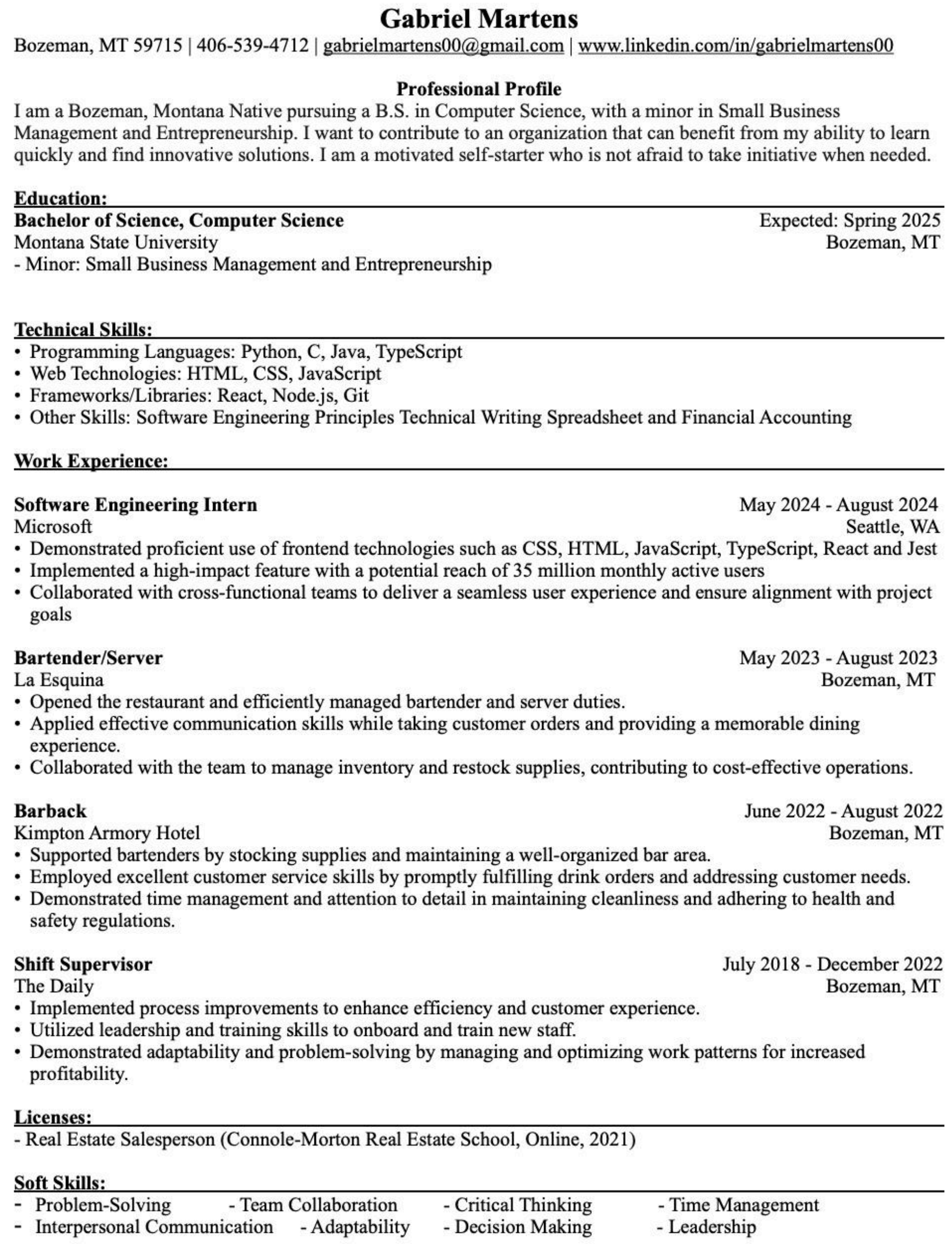
P.A.S.S.

Gabe Martens
Mason Watamura
Shane Costello

Introduction

Participating in a stock exchange is a well-known method of increasing personal wealth. Nearly all American adults are familiar with the concept, yet only 62% are invested in stocks. Breaking that figure down by income, we see that 84% of Americans earning $100,000+ annually own stocks. This is true for only 29% of those earning less than $40,000 annually. A similar positive correlation exists with the highest level of education completed. These statistics make apparent the reason why Americans are not invested in the stock market. It is due to a lack of exposure and relevant education. When engaging with the stock market, people face a plethora of industry jargon that only someone with knowledge of the field would understand. The process of purchasing and trading a stock for the first time is subsequently confusing. With money on the line, people are nervous about making uninformed decisions and opt not to participate at all. It is the mission of our software solution, the Portfolio Automation Software System (PASS), to lower the barrier of entry for Americans investing in the stock market. It is our goal to simplify the process and get more Americans invested in the stock market. To do this, we have developed a unique stock selection algorithm. The user inputs how long they'd like to be invested and how much they'd like to invest, and our algorithm does the rest. Pulling real-time stock data from the NASDAQ index (via a news API), our algorithm considers many factors, including sentiment analysis, to develop a diverse, high-yielding portfolio. Users are presented with information regarding their generated portfolio, including relevant news stories and pertinent financial metrics, enabling them to become more informed. Coupled with an intuitive user interface, PASS is the answer for Americans looking to make their first investments.

Qualifications



Gabriel Martens

Bozeman, MT 59715 | 406-539-4712 | gabrielmartens00@gmail.com | www.linkedin.com/in/gabrielmartens00

Professional Profile

I am a Bozeman, Montana Native pursuing a B.S. in Computer Science, with a minor in Small Business Management and Entrepreneurship. I want to contribute to an organization that can benefit from my ability to learn quickly and find innovative solutions. I am a motivated self-starter who is not afraid to take initiative when needed.

Education:

Bachelor of Science, Computer Science Expected: Spring 2025
Montana State University Bozeman, MT
- Minor: Small Business Management and Entrepreneurship

Technical Skills:

- Programming Languages: Python, C, Java, TypeScript
- Web Technologies: HTML, CSS, JavaScript
- Frameworks/Libraries: React, Node.js, Git
- Other Skills: Software Engineering Principles Technical Writing Spreadsheet and Financial Accounting

Work Experience:

Software Engineering Intern May 2024 - August 2024
Microsoft Seattle, WA
- Demonstrated proficient use of frontend technologies such as CSS, HTML, JavaScript, TypeScript, React and Jest
- Implemented a high-impact feature with a potential reach of 35 million monthly active users
- Collaborated with cross-functional teams to deliver a seamless user experience and ensure alignment with project goals

Bartender/Server May 2023 - August 2023
La Esquina Bozeman, MT
- Opened the restaurant and efficiently managed bartender and server duties.
- Applied effective communication skills while taking customer orders and providing a memorable dining experience.
- Collaborated with the team to manage inventory and restock supplies, contributing to cost-effective operations.

Barback June 2022 - August 2022
Kimpton Armory Hotel Bozeman, MT
- Supported bartenders by stocking supplies and maintaining a well-organized bar area.
- Employed excellent customer service skills by promptly fulfilling drink orders and addressing customer needs.
- Demonstrated time management and attention to detail in maintaining cleanliness and adhering to health and safety regulations.

Shift Supervisor July 2018 - December 2022
The Daily Bozeman, MT
- Implemented process improvements to enhance efficiency and customer experience.
- Utilized leadership and training skills to onboard and train new staff.
- Demonstrated adaptability and problem-solving by managing and optimizing work patterns for increased profitability.

Licenses:

- Real Estate Salesperson (Connole-Morton Real Estate School, Online, 2021)

Soft Skills:

- Problem-Solving - Team Collaboration - Critical Thinking - Time Management
- Interpersonal Communication - Adaptability - Decision Making - Leadership

Mason Watamura

3532 72nd Ave Ct W, University Place, WA 98466
Ph: 2533804549
2003xcrun@gmail.com

GitHub Account

Education

Montana State University- Bozeman, Montana

Computer Science Major - Interdisciplinary Studies; Data Science Minor; GPA: 3.64

Languages learned

- Web Based (HTML, PHP, etc.)
- R
- Python
- C/C++
- Java
- C#
- Go
- Typescript

Work Experience

T-Seal Employee *Oct 2022- Current*

Techlink- Bozeman, Montana

Wrote and tested automations that tested different features of a sustainment management system website for the Department of Defense using C# and Visual Studio's Selenium WebDriver framework. Used knowledge of x-paths and html code to create the automations used to test the features. Also used a large knowledge of git to work with a team while creating the automations.

Side Projects

Interdisciplinary Project – Automated Portfolio Project

A year long project for the capstone course for an automated stock creating portfolio. This involved using news APIs to check an input from the user for stocks the user was curious about and using a data checking model to make sure working with those stocks will work in the users' favor or not.

Bridger Solar Club Website

Worked with the club on campus to build a fully functioning website for their needs. This involved communication about content needs and making sure that they were happy with the overall view of how the website appeared as a user. It was also good experience working with clients who may or may not be on top of what they may need to give the development team and work arounds to that.

Volunteer Experience

Camp Agape Northwest *2017- Current*

Raft Island, Washington

Worked with the organization to provide a week long summer camp experience for kids with cancer and their families. During the week, time is spent with the kids and families helping to create memories while their child/sibling faces cancer and allowing families to have a safe space to talk about their journeys. Throughout the year, time is spent fundraising to be able to continue supporting the organization and to allow the camp to continue its effect on family's lives.

Shane M. Costello

Bozeman, Montana · 631-680-1988 · shanecost2002@gmail.com · github.com/ShaneCost

Education:

Montana State University, ***Bozeman, MT (GPA 3.98)*** **Spring 2022 - Spring 2025** ***(expected)***

- Pursuing a Bachelor's Degree in Computer Science
- Minor studies in Finance

Suffolk County Community College, ***Riverhead, NY (GPA 4.0)*** **Fall 2020 - Fall 2021**

- Associated Degree in Liberal Studies

Relevant Coursework:

Machine Learning*, Database Systems*, Computer Security*, Embedded Systems: Robotics, Computer Science Theory, Software Engineering Applications, Human-Computer Interaction, Web Development, Data Structures and Algorithms, Programming with C, Linear Algebra
**in-progress as of Fall 2024*

Relevant Projects:

Internal Guest Management System, ***HRDC Warming Shelter*** github.com/423s24/Group_3
Software Engineering Applications

- Developed a web application used to check guests in, store, and update relevant guest information
- Worked directly with the client to develop requirement specifications
- Collaborated in a group of four throughout the entire software lifecycle
- Wrote thorough user and developer documentation

Spotify API Web Application github.com/ShaneCost/spotifyInte
Web Development

- Integrated the Spotify API in a web application to retrieve and display user information
- Worked with Flask, a Python framework
- Collaborated in a two-person group throughout the entire software lifecycle

Professional Experience:

Software Engineer Intern, ***Montana State University*** **December 2023 - present**
Bozeman, MT

- Developed and deployed multiple Django web applications
- Collaborated with clients to create requirement specifications
- Individually responsible for design, development, testing, deployment, and maintenance

SmartyCats Tutor, ***Montana State University*** **September 2023 - May 2024**
Bozeman, MT

- Topics taught range from printing "Wello, World!" to using advanced data structures
- Developed numerous teaching techniques to meet the needs of students with varying skill levels
- Host group, as well as 1-on-1 tutoring sessions

BOH, ***Ted's Montana Grill*** **June 2022 - August 2023**
Bozeman, MT

- Prepare ingredients accurately and efficiently
- Organize and maintain a tidy station
- Ensure adherence to recipe standards for consistent quality

Ramp Agent, ***Menzies Aviation*** **January 2021 - August 2021**
Bozeman, MT

- Collaborated with team to safely load and unload baggage from commercial aircraft
- Conducted thorough safety inspections as part of routine procedures
- Ensured efficient operations and maintained cleanliness of the work area

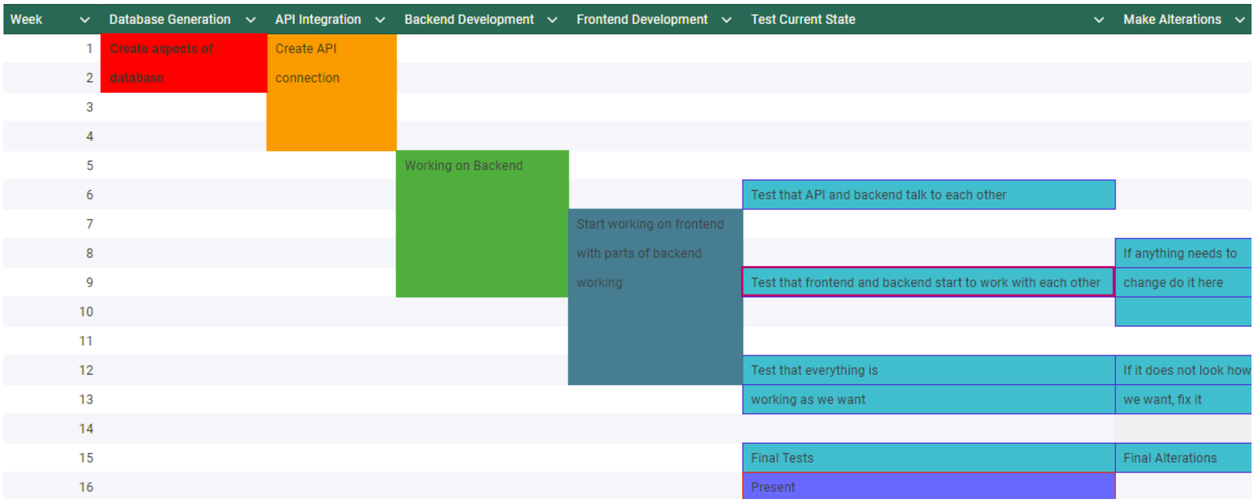
Technical Skills:

- **Java**
- **Python**
- **C / C++**
- **HTML**
- **CSS**
- **JavaScript**
- **UML**
- **Agile Development**
- **Git**
- **Django**
- **SQL**

Background

As of right now “61% of Americans [report] that they own stock… This is up from the 56% measured in 2021 and 55% in 2020” (Gallup). These statistics are from a study that was updated in May of 2023 and continues to show a trend that more Americans are investing in the stock market. This increasing number presents a unique market opportunity to capture new investors who may not know where to start when it comes to stock market investing. A study from the FCA found that “only 2% of investors have a timeframe of more than 5 years…and 14% have no timeline in mind at all” and, “31% of people [invest] to earn more money than they would in a savings account” (FCA). People may see videos on social media or the news of investors and traders making a large amount of money in a short amount of time and want to capitalize on the same opportunities. Individuals often place themselves in a position of risk in which they spend money and time trying to understand how to invest in the stock market due to the difficulty of predicting trends and making educated financial decisions in the stock market. The movie “Dumb Money” chronicles the tale of the 2021 short squeeze of Gamestop stock in which the stock price rose by 1,700 percent from a price of around $20 to a pre-market price of $500 in only 6 months. The story highlights the complexity of the stock market and the power of retail traders. It culminates in January 2021 when “Equity and options trading volume in the U.S. reaches its highest-ever single-day level” (Salvucci). This prompted Robinhood and other popular trading apps to halt the buying of the stock. While some may look at this story as a victory, the stock quickly fell again and today (November 7th, 2024), sits at a price of $23. Many retail investors not armed with the information or criteria to make smart financial decisions did not sell their stock which resulted in large financial losses. This is where our website would differ from all of the other stock trading platforms. Our web application will automate the majority of the investing process and help inform users decision-making making criteria that lead to the minimization of risk when investing. By doing this, much of the stress is taken out of investing and what remains is a gamified way to trade stocks responsibly. This differs from other websites that place 100% of the responsibility on the user. Our website will automate portfolio creation, based on a user-determined timeline, through financial and sentiment analysis. Traders will be able to filter the market by relevant categorical data and manipulate their portfolios as they see fit. Our system intends to grant the user enough freedom to determine which types of stocks they want to work with and for how long they choose to invest in those stocks. Through the transparency of the automation process, the user will become educated on what kinds of criteria go into trading not only profitably, but responsibly. This in turn will lead to a deeper understanding of the market.

Work Schedule



Week	Database Generation	API Integration	Backend Development	Frontend Development	Test Current State	Make Alterations
1	Create aspects of	Create API				
2	database	connection				
3						
4						
5			Working on Backend			
6					Test that API and backend talk to each other	
7				Start working on frontend		
8				with parts of backend		If anything needs to
9				working	Test that frontend and backend start to work with each other	change do it here
10						
11						
12					Test that everything is	If it does not look how
13					working as we want	we want, fix it
14						
15					Final Tests	Final Alterations
16					Present	

Proposal Statement

Our project will create a gamified stock trading web application. After the user determines categorical criteria (e.g. Industry, Company Size, Geographic Location, etc.) and a timeframe, our system will automate the creation of a portfolio for the user. Our system will utilize data from financial APIs to create multiple algorithms to analyze the finances and news sentiment for a large pool of stocks. Information including user data, API data, and system-generated data such as generated portfolios will be stored in a database. More specifics to the project will be explained below.

Functional Requirements

1. **User Authentication**
    1.1. Trader registers: The system shall allow a user to create an account with a unique username and password.
    1.2. Trader login: The system shall authenticate users to provide personal and secure access.
2. **Initial Stock Integration**
    2.1. Initial Stock Loading: The system shall load an initial set of stocks by calling external APIs.
    2.2. Financial Metrics: The system shall be able to reference financial metrics for stocks in the database.
    2.3. Categorical Metrics: The system shall be able to reference categorical metrics (e.g. Industry, Company Size, Geographic Location, etc.) for stocks in the database.
    2.4. News Stories: The system shall be able to reference news stories for stocks in the database.

3. **Portfolio Creation**
    - **3.1.** Timeline Selection: The system shall require the user to select a timeframe (e.g. 1 week - 1 quarter) and shall filter stocks based on upcoming earnings.
    - 3.2. User Filters: The system shall allow users to filter stocks based on preferences in categorical metrics.
    - 3.3. Metric Analysis: The system shall use the metric analysis algorithm to evaluate, weigh, remove, and perform a risk assessment on stocks.
    - 3.4. Sentiment Analysis: The system shall use the sentiment analysis algorithm to provide further insight into each stock.
4. **Metric Analysis**
    - 4.1. Stock evaluation: The system shall use point-based ranking to weigh the stocks based on common financial metrics, removing stocks that do not meet a certain threshold.
    - 4.2. Weight Assignment: The system shall use the point-based ranking of the stocks to create a weighted portfolio.
    - 4.3. Risk Assignment: The system shall use the point-based ranking of the stocks to determine the risk rating of the portfolio.
5. **Sentiment Analysis**
    - 5.1. News Article Retrieval: The system shall retrieve recent news articles for all stocks in the list.
    - 5.2. Sentiment Assessment using AlphaVantage: The system shall assess the sentiment of the articles by calling external APIs.
    - 5.3. Sentiment Assessment using chatGPT: The system shall use an NLP API to assess the overall sentiment of a stock based on recent news articles.
    - 5.4. Sentiment Overview: The system shall provide an overview of its decision criterion.
6. **Back-Testing**
    - 6.1. Test Mode: The system shall be able to be placed in testing mode which allows an arbitrary date to be chosen. This date must be at least as far from the current date as the timeline chosen to analyze.
    - 6.2. Monetary Input: The system shall allow the user to enter a monetary amount that is for testing purposes only.
    - 6.3. System Simulation: The system shall perform as normal but when the user is prompted to invest the investing timeline will be simulated and reported.
    - 6.4. System Analysis: The system shall display relevant metrics (e.g. overall loss/gain, percent loss/gain, average risk rating, average sentiments score).

7. **Portfolio Purchase**
    7.1. Investment Amount Input: The system shall allow the user to enter the amount they would like to invest.
    7.2. Investing Disclaimer: The system shall notify users of the risk of investing prior to allowing the user to complete the purchase.
    7.3. Portfolio Analysis: The system shall allow the user to view financial metrics, categorical metrics, news stories, sentiment, and risk ratings of each stock within the portfolio.
    7.4. Portfolio Editing: The system shall allow the user to change the weights of the portfolio or remove stocks from the portfolio.
    7.5. Portfolio Cancelation/Deletion: The system shall allow the user to cancel the creation of a portfolio and restart or exit.
    7.6. Final Purchase: The system shall allow the user to finalize the creation of their portfolio and invest the specified amount.
8. **Portfolio Management**
    8.1. Realtime Overview: The system shall track the movement of markets in real time showing real-time gains and losses of assets held in the app.
    8.2. Historical Overview: The system shall allow a user to see the historical performance of personal assets they have invested within the system.
    8.3. View Current Portfolios: The system shall allow the user to view currently active portfolios that includes financial metrics, categorical metrics, news stories, sentiment, and risk rating of each stock within the portfolio.
    8.4. View Previous Portfolios: The system shall allow the user to view previously active portfolios that includes financial metrics, categorical metrics, news stories, sentiment, and risk rating of each stock within the portfolio.
    8.5. Delete Portfolio: The system shall allow the user to terminate the portfolio prior to the completion of its timeline.

1: User Authentication

US 1.1: As a user of the portfolio automation system, I want to create an account, so that my data can be securely stored.

Use Case Name: Trader registers

Related requirements:
- R1.2 User Login

Goal In Context: The system shall allow a user to create an account with a unique username and password.

Preconditions:

1. The user has access to the system’s registration page.
2. The system is connected to a backend database to store the user’s personal information.

Successful End Condition:

- The user successfully creates an account with a unique username and password.
- The user’s personal information is successfully stored in the database.
- The user is notified that account creation is successful.

Failed End Condition:

- The user is unable to create an account due to invalid input, pre-existing email, or password mismatch.
- The system displays the appropriate error message and data is not stored.

Primary Actors:

- New Trader: An individual attempting to create an account in the portfolio automation system

Secondary Actors:

- Database: Responsible for storing and validating user information

Main Flow of Events:

1. The user navigates to the account creation page
2. The system displays the registration form
3. The system validates the input
4. The system checks the database
5. The user submits the form
6. The system stores the data
7. The system notifies the user

Extensions: None

User Story:

US 1.2: As a user of the portfolio automation system, I want to log in to my account, so that I can access my data

Use Case Name: Trader login

Related requirements:

- R1.1: User Account Creation

Goal In Context: The system shall authenticate users to provide personal and secure access.

Preconditions:

1. The user has created an account in the system.
2. The user has accessed the system login page.
3. The system is connected to the database for authentication.

Successful End Condition:

- The user is authenticated and granted access to their account.

Failed End Condition:

- The user is not authenticated and access is rejected.

Primary Actors:

- Registered Trader: A registered user of the portfolio automation system.

Secondary Actors:

- Database: Stores and validates user credentials.

Main Flow of Events:

1. The user accesses the login page
2. The system displays the login form
3. User enters information
4. System validates
5. The user is granted access to their account

Extensions: None

2: Initial Stock Integration

User Story:

US 2.1: As a user of the portfolio automation system, I want the system to have access to a robust selection of stocks so that my portfolio can be diversified optimally and based on my preferences.

Use Case Name: Initial Stock Loading

Related requirements:

- R2.2: Financial Metrics
- R2.3: Categorical Metrics
- R2.4: News Stories

Goal In Context: The system shall load an initial set of stocks by calling external APIs.

Preconditions:

1. The system has access to external stock data API.
2. The system is connected to a database for storing the stock data.

Successful End Condition:

- The system successfully retrieves an initial set of stock data and stores it in the database.

Failed End Condition:

- The system is unable to retrieve stock data from the external API

Primary Actors:

- Investor: a stakeholder who wants to look at stocks.

Secondary Actors:

- API: The external service providing stock data.
- Database: Stores the fetched stock data for future use.

Main Flow of Events:

1. The system initiates the stock-loading process.
2. The system connects to the external stock API
3. The system requests stock data.
4. The API returns stock data.
5. The system stores the stock data.
6. The system confirms successful loading.

Extensions: None.

User Story:
US 2.2: As a user of the portfolio automation system, I want each stock in the system to be associated with a variety of financial metrics so that stocks can be effectively evaluated.

Use Case Name: Financial Metrics

Related requirements:

- R2.1: Initial Stock Loading
- R2.3: Categorical Metrics
- R2.4: News Stories

Goal In Context: The system shall be able to reference financial metrics for stocks in the database.

Preconditions:

1. The system has access to external stock data API.
2. The system is connected to a database for storing the stock data.
3. The system has loaded initial stocks into the database.

Successful End Condition:

- The system successfully retrieves financial metrics for stocks and stores them in the database.

Failed End Condition:

- The system is unable to retrieve financial data from the external API

Primary Actors:

- Investor: a stakeholder who wants to look at financial metrics of the stocks.

Secondary Actors:

- API: The external service providing stock data.
- Database: Stores the fetched stock data for future use.

Main Flow of Events:

1. The system initiates the stock-loading process.
2. The system connects to the external stock API
3. The system requests stock data.
4. The API returns stock data.
5. The system stores the stock data.

6. The system confirms successful loading.

Extensions: None.

User Story:
US 2.3: As a user of the portfolio management system, I want each stock to be associated with categorical metrics so that I can filter stocks based on my personal preferences.

Use Case Name: Categorical Metrics

Related requirements:
- R2.1: Initial Stock Loading
- R2.2: Financial Metrics
- R2.4: News Stories

Goal In Context: The system shall be able to reference categorical metrics (e.g. Industry, Company Size, Geographic Location, etc.) for stocks in the database.

Preconditions:
1. The system has access to external stock data API.
2. The system is connected to a database for storing the stock data.
3. The system has loaded initial stocks into the database.

Successful End Condition:
- The system successfully retrieves categorical metrics for stocks and stores them in the database.

Failed End Condition:
- The system is unable to retrieve categorical data from the external API

Primary Actors:
- Investor: A stakeholder who wants to look at categorical metrics of the stocks.

Secondary Actors:
- API: The external service providing stock data.
- Database: Stores the fetched stock data for future use.

Main Flow of Events:
1. The system initiates the stock-loading process.

2. The system connects to the external stock API
3. The system requests stock data.
4. The API returns stock data.
5. The system stores the stock data.
6. The system confirms successful loading.

Extensions: None.

User Story:
US 2.4: As a user of the portfolio management system, I want each stock to be associated with relevant news stories and current events so that an overall sentiment can be derived related to an individual stock.

Use Case Name: News Stories

Related requirements:
- R2.1: Initial Stock Loading
- R2.2: Financial Metrics
- R2.3: Categorical Metrics

Goal In Context: The system shall be able to reference news stories for stocks in the database.

Preconditions:
1. The system has access to external stock data API.
2. The system is connected to a database for storing the stock data.
3. The system has loaded initial stocks into the database.

Successful End Condition:
- The system successfully retrieves news stories for stocks and stores them in the database.

Failed End Condition:
- The system is unable to retrieve news data from the external API

Primary Actors:
- Investor: A stakeholder who wants to look at news stories pertaining to different stocks.

Secondary Actors:

- API: The external service providing stock data.
- Database: Stores the fetched stock data for future use.

Main Flow of Events:

1. The system initiates the stock-loading process.
2. The system connects to the external stock API
3. The system requests stock data.
4. The API returns stock data.
5. The system stores the stock data.
6. The system confirms successful loading.

Extensions: None.

3. Portfolio Creation

User Story:

US 3.1: As a user of the automated portfolio system, I want to select a duration I intend for my portfolio to be active so that the system can filter stocks by earnings dates.

Use Case Name: Timeline Selection

Related requirements:

- R2.2: Financial Metrics

Goal In Context: The system shall require the user to select a timeframe (e.g. 1 week - 1 quarter) and shall filter stocks based on upcoming earnings.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.

Successful End Condition:

- An initial dataset is created consisting of stocks whose earrings fall within the selected timeframe.

Failed End Condition:

- The initial portfolio is empty and a user must select broader criteria.
- The system is unable to fetch earnings dates.

Primary Actors:

- Investor: the user of the system selects a timeframe.

Secondary Actors:

- External API: Provides earnings data for stocks.
- Database: If the database is being populated with metrics the system will fetch data from the database to reduce the amount of API calls.

Main Flow of Events:

1. The user navigates to the portfolio creation page.
2. The system prompts the user to select a timeline for their portfolio.
3. The user selects a duration (e.g., 1 week, 1 month, 1 quarter).
4. The system retrieves earnings data for all stocks in the database.
5. The system filters the stocks by those with earnings dates within the selected timeframe.

Extensions: None.

User Story:

US 3.2: As a user of the portfolio automation system, I want to select specific categorical metrics (i.e. Industry, Geographical Location, Company Size, Stock Price Range, etc.) so that the system can reduce the initial portfolio based on my preferences.

Use Case Name: User Filters

Related requirements:

- R2.3: Categorical Metrics

Goal In Context: The system shall allow users to filter stocks based on preferences in categorical metrics.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.

Successful End Condition:

- The user selects “Filter by” and is able to filter the initial portfolio by categorical metrics.

Failed End Condition:

- The user is unable to filter the initial dataset.

Primary Actors:

- Investor: The investor is responsible for selecting the option to filter the initial portfolio

Secondary Actors:

- Database: If the database is being populated with metrics the system will fetch data from the database to reduce the amount of API calls.

Main Flow of Events:

1. The user navigates to the portfolio creation page.
2. The system prompts the user to select a timeline for the portfolio.
3. The system generates an initial list of stocks based on the earnings dates that fall within the selected timeline.
4. The system displays a "Filter by" option on the portfolio creation page.
5. The user selects the "Filter by" option and chooses one or more categorical metrics (e.g., industry, company size, geographic location).
6. The system retrieves the categorical data and filters the stocks according to the user's selections.

Extensions: None.

User Story:
US 3.3: As a user of the portfolio automation system, I want the system to use thorough metric analysis so that I can trust the portfolio being created.

Use Case Name: Metric Analysis

Related requirements:

- R2.2: Financial Metrics
- R4.1: Stock evaluation
- R4.2
- R4.3

Goal In Context: The system shall use the metric analysis algorithm to evaluate, weigh, remove, and perform a risk assessment on stocks

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.

Successful End Condition:

- A weighted portfolio is created.
- The weighted portfolio as an associated risk assessment.

Failed End Condition:

- A portfolio is unable to be created.

Primary Actors:

- Investor: The investor is responsible for prompting the system to create the portfolio.

Secondary Actors:

- External API: Provides quantifiable metrics data for stocks.
- Database: If the database is being populated with metrics the system will fetch data from the database to reduce the amount of API calls.

Main Flow of Events:

1. The user prompts the system to create the portfolio.
2. The system analyzes each stock using the algorithm.
3. An initial portfolio is created.

Extensions: None.

User Story:
US 3.4: As a user of the portfolio automation system, I want to view the sentiment related to stocks in my automated portfolio so that I can understand the overall opinion of the stock.

Use Case Name: Sentiment Analysis

Related requirements:

- R2.4: News Stories
- R5.1:

- R5.2:
- R5.3:
- R5.4:

Goal In Context: The system shall use the sentiment analysis algorithm to provide further insight into each stock.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.
5. The user has prompted the system to create the portfolio.
6. The system has created an initial portfolio based on financial analysis.

Successful End Condition:

- Each stock in the portfolio receives a sentiment score based on recent news.

Failed End Condition:

- The system is unable to fetch news stores.
- The system is unable to create sentiment scores for the stocks in the initial portfolio.

Primary Actors:

- Investor: The investor is responsible for prompting the system to display the sentiment scores.

Secondary Actors:

- External API: Either a sentiment API or chatGPT via prompt engineering will be used to establish the sentiment of a stock based on recent news stories.
- Database: If the database is populated with news stories the system will fetch data from the database to reduce the amount of API calls.

Main Flow of Events:

1. The system calls the get news stories function.
2. The system retrieves news stories either from the database or from the API
3. If the system retrieved news stories from the API the database is populated with those. News stories.

4. For each stock in the initial portfolio, the system calls the sentiment analysis function which is either a sentiment API or a ChatGPT API that utilizes prompt engineering to evaluate each news story.
5. The sentiment of each stock in the portfolio is displayed to the user
6. The user is provided with the option to remove stocks with low sentiment.

Extensions: None.

4: Metric Analysis

User Story:

US 4.1: As a user of the portfolio automation system, I want the system to remove stocks that don't meet a certain financial validity so that my risk is minimized.

Use Case Name: Stock evaluation

Related requirements:

- R3.3: Metric Analysis
- R4.2: Weight Assignment
- R4.3: Risk Assignment

Goal In Context: The system shall use point-based ranking to weigh the stocks based on common financial metrics, removing stocks that do not meet a certain threshold.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.

Successful End Condition:

- The system successfully removes stocks that do not meet the threshold.

Failed End Condition:

- The system fails to run the metric analysis algorithm
- The system contains stocks that do not meet the threshold requirement.

Primary Actors:

- Investor: Picks stocks that can later be removed from the created portfolio.

Secondary Actors:

- External API: Provides quantifiable metrics data for stocks.
- Database: If the database is being populated with metrics the system will fetch data from the database to reduce the amount of API calls.

Main Flow of Events:

1. The system fetches each relevant metric based on the algorithm.
2. The system applies the point ranking algorithm to each stock in the dataset.
3. The system removes stocks that fall below a certain point threshold.

Extensions: None.

User Story:
US 4.2: As a user of the portfolio automation system, I want the system to weigh stocks based on financial analysis of common financial metrics so that my portfolio can be diversified in a way that minimizes risk.

Use Case Name: Weight Assignment

Related requirements:

- R3.3: Metric Analysis
- R4.1: Stock evaluation
- R4.3: Risk Assignment

Goal In Context: The system shall use the point-based ranking of the stocks to create a weighted portfolio.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.
5. The system has removed stocks that do not meet the criteria threshold.

Successful End Condition:

- The system creates an initial diversified, weighted portfolio composed of stocks whose companies meet the minimum criteria.

Failed End Condition:

- The system fails to create a weighted portfolio.

Primary Actors:

- Investor: Picks the stocks that will be weighed.

Secondary Actors:

- External API: Provides quantifiable metrics data for stocks.
- Database: If the database is being populated with metrics the system will fetch data from the database to reduce the amount of API calls

Main Flow of Events:

1. The system fetches each relevant metric based on the algorithm
2. The system applies the point ranking algorithm to each stock in the dataset
3. The system takes the stocks remaining after stock evaluation and creates a weighted portfolio in which the percentage of each stock in the portfolio is based on the number of points received out of the total points awarded.

Extensions: None.

User Story:
US 4.3: As a user of the portfolio automation system, I want to be provided with an overall risk assessment of the generated portfolio so that my decision on whether or not to invest in the stock can be well-informed.

Use Case Name: Risk Assignment

Related requirements:

- R3.3: Metric Analysis
- R4.1: Stock evaluation
- R4.2: Weight Assignment

Goal In Context: The system shall use the point-based ranking of the stocks to determine the risk rating of the portfolio.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.

4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.
5. The system has removed stocks that do not meet the criteria threshold.
6. A weighted portfolio based on the remaining stocks has been created.

Successful End Condition:

- The user is displayed a risk assessment for the generated portfolio

Failed End Condition:

- The system is unable to generate a risk assessment.

Primary Actors:

- Investor: Picks the stocks that will later have a risk assigned to them.

Secondary Actors:

- External API: Provides quantifiable metrics data for stocks.
- Database: If the database is being populated with metrics the system will fetch data from the database to reduce the amount of API calls

Main Flow of Events:

1. The system fetches each relevant metric based on the algorithm
2. The system applies the point ranking algorithm to each stock in the dataset
3. Based on predetermined criteria the system assesses the risk of each stock based on score (and maybe sentiment) and the risk of the entire portfolio based on total score. (ex. If the majority of the stock in a portfolio scores well during financial analysis then the portfolio has a better risk rating than a portfolio whose stocks score low during financial analysis).

Extensions: None.

5: Sentiment Analysis

User Story:

US 5.1: As a user of the portfolio automation system, I want the system to retrieve relevant news articles for all stocks in the initial portfolio so that the sentiment of the stock can be assessed by the system.

Use Case Name: News Article Retrieval

Related requirements:

- R3.4: Sentiment Analysis

- R5.3: Sentiment Assessment using chatGPT
- R5.4: Sentiment Overview

Goal In Context: The system shall retrieve recent news articles for all stocks in the list.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.
5. The user has prompted the system to create the portfolio.
6. The system has created an initial portfolio based on financial analysis.

Successful End Condition:

- The system successfully retrieves news articles for the initial portfolio.

Failed End Condition:

- The system fails to retrieve recent news articles for the initial portfolio.

Primary Actors:

- Investor: The investor wants to see the relevant news stories so that they can make the final decision to continue with the stock.

Secondary Actors:

- External API: Either a sentiment API or chatGPT via prompt engineering will be used to establish the sentiment of a stock based on recent news stories.
- Database: If the database is populated with metrics the system will fetch data from the database to reduce the amount of API calls.

Main Flow of Events:

1. The system calls the get news stories function.
2. The system retrieves news stories either from the database or from the API
3. If the system retrieved news stories from the API the database is populated with those. News stories.

Extensions: None.

User Story:
US 5.2: As a user of the portfolio analysis system, I want the system to return a sentiment score after the initial portfolio is created so that I can further inform my decision on whether to remove a specific stock from the generated portfolio.

Use Case Name: Sentiment Assessment using AlphaVantage

Related requirements:

- R3.4: Sentiment Analysis
- R5.4: Sentiment Overview

Goal In Context: The system shall assess the sentiment of the articles by calling external APIs.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.
5. The user has prompted the system to create the portfolio.
6. The system has created an initial portfolio based on financial analysis.

Successful End Condition:

- The system retrieves a sentiment score for each stock in the initial portfolio;

Failed End Condition:

- The system fails to retrieve a sentiment score for stocks in the initial portfolio.

Primary Actors:

- Investor: The investor initiates the process for AlphaVantage to perform the sentiment analysis.

Secondary Actors:

- External API: Either a sentiment API or chatGPT via prompt engineering will be used to establish the sentiment of a stock based on recent news stories.

Main Flow of Events:

1. The system calls the get news stories function.

2. The system retrieves news stories either from the database or from the API
3. If the system retrieved news stories from the API the database is populated with those. News stories.
4. For each stock in the initial portfolio, the system calls the sentiment analysis function which utilizes the alphaVantage API to return a sentiment score for each stock in the initial portfolio.
5. Information is stored in the database.

Extensions: None.

User Story:
US 5.3: As a user of the portfolio management system, I want the system to employ prompt engineering via chatGPT so that I can further inform my decision on whether to remove a specific stock from the generated portfolio.

Use Case Name: Sentiment Assessment using chatGPT

Related requirements:
- R3.4: Sentiment Analysis
- R5.1: News Article Retrieval
- R5.4: Sentiment Overview

Goal In Context: The system shall use an NLP API to assess the overall sentiment of a stock based on recent news articles.

Preconditions:
1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.
5. The user has prompted the system to create the portfolio.
6. The system has created an initial portfolio based on financial analysis.

Successful End Condition:
- The system successfully creates a sentiment score for each stock in the initial portfolio.

Failed End Condition:
- The system fails to create a sentiment score for the stocks in the initial portfolio.

Primary Actors:

- Investor: The investor initiates the process for ChatGPT to perform a sentiment analysis.

Secondary Actors:

- External API: Either a sentiment API or chatGPT via prompt engineering will be used to establish the sentiment of a stock based on recent news stories.
- Database: If the database is populated with news stories the system will fetch data from the database to reduce the amount of API calls.

Main Flow of Events:

1. The system calls the get news stories function.
2. The system retrieves news stories either from the database or from the API
3. If the system retrieved news stories from the API the database is populated with those. News stories.
4. For each stock in the initial portfolio, the system calls the sentiment analysis function which utilizes the chatGPT API to run each recent news story associated with the stock through a set of prompts (i.e "Is the overall sentiment of the article negative or positive?", "on a scale of 1-10 how negative or positive?").
5. From this, a sentiment score is deduced along with relevant conclusions surrounding the news of a specific stock.
6. Information is stored in the database.

Extensions: None.

User Story:
US 5.4: As a user of the portfolio automation system, I want to be able to view the sentiment of each news article in the initial portfolio so that I can better understand why the portfolio was created the way it was, and further inform my decision on whether to remove a specific stock from the generated portfolio.

Use Case Name: Sentiment Overview

Related requirements:

- R3.4: Sentiment Analysis
- R5.1: News Article Retrieval
- R5.2: Sentiment Assessment using AphaVantage
- R5.3: Sentiment Assessment using chatGPT

Goal In Context: The system shall provide an overview of its decision criterion.

Preconditions:

1. The system has loaded stocks and their respective metrics.
2. The user has initiated a portfolio creation.
3. The user has selected a timeframe for their portfolio.
4. The system has created an initial dataset with stocks whose earnings dates fall within the selected timeframe.
5. The user has prompted the system to create the portfolio.
6. The system has created an initial portfolio based on financial analysis.
7. The system has returned the sentiment score of each stock provided by alphaVantage or the sentiment score and other relevant conclusions from the chatGPT API.

Successful End Condition:

- The system displays the user with all relevant information regarding sentiment in an efficient and digestible way.

Failed End Condition:

- The system fails to display the user with the sentiment information.

Primary Actors:

- Investor: The investor can view the information regarding the sentiment of each stock in the initial portfolio and is provided with the option to remove specific stocks.

Secondary Actors:

- External API: Either a sentiment API or chatGPT via prompt engineering will be used to establish the sentiment of a stock based on recent news stories.
- Database: If the database is populated with news stories the system will fetch data from the database to reduce the amount of API calls.

Main Flow of Events:

1. The system fetches the sentiment and/or relevant sentimental conclusions from the database.
2. The system updates the UI to show the sentiment of each stock in the initial portfolio.
3. The user is able to view the sentiment score associated with each stock in the initial portfolio and remove stocks if they wish.

Extensions: None.

6: Back-Testing

User Story:

US 6.1: As an administrator of the portfolio automation system, I want to be able to place the system in a historical state so that I can ensure the system is functioning as intended.

Use Case Name: Test Mode

Related requirements:

- R1.1: User Account Creation
- R1.2 User Login

Goal In Context: The system shall be able to be placed in testing mode which allows an arbitrary date to be chosen. This date must be at least as far from the current.

Preconditions:

1. The administrator has created an account
2. The administrator has logged into the system

Successful End Condition:

- The system is placed in a historical, administrative state.

Failed End Condition:

- The system fails to verify the user has the privileges to enter an administrative state.
- The system fails to be placed in a historical, administrative state.

Primary Actors:

- Administrator: The admin is responsible for triggering the system into an administrative state.

Secondary Actors:

- None.

Main Flow of Events:

1. The administrator prompts the system into an administrative state.
2. The system is placed in an administrative state.
3. The administrator prompts the system into a historical state.
4. The system prompts an administrator to choose a date.
5. The system is placed in a historical state.

Extensions: None.

User Story:
US 6.2: As an administrator of the portfolio automation system, I want to be able to use fake money to test the system so that I can verify the system is working without loss of capital.

Use Case Name: Monetary Input

Related requirements:

- R3.1: Timeline Selection
- R7.1: Investment Amount Input

Goal In Context: The system shall allow the user to enter a monetary amount that is for testing purposes only.

Preconditions:

1. The system is in a historical state.
2. The administrator has prompted the system to create a portfolio.
3. The administrator has selected a timeline for the portfolio.

Successful End Condition:

- The system successfully accepts the monetary input and the portfolio automation process begins.

Failed End Condition:

- The system fails to accept monetary input.

Primary Actors:

- Administrator: The administrator is responsible for monetary input.

Secondary Actors:

- None.

Main Flow of Events:

1. The system prompts the user to enter a monetary input for the automated portfolio
2. The user enters a specified amount.

Extensions: None.

User Story:
US 6.3: As an administrator of the portfolio automation system, I want the system to simulate mock portfolios so that I can verify that the system is operating as intended.

Use Case Name: System Simulation

Related requirements:

Goal In Context: The system shall perform as normal but when the user is prompted to invest the investing timeline will be simulated and reported.

Preconditions:

1. The system is in a historical state.
2. The administrator has selected a timeframe.
3. The administrator has created a portfolio.
4. The administrator has entered an amount for the portfolio to trade with.

Successful End Condition:

- The system successfully simulates the portfolio.

Failed End Condition:

- The system fails to simulate the portfolio.

Primary Actors:

- Administrator: The admin is responsible for prompting the system to simulate the portfolio.

Secondary Actors:

- API: The API fetches historical metrics for stocks in the portfolio.

Main Flow of Events:

1. The user prompts the system to simulate the portfolio.
2. The system fetches historical data for the given timeframe.
3. The system simulates the portfolio over the entire timeframe.

Extensions: None.

User Story:
US 6.4: As an administrator of the portfolio automation system, I want the system back-testing to be robust and extensible so that I can test the system in a timely way that conveys validity.

Use Case Name: System Analysis

Related requirements:

Goal In Context: The system shall display relevant metrics (e.g. overall loss/gain, percent loss/gain, average risk rating, average sentiments score).

Preconditions:

1. The system is in a historical state.
2. The administrator has prompted the system to simulate a portfolio.

Successful End Condition:

- The user is displayed with a comprehensive overview of the simulation that took place.

Failed End Condition:

- The system fails to display the simulation overview.

Primary Actors:

- Administrator: The administrator is presented with results in the form of graphs, plots, infographics, etc.

Secondary Actors:

- None.

Main Flow of Events:

1. The system fetches relevant pre-simulation data for the simulated portfolio (i.e. weights, metrics, risk, etc.)
2. The system creates loss/gain metrics.
3. The system displays the overview with the metrics created.

Extensions: None.

7: Portfolio Purchase

User Story:

US 7.1: As a user of the portfolio automation system, I want to specify an amount of money to be invested in the stock, so that I can ensure the portfolio aligns with my investment goals.

Use Case Name: Investment Amount Input

Related requirements:

- R3.1: Timeline Selection
- R3.2: User Filters
- R6.2: Monetary Input

Goal In Context: The system shall allow the user to enter the amount they would like to invest.

Preconditions:

1. The user has prompted the system to create a portfolio.
2. The user has specified a timeframe.
3. The user has filtered by categorical metrics or opted out of filtering.
4. The system has created a portfolio.

Successful End Condition: The system accepts the user's monetary input.

Failed End Condition: The system fails to accept the user's input.

Primary Actors:

- Investor: The investor is responsible for entering the amount of money they would like to invest.

Secondary Actors:

- Database: The database keeps track of available investment funds.

Main Flow of Events:

1. The system prompts the user to enter a monetary value.
2. The user enters an amount of money.
3. The system verifies the money is available.
4. The system accepts the input.

Extensions: None.

User Story:
US 7.2: As a user of the portfolio automation system, I want the system to inform me of the risks of investing prior to investing in a portfolio so that I can make informed and educated financial decisions.

Use Case Name: Investing Disclaimer

Related requirements:

- R7.6: Final Purchase

Goal In Context: The system shall notify users of the risk of investing prior to allowing the user to complete the purchase.

Preconditions:

1. The user has prompted the system to create a portfolio.
2. The user has specified a timeframe.
3. The user has filtered by categorical metrics or opted out of filtering.
4. The system has created a portfolio.
5. The user has entered monetary input.
6. The user has completed portfolio analysis.
7. The user has prompted the system to invest.

Successful End Condition: The user is informed of the risks, liabilities, and consequences of investing and verifies that they understand said risks.

Failed End Condition: The system fails to inform the user of the various risks of investing.

Primary Actors:

- Investor: The investor is displayed the warning.

Secondary Actors:

- None.

Main Flow of Events:

1. The user prompts the system to invest in the portfolio.
2. The system displays the notification.
3. The user notifies the system that they have read and understands the notification.

Extensions: None.

User Story:
US 7.3: As a user of the portfolio automation system, I want to be able to view all information related to my portfolio so that I can stay informed about information related to my investment.

Use Case Name: Portfolio Analysis

Related requirements:

- R2.2: Financial Metrics
- R2.3: Categorical Metrics
- R2.4: News Stories
- R3.3: Metric Analysis
- R3.4: Sentiment Analysis

Goal In Context: The system shall allow the user to view financial metrics, categorical metrics, news stories, sentiment, and risk ratings of each stock within the portfolio.

Preconditions:

1. The user has prompted the system to create a portfolio.
2. The user has specified a timeframe.
3. The user has filtered by categorical metrics or opted out of filtering.
4. The system has created a portfolio.

Successful End Condition: The user can view all information related to the portfolio in an organized and appealing way.

Failed End Condition: The system fails to display information related to the portfolio.

Primary Actors:

- Investor: The investor views the detailed synopsis of the system.

Secondary Actors:

- Database: The database fetches relevant information to be displayed by the system.

Main Flow of Events:

1. The user prompts the system by clicking on the portfolio.
2. The system fetches relevant information for the portfolio.

3. The system expands the portfolio view displaying all relevant information.

Extensions: None.

User Story:
US 7.4: As a user of the portfolio automation system, I want to be able to modify the weights of the generated portfolio or entirely remove stocks from the generated portfolio so that the finalized portfolio can be one that I am comfortable and content with.

Use Case Name: Portfolio Editing

Related requirements:

- R7.3: Portfolio Analysis

Goal In Context: The system shall allow the user to change the weights of the portfolio or remove stocks from the portfolio.

Preconditions:

1. The user has prompted the system to create a portfolio.
2. The user has specified a timeframe.
3. The user has filtered by categorical metrics or opted out of filtering.
4. The system has created a portfolio.
5. The user has prompted the system into the portfolio analysis view.

Successful End Condition: The user successfully modifies the portfolio.

Failed End Condition: The system fails to allow the user to modify the portfolio.

Primary Actors:

- Investor: The investor is responsible for prompting the system to edit or delete a stock from the portfolio.

Secondary Actors:

- None.

Main Flow of Events:

1. The user prompts the system to edit a specific stock from the portfolio analysis view.
2. The system prompts the user with the ability to scale the weight up or down or entirely remove the stock.

3. The user enters the modification.
4. The system finalizes the modification.

Extensions: None.

User Story:
US 7.5: As a user of the portfolio automation system, I want to be able to cancel and delete a portfolio in the process of being created anytime after the creation process was initiated so that I do not have to finalize a portfolio I am unhappy with.

Use Case Name: Portfolio Cancelation/Deletion

Related requirements:

Goal In Context: The system shall allow the user to cancel the creation of a portfolio and restart or exit.

Preconditions:

1. The user has prompted the system to create a portfolio.

Successful End Condition: The system exits the portfolio creation flow to the home page.

Failed End Condition: The system fails to exit the portfolio creation flow.

Primary Actors:

- Investor: The investor is responsible for prompting the system to cancel and delete the portfolio that is in the process of being created.

Secondary Actors:

- None.

Main Flow of Events:

1. The user prompts the system to cancel
2. The system notifies the user that data involved in the process of creating said portfolio will not be saved.
3. The user verifies they understand the consequences of early termination.
4. The system exits the portfolio creation flow to the home page.

Extensions: None.

User Story:
US 7.6: As a user of the portfolio automation system, I want to be able to finalize my purchase of a portfolio so that I know that the portfolio creation process is complete and the specified amount of money has been invested.

Use Case Name: Final Purchase

Related requirements:
- R7.2: Investing Disclaimer

Goal In Context: The system shall allow the user to finalize the creation of their portfolio and invest the specified amount.

Preconditions:
1. The user has prompted the system to create a portfolio.
2. The user has specified a timeframe.
3. The user has filtered by categorical metrics or opted out of filtering.
4. The system has created a portfolio.
5. The user has entered monetary input.
6. The user has completed portfolio analysis.

Successful End Condition: The system successfully invests the specified amount of money into the portfolio.

Failed End Condition: The system fails to invest the user's money.

Primary Actors:
- Investor: The investor is responsible for prompting the system to invest after they complete portfolio analysis.

Secondary Actors:
- None

Main Flow of Events:
1. The user prompts the system to invest in the portfolio.
2. The user completes the investing disclaimer.
3. The system invests in the portfolio.

Extensions: None.

8: Portfolio Management

User Story:

US 8.1: As a user of the portfolio automation system, I want to be able to visualize the market and my portfolios in real time from the home page. So that I can understand what is happening in the market.

Use Case Name: Realtime Overview

Related requirements:

- 8.2 Historical Overview

Goal In Context: The system shall track the movement of markets in real time showing real-time gains and losses of assets held in the app.

Preconditions:

1. The user is logged into the system.
2. The system has loaded the homepage.

Successful End Condition: If the user has no portfolios the system displays the S&P500 in real time, otherwise the plot displays real-time movement of the aggregation of the user's portfolio.

Failed End Condition: The system fails to visualize data.

Primary Actors:

- Investor: The investor views the data.

Secondary Actors:

- Database: The system fetches relevant stock information from the database.
- API: Any information not found in the database will be fetched via API.

Main Flow of Events:

1. On load, the system fetches market data and displays 1 day stock information by default
2. The user can prompt the system to display plots for a specific portfolio.

Extensions: None.

User Story:
US 8.2: As a user of the portfolio automation system, I want to not only be able to view the live data but also the historical data associated with my account so that I can see how my account has performed over time.

Use Case Name: Historical Overview

Related requirements:

- 8.1: Realtime Overview

Goal In Context: The system shall allow a user to see the historical performance of personal assets they have invested within the system.

Preconditions:

1. The user has logged into the system.
2. The system has loaded the homepage.
3. On load, the system fetches market data and displays 1-day stock information by default.

Successful End Condition: The system displays the historical overview requested by the user.

Failed End Condition: The system fails to display the historical view.

Primary Actors:

- Investor: The investor is responsible for prompting the specific view from the system.

Secondary Actors:

- Database: The system fetches relevant stock information from the database.
- API: Any information not found in the database will be fetched via API.

Main Flow of Events:

1. The user can prompt the system to display plots for (1-day, 1-month, 1-week, 1-month, 1-quarter, year-to-date, and 1-year).
2. The user can prompt the system to display plots for a specific portfolio.

Extensions: None.

User Story:
US 8.3: As a user of the portfolio automation system, I want to view all currently active portfolios comprehensively and aesthetically so that I can understand what stocks my account consists of and view each portfolio in greater detail.

Use Case Name: View Current Portfolios

Related requirements:

- R7.3: Portfolio Analysis

Goal In Context: The system shall allow the user to view currently active portfolios this includes financial metrics, categorical metrics, news stories, sentiment, and risk rating of each stock within the portfolio

Preconditions:

1. The user has logged onto the system.
2. The system has loaded the homepage.

Successful End Condition: The user is displayed all currently active portfolios.

Failed End Condition: The system fails to display currently active portfolios.

Primary Actors:

- Investor: The investor views the currently active portfolios.

Secondary Actors:

- Database: The system fetches relevant stock information from the database.
- API: Any information not found in the database will be fetched via API.

Main Flow of Events:

1. The system loads the homepages
2. The system fetches current stock information
3. The system displays current stock information.

Extensions: None.

User Story:
US 8.4: As a user of the portfolio automation system, I want to be able to view previous portfolios I have invested in so that I can reference my trading history to enhance my knowledge and decision-making processes.

Use Case Name: View Previous Portfolios

Related requirements:

- R8.3: View Current Portfolios

Goal In Context: The system shall allow the user to view previously active portfolios this includes financial metrics, categorical metrics, news stories, sentiment, and risk rating of each stock within the portfolio.

Preconditions:

1. The user has logged into the system

Successful End Condition: The user accesses the portfolio page and prompts the system to show all inactive portfolios.

Failed End Condition: The system fails to access the portfolio page or fails to fetch inactive portfolios.

Primary Actors:

- Investor: the investor is responsible for prompting the system to show inactive portfolios.

Secondary Actors:

- Database: The system fetches relevant stock information from the database.
- API: Any information not found in the database will be fetched via API.

Main Flow of Events:

1. The user prompts the system to navigate to the portfolios page.
2. The system navigates to the portfolios page and displays the active portfolios that are also displayed on the homepage.
3. The user toggles the system to either view all or view inactive portfolios.
4. The system fetches data related to inactive portfolios.
5. The system displays inactive portfolios.

Extensions: None.

User Story:
US 8.5: As a user of the portfolio automation system, I want to be able to terminate an active portfolio before its initial timeline so that I can maximize the control I have over my finances.

Use Case Name: Delete Portfolio.

Related requirements:

- R8.3: View Current Portfolios

Goal In Context: The system shall allow the user to terminate the portfolio prior to the completion of its timeline

Preconditions:

1. The user has logged into the system
2. The system has one or more active portfolios.

Successful End Condition: The system successfully terminates an active portfolio.

Failed End Condition: The system fails to terminate the specified active portfolio.

Primary Actors:

- Investor: the investor is responsible for prompting the system to terminate an active portfolio.

Secondary Actors:

- None.

Main Flow of Events:

1. The user accesses portfolio analysis from the home page or prompts the system to navigate to the portfolios page.
2. The system navigates to the portfolios page and displays the active portfolios.
3. The user prompts the system to terminate an active portfolio.
4. The system terminates the portfolio and removes it from the list of active portfolios.

Extensions: None.

Non-Functional Requirements

1. **Usability**
    - The system shall be intuitive and easy to use for users with minimal knowledge of financial markets and stock trading.
    - The system shall prioritize clean, informative and navigable interfaces.
2. **Reliability**
    - The system shall be up {percent} of the time to ensure users can access their portfolios without disruption.
    - The system shall back up user portfolios and transactions to prevent data loss.
3. **Scalability**
    - The system shall be able to increase user numbers without degradation of performance.
    - The system shall be extensible allowing for future integrations.
4. **Security**
    - The system shall use secure protocols to encrypt all data transfers between users and the server.
    - The system shall encrypt user passwords using industry-standard encryption.
5. **Maintainability**
    - The system shall follow software engineering design principles.
    - The system shall be well documented.
    - The system shall incorporate automated testing to verify functionality.
6. **Compliance**
    - The system shall comply with industry-standard regulations for data privacy.

Performance Requirements

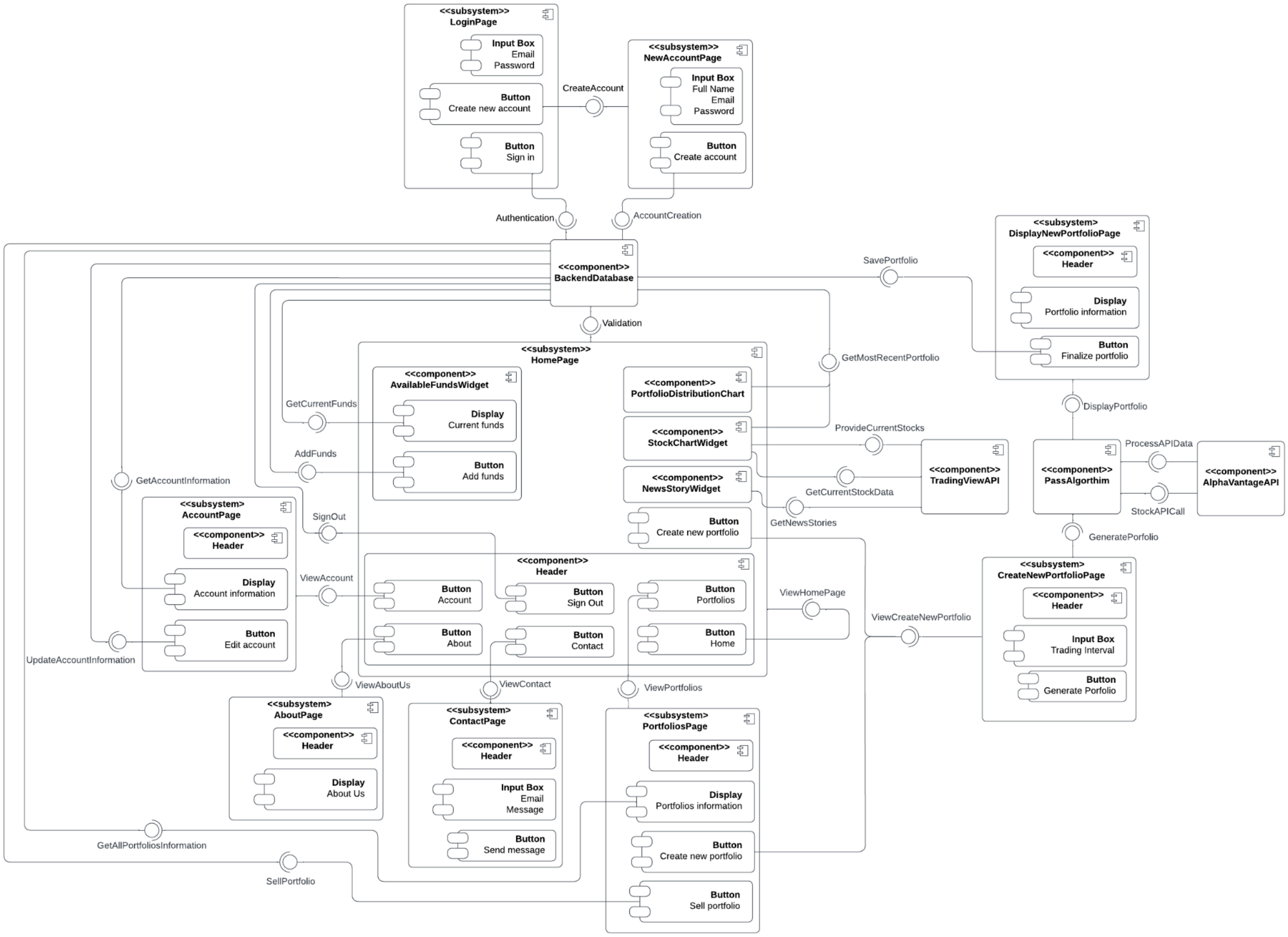
1. **Response Time**
    - The system shall retrieve stock data within 3 seconds of a user query.
    - The system shall process portfolio creation within 10 seconds of creation prompt.
    - The system shall complete authentication operations (e.g. login, account creation, fund verification) within 2 seconds.
2. **Capacity**
    - The system shall support up to 1000 registered users.
    - The system shall support up to 1000 active users.
    - The system shall be capable of storing up to 10,000 portfolios.
    - The system shall be capable of storing up to 100,000 transactions.

3. **Availability**

- The system shall be available 100% of the time during market hours.
- The system shall be available 98% of the time.
- The system shall implement redundancy for critical components to minimize downtime in case of failure.

4. **Scalability**
    - The system shall scale server-side resources based on load to ensure consistent response times under heavy usage.
5. **Data Consistency**
    - The system shall execute all database transactions in a fully ACID-compliant manner to ensure data integrity.
    - The system shall ensure that, in the event of system failure data is consistent across all systems upon recovery.
6. **Fault Tolerance**
    - The system shall retry failed API calls up to 3 times before displaying an error message.
    - The system shall notify users that real-time data is unavailable but still be accessible in case of partial failure (i.e. API failure).

Interface Requirements



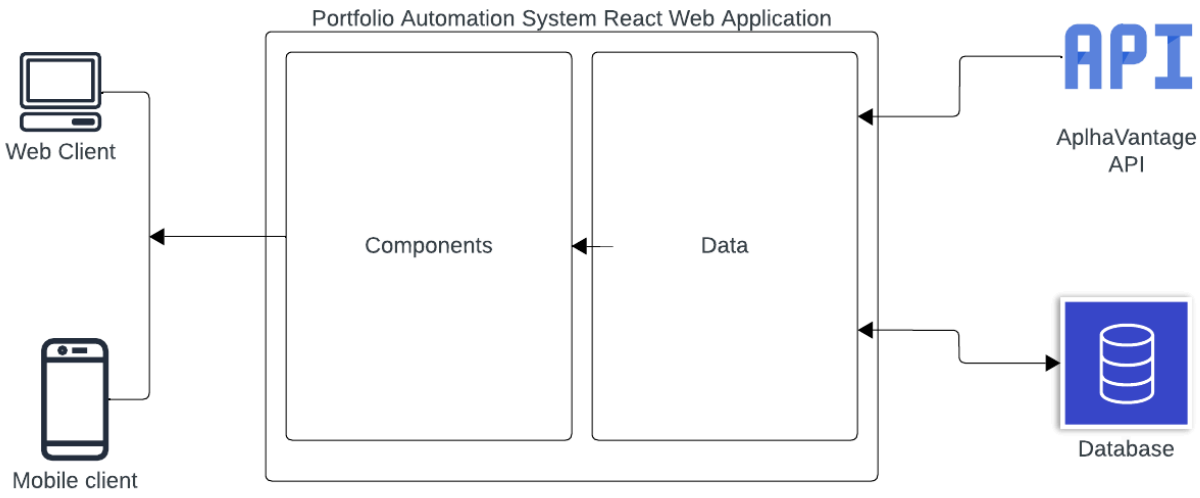
1. The system shall contain a LoginPage
    1. The system shall display an input box for the user to enter their email and password on the LoginPage.
    2. The system shall display a “Create new account” button on the LoginPage that navigates to the NewAccountPage.
    3. The system shall display a “Sign in” button that submits the user’s information for authentication with the BackendDatabase.
2. The system shall contain a NewAccountPage
    1. The system shall display an input box for the user to enter their full name, email, and password on the NewAccountPage.
    2. The system shall display a “Create account” button that submits the users information for account creation in the BackendDatabase.
3. The system shall contain a HomePage
    1. The system shall display a header on the HomePage.
        1. The system shall display a “Portfolios” button in the header that navigates to the AllPortfoliosPage.
        2. Thes system shall display a “Home” button in the header that navigates to the HomePage.
        3. The system shall display a “About” button in the header that navigates to the AboutPage.
        4. The system shall display a “Contact” button in the header that navigates to the ContactPage.
    2. The system shall display information retrieved from the BackendDatabase in the CurrentPortfolioWidget on the HomePage.
        1. The system shall display a line graph demonstrating the performance of the current portfolio in the CurrentPortfolioWidget.
        2. The system shall display a bar diagram demonstrating the diversity of the portfolio in the CurrentPortfolioWidget.
        3. The system shall display a table detailing relevant stock information in the CurrentPortfolioWidget.
    3. The system shall display information retrieved from the BackendDatabase in the HomePageWidget on the HomePage.
        1. The system shall display the total dollar amount of current investments in the HomePageWidget.
        2. The system shall display the date the latest portfolio was generated in the HomePageWidget.
    4. The system shall display a NewStoryWidget with recent articles fetched via the AlphaVantageAPI on the HomePage.
    5. The system shall display a “Create new portfolio” button that navigates to the NewPortfolioPage on the HomePage.

4. The system shall contain a NewPortfolioPage
    a. The system shall display a header on the NewPorfolioPage.
    b. The system shall display an input box fro the user to enter their desired investment dollar amount and desired investment duration on the NewPortfolioPage.
    c. The system shall display a “Generate portfolio” button that submits users input to the PassAlgorthim on the NewPortfolioPage.
5. The system shall contain a PortfolioDisplayPage
    a. The system shall display a header on the PortfolioDisplayPage.
    b. The system shall display a bar diagram demonstrating the diversity of the portfolio generated by the PassAlgorthim on the PortfolioDisplayPage.
    c. The system shall display a table detailing relevant stock information of the portfolio generated by the PassAlgorthim on the PortfolioDisplayPage.
    d. The system shall display a “Modify portfolio” button that navigates to the PortfolioEditingPage on the PortfolioDisplayPage
    e. The system shall display a “Accept portfolio” button that submits the portfolio information for storage in the BackendDatabase on the PortfolioDisplayPage.
6. The system shall contain a PortfolioEditingPage
    a. The system shall display a header on the PortfolioEditingPage.
    b. The system shall display a “Remove stock” button to allow users to remove stocks from the portfolio on the PortfolioEditingPage.
    c. Thes system shall display a “Add stock” button to allow users to add new stocks to the portfolio on the PortfolioEditingPage.
7. The system shall contain an AllPortfoliosPage
    a. The system shall display a header on the AllPortfoliosPage
    b. The system shall display information about all portfolios retrieved from the BackendDatabase on the AllPortfoliosPage.
8. The system shall contain an AboutPage
    a. The system shall display a header on the AboutPage.
    b. The system shall display content relevant to the history and mission of the company on the AboutPage.
9. The system shall contain a ContactPage
    a. The system shall display a header on the ContactPage.
    b. The system shall display content relevant to contacting the company on the ContactPage.

Development Standards

For this project, we will be using React.js to complete the front end development of the project and create a user interface that the user will be able to interact with. The front end will work with the back end of the project which will most likely be python. For a news API that we will use to grab the data to populate the website, we will use AlphaVantage which has 25 free uses a day. However, since we are going to use the observer pattern we will most likely need the premium version for 75 API requests per minute at $50 a month. This ensures that the data we get can be updated accordingly and investors get the updated knowledge they need to make their decisions. To perform our own sentiment analysis we will use VADER which is a python extension and for a second sentiment analysis, we would be planning to use OpenAI. However, the pricing depends on how many API calls are made with a key which could change depending on how much testing and usage there is in a month. Finally, our data storage will be kept in a database using MySQL with the potential of needing Redis if MySQL doesn't meet the needs that we would need it to.

Architectural Design



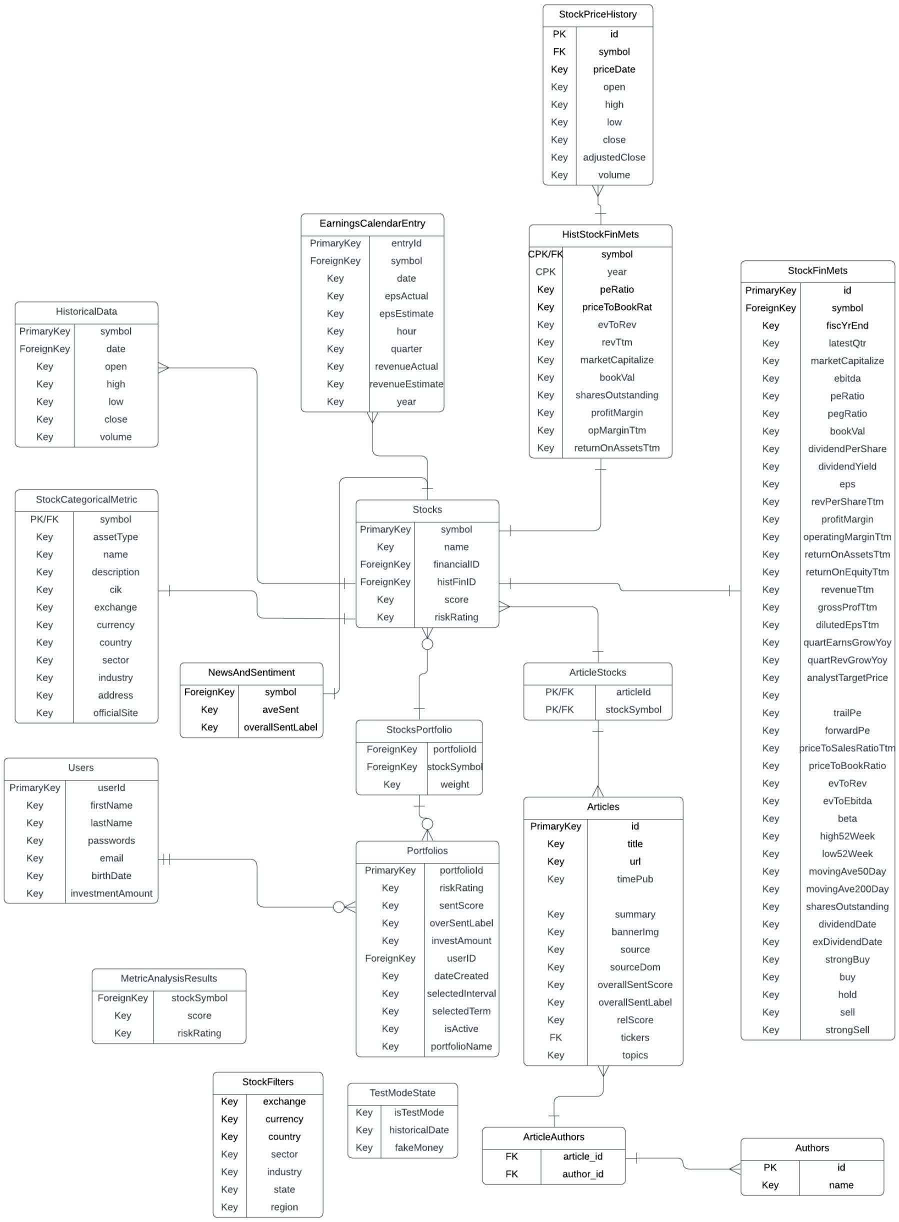
UML Diagrams

Class Diagram

*Note: PDF must be downloaded to view

https://drive.google.com/file/d/1avpMNx7MB2umsnRF4ho6OOT_T3cKeINm/view?usp=share_link

ER Diagram



StockPriceHistory	
PK	id
FK	symbol
Key	priceDate
Key	open
Key	high
Key	low
Key	close
Key	adjustedClose
Key	volume

EarningsCalendarEntry	
PrimaryKey	entryId
ForeignKey	symbol
Key	date
Key	epsActual
Key	epsEstimate
Key	hour
Key	quarter
Key	revenueActual
Key	revenueEstimate
Key	year

HistStockFinMets	
CPK/FK	symbol
CPK	year
Key	peRatio
Key	priceToBookRat
Key	evToRev
Key	revTtm
Key	marketCapitalize
Key	bookVal
Key	sharesOutstanding
Key	profitMargin
Key	opMarginTtm
Key	returnOnAssetsTtm

StockFinMets	
PrimaryKey	id
ForeignKey	symbol
Key	fiscYrEnd
Key	latestQtr
Key	marketCapitalize
Key	ebitda
Key	peRatio
Key	pegRatio
Key	bookVal
Key	dividendPerShare
Key	dividendYield
Key	eps
Key	revPerShareTtm
Key	profitMargin
Key	operatingMarginTtm
Key	returnOnAssetsTtm
Key	returnOnEquityTtm
Key	revenueTtm
Key	grossProfTtm
Key	dilutedEpsTtm
Key	quartEarnsGrowYoy
Key	quartRevGrowYoy
Key	analystTargetPrice
Key	
Key	trailPe
Key	forwardPe
Key	priceToSalesRatioTtm
Key	priceToBookRatio
Key	evToRev
Key	evToEbitda
Key	beta
Key	high52Week
Key	low52Week
Key	movingAve50Day
Key	movingAve200Day
Key	sharesOutstanding
Key	dividendDate
Key	exDividendDate
Key	strongBuy
Key	buy
Key	hold
Key	sell
Key	strongSell

HistoricalData	
PrimaryKey	symbol
ForeignKey	date
Key	open
Key	high
Key	low
Key	close
Key	volume

StockCategoricalMetric	
PK/FK	symbol
Key	assetType
Key	name
Key	description
Key	cik
Key	exchange
Key	currency
Key	country
Key	sector
Key	industry
Key	address
Key	officialSite

Stocks	
PrimaryKey	symbol
Key	name
ForeignKey	financialID
ForeignKey	histFinID
Key	score
Key	riskRating

NewsAndSentiment	
ForeignKey	symbol
Key	aveSent
Key	overallSentLabel

ArticleStocks	
PK/FK	articleId
PK/FK	stockSymbol

StocksPortfolio	
ForeignKey	portfolioId
ForeignKey	stockSymbol
Key	weight

Users	
PrimaryKey	userId
Key	firstName
Key	lastName
Key	passwords
Key	email
Key	birthDate
Key	investmentAmount

Portfolios	
PrimaryKey	portfolioId
Key	riskRating
Key	sentScore
Key	overSentLabel
Key	investAmount
ForeignKey	userID
Key	dateCreated
Key	selectedInterval
Key	selectedTerm
Key	isActive
Key	portfolioName

Articles	
PrimaryKey	id
Key	title
Key	url
Key	timePub
Key	summary
Key	bannerImg
Key	source
Key	sourceDom
Key	overallSentScore
Key	overallSentLabel
Key	relScore
FK	tickers
Key	topics

MetricAnalysisResults	
ForeignKey	stockSymbol
Key	score
Key	riskRating

StockFilters	
Key	exchange
Key	currency
Key	country
Key	sector
Key	industry
Key	state
Key	region

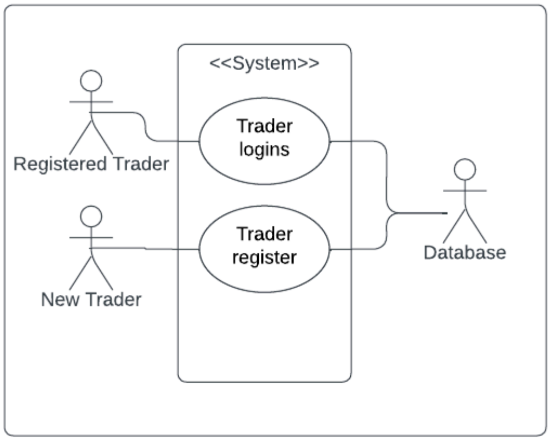
TestModeState	
Key	isTestMode
Key	historicalDate
Key	fakeMoney

ArticleAuthors	
FK	article_id
FK	author_id

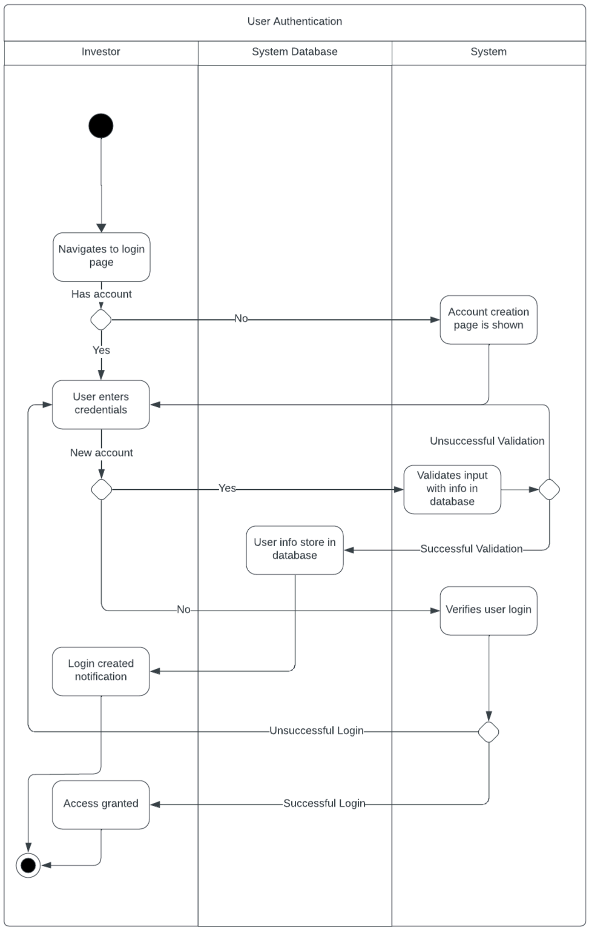
Authors	
PK	id
Key	name

R1: User Authentication

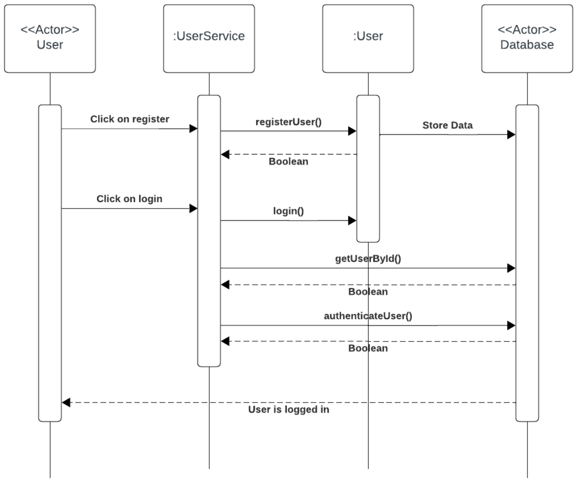
Use Case Diagram



Activity Diagram

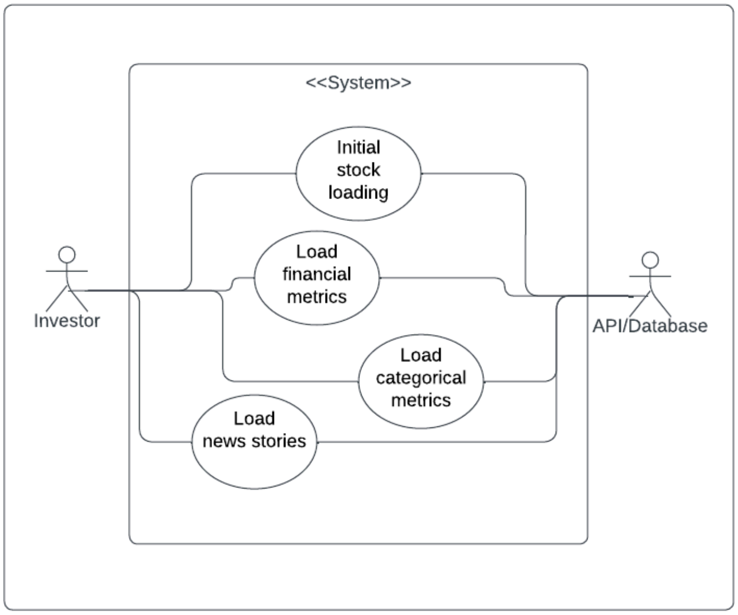


Sequence Diagram

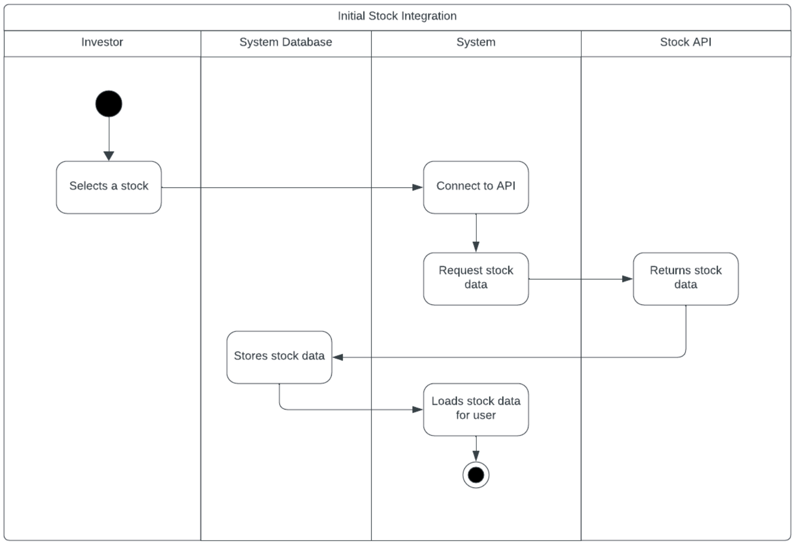


R2: Initial Stock Integration

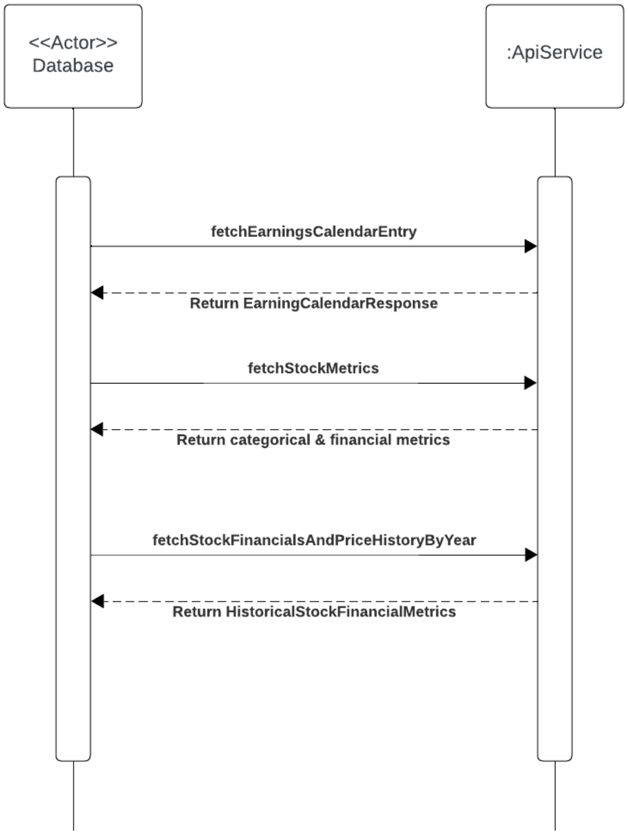
Use Case Diagram



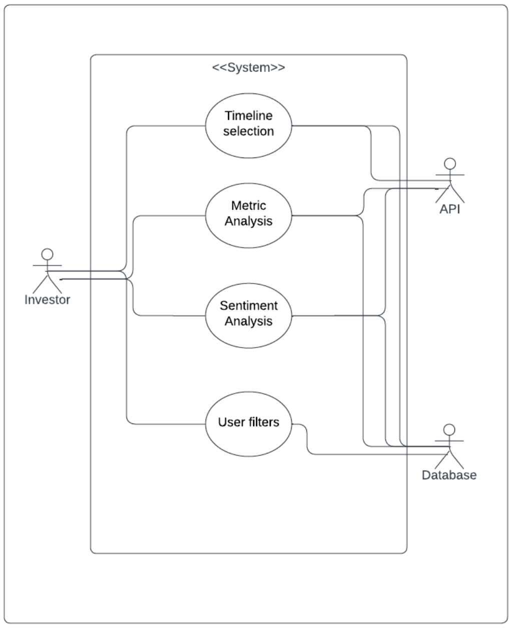
Activity Diagram



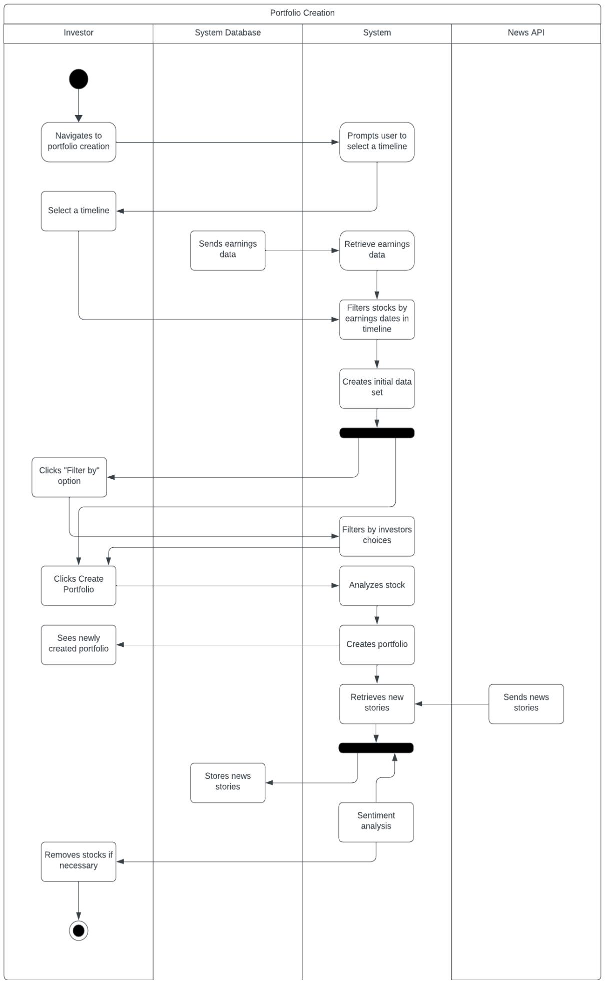
Sequence Diagram


<<Actor>> Database
:ApiService
fetchEarningsCalendarEntry
Return EarningCalendarResponse
fetchStockMetrics
Return categorical & financial metrics
fetchStockFinancialsAndPriceHistoryByYear
Return HistoricalStockFinancialMetrics

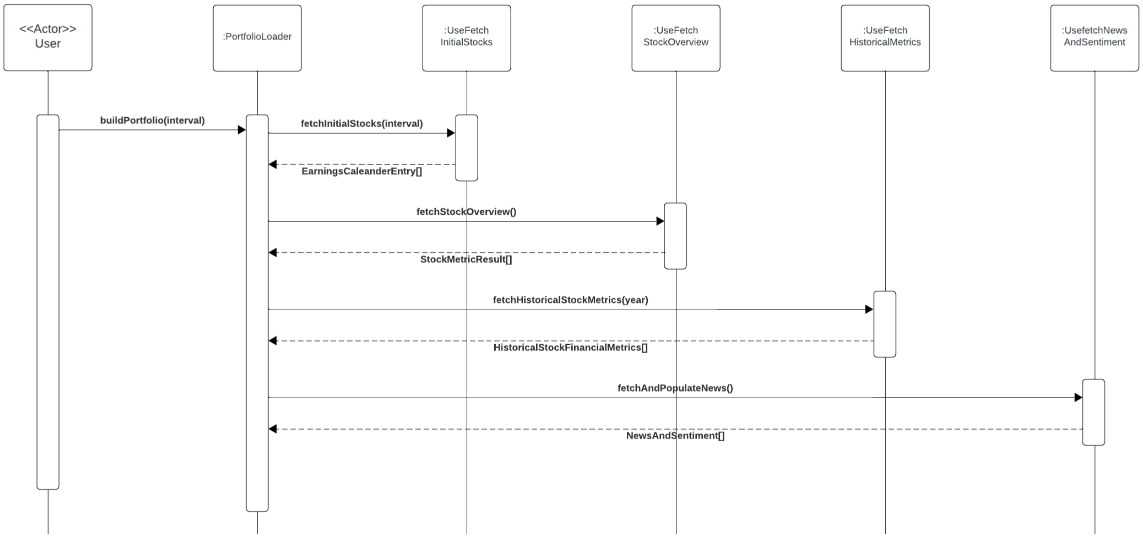
R3: Portfolio Creation
Use Case Diagram



Activity Diagram

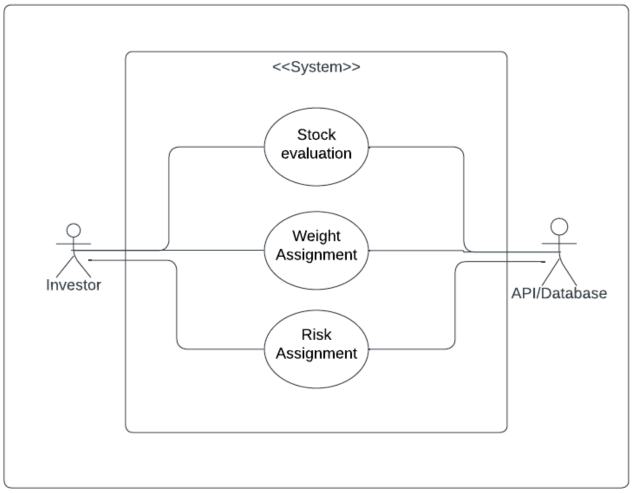


Sequence Diagram

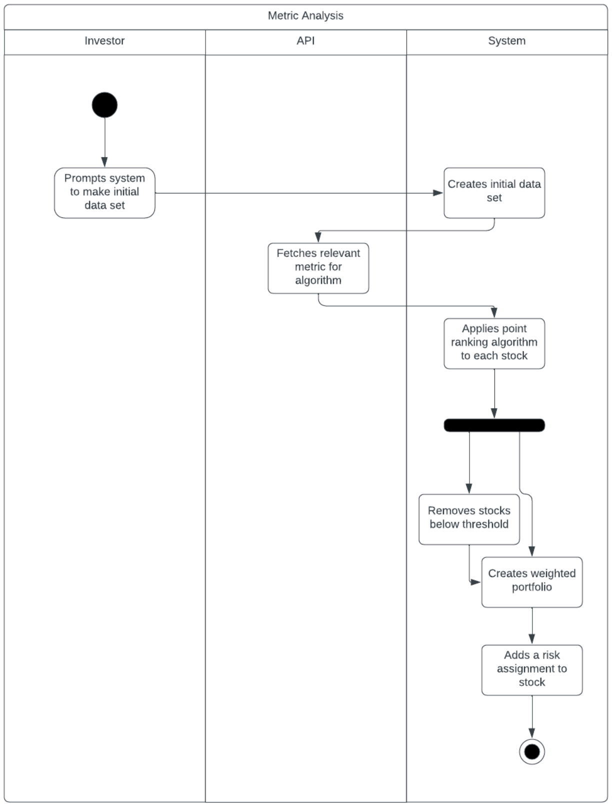


R4: Metric Analysis

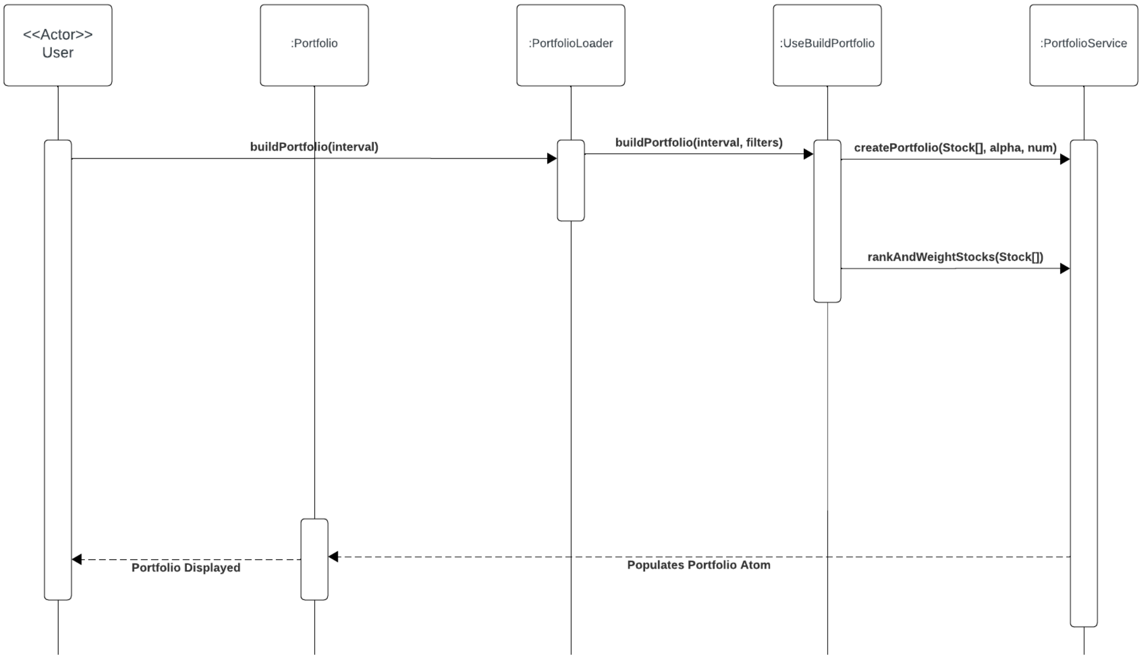
Use Case Diagram



Activity Diagram

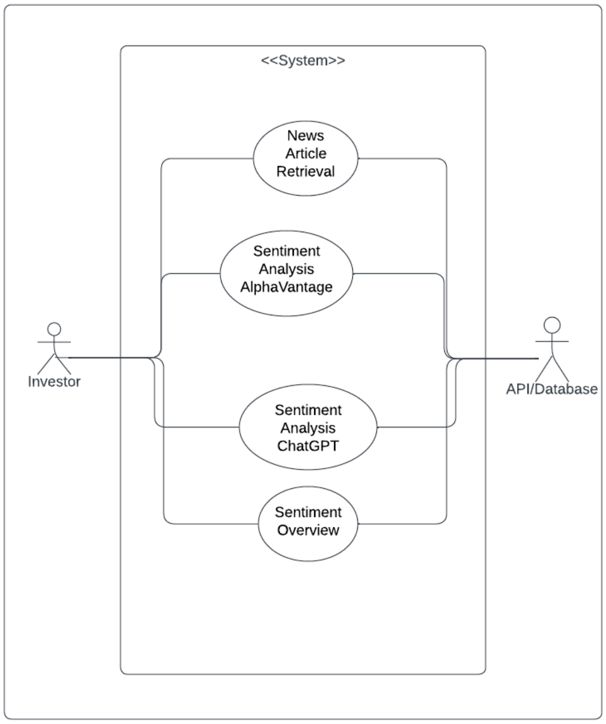

Metric Analysis
Investor
API
System
Prompts system to make initial data set
Creates initial data set
Fetches relevant metric for algorithm
Applies point ranking algorithm to each stock
Removes stocks below threshold
Creates weighted portfolio
Adds a risk assignment to stock

Sequence Diagram

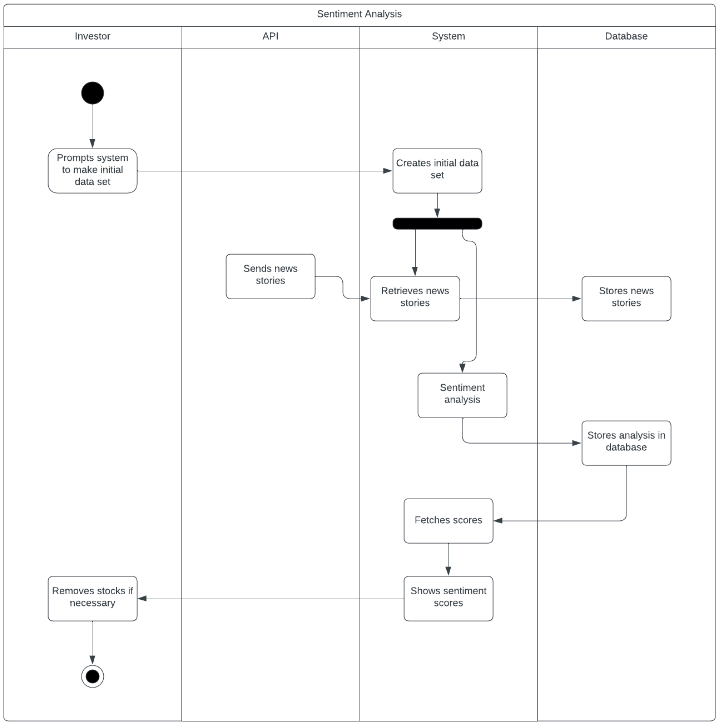


R5: Sentiment Analysis

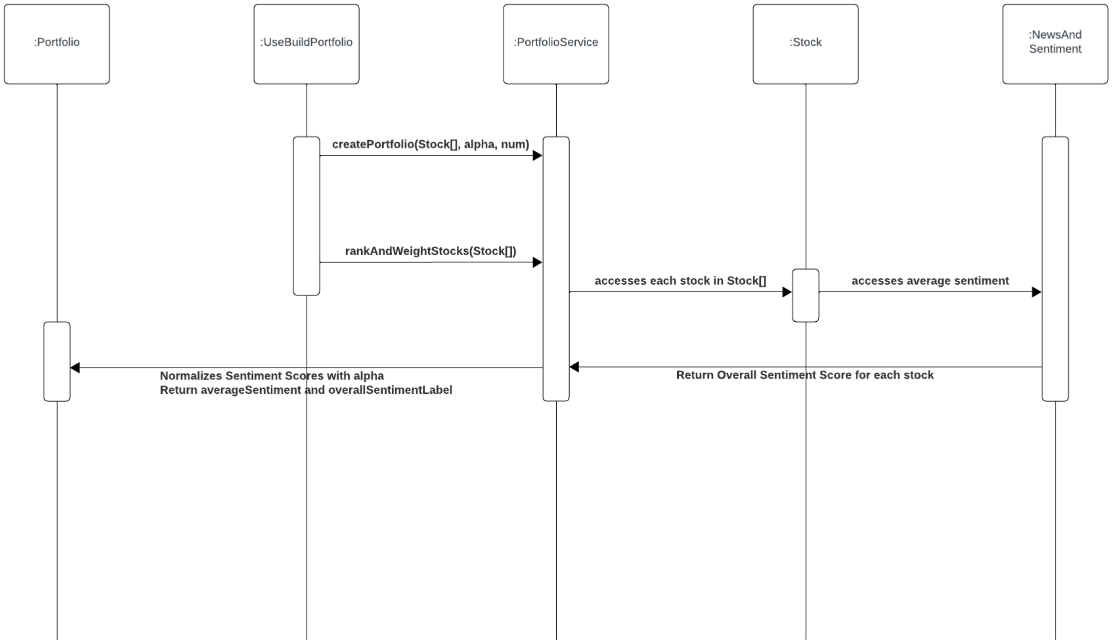
Use Case Diagram



Activity Diagram

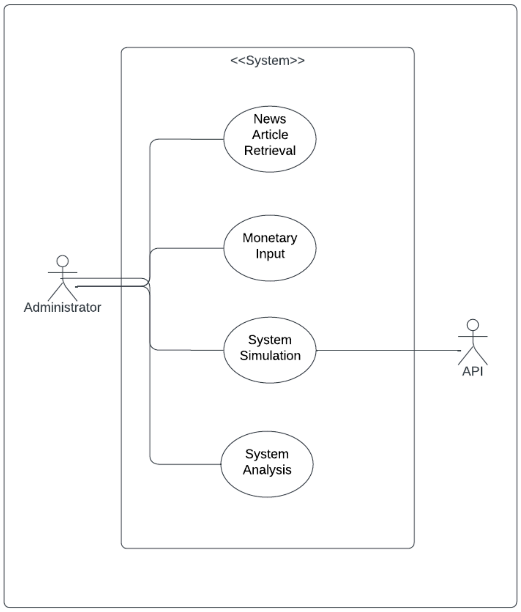


Sequence Diagram

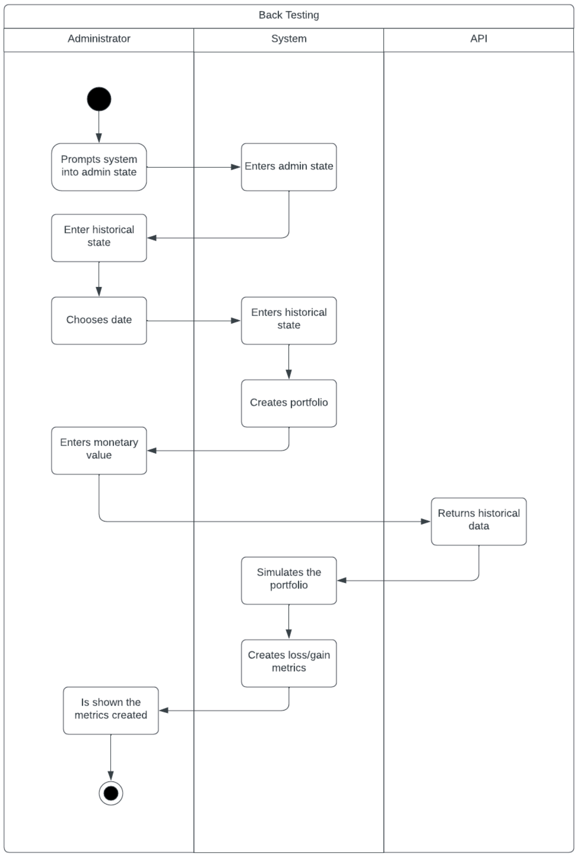

:Portfolio
:UseBuildPortfolio
:PortfolioService
:Stock
:NewsAnd Sentiment
createPortfolio(Stock[], alpha, num)
rankAndWeightStocks(Stock[])
accesses each stock in Stock[]
accesses average sentiment
Normalizes Sentiment Scores with alpha
Return averageSentiment and overallSentimentLabel
Return Overall Sentiment Score for each stock

R6: Back-Testing

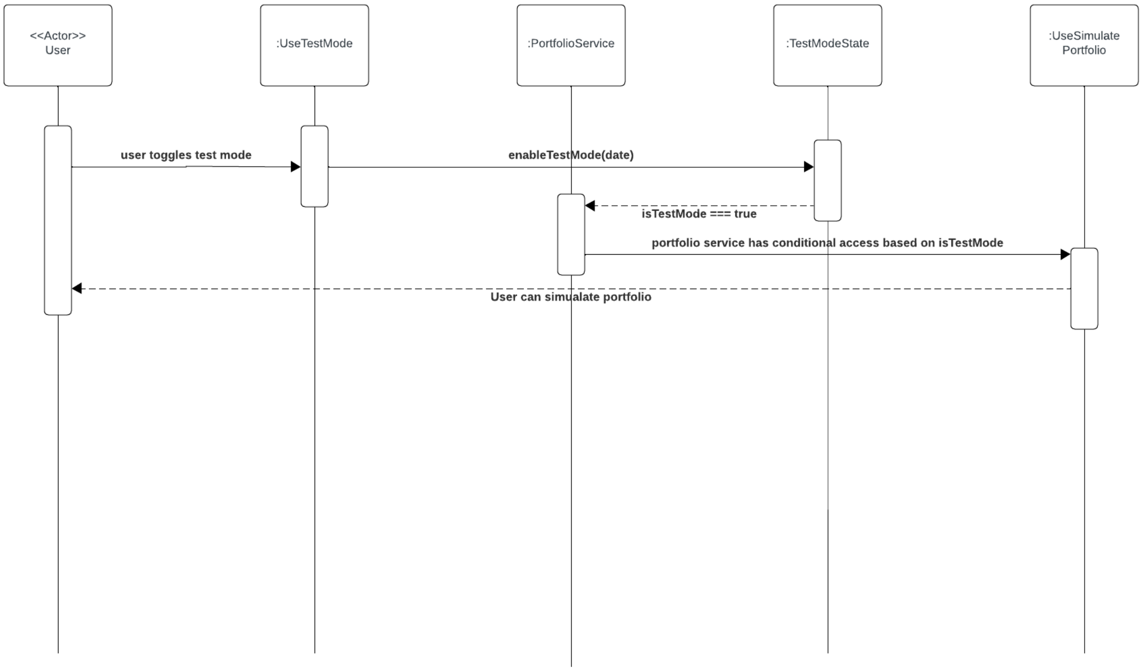
Use Case Diagram



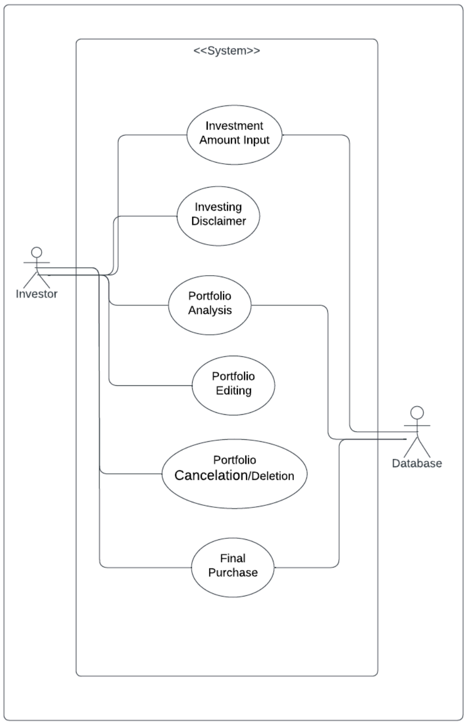
Activity Diagram



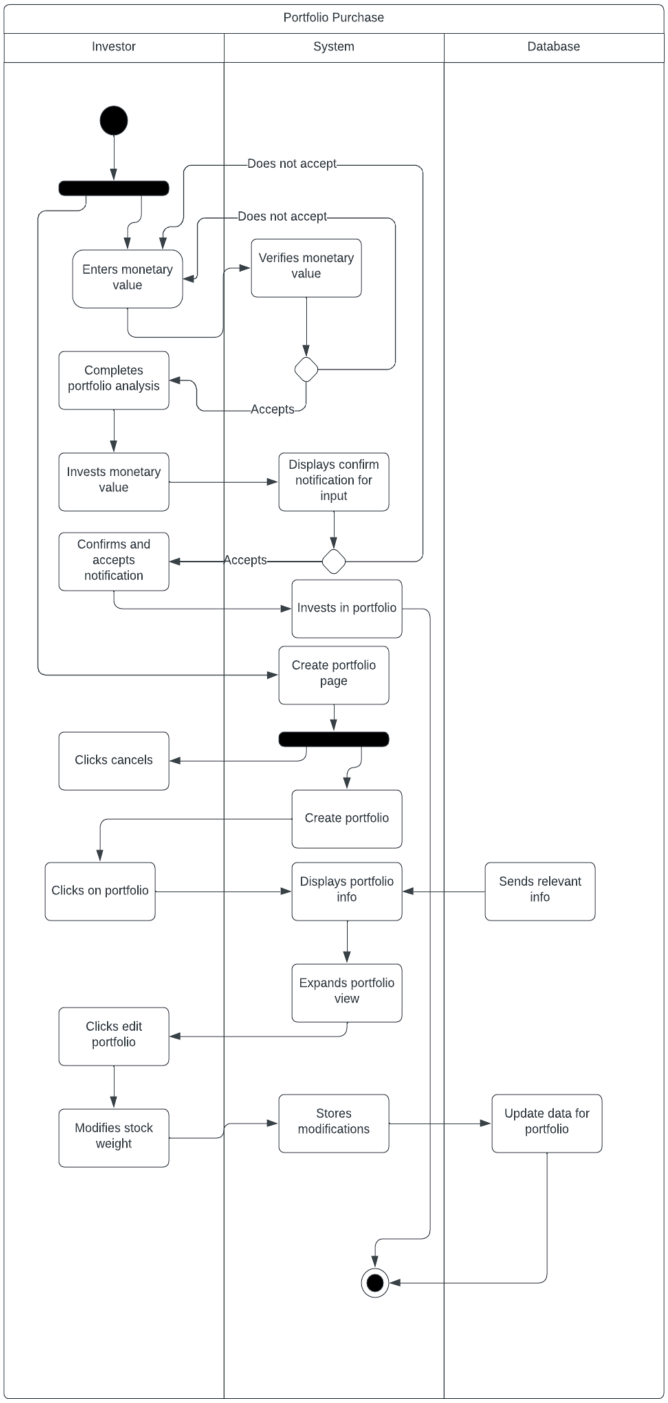
Sequence Diagram



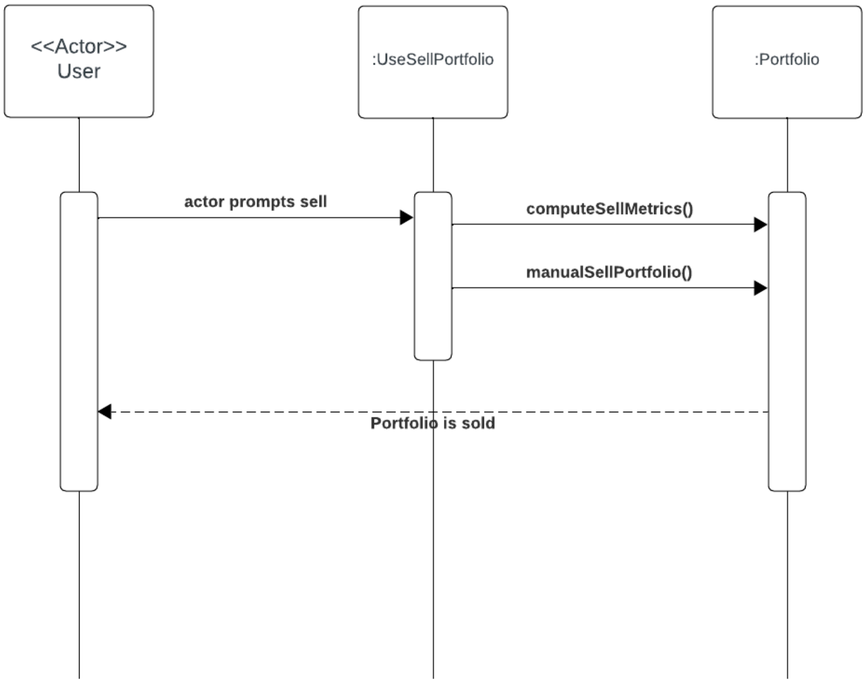
R7: Portfolio Purchase Use Case Diagram



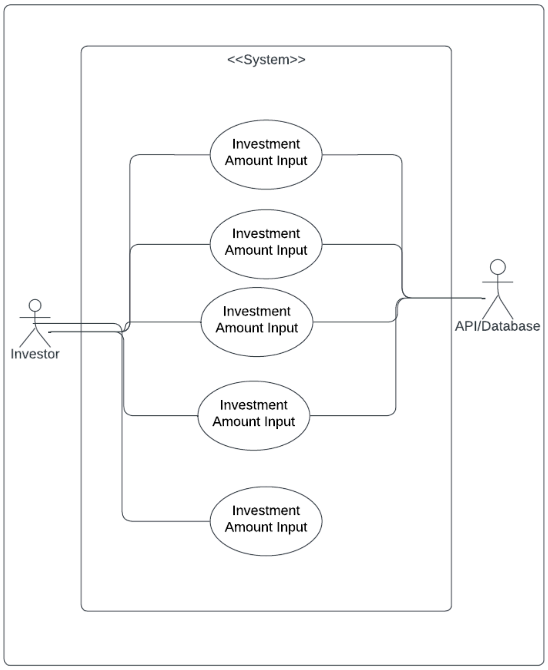
Activity Diagram



Sequence Diagram

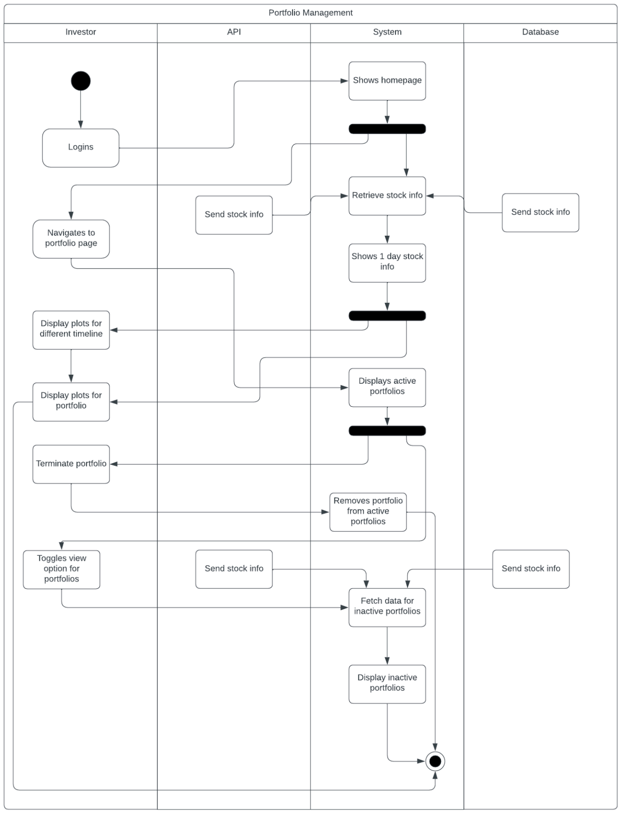


R8: Portfolio Management
Use Case Diagram

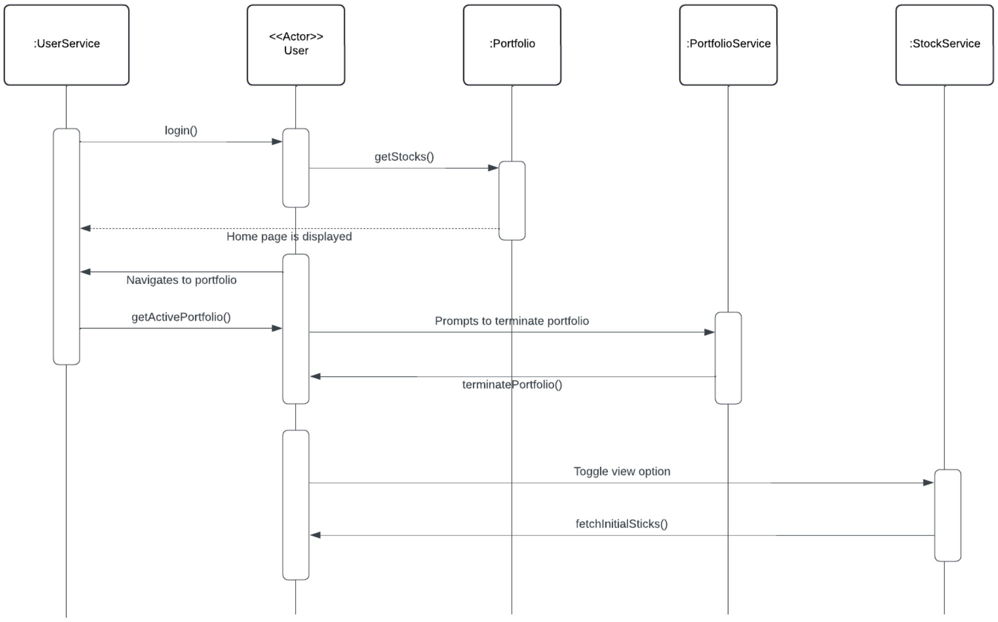


Activity Diagram

Functional Requirements 8


Portfolio Management
Investor
API
System
Database
Shows homepage
Logins
Retrieve stock info
Send stock info
Send stock info
Navigates to portfolio page
Shows 1 day stock info
Display plots for different timeline
Displays active portfolios
Display plots for portfolio
Terminate portfolio
Removes portfolio from active portfolios
Toggles view option for portfolios
Send stock info
Send stock info
Fetch data for inactive portfolios
Display inactive portfolios

Sequence Diagram



Design Patterns

We will be using two design patterns as our main patterns. Firstly, we will use the model view controller (MVC) pattern because our product involves a web interface that investors will be interacting with. The MVC pattern will contain the three parts symbolized in the name, which will handle the different parts of the website itself. The model handles the data that the website will obtain from the API, the view, which would be what the user sees, and the controller, which would connect the model and view elements. As we develop, we can work on the different parts separately, and this pattern would make adding new features later down the road much easier. The second pattern we will be using is the observer pattern. We need to use the observer since we are working with stocks that are being updated at all times. The website can monitor information and update the information that needs to be updated as the information changes.

Trade-Offs

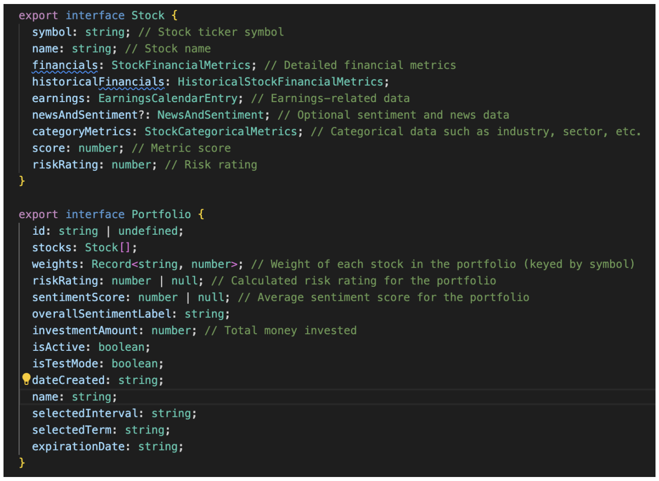
For this project, we could have gone a variety of ways since we are working with a website product. We could have used patterns such as the adapter pattern, the strategy pattern, or the command pattern. We chose the two patterns we did for ease of creating a web application that heavily handles data and data that updates on a very regular basis. One of the tradeoffs to using the MVC pattern is that because the pattern would break up the code, which could easily become a buildup of files, and the code could increase in complexity. In addition, one drawback to using the observer pattern is that the website could have performance issues and run at a slower speed with the multitude of data.

If we had wanted to use multiple APIs to pull multiple sources of information, we could have used the adapter pattern, which could have adapted to the different API formatting that could be found. Instead, because we are only going to use one, we chose not to use this pattern. It also would've caused problems if we had used both the observer and adapter pattern because there could've been too many API calls going to the API trying to get the updated data as the observer detects a change. The problem with the strategy pattern is that it is useful for multiple algorithms. While there are many parts to our website, there will most likely only be one main algorithm that will compute our version of a sentiment analysis, while the sentiment analysis will most likely come from either AlphaVantage or ChatGPT. The other portions of the code will help with user interaction with one algorithm to compute its own value.

Expected Results

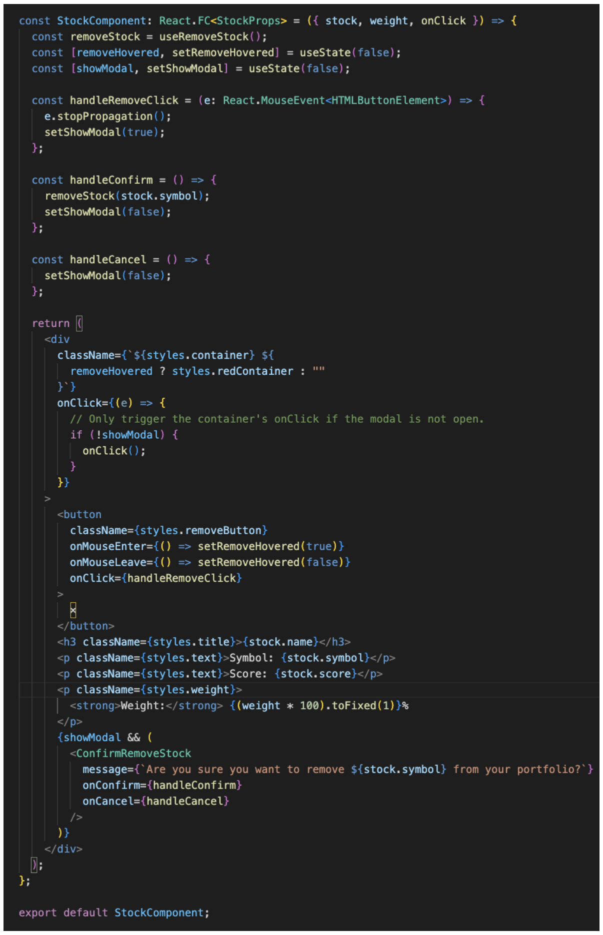
This project proposal outlines an optimistic implementation of the portfolio automation system. Based on various constraints such as timeline, resources, scope, and compliance our expected results will be limited. The project's primary goal is to implement a working prototype; In order to accomplish this goal in the given timeline, the scope of this project limits our implementation to only the necessary requirements needed to accomplish this goal. Furthermore, compliance limits our implementation to requirement 6, back-testing and no real money will be integrated into the system.

Appendix A. Code



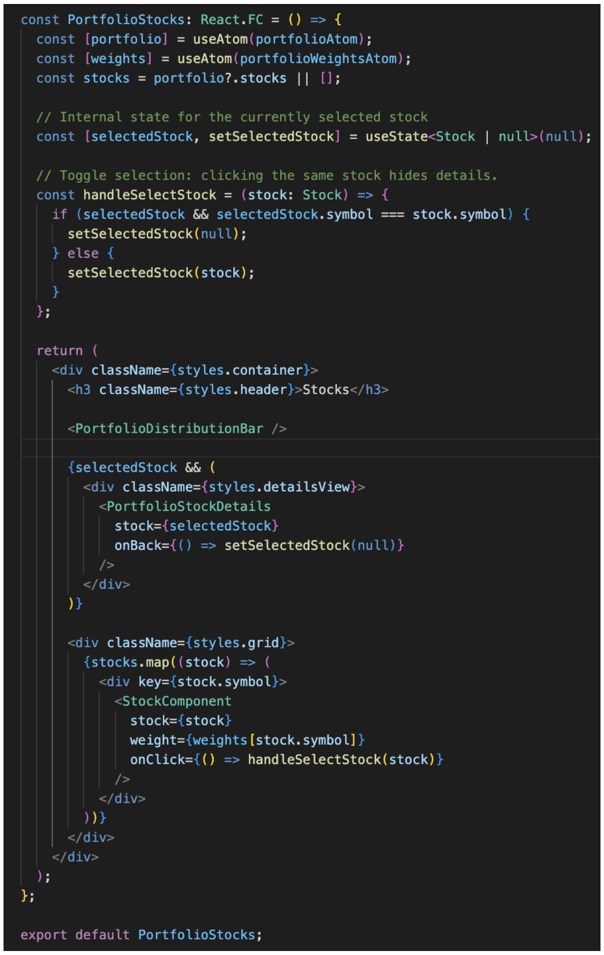
```typescript
export interface Stock {
  symbol: string; // Stock ticker symbol
  name: string; // Stock name
  financials: StockFinancialMetrics; // Detailed financial metrics
  historicalFinancials: HistoricalStockFinancialMetrics;
  earnings: EarningsCalendarEntry; // Earnings-related data
  newsAndSentiment?: NewsAndSentiment; // Optional sentiment and news data
  categoryMetrics: StockCategoricalMetrics; // Categorical data such as industry, sector, etc.
  score: number; // Metric score
  riskRating: number; // Risk rating
}

export interface Portfolio {
  id: string | undefined;
  stocks: Stock[];
  weights: Record<string, number>; // Weight of each stock in the portfolio (keyed by symbol)
  riskRating: number | null; // Calculated risk rating for the portfolio
  sentimentScore: number | null; // Average sentiment score for the portfolio
  overallSentimentLabel: string;
  investmentAmount: number; // Total money invested
  isActive: boolean;
  isTestMode: boolean;
  dateCreated: string;
  name: string;
  selectedInterval: string;
  selectedTerm: string;
  expirationDate: string;
}
```

Figure 1.



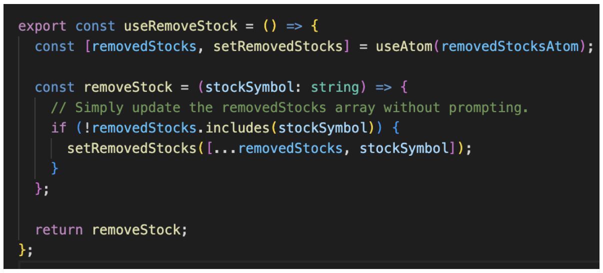
```typescript
const StockComponent: React.FC<StockProps> = ({ stock, weight, onClick }) => {
  const removeStock = useRemoveStock();
  const [removeHovered, setRemoveHovered] = useState(false);
  const [showModal, setShowModal] = useState(false);

  const handleRemoveClick = (e: React.MouseEvent<HTMLButtonElement>) => {
    e.stopPropagation();
    setShowModal(true);
  };

  const handleConfirm = () => {
    removeStock(stock.symbol);
    setShowModal(false);
  };

  const handleCancel = () => {
    setShowModal(false);
  };

  return (
    <div
      className={`${styles.container} ${
        removeHovered ? styles.redContainer : ""
      }`}
      onClick={(e) => {
        // Only trigger the container's onClick if the modal is not open.
        if (!showModal) {
          onClick();
        }
      }}
    >
      <button
        className={styles.removeButton}
        onMouseEnter={() => setRemoveHovered(true)}
        onMouseLeave={() => setRemoveHovered(false)}
        onClick={handleRemoveClick}
      >
        ×
      </button>
      <h3 className={styles.title}>{stock.name}</h3>
      <p className={styles.text}>Symbol: {stock.symbol}</p>
      <p className={styles.text}>Score: {stock.score}</p>
      <p className={styles.weight}>
        <strong>Weight:</strong> {(weight * 100).toFixed(1)}%
      </p>
      {showModal && (
        <ConfirmRemoveStock
          message={`Are you sure you want to remove ${stock.symbol} from your portfolio?`}
          onConfirm={handleConfirm}
          onCancel={handleCancel}
        />
      )}
    </div>
  );
};

export default StockComponent;
```

Figure 2.



```typescript
const PortfolioStocks: React.FC = () => {
  const [portfolio] = useAtom(portfolioAtom);
  const [weights] = useAtom(portfolioWeightsAtom);
  const stocks = portfolio?.stocks || [];

  // Internal state for the currently selected stock
  const [selectedStock, setSelectedStock] = useState<Stock | null>(null);

  // Toggle selection: clicking the same stock hides details.
  const handleSelectStock = (stock: Stock) => {
    if (selectedStock && selectedStock.symbol === stock.symbol) {
      setSelectedStock(null);
    } else {
      setSelectedStock(stock);
    }
  };

  return (
    <div className={styles.container}>
      <h3 className={styles.header}>Stocks</h3>

      <PortfolioDistributionBar />

      {selectedStock && (
        <div className={styles.detailsView}>
          <PortfolioStockDetails
            stock={selectedStock}
            onBack={() => setSelectedStock(null)}
          />
        </div>
      )}

      <div className={styles.grid}>
        {stocks.map((stock) => (
          <div key={stock.symbol}>
            <StockComponent
              stock={stock}
              weight={weights[stock.symbol]}
              onClick={() => handleSelectStock(stock)}
            />
          </div>
        ))}
      </div>
    </div>
  );
};

export default PortfolioStocks;
```

Figure 3.



```typescript
export const useRemoveStock = () => {
  const [removedStocks, setRemovedStocks] = useAtom(removedStocksAtom);

  const removeStock = (stockSymbol: string) => {
    // Simply update the removedStocks array without prompting.
    if (!removedStocks.includes(stockSymbol)) {
      setRemovedStocks([...removedStocks, stockSymbol]);
    }
  };

  return removeStock;
};
```

Figure 4.

Figures 1-4 show an implementation of the Model-View-Controller design pattern in our code base. Figure 1 depicts two data models used frequently throughout our project, the Portfolio and Stock interfaces. These objects store data retrieved through API calls. Figures 2 and 3 contain the code for our user interface, or the view. Specifically, Figure 2 is depicts the user interface for displaying stock data to the user. On this component, there is an option to remove the stock from your portfolio. Figure 4 contains the logic for removing said stock, which will update the Portofolio model, specifically the Stocks[] attribute. This change will then be displayed to the user in Figure 3, which displays the entire portfolio.

The rest of our code base can be found at this GitHub link:

github.com/gabrielmartens00/portfolio-automation-software-system

References

FCA finds young investors are more likely to have long-term goals in mind when dating than when investing. (2023, May 24). FCA. https://www.fca.org.uk/news/press-releases/young-investors-more-likely-have-long-term-goals-mind-dating-when-investing

Gallup. (2024, October 16). *What percentage of Americans own stock?* Gallup.com. https://news.gallup.com/poll/266807/percentage-americans-owns-stock.aspx

Salvucci, Jeremy. “An In-Depth Timeline of the Gamestop Short Squeeze Saga - Thestreet.” *TheStreet.*, 15 Sept. 2023, www.thestreet.com/investing/stocks/a-timeline-of-the-gamestop-short-squeeze.