# **InvestMate Software Architecture**

**Document Number:**

**INVM-DSGN-1**

**Version: 1.2**

**10/6/2019 5:47 PM**

**Prepared by:**

**David Casente,**

**Mike O’Brien,**

**Remy Van der Ploeg**

**Table Of Contents**

**1. Introduction 3**

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, acronyms, and abbreviations 3

1.4 References 3

1.5 Overview 3

**2. Applicable Documents 4**

2.1 InvestMate Documents 4

2.2 Non-InvestMate Documents 4

**3. Use-Case View 4**

3.1 System Inputs and Outputs 4

3.1.1 Inputs 4

3.1.2 Outputs 4

3.2 Use Case Realizations 5

3.2.1 Importing Portfolio 5

3.2.2 Edit Portfolio 5

3.2.3 View Overall Portfolio 5

3.2.4 View Specific Stock 5

3.2.5 View Projection 6

3.2.6 ViewAnalysis Answer 6

3.2.7 Stock of the Week 6

3.2.8 Analysts Stock of the Week 7

3.2.9 Search for Stocks 7

3.2.10 Metric Calculation and Explanation 7

3.2.11 Projection Learning Tool 8

3.2.12 View Stock News 8

3.2.13 View FAQ’s 8

3.2.14 View Portfolio Guidance 9

3.2.15 Buy Stock 9

3.2.16 Sell Stock 10

3.2.17 Submit Question for Analysis 10

**4. Class Interfaces 10**

4.1 Class Analyst 11

4.1.1 Public Method getAnswer 11

4.1.2 Public Method setAnswer 11

4.1.3 Public Method submitAnswer 11

4.1.4 Public Method submitStockPick 11

4.2 Class Graphics 11

4.2.1 Public Method loadPage 11

4.2.2 Public Method getPageinfo 12

4.3 Class Individual Stock Assessment 12

4.3.1 Public Method getAssessment 12

4.3.2 Public Method setAssessment 12

4.3.3 Public Method analysisToAssessment 12

4.4 Class Portfolio 12

4.4.1 Public Method getPortfolio 12

4.4.2 Public Method setPortfolio 13

4.4.3 Public Method buyStock 13 4.4.4 Public Method sellStock 13

4.5 Class Portfolio 13

4.5.1 Public Method getAnalysis 13

4.5.2 Public Method setAnalysis 13

4.5.3 Public Method analyze 14

4.6 Class Printer 14

4.6.1 Public Method ImportPortfolio 14

4.6.2 Public Method savePortfolio 14

4.6.3 Public Method saveProjectionAnalysis 14

4.6.4 Public Method saveStockAssessment 14

4.7 Class Project Learning Tool 15

4.7.1 Public Method getProjectionAnalysis 15

4.7.2 Public Method projectionAnalysis 15

4.8 Class Question 15

4.8.1 Public Method setQuestion 15

4.8.2 Public Method getQuestion 15

4.9 Class Stock 15

4.9.1 Public Method getStock 16

4.9.2 Public Method setStock 16

4.10 Class Stock Analysis 16

4.11 Class Stock Retrieval 16

4.11.1 Public Method retrieveStocks 16

4.11.2 Public Method converToStocks 16

4.12 Class Stock Search Recommender 16

4.13 Class User 17

**5. Class Diagrams 18**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Change** | **Version** |
| **Mike, David, and Remy** | **10/3/19** | **Initial draft** | **1.0 draft 1** |
| **Mike, David, and Remy** | **10/7/19** | **Baseline changes after inspection** | **1.1 draft 2** |
| **Mike, David, and Remy** | **10/9/19** | **Final Initial Draft Revisions** | **1.2 draft 3** |

**1. Introduction**

**1.1 purpose**

This document details a comprehensive overview of the architecture of the InvestMate program, showcasing the design of different aspects of the system. It is intended to convey the critical architectural decisions that have been made about the system.

**1.2 Scope**

The program will be used as an aid for beginning to intermediate level investors. The primary features of this program are the following. It will allow the user to onboard their portfolio when the application is first opened. It will then save the user's portfolio data so that every time the application is opened afterward, the system will be able to retrieve the data.

The three main analysis features that the program will have are portfolio assessment, a stock search recommender, and an individual stock assessment tool. These will allow the user to learn the basics of stock trading shortly and efficiently.

Lastly, the program will allow the user to ask an analyst to answer and qualitative questions about specific stocks in the market.

**1.3 Definitions, acronyms, and abbreviations**

Stock- an equity investment that represents a small piece of ownership in a corporation.

Portfolio- A person’s collection of investments, usually stocks.

Projection- The calculated stock price prediction for future dates.

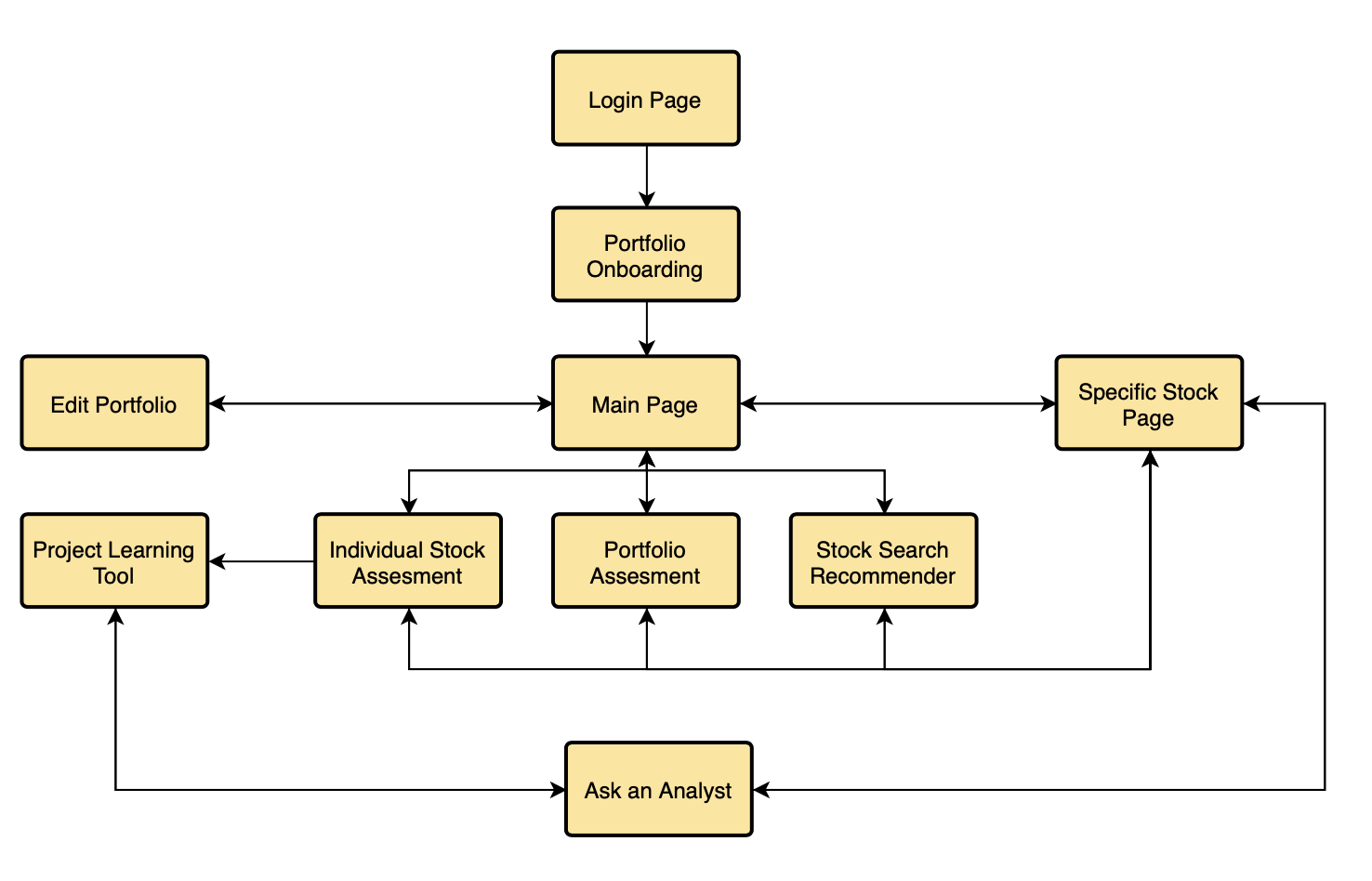
Purchasing Power- The amount of money a person could potentially spend at a given time.

**1.4 References**

None.

**1.5 Overview**

For this project, we will employ an Event-Driven Design architecture. The basic principle of this architecture is whenever a change of state occurs in the system, and the event is triggered. A change of state typically is caused by a user action. A simple example of this is when the user clicks a button resulting in a change in the next screen to be displayed.

**Figure 1**: Application architecture flow model 1.0 of the InvestMate program.

**2. Applicable Documents**

**2.1 InvestMate Documents**

InvestMate Software Requirements Specifications, 19 September 2019

**2.2 Non-InvestMate Documents**

The Intelligent Investor, Benjamin Graham 1949

A Random Walk Down Wall Street, Burton Malkiel 1973

**3. Use-Case view**

**3.1 System Inputs and Outputs**

**3.2.1 Inputs**

The application will take in relevant stock data from the internet to create a stock database.

The application will receive the portfolio and profile information from the user.

**3.2.2 Outputs**

The application will output the user to the internet for stock news articles

The application will output to excel for portfolio information

The application will output items such as an analysts answer and detailed projection calculations to a text document for printing purposes

**3.2 Use Case Realizations**

**3.2.1 Importing Portfolio**

* + **Actor:**
    - Investor
  + **Brief Description:**
    - In this use case the user submits its portfolio information to the application.
  + **Flow of Events:**

1. The use case begins with the user signing up on the login page.
2. The application returns the Portfolio Onboarding page with input spaces for stocks, purchasing power, and investing interests.
3. The user completes the input page.
4. The application takes the user’s input and inserts it into the user object.
5. End of case

**3.2.2 Editing Portfolio**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this use case, the user makes changes to its portfolio information.
  + **Flow of Events:**

1. The use case begins with the user clicking the “Edit Portfolio” button which is located on the Main page.
2. The application then returns the Edit Portfolio page, which displays all stocks currently in a user portfolio and an empty section for adding stocks.
3. The user can then edit the portfolio either removing or adding stocks and submit the changes.
4. The application takes the input changes and updates the user object.
5. The application returns the Main page.
6. End of case

**3.2.3 View Overall Portfolio**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this use case, the user views the vital metrics of individual stocks they own and overall performance of their portfolio.
  + **Flow of Events:**

1. The use case begins with the user clicking the “Return Home” button on any other page in the application.
2. The application then returns the home page and displays the user’s portfolio.
3. End of case

**3.2.4 View Specific Stock**

* + **Actor:**
    - Investor
  + **Brief Description:**
    - In this use case, the user views a specific stock and more detailed metrics than provided on the Main page.
  + **Flow of Events:**

1. The use case begins with a user clicking on a “more information” button designated to a specific stock on either the Main Page, Portfolio Assessment Page, or Stock Recommender Page.
2. The application then returns the Specific Stock Page, using the input stock
3. End of case

**3.2.5 View Projection**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this use case, the user views a specific stock’s projection as well as the detailed steps the application took to calculate it.
  + **Flow of Events:**

1. The use case begins with a user clicking on a “View Projection Specifics” button designated to a specific stock the Main Page, Portfolio Assessment Page, or Stock Recommender Page.
2. The application then returns the Projection Learning tool, using the input stock.
3. End of case

**3.2.6 View Analyst Answer**

* + **Actor:** 
    - Investor, Analyst
  + **Brief Description:**
    - In this use case, the analyst submits an answer to a question that the user asks. The user views the answer provided by an analyst to a previously submitted question.
  + **Flow of Events:**

1. The use case begins with the analyst submitting an answer to a users previously submitted question.
2. The user can then click on the “View Analyst Message” that is designated to a certain question on the Ask an Analyst page.
3. The application will then return a pop up window with the analyst’s message based on the designated question to the user.
4. The user can close the window once they are finished reading the message.
5. End of case

**3.2.7 Stock of the Week**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user views the application’s stock recommendation for the particular week
  + **Flow of Events:**

1. The use case begins by the user clicking on the stock search recommender tab on the Main Page.
2. The application then returns the stock search recommender page which displays at the top the application’s pick of the week, based on examination of all stocks in the database.
3. The user can view the stock pick of the week
4. End of case

**3.2.8 Analyst Stock of the Week**

* + **Actor:** 
    - Investor, Analyst
  + **Brief Description:**
    - In this case, the analyst submits a stock recommendation for a particular week. The user views the stock recommended by the analyst for a particular week.
  + **Flow of Events:** 
    - 1. The use case begins with the analyst submitting their stock pick of the week to the application on the analyst page
      2. The application will update the information on the stock pick object
      3. The user can then click on the stock search recommender tab on the Main Page
      4. The application then returns the stock search recommender page which displays at the top the analyst pick of the week
      5. The user can view the analyst stock pick of the week
      6. End of case

**3.2.9 Search for Stocks**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case the user utilizes the Stock recommender to search for a stock to invest in given filters.
  + **Flow of Events:** 
    - 1. The use case begins with the user clicking on the stock search recommender tab on the Main page
      2. The application returns the stock recommender page with the stock search at the bottom
      3. The user can fill out the filter fields and submit a search
      4. The application will take the input filters and determines using an algorithm the stock(s) that are recommendable or none at all and will display it on the screen for the user
      5. End of case

**3.2.10 Metric Calculation and Explanation**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user views the metric calculation and explanation for a specific stock.
  + **Flow of Events:** 
    - 1. The use begins with the user clicking on the individual stock assessment tab and then selecting a specific stock in the input filter fields and clicks the “Learn Metric Calculation” button or if the user clicks on the button “Learn Metric Calculation” on the specific stock page.
      2. The application uses the designated stock and displays the detailed calculation for the important investing metrics
      3. End of case

**3.2.11 Projection Learning Tool**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user views the projection calculation and explanation for a specific stock.
  + **Flow of Events:**
    - 1. The use begins with the user clicking on the individual stock assessment tab and then selecting a specific stock in the input filter fields and clicks the “Learn Projection Calculation” button or if the user clicks on the button “Learn Projection Calculation” on the specific stock page.
      2. The application uses the designated stock and displays the detailed calculation for the stock price projection
      3. End of case

**3.2.12 View Stock News**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user views recent news articles for a specific stock.
  + **Flow of Events:** 
    - 1. The use case begins with the user clicking on the “show recent stock news” button on the specific stock page
      2. The application will take the specific stock input and display the titles of the top three articles regarding the stock.
      3. The user can then click on the title of the article they want to read
      4. The application will open up the article in a user web browser
      5. End of case

**3.2.13 View FAQ’s**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user views the most frequent questions and answers for Ask an Analyst.
  + **Flow of Events:** 
    - 1. This use case begins with the user clicking on the ask an analyst tab from the main page
      2. The application then displays the question submitting section at the top of the ask an analyst page and below there is the FAQ section
      3. End of case

**3.2.14 View Portfolio Guidance**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user views the applications recommendation to balance their portfolio given user input.
  + **Flow of Events:** 
    - 1. The use case begins with the user clicking on the Portfolio Guidance tab on the Main Page.
      2. The Application then will display the Portfolio Guidance page with input fields to specify the guidance a user wants
      3. The user will fill out the input fields and submit
      4. The application will use the input to determine how the user should allocate its purchasing power and display that advice
      5. The user can read the advice
      6. End of case

**3.2.15 Buy stock**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user adds a new stock or more of a stock to their portfolio.
  + **Flow of Events:** 
    - 1. The use case begins with a user clicking on a “more information” button designated to a specific stock on either the Main Page, Portfolio Assessment Page, or Stock Recommender Page.
      2. The application then returns the Specific Stock Page, using the input stock, with a Buy Stock button at the bottom
      3. The user can click on the button.
      4. The application will then display a few input fields such as number of stocks.
      5. The user will complete the fields and submit
      6. The application will use the input fields and add the specific stock to the user’s portfolio (note: this does not actually purchase the stock).
      7. End of case

**3.2.16 Sell stock**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user removes some or all of a stock from its portfolio.
  + **Flow of Events:** 
    - 1. The use case begins with a user clicking on a “more information” button designated to a specific stock that they own on either the Main Page, Portfolio Assessment Page, or Stock Recommender Page.
      2. The application then returns the Specific Stock Page, using the input stock, with a Sell Stock button at the bottom if they own the stock in their portfolio
      3. The user can click on the button.
      4. The application will then display a few input fields such as number of stocks.
      5. The user will complete the fields and submit
      6. The application will use the input fields and remove the amount of stock or the specific stock if they remove all, from the user’s portfolio (note: this does not actually sell the stock).
      7. End of case

**3.2.17 Submit question for analyst**

* + **Actor:** 
    - Investor
  + **Brief Description:**
    - In this case, the user submits a question for an Analyst regarding investing
  + **Flow of Events:** 
    - 1. The use case begins with a user clicking on a “Ask an Analyst” button on either the specific stock page or the Projection Learning Tool
      2. The application then displays the ask an analyst page with the section for submitting questions at the top, which includes input fields for type of question, description, etc.
      3. The user then fills out the fields and submits their question
      4. The application will take the user’s submission and puts it into the question object and will send it to the Analyst
      5. End of case

**4. Class Interfaces**

This section will detail the interfaces of the object classes within InvestMates software architecture. These classes are not finalized and subject to change. Constructors, Deconstructors, and exceptions are not included. Classes are shown in alphabetical order.

**4.1 Class Analyst**

This will be the Analyst object that holds all the information for each Analyst. The object will store the profile of the analyst and will handle the process of answering user questions and submitting weekly stock picks

**4.1.1 Public Method getAnswer**

Answer getAnswer()

This method returns the Answer object

**4.1.2 Public Method setAnswer**

Void setAnswer(String, Analyst)

This method sets the Answer object given the input of answer text and analyst who wrote it.

**4.1.3 Public Method submitAnswer**

Void submitAnswer(Answer, Question)

This method will update the question object with the analyst’s answer and will also update the user object so that the user knows the question has been answered.

**4.1.4 Public Method submitStockPick**

Void submitStockPick(Stock, StockSearchRecommender)

This method will take an input stock and will update the analyst stock pick in the stock search recommender with it.

**4.2 Class Graphics**

This class will handle the display of the application, loading the pages and within that all the text as well as graphics.

**4.2.1 Public Method loadPage**

Void loadPage(Object)

Given the page object, this method will create the specific display page of the application.

**4.2.2 Public Method getPageInfo**

Page getPageInfo()

This method returns the Page object for the specified page.

**4.3 Class Individual Stock Assessment**

This class will contain information about the viability and projected trend of a stock.

**4.3.1 Public Method getAssessment**

String IndividualStockAssessment.getAssessment()

This method returns a string with information about the stock that it holds and it’s outlook.

**4.3.2 Public Method setAssessment**

void IndividualStockAssessment.setAssessment()

This is a method that is used by the stock analysis class in order to populate the stock assessment object with information about that stock’s projection.

**4.3.3 Public Method analysisToAssessment**

Void IndividualStockAssessment.analysisToAssessment()

This method converts a stock’s analysis into an assessment that can be retrieved from the IndividualStockAssessment.

**4.4 Class Portfolio**

This will be the portfolio object that holds all information for each User’s Portfolio. This will hold a collection of stocks, as well as information about the portfolio as a whole, (diversity, total value, portfolio analysis).

**4.4.1 Public Method getPortfolio**

Portfolio getPortfolio()

This method will return the specified portfolio object.

**4.4.2 Public Method setPortfolio**

Void setPortfolio(Portfolio)

This method will update the specified portfolio with the input information.

**4.4.3 Public Method buyStock**

Void buyStock(Stock, Integer)

This method will add the input stock to the portfolio object and will use the input integer to fill in the number of stocks purchased if the stock is not already in the portfolio. If the stock was purchased previously, then only the number of stock variable will be updated.

**4.4.4 Public Method sellStock**

Void sellStock(Stock, Integer)

If the user has more numbers of stock than the input integer, then this method will update the number of stock variable. Otherwise, the method will completely remove the input stock. If the input stock is not listed in the portfolio, then the method will do nothing.

**4.5 Class Portfolio Analysis**

This class will handle all activities related to the portfolio guidance. This will include creating portfolio balance analysis and displaying the breakdown of the user’s monetary allocation.

**4.5.1 Public Method getAnalysis**

StringPortfolioAnalysis.getAnalysis()

This method will return a string description of the current outlook on the User’s portfolio.

**4.5.2 Public Method setAnalysis**

void PortfolioAnalysis.setAnalysis(Analysis)

This method will be used by the API to set the analysis for the User’s portfolio.

**4.5.3 Public Method analyze**

void PortfolioAnalysis.analyze()

This method will be called in order to analyze the current portfolio and determine it’s outlook.

**4.6 Class Printer**

This class is used for data ingress/egress, allowing the application to take in data and export the User’s portfolio and analysis to the user’s device.

**4.6.1 Public Method importPortfolio**

void Printer.importPortfolio(String)

This method will intake a file path to a spreadsheet and allow Users to import a portfolio from their device.

**4.6.2 Public Method savePortfolio**

void Printer.savePortfolio()

This method will output the current User’s portfolio to a spreadsheet, allowing them to load it and continue to work on it at a later date.

**4.6.3 Public Method saveProjectionAnalysis**

void Printer.saveProjectionAnalysis()

This method will save a text document with the projection analysis of their portfolio to their device.

**4.6.4 Public Method saveStockAssessment**

void Printer.saveStockAssessment(Stock)

This method will save an assessment of an input User’s stock to the User’s device.

**4.7 Class Projection Learning Tool**

This class will handle all of the activity on the Projection Learning tool page. The main function is to take an input stock and create the analysis and explanation of the applications projection of the stock price.

**4.7.1 Public Method getProjectionAnalysis**

String getProjectionAnalysis()

This method returns the specified projection analysis

**4.7.2 Public Method projectionAnalysis**

Void projectionAnalysis(Stock)

This method takes the input stock and created the Projection Analysis to be displayed on the page.

**4.8 Class Question**

This class will handle all of the questions from users for the Ask an Analyst page. The object will contain information such as the question text, user, and type of question.

**4.8.1 Public Method setQuestion**

void Question.setQuestion(String)

This method saves an input question from a User and can return it to an expert to be answered later.

**4.8.2 Public Method getQuestion**

String Question.getQuestion()

This method returns the question previously input by a User.

**4.9 Class Stock**

This will be the Stock object that holds all information for each specific Stock. Stocks will hold their ticker, exchange, title and API key.

**4.9.1 Public Method getStock**

String Stock.getStock()

This method returns information about the specified stock object.

**4.9.2 Public Method setStock**

void Stock.setStock(Ticker)

This method intakes the entered stock Ticker and keeps it within the Stock object.

**4.10 Class Stock Analysis**

This class will handle the metric calculation and analysis for a specific stock. Functions that will be carried out will include the calculation of a stocks score, which will be used in the stock search algorithm. The interface of this class is not yet defined.

**4.11 Class Stock Retrieval**

This class gets all the stock information using the API.

**4.11.1 Public Method retrieveStocks**

voidretrieveStocks(Portfolio)

This method intakes a portfolio and retrieves information about all of the stocks in it, and adds that information to the input portfolio.

**4.11.2 Public Method convertToStocks**

Stock convertToStocks(String)

This method generates Stock objects from a ticker using the stock API.

**4.12 Class Stock Search Recommender**

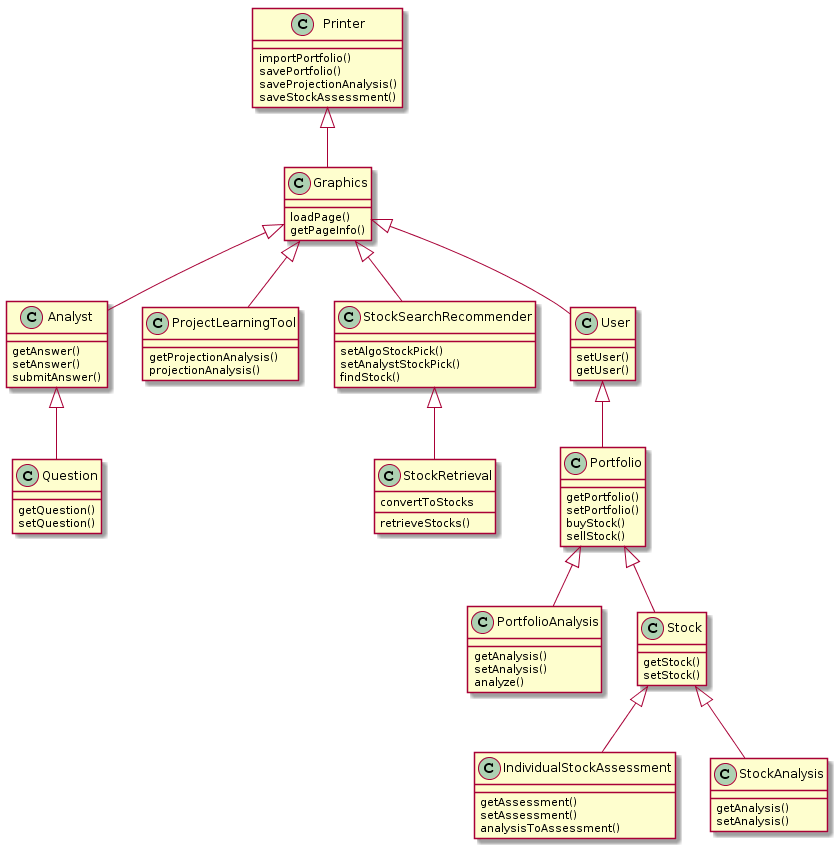
This class will handle all activities related to the stock search recommender page.

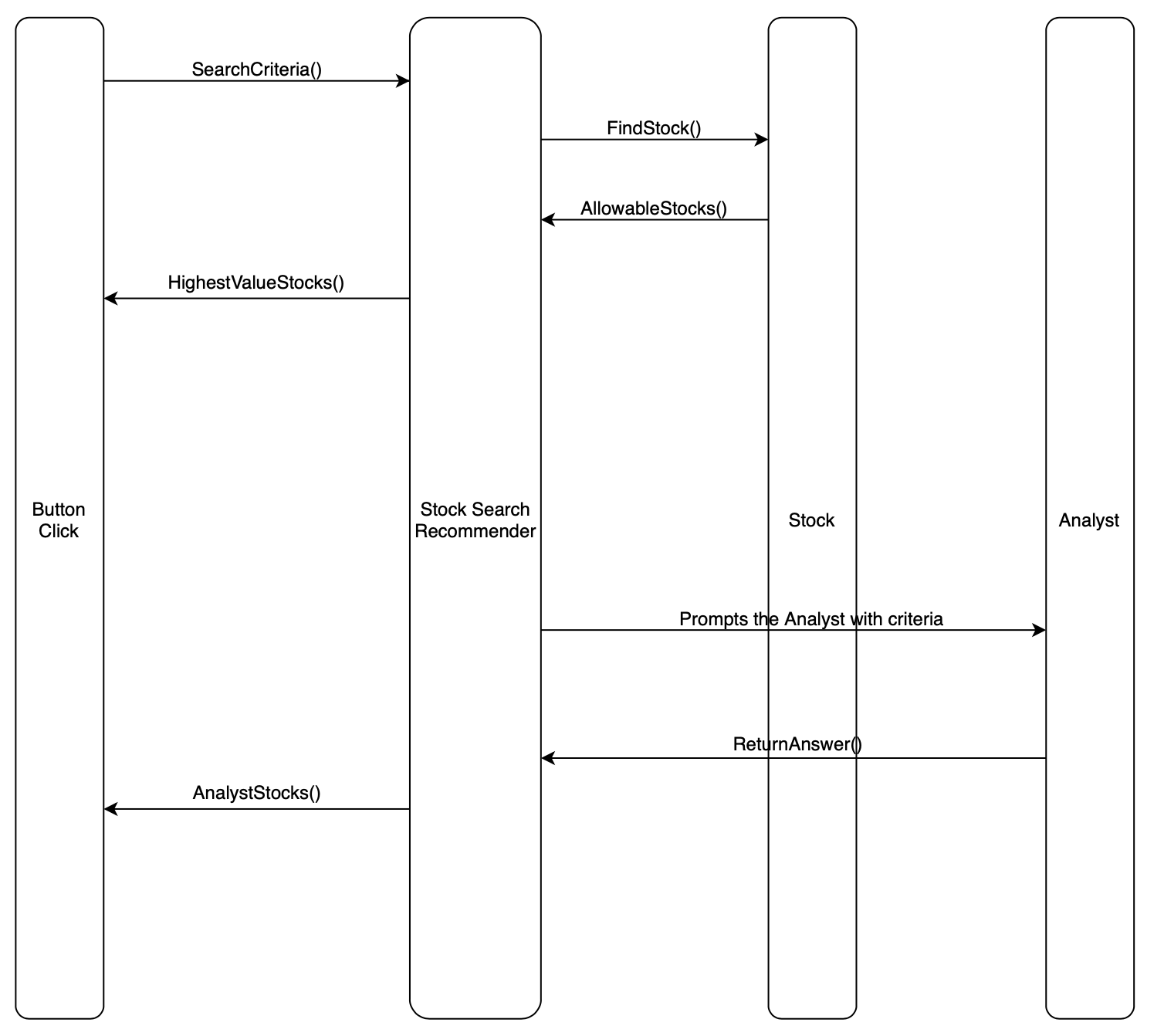
The class will store the stock picks of the week from the application and analyst, and functions carried out will include the stock search, which finds the stock recommendations based on user criteria. The interface of this class is not yet defined.

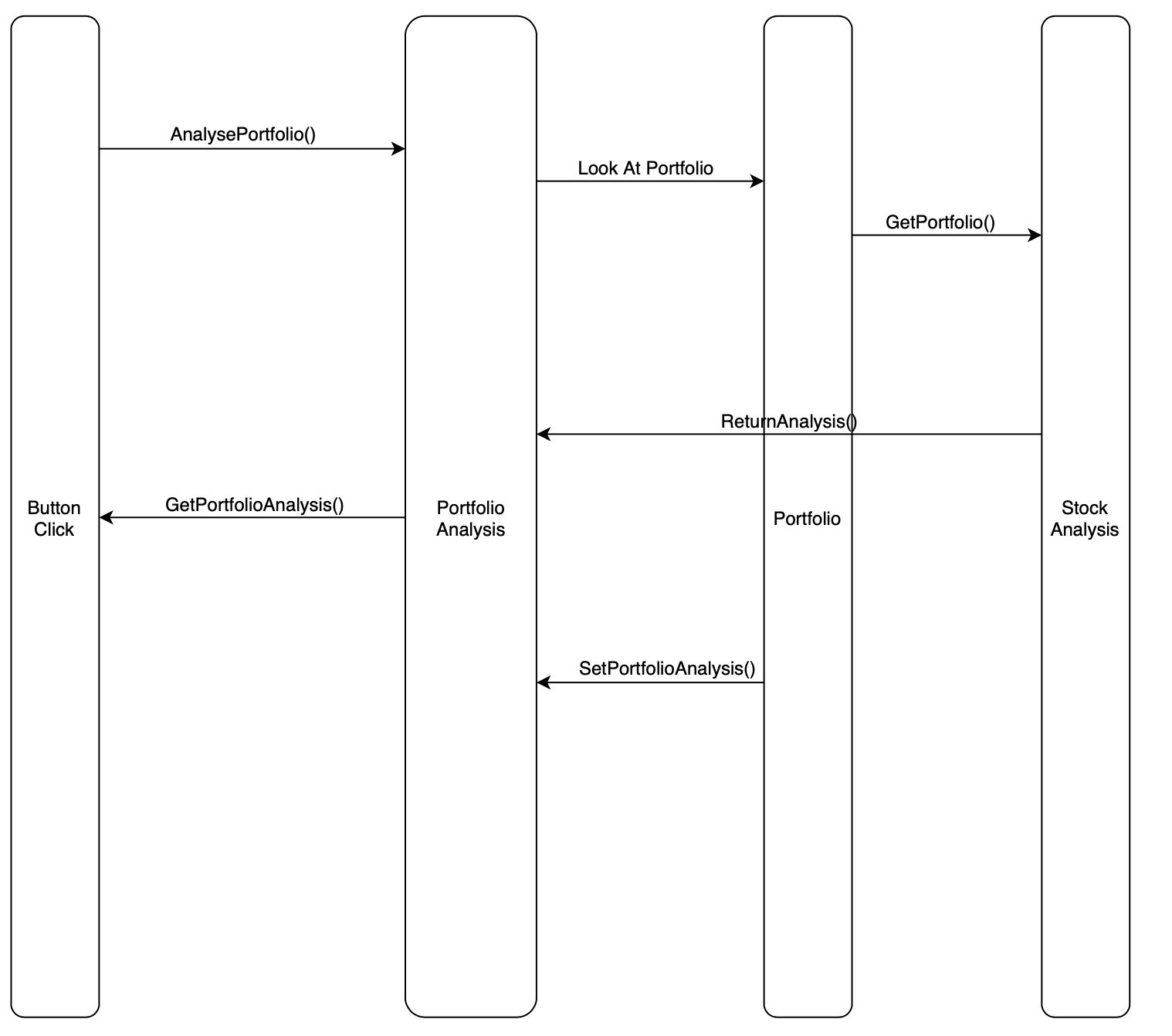
**4.13 Class User**

An instance of class User represents all the information about the user that is being processed. User objects are used internally by the program. This will be the user object that holds all the information for each user. The interface of this class is not yet defined.

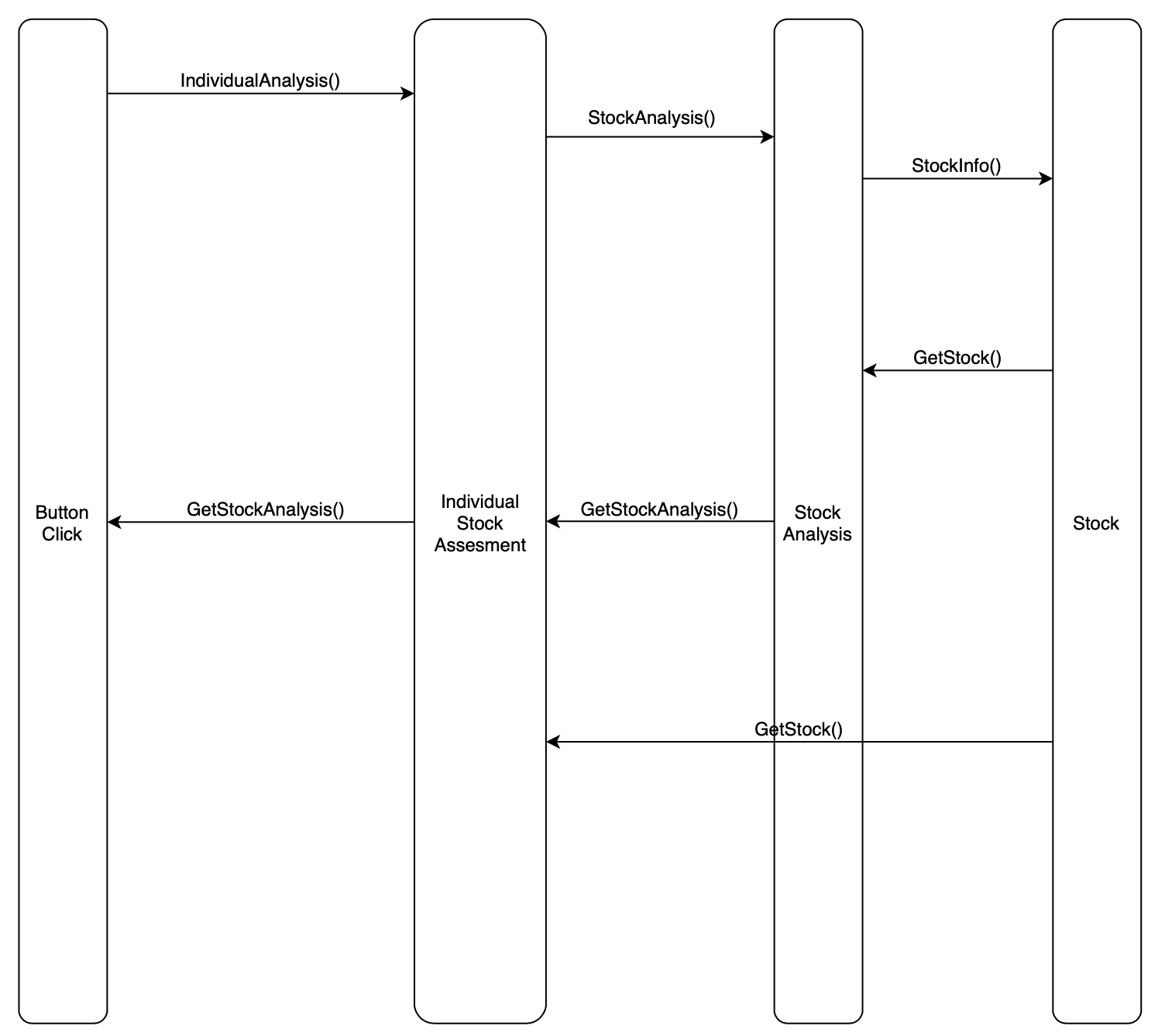
**5. Class Diagrams**

**Figure 2:** Class Hierarchy Diagram

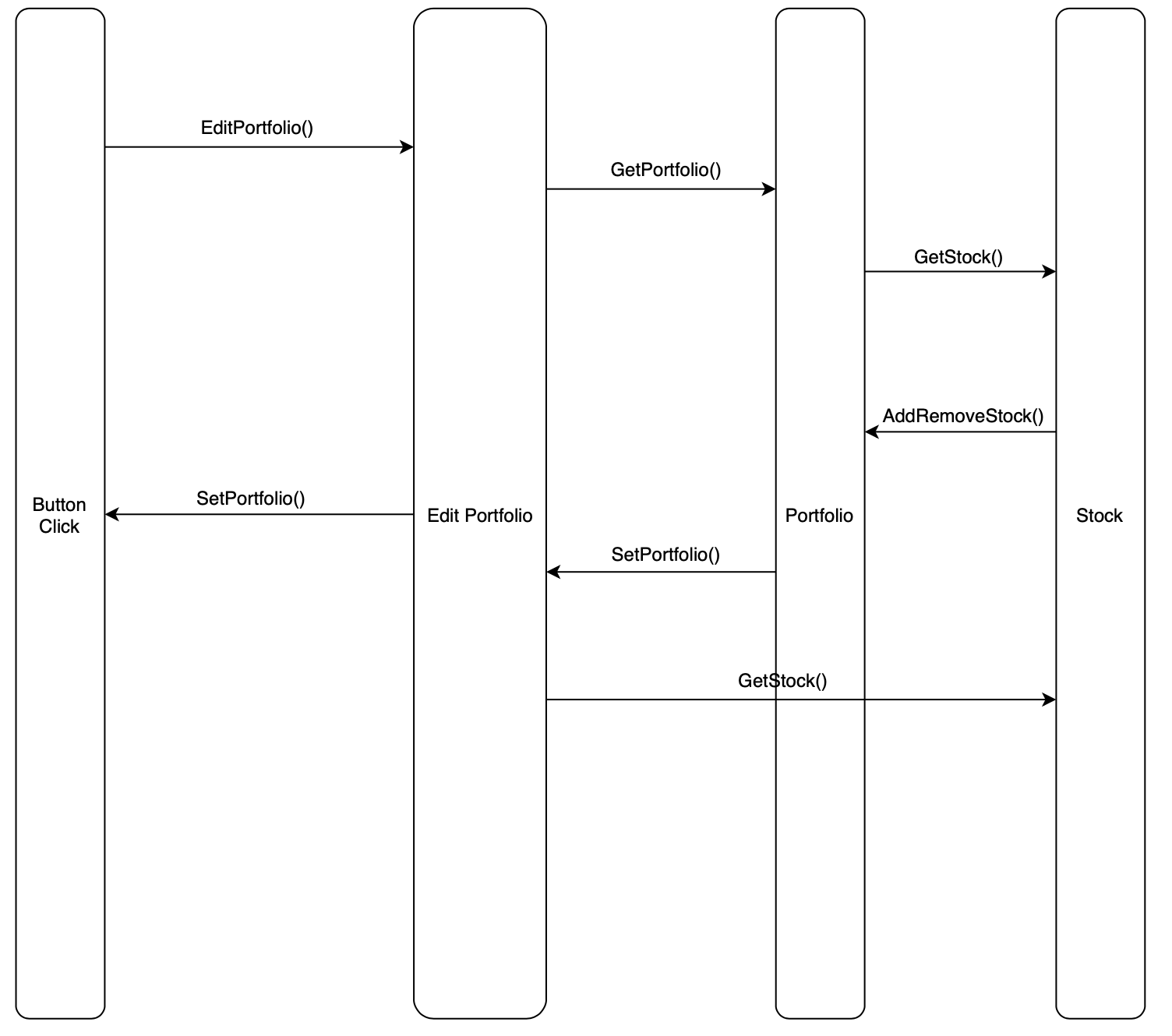


**Figure 3:** Stock Search Recommender Interaction Diagram

**Figure 4:** Portfolio Analysis Interaction Diagram



**Figure 5:** Individual Stock Assessment Interaction Diagram



**Figure 6:** Edit Portfolio Interaction Diagram