

CIS 4951

Masoud Sadjadi, Florida International University

CAPSTONE I

StockOverflow

Team Members

**Alex Gomez, Brandon Rodriguez, Lorenzo
Zaidowick, Gabriel Guzman**

Introduction

Our project for this semester is a stock market simulation tool designated to merge mathematical modeling with the real world-financial field. The majority of financial applications fail to demonstrate the real market trends using statistical methods. StockOverflow drastically fulfills the gap through the implementation of Geometric Motion and Natural Language Processing.

This simulator educates users of all fields on how market volatility, drift, and emotional reactions to real-world events influence stock behavior. StockOverflow provides a unique and educational experience by giving interactive simulation.

Current System

The current system's simulation engine ('/spi/dimulate') implements Geometric Brownian Motion. The Geometric Brownian Motion equation is used to properly simulate the application, which takes both drift (slope) and volatility (randomness) to account.

$$dS(t)=\mu S(t)dt+\sigma S(t)dW(t)^*$$

(where μ is drift and σ is volatility)

The [next.js](#) framework powers the application, the React front end fetches the JSON generated by the simulation and renders it as an interactive chart with [Chart.js](#)

Purpose of New System

The goal of our system is to fulfill the gap between financial models and real world market behavior through an interactive web application. This

system is developed to help students and investors to have a better understanding of how stock prices evolve over time.

By this simulation of the stock price using the Geometric Brownian Motion, the application introduces users to the mathematical foundation behind the financial model. It also demonstrates how changes in the market sentiment can make an impact on the price direction in real time.

User Stories

These are the user stories that were implemented in the StockOverflow project. It helps us stay organized during the development of our project and introduces the basis for the implementation of features of the project.

Implemented User Stories

User Story#1: Finalize any conceptual understanding related to the project: StockOverflow.

User Story#2: Create a demo of a stock simulator, with the brownian motion equation used to simulate randomness.

User Story#3: Implement a loop timer which updates the stock and incorporates a real time simulation.

User Story#4: Research possible frontends and frameworks to incorporate the simulation into.

User Story#5: Begin experimenting with different types of zero-shot classifiers to create a sentiment analyzer.

User Story#6: Begin creating the front-end website used to display the charts.

User Story#7: Finalize any conceptual understanding related to the project: StockOverflow.

User Story#8: Improve accuracy and validation of Brownian Motion Simulation.

User Story# 9: Implement a loop timer which updates the stocks and incorporates a real time simulation.

User Story#10: Build a minimal interactive frontend displaying live simulated stocks.

User Story#11: Finish a demo of a stock simulation, with the brownian motion equation used to simulate randomness.

User Story#12: Research possible ways to implement a zero-shot classifier into the program.

User Story#13: Fix any git related bugs.

User Story#14: Implement the AI portion.

User Story#15: Create a front-end using [next.js](#)

User Story#16: Finalize any conceptual understanding related to the project.

User Story#17: Further improve the frontend of the program.

User Story#18: Research and start to implement possible sentiment analysis.

User Story#19: Further improve stock format, making sure there are no bugs related to the real time feed.

User Story#20: Conduct testing to ensure the new version of the program has no issues.

User Story#21: Research new algorithms to improve our model.

User Story#22: Keep improvement of the stock format.

User Story#23: Finish toucher on front-end using [next.js](#)

User Story#24: Implement a zero-shot classifier into the program based on best research findings.

User Story#25: Finish new features for the frontend of the program.

User Story#26: End research on new algorithms to improve our model.

User Story#27: Add a new CSV file feature in our program.

User Story#28: Finish a demo of a stock simulation with the Brownian motion equation used to simulate randomness.

User Story#29: : Improve the font-end using [next.js](#)

User Story#30: Test app deployment using Vercel.

Project Plan

Agile technique is applied for the software development of this project and sprint planning are detailed in this section as well as description of system requirements used for this project.

System Requirements

- Nodev16.26.0 (LTS)
- React 18x
- Git (version control)
- Vercel (deployment)
- [Chart.js](#)
- Hugging Face

Sprint Plan

Sprint 1

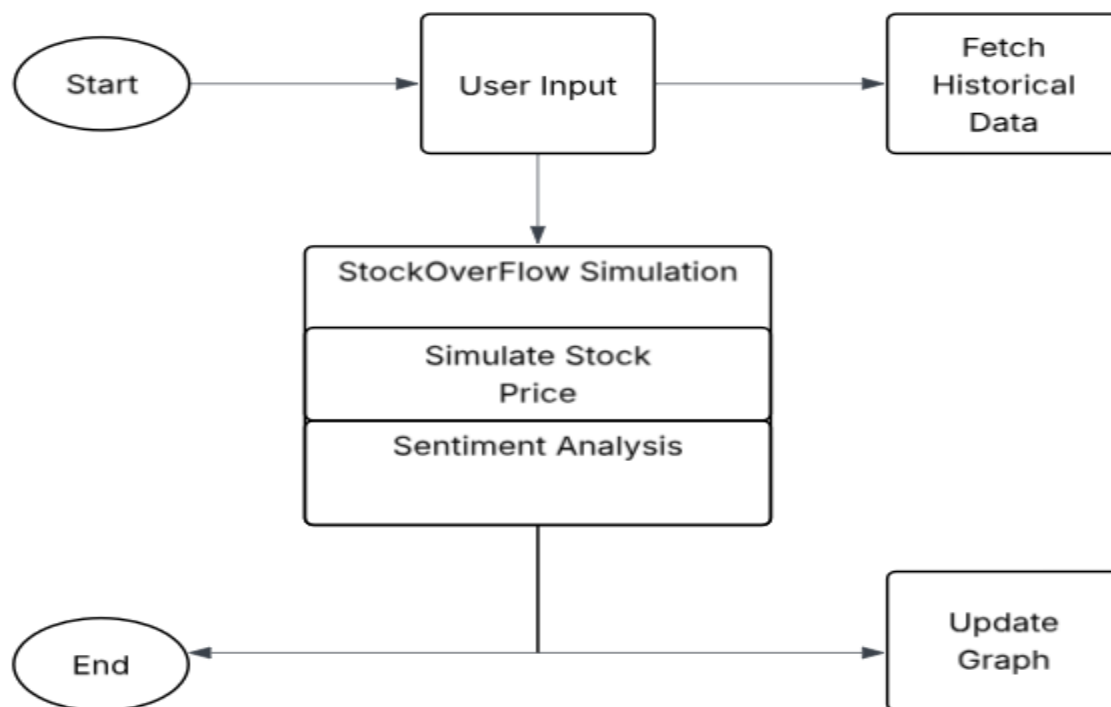
Sprint 2

Sprint 3

Sprint 4

Sprint 5

Architectural Pattern



Key Features:

Our stock simulator works are based on the Brownian Motion Simulation using the equation below. Drift and volatility influence the price movement over time. Another feature that we implemented in our project was the sentiment-driven Market response; positive sentiments increase the drift, and negative sentiment decreases the drift. Market line color changes depending on green for rising trends and red for opposite direction.

Dynamic Visualization:

The stock price simulation is displayed using Chart.js, which updates the graph dynamically every two seconds. Users have control over several key simulation parameters including initial stock price, Drift (%), Volatility (%),

and Simulations duration in days. This layout allows users to explore how different factors influence market behavior in real time, offering flexibility and education insights on the financial market.

Real Market Metadata:

StockOverflow grabs historical stock data from Yahoo Finance through the custom `/api/stockstats` endpoint to ensure that simulations are truly in it fact. This system computes logarithmic returns from this data and uses them to derive the drift and volatility. The metadata gives the simulator realistic starting points and helps users analyze how past market behavior can inform future trends.

Implementation

The implementation of StockOverflow integrates a range of modern frontend and backend technologies to deliver a responsive, data-driven stock market simulation experience. The system was built using a dynamic frontend interface and a backend powered by external APIs. The frontend is developed using React along with Next.js framework enabling quick rendering and smooth navigation. Data visualization was made with Chart.js integrated via react-chartjs wrapper allowing smooth, real time graph updates based on simulation results.

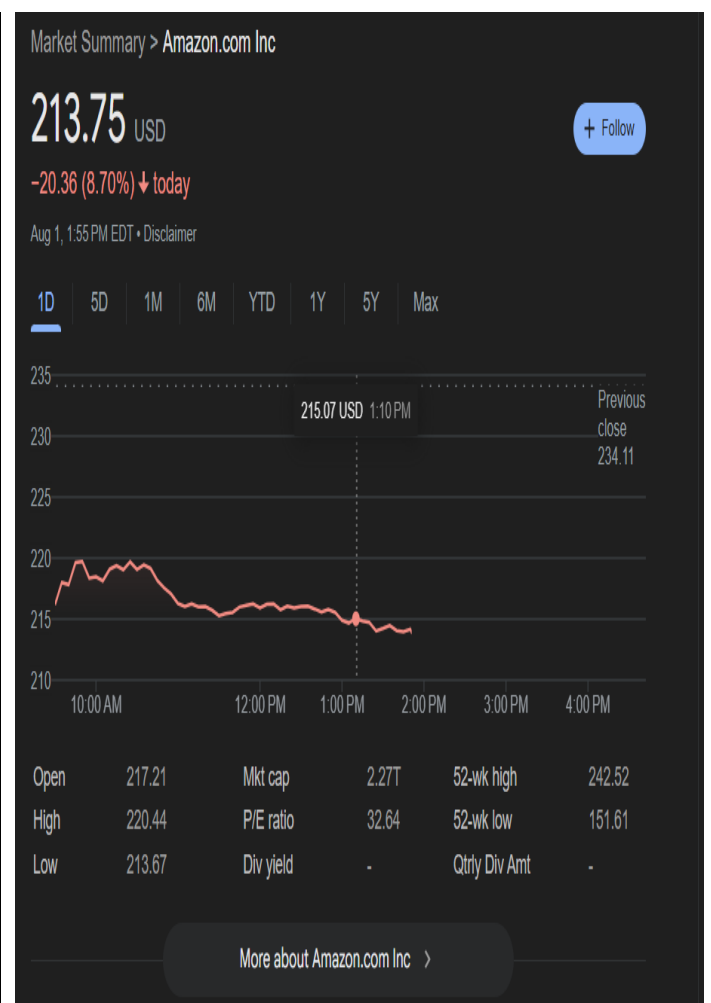
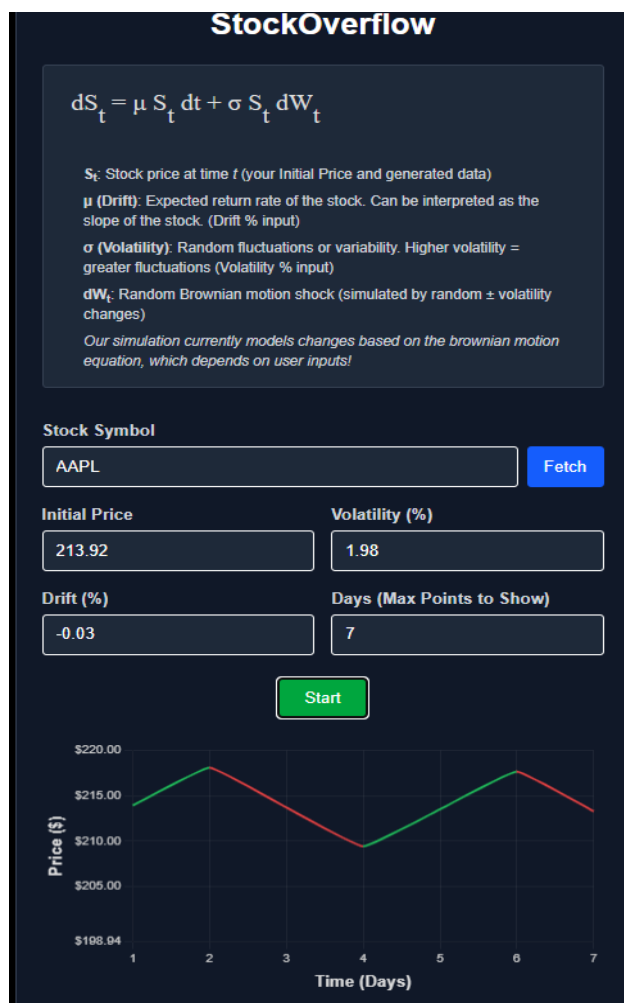
The backend consists of Nodes API routes which are used to handle requests and process stock simulation logic. This project uses data pulled from Yahoo Finance API, where historical price movements are analyzed. The backend derives statistical parameters like the drift and volatility by computing log returns, and these are inputs to the Geometric Brownian Motion simulation.

The layout of this application was styled with Tailwind CSS giving a friendly user interface. Hosting is handled through Vercel that supports smooth

deployment and less launching time using serverless infrastructure. StockOverflow delivers an educational and engaging user experience that delivers how price trends and sentiment dynamics can influence market behavior.

Verification

Our application successfully simulates a stock with a given start (USD), drift, volatility, and length (days). As shown through implementation, our application has a 2 second timer which updates the daily price of the stock. The fetch system was tested with multiple different stock symbols, which granted a 100% success rate.



APP Structure

- **app/page.js** – landing page with a button that routes to /brownian.
- **app/brownian/page.js** – main simulation UI built with React and Chart.js.
- **app/api/stockstats/route.js** – API endpoint that fetches drift and volatility for a ticker.
- **app/layout.js** – global layout and fonts.
- **app/globals.css** – Tailwind CSS styles.

Brownian Motion Simulation

The simulator maintains state for the current price, drift, volatility, and number of days. Every two seconds it generates the next price using a basic Brownian motion step:

```
const generateNextPrice = () => {  
  setDataPoints((prev) => {  
    const lastPrice = prev[prev.length - 1] ?? initialPrice;  
    const volDecimal = volatility / 100;  
    const driftDecimal = drift / 100;  
    const change = Math.random() < 0.5 ? -volDecimal : volDecimal;  
    let newPrice = lastPrice * (1 + driftDecimal + change);  
    if (newPrice < 0.01) newPrice = 0.01;  
    return [...prev, newPrice].slice(-days);  
  });  
};
```

- InitialPrice is the starting value.
- Volatility and drift are user supplied percentages.
- Change randomly adds or subtracts volatility.
- The new price is clamped at 0.01 and stored in the dataPoints array.

Stock Statistics Endpoint

The API route /api/stockstats retrieves recent daily prices for a symbol from Yahoo Finance, computes log returns, and returns the mean (drift) and standard deviation (volatility):

```
const res = await fetch(
  `https://query1.finance.yahoo.com/v8/finance/chart/${symbol}?range=1y&interval=1d`,
  { headers: { 'User-Agent': 'Mozilla/5.0' } }
);
const json = await res.json();
const prices = json.chart?.result?.[0]?.indicators?.adjclose?.[0]?.adjclose || [];
const returns = [];
for (let i = 1; i < prices.length; i++) {
  const prev = prices[i - 1];
  const curr = prices[i];
  if (prev != null && curr != null && prev > 0) {
    returns.push(Math.log(curr / prev));
  }
}
const mean = returns.reduce((a, b) => a + b, 0) / returns.length;
const variance = returns.reduce((a, b) => a + (b - mean) ** 2, 0) / returns.length;
const drift = mean * 100;
const volatility = Math.sqrt(variance) * 100;
```

User Interface

Home Page - A button that navigates to the simulator:

```
const goToBrownian = () => {
  router.push("/brownian");
};
```

Layout - Fonts and global styles are configured in app/layout.js

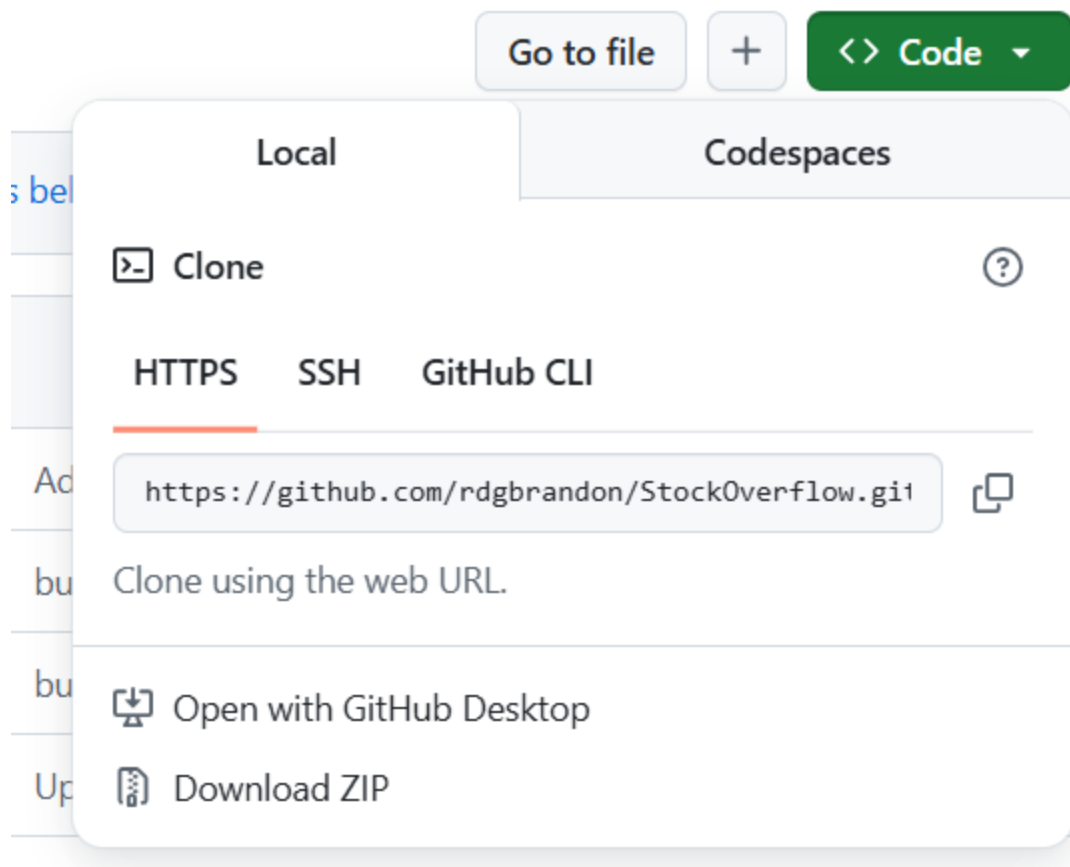
```
export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body className={` ${geistSans.variable} ${geistMono.variable} antialiased`} >
        {children}
      </body>
    </html>
  );
}
```

Simulation Controls – Users can adjust initial price, volatility, drift, and the number of points displayed. When running, the graph updates every two seconds.

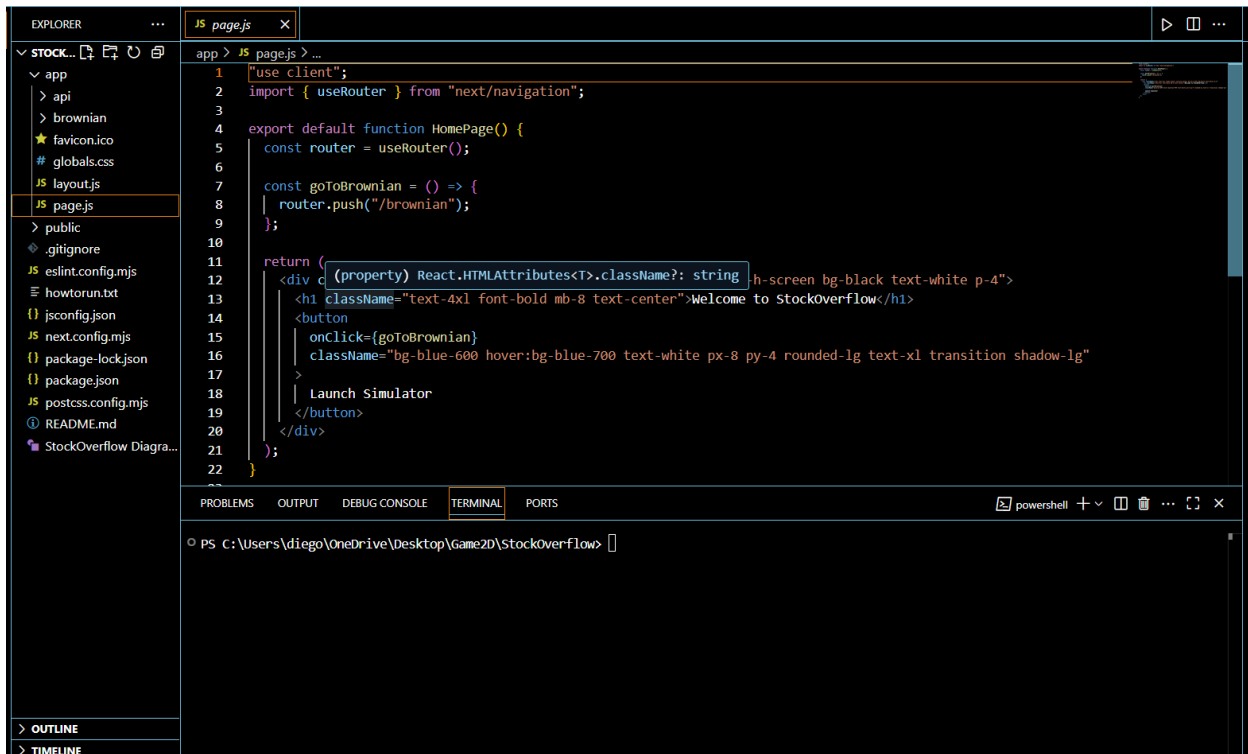
User Manual, Installation Document

Installation

Browse <https://github.com/rdgbrandon/StockOverflow.git> and press the green button “Code”. Either clone the repository through git by copying the link, or press “Download ZIP” to have the zip file of OverStockflow.



Select any folder as repository destination, and it should open as below:



Go to terminal and install all the following using these commands: `npm install`, `npm install react-chartjs-2` [chart.js](#)

npm install - This command installs all the dependencies listed in package.json file, since cloning the [Node.js](#) from Github would not include the packages but only version and names are stored in package.json.

react-chartjs-2 - A react wrapper for [chart.js](#) that gives you a smooth use of charts in React.

[chart.js](#) - Javascript library which has react-chartjs-2 as a dependable.

npm install react-chartjs-2 [chart.js](#)

Adds those two packages to the project and updates the packages json.

Overview

```
PS C:\Users\diego\OneDrive\Desktop\Game2D\StockOverflow> npm install

added 337 packages, and audited 338 packages in 32s

136 packages are looking for funding
  run `npm fund` for details

2 low severity vulnerabilities

To address all issues, run:
  npm audit fix
```

```
PS C:\Users\diego\OneDrive\Desktop\Game2D\StockOverflow> npm install react-chartjs-2 chart.js

changed 1 package, and audited 338 packages in 2s

136 packages are looking for funding
  run `npm fund` for details

2 low severity vulnerabilities
```

Start the development server

Npx next dev

Open <http://localhost:3001> in your browser and launch simulator to access the Brownian page.

```
PS C:\Users\diego\OneDrive\Desktop\Game2D\StockOverflow> npx next dev
⚠ Port 3000 is in use, using available port 3001 instead.
▲ Next.js 15.3.3
- Local:      http://localhost:3001
- Network:    http://10.0.0.123:3001

✓ Starting...
✓ Ready in 1767ms
o Compiling / ...
✓ Compiled / in 2.5s (722 modules)
```

Overview of website

StockOverflow

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

S_t : Stock price at time t (your Initial Price and generated data)

μ (**Drift**): Expected return rate of the stock. Can be interpreted as the slope of the stock. (Drift % input)

σ (**Volatility**): Random fluctuations or variability. Higher volatility = greater fluctuations (Volatility % input)

dW_t : Random Brownian motion shock (simulated by random $\pm$ volatility changes)

Our simulation currently models changes based on the brownian motion equation, which depends on user inputs!

Stock Symbol

Fetch

Initial Price

Volatility (%)

Drift (%)

Days (Max Points to Show)

Start