// Add title page

// Add revision history

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Change** | **Version** |
| **Mike, David, and Remy** | **11/1/19** | **Initial draft** | **1.0 draft 1** |
| **Mike, David, and Remy** | **11/4/19** | **Baseline changes after inspection** | **1.1 draft 2** |
| **Mike, David, and Remy** | **11/6/19** | **Final Initial Draft Revisions** | **1.2 draft 3** |

1. Introduction

This document serves as an outline of the goals for testing of the InvestMate application. It will entail all the tests for the methods within the application and how we plan on implementing them. Functional testing areas include stock data retrieval, stock/portfolio analysis and professional analyst response. The objectives of this document are to outline tests and create detailed guidelines for function testing plan. The other objective is to confirm that each test case verifies the software and fulfills the requirements that we talked about in our SRS document.

2. Project and Scope

This document will present the goals which our group will accomplish so that the InvestMate application can be fully tested. it will contain brief descriptions of how each of the use cases will be tested. We will mainly test the InvestMate application using Junit tests, but we will also use some special testing for the data retrieval. For specific tests, reference the Test Specifications section of this document. We have made it so all the user can input into the program is through pre selected data, so that we can control the range of tests that we run on the program. Therefore, we will be testing a limited but comprehensive sample size. After completing the code for each of the sections of our program we will begin to test it. Once we have finished testing that one section of our program, we will continue to implement another section of our project.

3. Test Design Specifications

In this document, we will have specific tests for each of our different classes. Each test will be simulated with a user input and then go through steps to get the expected output of the program. If all the steps of the program are have the correct output for the input, the n the test will count as being passed. Any test that fails somewhere along the way to the final step of testing will be considered as a failed test.

4. Test Specifications**- elaborate on how the application will function specifically**

**Importing Portfolio**

1. user signs up on the login page.
2. The application returns the Portfolio Onboarding page with input spaces for stocks, with parameters for stock name, amount, and date purchased.
3. The user completes the input page.
   1. If the user does not fill out a section for the stock, or inputs an invalid value, the application will prevent the user from continuing until the mistake is fixed
4. The application takes the user’s inputs as arrays of string for stock name, int for amount and string for date purchased. The application creates OwnedStock objects for each stock inputted, uses them to create a Portfolio object and creates a User object for the user, with a variable pointing to the portfolio object.
5. The application returns the main page, with the Portfolio object displayed.

**Editing Portfolio**

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

**View Overall Portfolio**

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

**View Specific Stock**

* + - 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.

1. The application then returns the Specific Stock Page, using the input stock
2. End of case

**View Projection**

* + - 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.

1. The application then returns the Projection Learning tool, using the input stock.
2. End of case

**View Analyst Answer**

* + - The use case begins with the analyst submitting an answer to a users previously submitted question.

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

**Stock of the Week**

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

**Analyst Stock of the Week**

* + The use case begins with the analyst submitting their stock pick of the week to the application on the analyst page
    - 1. The application will update the information on the stock pick object
      2. The user can then click on the stock search recommender tab on the Main Page
      3. The application then returns the stock search recommender page which displays at the top the analyst pick of the week
      4. The user can view the analyst stock pick of the week
      5. End of case

**Search for Stocks**

* + - 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

**Metric Calculation and Explanation**

* + - 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.
      1. The application uses the designated stock and displays the detailed calculation for the important investing metrics
      2. End of case

**Projection Learning Tool**

* + - 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.
      1. The application uses the designated stock and displays the detailed calculation for the stock price projection
      2. End of case

**View Stock News**

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

**Buy stock**

* + 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.
    - 1. The application then returns the Specific Stock Page, using the input stock, with a Buy Stock button at the bottom
      2. The user can click on the button.
      3. The application will then display a few input fields such as number of stocks.
      4. The user will complete the fields and submit
      5. 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).
      6. End of case

**Sell stock**

* + 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.
    - 1. 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
      2. The user can click on the button.
      3. The application will then display a few input fields such as number of stocks.
      4. The user will complete the fields and submit
      5. 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).
      6. End of case

**Submit question for analyst**

* + - 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
      1. 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.
      2. The user then fills out the fields and submits their question
      3. The application will take the user’s submission and puts it into the question object and will send it to the Analyst
      4. End of case

5. Functional Tests

1. Analyst Class Tests

Test: ANL-1

Description: Determine a valid Analyst Object

Setup: Pass parameters to create an Analyst object

Expected Result: no error should occur when creating the object

Test: ANL-2

Description: Return the name of the Analyst

Setup: Create an Analyst object with a name and use the GetName method

Expected Result: the name of the Analyst that we made

Test: ANL-3

Description: Pass a question to the Analyst

Setup: Create an analyst object and try to pass a question to it

Expected Result: the analyst should receive the question and be able to answer it

1. Individual Stock Assessment Class Tests

Test: ISACT-1

Description: Pass no stock to be analyzed

Setup: pass no stock to the stock analyzer

Expected Result: NoStockError should be raised because there is nothing to analyze

Test: ISACT-2

Description: Pass a stock to be analyzed

Setup: Pass any random stock to be analyzed

Expected Result: Data pertaining to that stock and what the user should do with it

Test: ISACT-3

Description: Pass a non existing stock to be analyzed

Setup: Pass a stock that doesn't exist to the analysis method

Expected Result: a NonExistingStockError should be raised because unreal stock.

1. Project Learning Tool Class Tests

Test: PLTCT-1

Description: Pass no stock to be helped learned

Setup: pass nothing to be helped learned

Expected Result: a NoStockError should be raised

Test: PLTCT-2

Description: Pass a non existent stock to be helped learned

Setup: Pass a non existent stock to be helped learned

Expected Result: a NonExistingStockError should be raised

Test: PLTCT-3

Description: Pass a stock to be helped learned

Setup: Pass a stock to the method

Expected Result: The program should show step by step how it is analyzed

**// Remi take these ones below**

1. Question Class Test

Test: QCT-1

Description: Pass no question

Setup: User inputs nothing for a question

Expected Result: The program should not send through an empty question, prompt user for a question. Throw NoQuestionError.

Test: QCT-2

Description: Pass a question

Setup: User inputs a question for our expert to respond to

Expected Result: The program should send through the question to the expert’s portal.

1. Stock Analysis Class Tests

Test : SACT-1

Description: No stock is passed to be analyzed

Setup: User attempts to analyze a stock without inputting a stock

Expected Result: Throw NoAnalyzedStock error and prompt the user to input a stock to be analyzed.

Test: SACT-2

Description: Stock is passed to ba analyzed

Setup: User provides a stock for the program to analyze

Expected Result: Program returns information about the stocks possible trends and viability based off of the stock’s prior data.

1. Stock Search Recommender Class Tests

Test: SSRCT-1

Description:

1. Stock Class Tests

Test: SCT-1

Description: Attempt to retrieve data for nonexistent date

Setup: The user attempts to get data about a specific stock on a specific date that does not exist

Expected Result: Program returns message that this is an invalid date and throws InvalidDateError.

Test: SCT-2

Description: Attempt to retrieve data for existing date

Setup: The user retrieves data about a specific stock at an existing date

Expected Result: Program returns data about that stock at that date.

Test: SCT-3

Description: Attempt to retrieve data without internet connection

Setup: The user attempts to retrieve data about a specific stock at a date without an internet connection.

Expected Result: Program messages user that they need an internet connection in order to use this part of InvestMate. Throws FailedURLConnectionError.

6. Defect Responsibility and Resolution

1. Issue Tracking
   1. We plan on using Github’s built-in issue tracking within our repository to track issues as it is easily accessible to all of our teammates and is a powerful enough tool to explain issues that need to be resolved.
2. Responsibilities

|  |  |  |
| --- | --- | --- |
| System Part | Developer | Responsibilities |
| Functional Testing and Investing Implementation | David Casente | Make sure testing has necessary coverage of code. Complete GUI functional tests. Handle backend material related to investing concepts and techniques. |
| Backend (API and Data Management) | Rembrandt van der Ploeg | Import stock information from the Alpha Vantage API. Process and handle data for consumption by the InvestMate. Handle user-side responsiveness through API usage optimization |
| Front End (GUI/ backend connecting) | Mike | Create an easy to use functional, GUI and make it aesthetically pleasing. Connect the front end to the back end. Feed data from the user to the back end |

7. Exit Criteria

1. All J-Unit Tests have run and passed inspection.
2. All API related tests that cannot be checked through J-Unit testing have been thoroughly tested (manually), and all information taken from the API have been compared to trusted market data.
3. No stoppages throughout execution of the program, including those that could be caused by delayed API connections.