);
      res.status(500).json({ error: 'Failed to retrieve data' });
      return;
    }
    console.log('Query result(balance):', result);
    res.json(result);
  });
});
```

Figure 10 Back-end Server Code

Database

I used MySQL as my database to store the user transaction information. There are three tables.

Field	Type	Null	Key	Default	Extra
uID	varchar(255)	NO	PRI	NULL	
userName	varchar(255)	NO		NULL	

Figure 11 User table

Field	Type	Null	Key	Default	Extra
uID	varchar(255)	NO	PRI	NULL	
companyName	varchar(255)	NO	PRI	NULL	
stockPrice	float	NO		NULL	
shareQuantity	int	NO		NULL	
cost	float	NO		NULL	

Figure 12 Inventory Table

Field	Type	Null	Key	Default	Extra
tID	int	NO	PRI	NULL	auto_increment
uID	varchar(255)	NO	MUL	NULL	
companyName	varchar(255)	NO		NULL	
dateTime	datetime	NO		NULL	
stockPrice	float	NO		NULL	
shareQuantity	int	NO		NULL	
action	varchar(4)	NO		NULL	
changes	float	NO		NULL	

Figure 13 Transaction Table

4. Result

Successes

My main goal of the project was achieved: users can practice trading stock with real-time stock prices without using real money. The web application allows users to store transaction data and keep tracking the performance of the user's portfolio. Let people start to learn investing earlier and have enough understanding of trading before spending their own money in the stock market in the future.

Limitation

The most significant limitation of the web simulator is I only used free APIs to get stock information. Therefore, my current website cannot provide additional information like earning reports, ROI, some financial factors, etc. With a bigger budget, I believe I can create some features like filtering companies with customized criteria or testing users' strategies.

5. Conclusions

Design Choices

The choice of using JavaScript to develop the web was a good one. It let me cover from the front end to the back end with the same language. SQL database is also a good choice for organized data to access frequently.

Deploying the web application on AWS is a great choice as well. It provides all the features I need on one site, which is beneficial for future development and management.

Future Work

Going forward, I would like to continue improving on the web development project.

Functions-wise, as mentioned in the limitation section, I would like to implement more features to let users filter out companies with their strategy. In addition, I want to have the ability to import users' actual stock inventory from other source and let user manages all their inventory in one place.

Techniques-wise, I want to implement CI/CD to optimize the project update process.