CS308: Folio Tracker General Analysis

Group 27: Ross Lyons (201724258), Syeda Ayela Gilani (201810821), Karl Rivett (201727785), Ismael Ahmed (201707978), Koni Watson (201710203)

Application Overview-

The general purpose of this application is to allow users to maintain and control their stock portfolios through a graphical user interface (GUI).
This user control being in the form of creating new folios where stocks can be added, edited or deleted from that current folio. The application even allows users to create or edit multiple folios simultaneously. The stocks added by the user are real stocks, whose prices are automatically updated after every 30 second interval. These stocks are retrieved from the server which then populates the table when added. The folios created can be saved and closed where the user can then re-open the folio at a later date if they do so wish. Folios can be entirely deleted, resulting in a complete removal of the folio from the system and the saved file if it had been saved initially.
This is done through a combination of Java and JavaFX, following the MVC (Model-View-Controller) Model.

Model:

The model is part of the system that handles all the data-related logic that will be used to populate the GUI, this being the back-end API designed. The model contains five classes; the StrathQuoteServer, Stock, Holding, Folio and FolioHolder.
The Stock class utilises the StrathQuoteServer class to get real Stock prices from the NYSE when provided with a ticker symbol. The Holding class contains a StockInterface object, the number of shares for a Stock object, and the total value of the Holding. From the Holding class you can get the price, total value, any profit made, and check if the price has risen/dropped recently for any Stock object. The Folio class contains a list of HoldingInterface objects. This class allows you to add/delete Holdings. The final class FolioHolder contains a list of FolioInterface objects. Within this class you can create/delete/save/load Folios. This class acts as the root to all the other classes. We decided to do this so that when more Folios/Holdings are created, the program would branch out and every object would be accessible via the single FolioHolder object. Initially, we intended to only have the FolioHolder, Folio and Stock as our main classes. However, later on, we decided to add the Holding class as in the long run, it made accessing details for the View much easier. We also have four interfaces, namely FolioHolder, Folio, Holding and Stock, which are used to achieve abstraction of the classes that implement them.

View:

This section is used for the user interface logic of the application. This obviously being the Graphical User Interface (GUI) in this case. This application uses the functionality of JavaFX in combination with Scene Builder to create the GUI. Within the Scene Builder software, an easy structure of a GUI can be constructed. From this program a .fxml file is produced that represents the GUI, along with a Controller.java class which is automatically created with the pre-defined elements from the .fxml file filled within. The View was decided to be built this way to enable us to create a simple and straight-forward user interface that users can navigate through with ease. It is also known that JavaFX can be styled very nicely and even use the additional styling from a .css file. Due to this we were able to produce a pleasant and easy to use GUI that follows the application purpose.

Controller:

The controller acts as the piping behind the scenes. It is the connection between the Model and View components. It processes things like incoming requests from the View and the data used within the Model, respectively concatenating the two, to allow for the GUI to communicate with the back end and vice versa. This then produces the final output that is rendered to the user. The controller is not only a vital part of the MVC model but is also required to help ensure that the back-end and GUI stay decoupled.

Through the methods described, our application covers the following basic functionality:

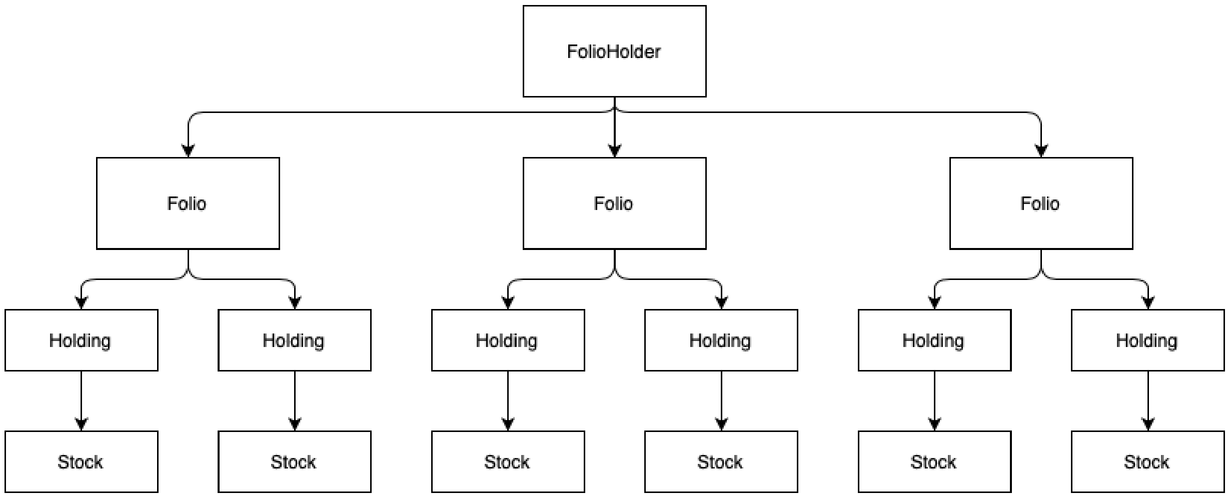
- Managing multiple portfolios
- Creation/Deletion of portfolios
- Viewing Ticker Symbol, Stock Name, No. of Shares, Price Per Share and the Value of the Holding
- Addition/Removal of stock
- Editing stocks, i.e. buying/selling shares
- Refreshing stock prices
- Total value of a folio as a whole

The application also covers these additional functions:

- Saving folio to an external file
- Indicating where a stock's price is increasing/unchanged/decreasing
- Sorting the stocks held in the table by its given value (Ticker Symbol, Stock Name etc)
- Keeping track of highs and lows and displaying them for user view.

Back-End API Design-

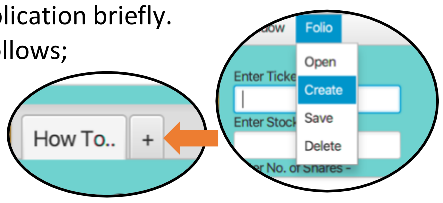
The back-end API's structure is as follows:



GUI Manual-

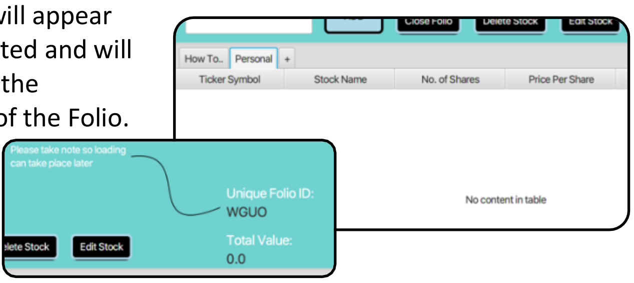
Our GUI looks like the following. The functionality of which is simple and easy to use/understand. The GUI itself has a "How To" tab which highlights how to use the application briefly.
A more in-depth instruction on how to use the application is as follows;

1. Start of by creating a new folio via the Folio tab in the top-left folio menu or the '+' tab. Fill in the folio name pop up with your desired name.



----

2. When the new Folio is created, a tab will appear with an empty table. Folio ID is generated and will be displayed on the right-hand side of the application along with the total value of the Folio.



Please take note so loading can take place later

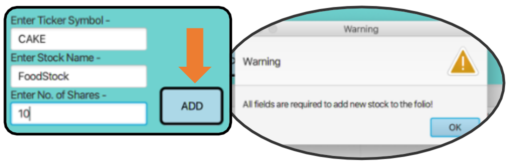
Unique Folio ID:
WGUO

Total Value:
0.0

No content in table

----

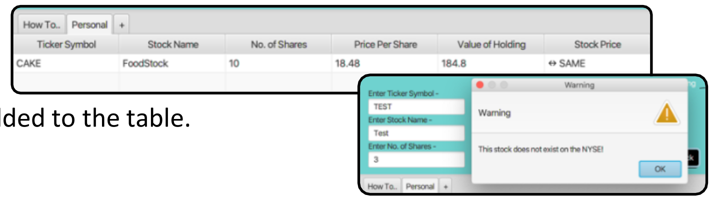
3. Stock can be added via the text boxes. These text boxes take in information for the ticker symbol, name of the stock and the number of shares to be purchased. Formats check help users to provide correct information.


Enter Ticker Symbol - CAKE
Enter Stock Name - FoodStock
Enter No. of Shares - 10

Warning
All fields are required to add new stock to the folio!

----

4. If the Stock exists, it will be added and the respective information about that Stock will be added to the table. If the Stock does not exist, the user will be informed, and nothing will be added to the table.



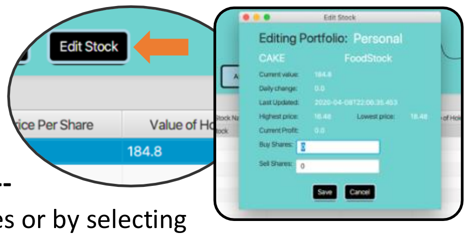
Ticker Symbol	Stock Name	No. of Shares	Price Per Share	Value of Holding	Stock Price
CAKE	FoodStock	10	18.48	184.8	↔ SAME

Enter Ticker Symbol - TEST
Enter Stock Name - Test
Enter No. of Shares - 3

Warning
This stock does not exist on the NYSE!

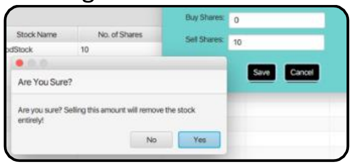
----

5. When the Folio is populated with Stock, the user is free to edit the Stock as they wish. This is done by selecting the desired Stock followed by the 'edit stock' button. When doing so, additional information about that Stock can be seen. This is also where the number of shares can be altered i.e. sold/bought.

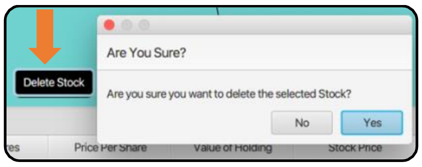

Editing Portfolio: Personal
CAKE FoodStock
Current value: 184.8
Daily change: 0.0
Last Updated: 2020-04-08T22:06:35.453
Highest price: 18.48 Lowest price: 18.48
Current Profit: 0.0
Buy Shares:
Sell Shares: 0

----

6. Stocks can be deleted completely by editing the Stock to have zero shares or by selecting the Stock and clicking the delete Stock button.

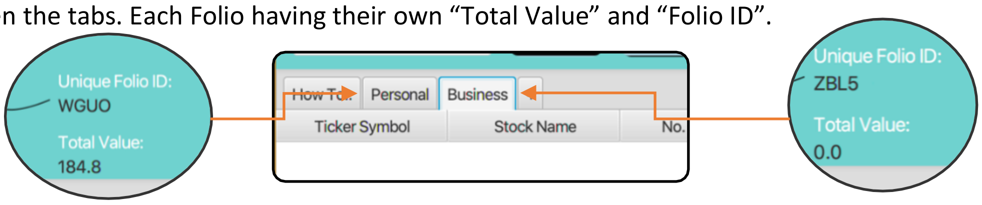

Buy Shares: 0
Sell Shares: 10

Are You Sure?
Are you sure? Selling this amount will remove the stock entirely!


Are You Sure?
Are you sure you want to delete the selected Stock?

----

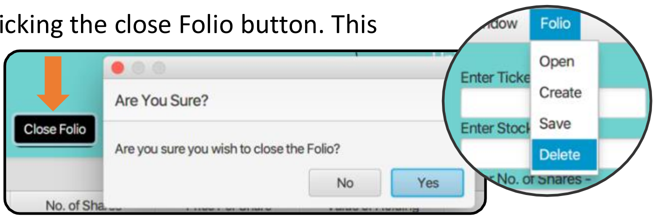
7. Multiple Folios can be maintained at the same time by just creating a new one and flicking between the tabs. Each Folio having their own "Total Value" and "Folio ID".


Unique Folio ID: WGUO
Total Value: 184.8

Unique Folio ID: ZBL5
Total Value: 0.0

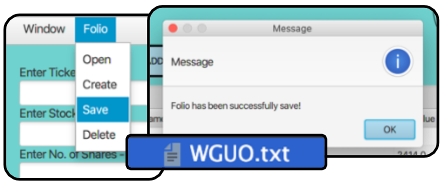
----

8. These folios can be closed at any time by clicking the close Folio button. This only closes the tab, however, to proper delete a folio, it is best to use the 'delete' option from the Folio menu in the top left.


Are You Sure?
Are you sure you wish to close the Folio?

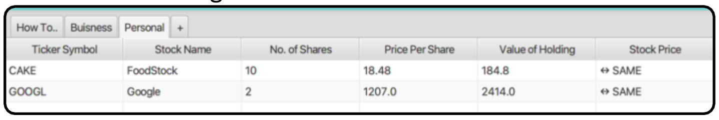
----

9. Within the Folio menu in the top left, a Folio can also be saved. This saves the Folio and the stocks within it to a .txt file.


Message
Folio has been successfully save!

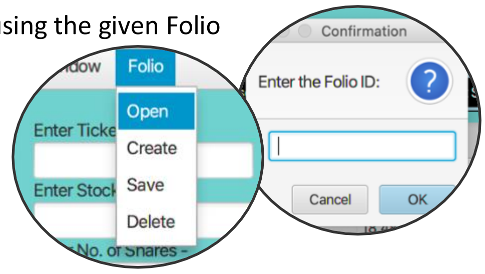
---

10. In this menu again, a Folio that has been saved can be opened, using the given Folio ID for that Folio, via the ‘open’ menu option. This opens up a new tab containing the saved Folios information.



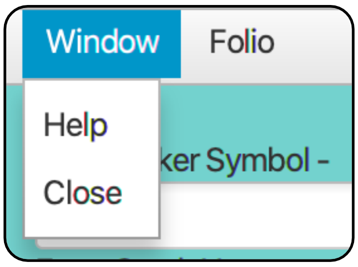
How To.. | Buisness | Personal | +

Ticker Symbol	Stock Name	No. of Shares	Price Per Share	Value of Holding	Stock Price
CAKE	FoodStock	10	18.48	184.8	⇔ SAME
GOOGL	Google	2	1207.0	2414.0	⇔ SAME



---

11. When you are finished, the application can be closed via the “window” menu in the top left. Clicking the ‘close’ menu option will quit the application.
There is also a ‘help’ option that will re-open the “How To” tab. This is to allow users who have accidentally closed the “How to” tab to regain access to helpful information.


Window
Folio
Help
Close
ker Symbol -
