Ruse Cruise Stock Ticker

Final Report Summary

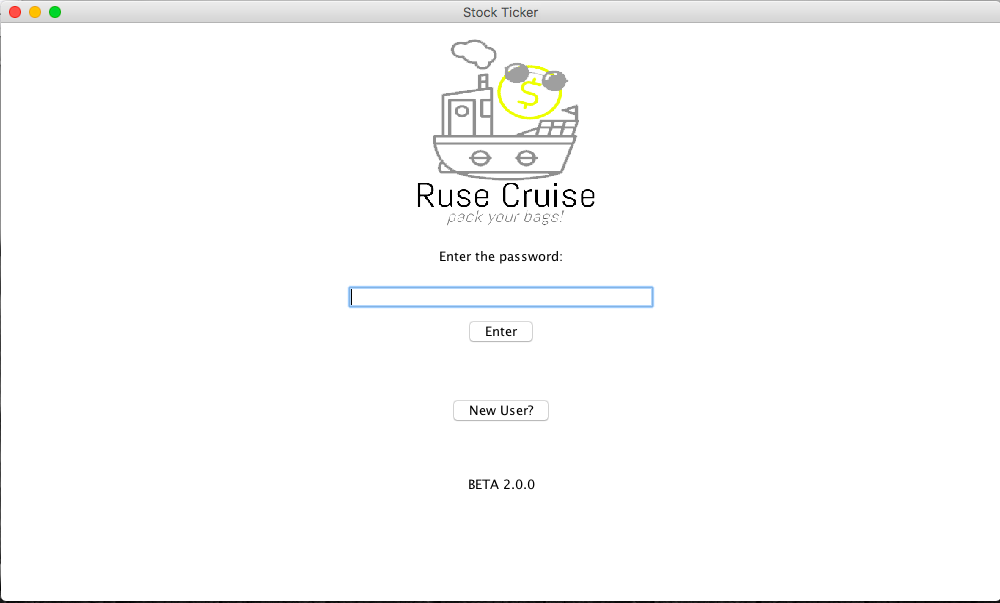
May 10, 2018

CSI 480

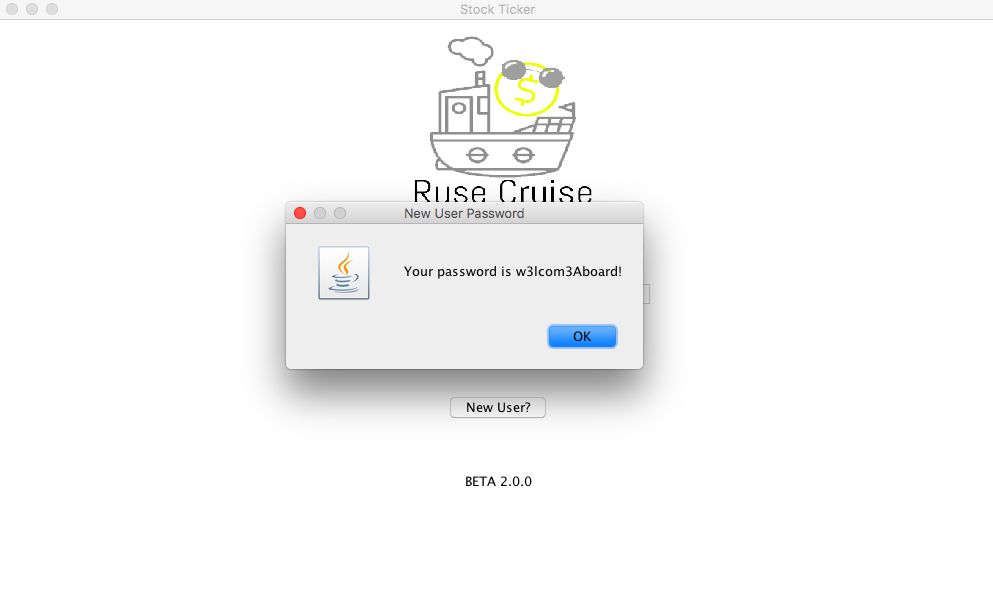
Bernero, Cox, Croke, Mi

1. Prologue
   1. Ruse Cruise: *Pack your bags!*
   2. Members: Matt Bernero, Caroline Cox, Caitlin Croke, Damien Mi
   3. Project Title: Ruse Cruise Stocks
2. Sprint I report:
   1. Project Overview
      1. Scrum Master: Caroline Cox
      2. Product Owner: Caitlin Croke
      3. Team members: Matt Bernero, Caitlin Croke, Caroline Cox, Damien Mi
      4. Estimate of time worked in total: 15 hours
   2. Product Overview
      1. All necessary and anticipated planning has been completed, including the structure of the API and User Interface (UI). Moving beyond planning, the basic API calls have been implemented and the UI has been started. Currently, this includes only the login screen with minimal functionality to type and press buttons.
      2. User stories addressed:
3. As a developer, I want to use an API to get the stock data, so that my team and I can more effectively use these resources to focus on the design aspects of our application.
4. Story Points: 8
5. Design: Calling JSON through a URL to parse into the result needed by the function. Each use of the API is performed with its own function and JSON parse. Currently some functions are not running as efficiently as initially intended. This is not a pressing issue but should be addressed in a future sprint, if time allows.
6. Implementer: Caroline Cox
7. As a user, I want to have my portfolio password protected, so that I can ensure my financial security.
8. Story Points: 8
9. Design: Using a JPasswordField and comparing input to a password variable.
10. Implementation Details: The program opens to a login screen that prompts the user to enter the password. If the password is incorrect, a dialog box appears indicating the password attempt was incorrect and the entry is cleared from the field. If the password is correct, the entry is cleared and the panel shifts to the home screen of the stock ticker.
11. Implementer: Caitlin Croke
    1. Integration Testing
       1. Integration between screens in the UI was done, currently with two screens switching between one another via buttons. Testing the switching between panels in the card layout was tested via using the buttons for their anticipated use and using the buttons when it should not work (such as if the password is not there or incorrect). Both cases performed as anticipated and no other edge cases exist for testing.
       2. No other major integration testing has been needed at this point since the API is currently not being used with the UI. This will be a task that will be important in the future sprints.
    2. State of the Quality of the Product
       1. Currently, the product is secure to the point that we were hoping for, and the base layout for the User Interface is detailed, with parts of it created in various stages of development. It is not a usable product for the purpose that it will eventually fulfil, but it is a working piece of software that behaves as expected for this point in time.
    3. Scrum I Retrospective:
       1. Overall, this sprint was satisfactory for the planning and initial development of the product. In future sprints, more workload can be assigned so all team members can have meaningful work to accomplish. Communication was good and allowed for the work that was completed to be completed well and with input from other team members. It will also be important that everyone on the team is familiar with all parts of the system so that there is not a limit to what certain people can work on due to lack of knowledge of certain aspects.
       2. Individual Perspectives
          1. Scrum Master’s Perspective: The team started off very well in planning and agreeing on the overall structure and design of the product. A large part of the sprint was spent setting up development environments and learning how to use GitHub, Eclipse, and Maven to automate the development process. Once this was completed, the team was able to complete necessary API functionality and start work on the UI. Total effort expended: 15 hours. Moving forward, more work needs to be done to put the UI in a position that it can be easily altered and all screens are available to be worked on.
          2. Product Owner’s Perspective: The design that was agreed upon by the team has a simple approach that encompasses all of the desired specifications put forward at the start of the project. Setting up took longer than expected and the third goal of displaying a graph of historical data was not reached. The UI that has been implemented so far is functional and will serve as an adequate foundation for future work.
          3. Matt Bernero’s Perspective: Much of my involvement in this sprint was based around the initial planning and design of the GUIs, as well as the initial programming for the password screen. To be truthful, I feel that I did not make much of an impact. I had no real assigned task to do during the sprint; while I have been working on odds and ends throughout, I do not have any visible accomplishments to claim and present as my own. This problem could be attributed to being too conservative with our time, and therefore not having a lot of tasks to work on. However, I believe it is for the best that we approached this project cautiously, rather than inundating ourselves with work that we could not hope to complete all in one sprint. For the next sprint, the team can hopefully use what we have learned about the Agile Process to more evenly distribute the workload.
          4. Caroline Cox’s Perspective: The first sprint has set the team up to efficiently and effectively continue working on the UI in future sprints. My contribution was creating the API class and coding all of the API functions to their current state, which is a large step to completing this project as a whole. I was also able to work on fixing a bug where an image we were showing would only appear if it was stored on the local machine.
          5. Caitlin Croke’s Perspective: The first sprint set up the overall design for the project and we got the API up and running and a basic UI. I set up the login screen and got the password field to check the user input and accept the correct password. Once the correct password is enter, the login screen changes to show the home screen that will be developed in future sprints.
          6. Damien Mi’s Perspective: The first sprint for our team is get a brief idea of the whole project and get a simple UI that everyone can see what it looks like. I was working on the home screen to create the research button. I am still reading something about how to get the key words and research it from the API database.
       3. Preparations for Sprint 2
          1. Scrum Master: Matt Bernero
          2. Product Owner: Damien Mi
          3. Goals:
       4. Display historical data on a graph (8 story points)
       5. Searching for stocks to add them to portfolio (16 story points)
       6. Visual indicators of stocks on home screen (2 story points)
       7. Keeping the data updated through the API (2 story points)
12. Sprint II report:
    1. Project Overview
       1. Scrum Master: Matt Bernero
       2. Product Owner: Damien Mi
       3. Team members: Matt Bernero, Caitlin Croke, Caroline Cox, Damien Mi
       4. Estimate of time worked in total: 25 hours
    2. Product Overview
       1. The three main components of the UI, the login, home, and information screens, have been started and implemented into our application. In addition, work on the SQL database has started to allow users to store data on the stocks they are interested in and a search bar has been implemented to browse available stocks.
       2. User Stories Addressed
13. As a user, I want to be able to view a graph of historical data so that I can easily visualize the data.
14. Story Points: 8
15. Design: Show a new screen with a stock's information and a graph with that stock's historical data.
16. Implementation Details: When a button is clicked in the Home Screen, the card changes to display the Information Screen. The stock's name, abbreviation, current value, opening value, high value, low value, close value, and the percent change of the current value from the previous day's value is shown for the stock. The graph displays the historical information for the time period chosen by the user via a combo box.
17. Implementer: Caitlin Croke
18. As a user, I want to be able to research the stocks of my choice and be able to add them to the portfolio so that I can easily access them in the future.
19. Story Points: 8
20. Design: Connected MySQL with Java to try to make the research button work to search for information through the database.
21. Implementer: Damien Mi
22. As a developer, I want to be able to store the data on any stock within a database so that a user can create a portfolio of stocks that they would like to keep track of.
23. Story Points: 16
24. Design: SQL Database accessed through the application using the JDBC DriverManager
25. Implementer: Matt Bernero
26. As a developer, I want to have the API update the database when called, so that I can have the most accurate data when I want it.
27. Story Points: 8
28. Design: Currently, the API is the main source of data at all points. I connected it to the database and it now stores current information into the database when the program is launched, and the program will pull data from the database in the case that the API fails. While working on this, I also sped up the sequence of API calls by eliminating the need for duplicate calls, though the number of calls still being made is still taking more than optimal time.
29. Implementer: Caroline Cox
    1. Integration Testing
       1. Integrating the API with the UI
          1. The UI now shows information obtained directly from the API through a call to the stocks from a given text file. Currently, everything works when the API is working, and everything but the history works when the API fails. The next step is making the historical data work in spite of an API failure.
       2. Integrating the API with the database
30. The API now updates the database when the program is initially run and will default to pulling from the database in the case that the API fails.
    1. State of the Quality of the Product
       1. Currently, the product has the same security measures in place from the previous sprint, and the base layout for the User Interface is detailed, with parts of it created in various stages of development. It does not yet have full functionality in this regard but will eventually be developed. At this time, it is a working piece of software that behaves as expected.
    2. Scrum II Retrospective:
       1. Overall, this sprint was devoted to the development of the product, and we have made excellent progress. Constant communication continues to prove instrumental to the project's successes. It will soon become very important that everyone on the team is familiar with all parts of the system; each team member will need to be able to understand all parts of the project if we are to be able to integrate the parts seamlessly.
       2. Individual Perspectives
          1. Scrum Master’s Perspective: Overall, this sprint was another solid performance by the team. Now that the team has had some experience with the Agile process, we were able to create a clear division of labor and made sure that each team member had enough to do. The group is very autonomous; they work diligently on their assigned tasks, and each team member offers regular updates on their progress. For the future, I foresee that much of our efforts will be devoted to the integration of our individual products, in addition to finishing touches and cleanup.
          2. Product Owner’s Perspective: Now our team works well on the project and just finish most part of the user story. We already have our database and our project is connected to the java so that user can find the stock name which they want in the database. We have our more info page and users can check the information in that page. User can check the graph and identify their stock directly.
          3. Matt Bernero’s Perspective: Aside from Scrum Master, I have also begun work on a database in which we can store the financial data for each stock. Said database can be accessed form the Java application via the JDBC DriverManager. Queries can be successfully executed on this database from our Java application; integrating the database into our screens will need to be implemented in the future. However, the database needs a server to be stored on; otherwise, the application would require potential users to install MySQL server on their machines just to access a database.
          4. Caroline Cox’s Perspective: We made great progress with this sprint in terms of the UI, especially in getting the info screen set up. Retrieving the historical information for the graph is still taking time that should be optimized if possible. I worked mostly on connecting the API to the information screen, as well as optimizing the API in other ways, including starting the integration with a database. The next task to be completed is making the info screen show the stock that is clicked. Currently, it is hard coded to show the information for GOOGL.
          5. Caitlin Croke’s Perspective: With all of the backend work completed in the first Sprint, we were able to make great strides in the second Sprint. I created the Information Screen with displays the stock information and a graph with the stock's historical information. At the moment, the information is static and being added as labels. Later, we will use the API data for each of the stocks to reflect the most current data. Much research was needed into implementing a JFreeChart, but it is currently reading and displaying current and historical data from the API.
          6. Damien Mi’s Perspective: In this sprint I mostly work on the search bar and connect it with the database. We did a good process on this part and user can see their more info page and get the data from the database as we have already connected them. In the next sprint, I will keep working, connect the research bar with the more info page, so that people can check the stock information, and add it to their portfolio. I am looking for a way to connect them smoothly so that user can easily use our system.
       3. Preparations for Sprint III
          1. Scrum Master: Caitlin Croke
          2. Product Owner: Caroline Cox
          3. Goals:
       4. Implement sort functions for the portfolio (4 Story Points)
       5. Integrate all components of the project together (16 Story Points)
       6. Perform unit and integration testing (8 Story Points)
31. Sprint III report:
    1. Project Overview
       1. Scrum Master: Caitlin Croke
       2. Product Owner: Caroline Cox
       3. Team members: Matt Bernero, Caroline Cox, Caitlin Croke, Damien Mi
       4. Estimate of time worked in total: 20 hours
    2. Product Overview
       1. The three main components of the UI, the login, home, and information screens, have been started and implemented into our application. In addition, work on the SQL database has started to allow users to store data on the stocks they are interested in and a search bar has been implemented to browse available stocks.
       2. User Stories Addressed
32. As a user, I want to be able to see the most up to date information possible, so that I have the most accurate information when accessing my financial portfolio.
33. Story Points: 2
34. Design: This required a switch from the AlphaVantage API to the IEX API, which greatly increased the efficiency of the product, and allows for information to be more easily updated. The most updated information can be acquired through using the refresh button, since this can sometimes take a bit of time and is better if the user initiates it.
35. Implementation Details: The JSON parser for the new API remained the same, all that had to switch were a few keys. By implementing the loading screen with the API calls running in the background, the refresh button easily updated all information. The information screen is created each time a button to see it is clicked, and thus displays the most recent information as of when you entered the screen.
36. Implementer: Caroline Cox
37. As a user, I want to see visual indicators for the status of the stock, so that I can quickly analyze the status of my stocks.
38. Story Points: 2
39. Design: When the stock prices have changed, it will show the percentage of its change. It will show the arrow that means whether the stock price goes up or down. And the percentage and the arrow will change, when the price increases, it is green. When it decreases, they will turn red.
40. Implementation Details: Set all the data as string and get rid of the bits which is two bits after the decimal point. Using ActionListener to change the color of the arrow and data under different situation. Using ImageIcon method to add the picture into the project.
41. Implementer: Damien Mi
42. As a user, I want my portfolio to be sortable in multiple ways so that I can customize the visualization of my portfolio.
43. Story Points: 4
44. Design: When a new sort is called, a specific sort function is called which sorts the portfolio according to the selection.
45. Implementation Details: Each sort function for Name, Abbreviation, Current Price, and Percentage is separate and creates a separate sorted array that is then translated into panels and displayed on the home screen.
46. Implementer: Caroline Cox
47. As a user, I want to be able to add, delete, browse, or edit stocks so that I can keep my portfolio up to date.
48. Story Points: 16
49. Design: The portfolio is stored within a text file that is manipulated with BufferedWriter.
50. Implementation Details: The Add/Update button on the More Info Screen allows a user to write the stock name and number of stocks to portfolio.txt using a BufferedWriter. The Delete button calls a function deleteStock; this function creates a temporary file and writes all the stocks to this file... except for the stock that has been chosen for deletion. This temporary file is then renamed portfolio.txt and replaces the old copy. If the Add/Update button was pressed, but the stock is already present in portfolio.txt, then deleteStock is called on said stock, and it is re-appended to the text file.
51. Implementer: Matt Bernero
    1. Integration Testing
       1. All of the components have now been implemented and are working together cohesively. The Home Screen allows the user to search the API using either the stock's name or abbreviation or returns a label indicating that no results were found if the user has entered a stock name or abbreviation that is not supported by the API or does not exist. The Home Screen also displays the user's saved stocks. The Info Screen allows the user to add a particular stock to their portfolio and update the number of shares owned. They may also choose to delete the stock from their portfolio on this screen. A Splash Screen has been implemented to improve the user experience when moving between screens that require information to be updated. It is shown when the API or user portfolio is being refreshed so the user has visual feedback as to what the program is doing.
    2. State of the Quality of the Product
       1. The product is now functionally complete. All of the user stories set forth at the beginning of the project have either been fulfilled or it was agreed upon to be dropped. This happened in the instance of the implementation of a database, as its usage and installation were deemed unwieldy and not optimal for the project. All members of the team, including the Product Owner and the Scrum Master, agreed that it was best to drop this story from the project.
       2. There are several minor fixes that may be performed, if time allows. However, these are mostly aesthetic alterations and minor non-critical bugs. Additional optimization may also be performed in formatting the code itself, such as removing print statements used for debugging and other commented out code that will no longer be used.
    3. Scrum III Retrospective:
       1. Overall, this sprint was very productive, and the final touches have been added. Regular meetings and open communication allowed for smooth coding and debugging. Each member was friendly and willing to explain their work to the other members of the team. In doing so, each member was able to integrate his or her own work seamlessly into the final version of the product.
       2. Individual Perspectives
          1. Scrum Master’s Perspective: Cooperation was paramount in this final sprint and the team worked very well together. There was a lot of communication about what each of the team members was doing, which reduced errors and any overlap in the code. Everyone was helpful and responsive to the critiques of their fellow team members on their code any bugs that arose. I am proud of what our team has accomplished and the product we have produced.
          2. Product Owner’s Perspective: As product owner, I believe that the final product created fits into the specifications that were laid out at the start. All requirements were met, and the User Interface is very intuitive, making for a pleasant user experience. By redesigning the API, the speed of the program improved greatly, and the addition of a loading screen helped with user experience when actions took a little bit longer. As product owner, I believe that the product is in a condition to be released.
          3. Matt Bernero’s Perspective: For our final sprint, I worked on implementing a portfolio that keeps track of the names and the number of shares of each stock a user invests in. Initially, I had created a SQL database last sprint with the intent of storing the data there, but I found that there was no way to access this database without first creating a SQL server to store the database on. The team agreed that it was simply unfeasible to expect a user to download and run server software just to keep track of their stocks; we decided to store the portfolio in a text file that the program can read and write to via a group of buttons on the Info Screen. While we lost some potential offline functionality, the team agreed that keeping our schedule was more important than what amounted to a "nice to have" feature.
          4. Caroline Cox’s Perspective: This sprint, I switched the API from using AlphaVantage to IEX. After researching options to optimize the time it took for the program to parse the AlphaVantage data correctly, I decided to see how easily I could switch to a new one that would require less API calls to get the same data. It ended up being a decently painless switch. Other than API changes, I worked on the ability to sort your portfolio on the main screen, as well as searching for stocks. I think that this sprint went very well and the product that we created meets all expectations.
          5. Caitlin Croke’s Perspective: Most of my work for the final sprint was fixing any GUI bugs and formatting the implementations in the Home Screen. I set up the two panels in the Home Screen to be of equal size, regardless of the dimensions of the frame. I also had to set up the format of the panels with in the ScrollPanes to display properly, as the number of results would affect the size of the panels due to the layout manager overriding the preferred sizing. I also edited the layout of the Splash Screen to be more proportionate and readable by the user.
          6. Damien Mi’s Perspective: In this sprint, I edited the format of the price that shows to the customer. I used String format to set that the price only shows two digits after the decimal point. Then I set the picture of the arrow as an ImageIcon and build a path to let the project get the image. As we need to change the color of the price, we add an ActionListener to it. When the price is decreasing, we change the color of the number into red and use the red arrow picture, and when the price is increasing, we change the color into green and use the green arrow picture.
52. Final Report
    1. Further Development
       1. Profile Management
          1. Multiple Users – Currently, the application only supports a single user. In a future update the application would support more than multiple users. A database would be set up to hold the users’ login information as well as their portfolio information. This would require the user to create an account by choosing a unique username and setting a password. This would also require changes on how user profiles are managed with the saved and deleted stocks.
          2. Update Password – Currently, there is no way to change or update the password from the default string. The update would allow the user to enter a new password, updating it from the default. In this way, the user would have a more secure and personalized experience with the product.
          3. Stock Summary – An additional feature to be added in the future would be a summary page or report. This report would display changes in the favorited stocks held within the user’s portfolio as well as a list of all of the stocks currently held within the user’s portfolio.
       2. Automatic Interface Updates: Currently, every time the user makes a change to his or her profile, the data within the code is updated but the refresh button on the home screen must be clicked before it is displayed to the user. A future update would still include a refresh button, but the UI would be updated without any action from the user. The refresh button would still update the stock information from the API, but any additions, deletions, or edits to the pre-existing stocks in the user profile would appear automatically.
    2. Final Reflections:
       1. Technical: The team learned a great deal about the workings of Git, Maven, and API implementation. Very few members of the team had previous experience working with Git. The experienced members were willing to instruct the unexperienced members on how push and pull code, as well as some of the tools and shortcuts that make using Git much more efficient. In doing so, all the members of the team were effectively able to use the Git repository, updating and uploading the code as it was created and implemented, allowing for the most up-to-date project to be available to the entire team. Creating a Maven project allowed the team to implement the use of .jar files within the project and utilize services, such as JFreeChart. Although some members of the team were initially unfamiliar as to how to use and run Maven projects, other members of the team were able to explain their function and walked through how to compile and run the project. Additionally, the team was able to become more familiar with using an API and integrating it into use with a Java-made GUI.
       2. Interpersonal: The team has learned much about the importance of communication, both written and verbal. In addition, each teammate learned about the importance of fulfilling their assigned tasks. Each individual component of the project is dependent on multiple, additional components; failing to accomplish our individual tasks put the rest of the team at a disadvantage. Communication allowed for each member of the team to inform the others as to the progress and/or status of each task. This was especially important towards the end of the project when the tasks being worked on by individual members began to overlap more and relied heavily on each other. All the members of the team were open to critiques and criticisms to his or her own code and professional when making those criticisms, each making their own changes and improvements when bugs and shortcomings were brought up.
    3. Github Repository URL: <https://github.com/ccroke2/RuseCruise>
    4. Walkthrough/Demonstration

* Starting up the program opens to the login screen. It contains a prompt for the user to enter his or her password:



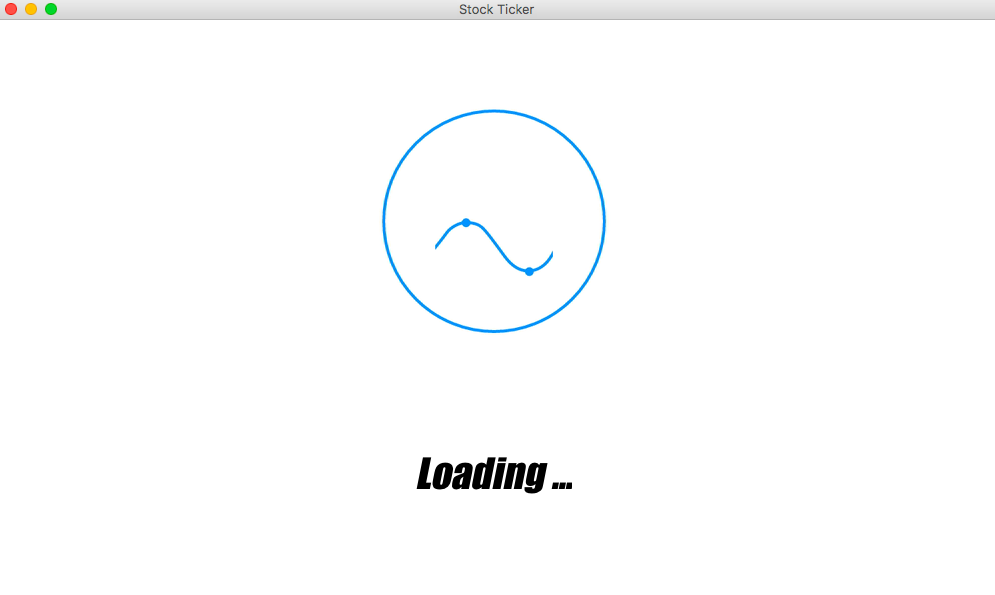
* Since there is only functionality for one user, click the "New User?" button to retrieve the password. A pop up will appear:



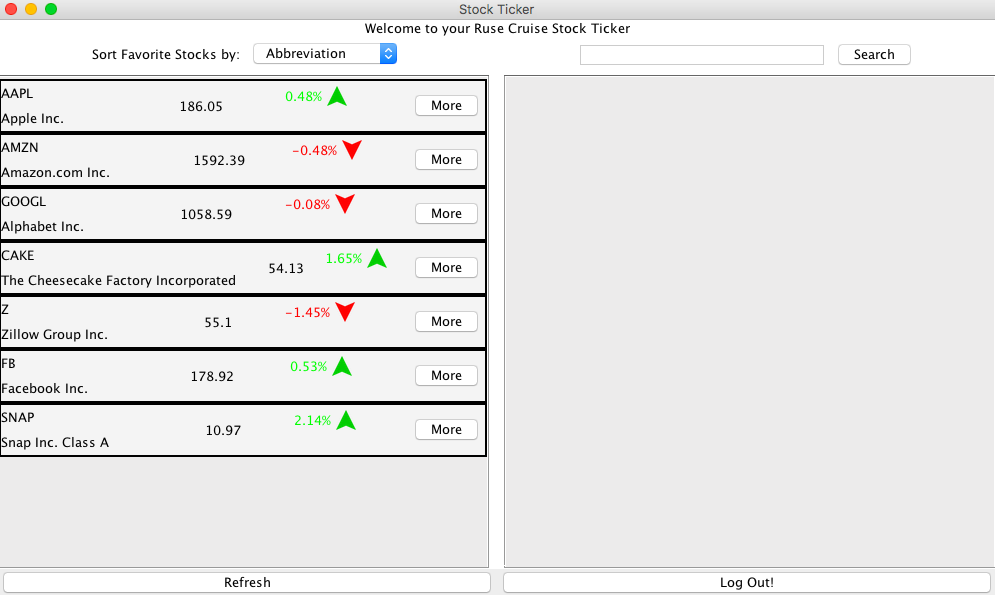
* If the password is entered incorrectly, a pop up will appear to indicate the password is incorrect and the password field is cleared:



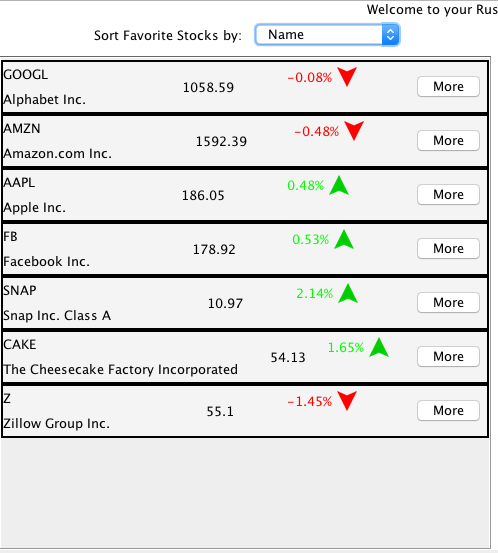
* Upon entering the password correctly, the loading screen will appear:



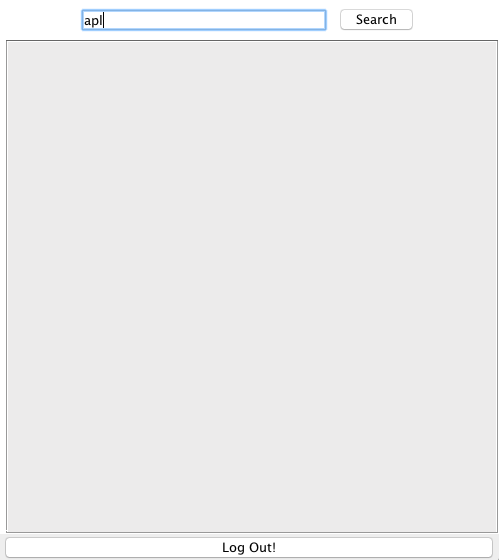
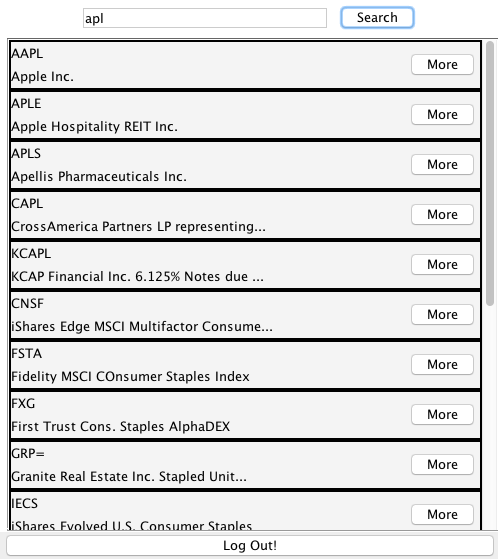
* Once loaded, the portfolio display is shown with the user’s favorite stocks on the left panel and the search functionality on the right panel:



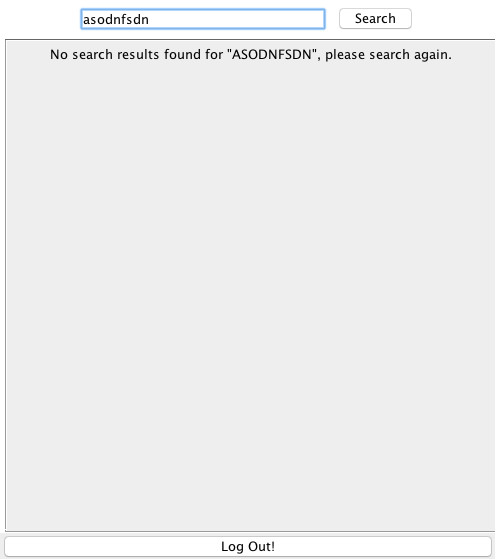
* Sort functionality for the portfolio:

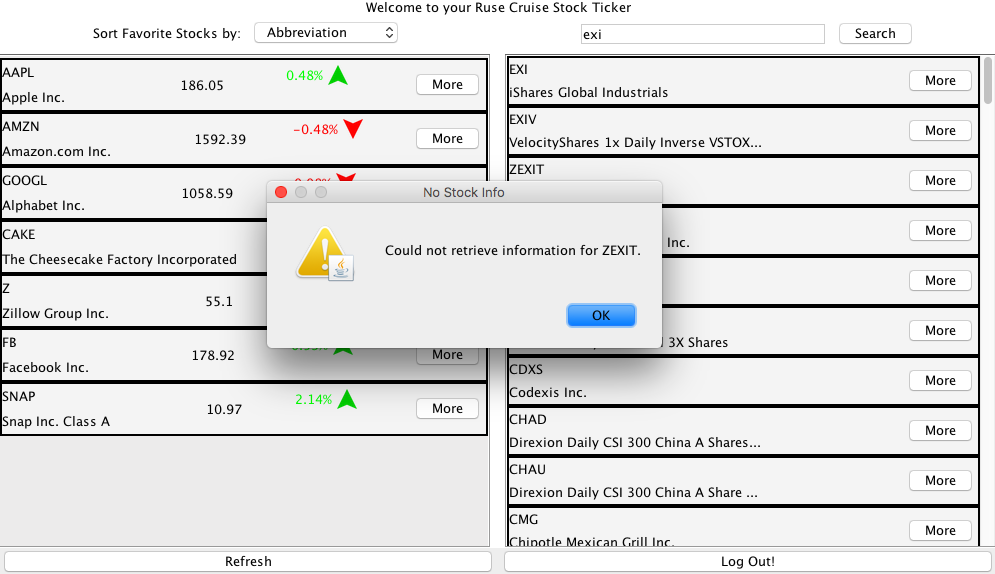
 

* Search functionality:

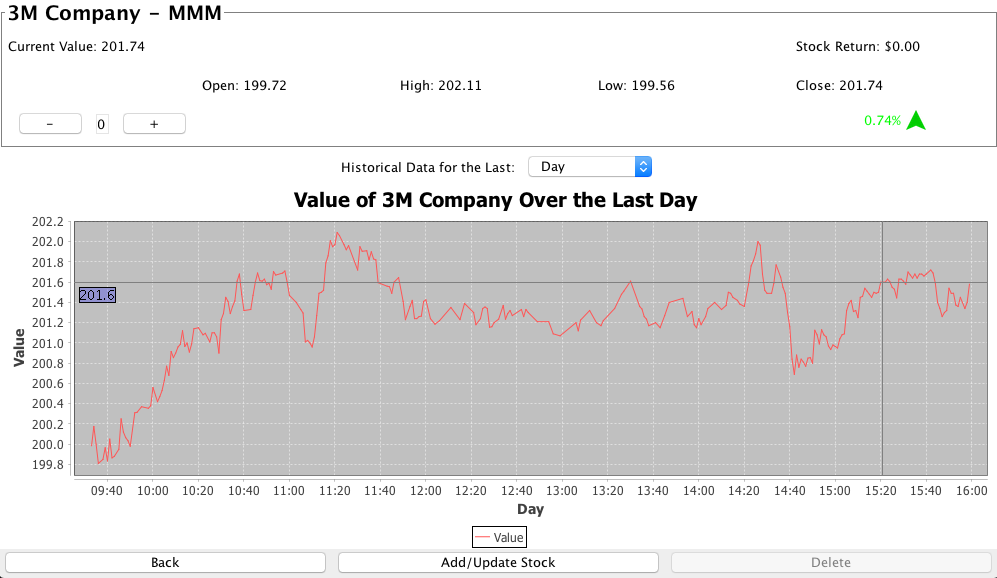
 

* If the search yields no results, or if the API returns a null value for the stock you select, appropriate error messages appear:

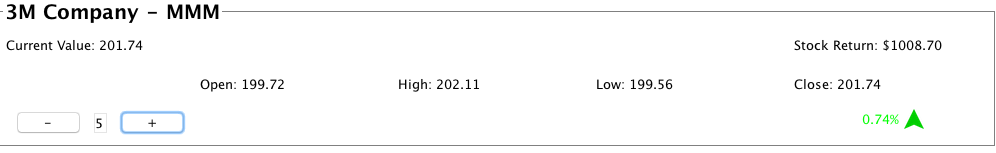




* Upon clicking "More" in either the search or portfolio panes, the information for the stock appears in a new screen:



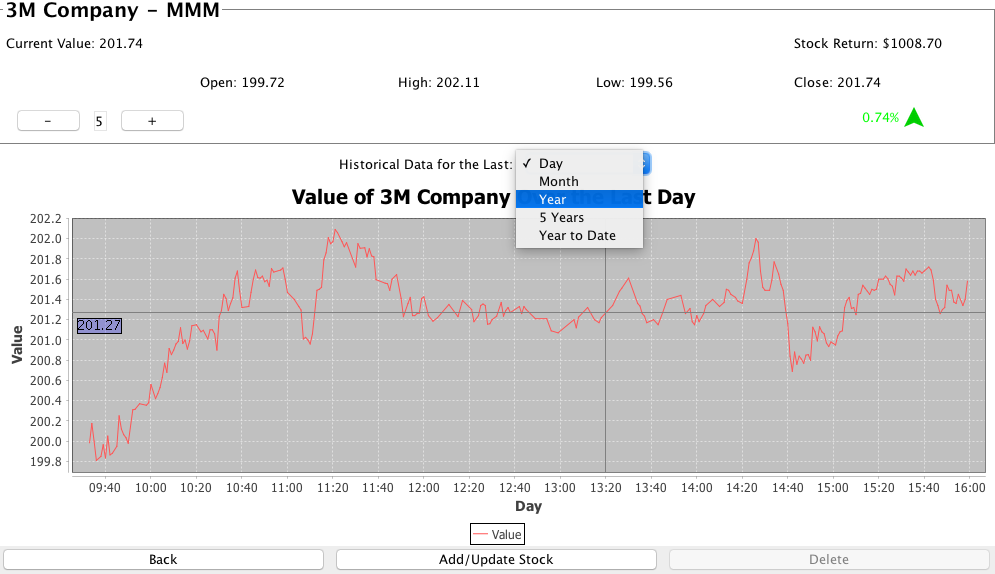
* The (+) and (-) buttons can be clicked to add or remove the number of shares from the portfolio and is updated with the use of the Add/Update Stock button:

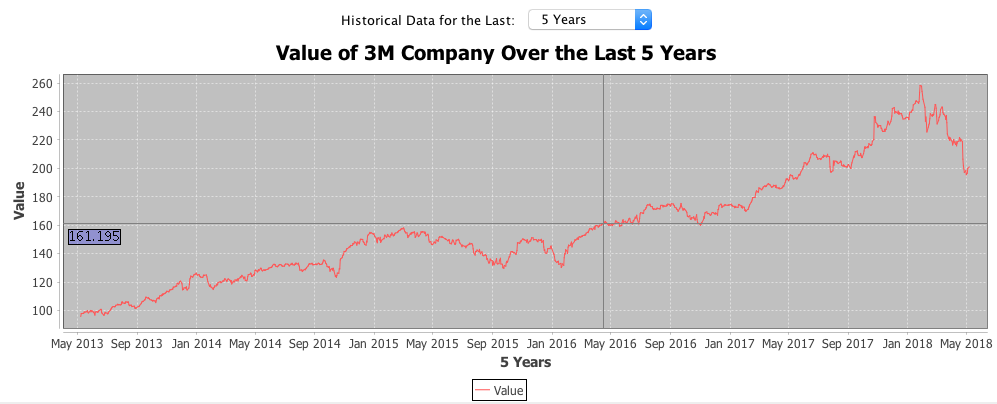


* Once added, refresh the portfolio (see loading screen) and it will be in the list of saved stocks:

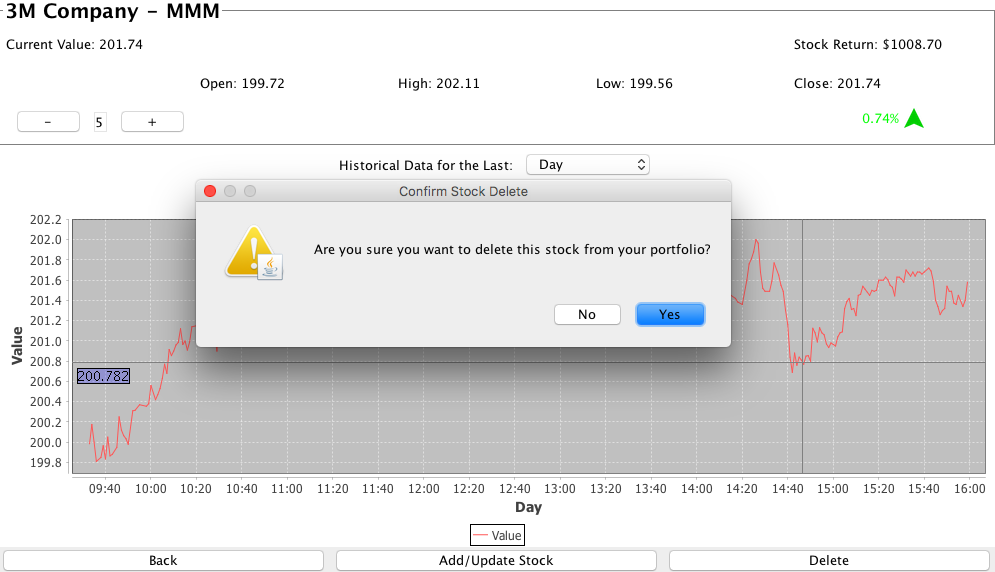


* Back in the info screen, the graphs are selectable for different lengths of time (day and 5 years shown below):





* If a stock is in the portfolio, the user may choose to delete it:



* Once the user is done with the application, he or she can log out by clicking the button labeled "Log Out!" on the home screen.