**Scrum Report Format**

* Prologue
  + Team Name: Nird
  + Team Slogan: Nirds of a feather
  + Team Members: Patrick Gallagher, Spencer Russell, Tianyi Zhang, and Hank Zhu
  + Project Name: Stock Ticker

Sprint 2 Report:

2.1 Teammates Roles:

* Scrum Master: Hank Zhu
* Product Owner: Patrick Gallagher
* Team Members: Spencer Russell and Tianyi Zhang
* Combined Hours on the job: ~20 hours

2.2 User Stories

* After the connection of the API and made the GUI we started to add functionality and data into the project. The functions in the project is allowing the user to buy the stocks after looking at the data of the graph. Also, the project includes the data of all the stocks in the API. Including the data for the users allows them to have a good understanding of the stock. The user stories that the team have worked on were:
  + As a user, I want multiple data displays when I am looking at a stock. So that I can have an easier time looking at a stock. – 4pts.
  + As a user, I want a search function when looking for stocks to add, so that I can find stocks easier. – 2pts.
  + As an investor, I want to keep hold of my personal shares. So that I can monitor my own financial gain. – 4pts

2.3 Design, Requirements, Test Plan (Data User Story)

* In Sprint 2, we plan to have another graph that presents the data in the past seven trading days, in the same format as the DailyGraph we developed in the last Sprint. We will revise the graph classes to panel classes so that they can be added to the frames. But we will still have method that can run the graph individually for the convenience of testing them.

2.3 Implementation details and issues (Data User Story)

* The team worked out a new java class called “GetStock.java”. It wrapps the API functions in three method, “getEverHour”, “getEverDay”, “getEveryMonth”. These methods return a object of List<StockData> at every given time nodes. It provides convenience for every team members in future development. Because the programmers can simply create a instance of object of GetStock and call the method to get the list of stock data they need.

2.3 Statement of outcomes (Data User Story)

* The Daily and Weekly graphs works fine.

2.3 Names of Implementer(s) (Data User Story)

* Tianyi Zhang, Spencer Russell

2.3 Design, Requirements, Test Plan (Search User Story)

* The search function is a pop-up window that allows users to enter the keywords that they can think about, even though it is not as powerful as Google search, but I managed to ignore the case-sensitive to allow users having more abstract searches.

2.3 Implementation details and issues (Search User Story)

* The implementation part is nearly perfect, we built a user-friendly search bar and nice display layout, with scroll bar on the both right and bottom that allows users have more flexible way of checking their search results. One of the issues we have right now is displaying the searching result as desired, since the database file is not perfectly organized. Another issues is the layout, when trying to implement the favorite function to the search results that allows users to add the stock to their favorites where the buttons do not line-up perfectly as desired.

2.3 Statement of outcomes and Names of Implementer(s) (Search User Story)

* The search function allows user to search repeatedly if the result was not desired, in order to minimize the latency of searching time, I wrote the code the read the file from the local database instead of online API, the file will only contain names of the stock, acronym of stock, and type of the stock, which are the only information required in order to proceed the next step, the step is get the detail information of the stock by reading the stock’s acronym.

2.3 Names of Implementer(s) (Search User Story)

* Hank Zhu

2.3 Design, Requirements, Test Plan (Add Stock User Story)

* During the second sprint, the team wanted to allow the users to add the stocks after they saw the data for the stock. They thought it was a user friendly action, the programmers did not want to fool the user into buying the stock first time searching the stock, but after they saw the information of the stock is when the users can buy the stocks. The requirement for the story is that the users will always have the information of the stock that they searched for before making the decision of the buying the stocks. The testing in this story was when the user hit the add button in the display class. Once the add button was hit, the stock would be written in a text file and be read in the Stocks class.

2.3 Implementation Details and Issues (Add Stock User Story)

* The group implemented text files into the project because they thought it would be easier to implement instead of a database. So far the project can only add the facebook stock, but in the next sprint the project will add the correct stock that the user inputted. The biggest issue of adding the stocks is that once the program comes back to the main menu, the list does not update itself and will not show the newly added stock. The user has to exit out of the program and run it again.

2.3 Statement of outcomes (Add Stock User Story)

* The outcome of the story is that after the user adds their stock, the stock name will be copied to a text file. The text file shows how much the user has bought and prints out to user in a form of buttons. The buttons will go to the display frame, displaying the information.

2.3 Names of Implementer(s) (Add Stock User Story)

* Patrick Gallagher, Spencer Russell

2.4 Integration Testing

* The integration testing that went into this sprint was that the graphs that were implemented into the display graph. After the testing of the graphs being a separate class, the program started to implement the graphs into the display frame. The second integration test was after testing the display frame writing stock names into a text file, the stock frame started to read from the text file and printing the buttons from the text file.

2.5 Scrum 2 Retrospective

* Spencer’s Perspective: In this sprint, I worked with Tianyi to display his graphs in the stock information window and to paint non-numeric data along the side of the graph. There were a few issues with populating the fields as well as the inability to switch between graphs. I am still working towards the ability to record new stocks to the user’s list which when completely implemented would allow us to begin working towards non-preset examples. I am working towards improving my java skills to no longer hinder my teammates. Team work has significantly improved from the first sprint, which has helped us with completing our tasks.
* Tianyi’s Perspective: In this sprint, my responsibility was to make the monthly graph. During the process, we encountered some difficulties such has having the stock information frame and the graph connected, and we still having difficulty reading data from the API. The GetStock.java class helps solving that trouble. The graphs can be improved in some way, for instance, having more time nodes so the trend lines will look more smooth. But it requires a lot of work. The team is not decided on whether we should implement this change.
* Patrick’s Perspective: In this sprint, I was working on the functionality of the add button in the display frame. I implemented a text file to keep track of the all of the added stocks that the user has bought. I am still working on how to update the stock frame after the user has bought the stock, showing the newly added stock button. The team have worked a lot in this sprint than last sprint.
* Hank’s Perspective: In this sprint, what I did looks pretty easy, because it just a search function that linked to the stock database, all cI did is creating a “for” loop that keep checking the user input, and tells whether it matches. I found a method that ignore the case-sensitive, which allows users to have more abstract search when users do not what exactly they wants. Secondly, when I trying to implement the “favorite” button to the search function, the display layout becomes the problems, I could not find the perfect way of lining up all the buttons. But I will fix it at the next spring.

2.5 Product Owner Statement:

* The end of this sprint the program can now print out some of the data of the stock, even though the program can only print a predetermined stock, but they are currently working on a way to have a connection between their two frames. The display frame also has a predetermined stock, but it also displays the daily graph and the current price of closing and the stock name. There are some choppy parts in this project that needs to be fixed like the ability to update the stock frame to correct number of stocks without closing and rerunning the application.

2.5 Scrum Master Statement:

* Everyone has worked on this project around 20 hours combined in this sprint, which demonstrates the commitment that everyone putted in this project.
* There are few details that we are focusing on during this sprint.
  + Connecting API to the every file of the project.
  + Making the user-interface looks better.
  + Displaying the graph based on hours, days, and months.
  + Integrating the search function to the detailed stock information panel.
* Overall, everyone did a wonderful job, especially Tianyi who did an excellent job of helping his teammates for the API and the stock graph.

2.5 Set up for Sprint 3:

* As a user, I want a help function for the application. So that I know exactly what the program is doing. – 8pts.
* As a user, I want to make a password for the application, so that I can protect my stocks. – 16pts.
* As a user, I want to compare the same stock from two different years. So that I can see how the stock does in long term. – 4pts.
* Also starting to debug and make the program look nicer than it is now

·

Signatures:

Scrum Master Name (Printed) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Scrum Master Name (Signature) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Product Owner Name (Printed) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Product Owner Name (Signature) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Member Name (Printed) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Member Name (Signature) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Member Name (Printed) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Team Member Name (Signature) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_