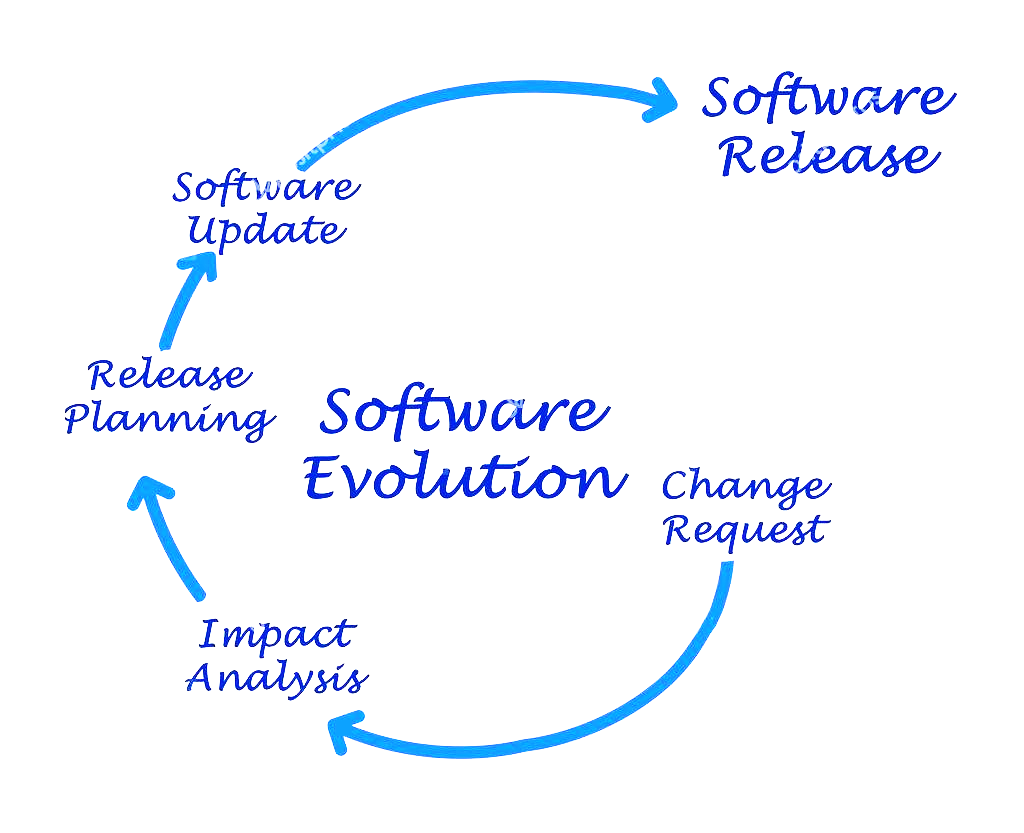
**(Group2**)’s **Software Evolution Project Final Report**

Group: 2  
Dr Adil Al-Yasiri  
Software Evolution   
University of Salford – 2018



|  |
| --- |
| **Table of Content** |

[**Table of Content 1**](#_Toc511928560)

[**Group Members 2**](#_Toc511928561)

[**Introduction 2**](#_Toc511928562)

[**Project Planning 3**](#_Toc511928563)

[**Sprint’s development processes 4**](#_Toc511928564)

[**System Design 5**](#_Toc511928565)

[**Appendices 11**](#_Toc511928566)

[**User’s Guide 12**](#_Toc511928567)

[**Solution implementation 12**](#_Toc511928568)

[**Conclusions and Reflections 13**](#_Toc511928569)

[**References 13**](#_Toc511928570)

|  |
| --- |
| **Group Members** |

|  |
| --- |
| * Zahera Mohamad |
| * Alweh Almohsin |
| * Uddin Forhad |

|  |
| --- |
| **Introduction** |

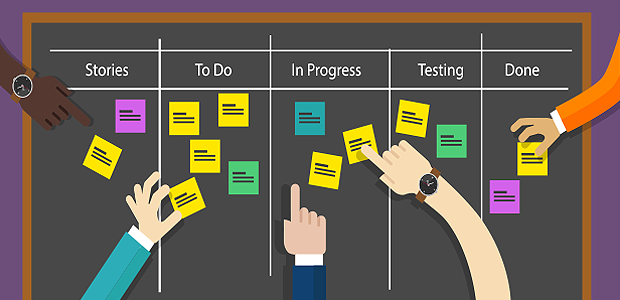
This report is an overview of the group teamwork of a software evolution and software maintenance project that has been practiced throughout the semester, to ensure the two software’s phases has been demonstrate in the way it should.

The essay basically divides into three main paragraphs firstly the introduction, then the main body and finally the conclusion. The body will be divided into subparagraphs that will demonstrate the whole journey of the project with the evidences of the work that has been done by the group members and all the modifications on the system.

The project is based on a security trading system (built in Python) that is used by a financial broker firm the customer and the traders of the firm, by way of a command line tool to submit orders to the market of execution. The team was provided with the system source code and all information needed to modify the system to an evolutional software that is qualified and eligible to maintenance in the future with new requirements and new technologies, in addition to that be able to have the longest lifecycle a software can have.

Throughout the essay, it will be explained how the group employed Agile methodologies to rebuild and modify the software in a three-week period. The software includes all requirements as requested and has been modified in the best way it could, to be able to carry the features of the software evolution and software maintenance.

This report has been written in a very simple, easy, organized and user-friendly way, mostly illustrated through tables, photos and diagrams.

Firstly, in the beginning of the body, there will be a calendar explaining how the group divided the sprints and how the team organised and managed the time, following by the section of “Sprint Development Processes” that demonstrating each of the group members’ tasks by a large table with all process of planning, designing, implementing and coding throughout the length of the project.

Secondly, in the next page, the software solution has been created by following the reverse engineering which produces an architectural diagram, use case diagrams and class diagrams of both before and after system’s modifications. Following by all evidences, screenshots and more information of the modified application.

|  |
| --- |
| **Project Planning** |

The group members did split the project into three sprints with each of a four-day long.

* **Sprint 0**
* 1 day
* As a start out, the group decided to have a sprint 0 to access the requirements documents provided by the lecturer and discuss it as a team, to ensure that everything is straight and clear to all members of group and that the team have a solid understanding of the brief. As a result, the backlogs for the sprints have been created and was ready to prepare it for splitting them off into tasks for each member of the group to tackle.
* **Sprint 1, 2 and 3**
* 4 days
* As the initial requirements document was very well detailed and included enough information to accurately rebuild and modify the software, the team did decide for three sprints, each with four-days in length.
* **Final Report**
* Maximum of 10 days

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **April 2018** | | | | | | | | | | | | | | | | | | | | | | |
| **Tu** | **We** | | **Th** | **Fr** | **Sa** | **Su** | **Mo** | **Tu** | **We** | | **Th** | **Fr** | **Sa** | **Su** | **Mo** | | **Tu** | | **We** | | **Tu** |
| 10  Start of the project  Sprint **0** | 11  Sprint **1** | | 12  Sprint **1** | 13  Sprint **1** | 14 | 15 | 16  Sprint **1** | 17  Sprint **2** | 18  Sprint **2** | | 19  Sprint **2** | 20  Sprint **2** | 21 | 22 | 23  Sprint **3** | | 24  Sprint **3** | | 25  Sprint **3** | | 26  Sprint **3** |
|  | | | | | | | | | | | | | | | | | | | | | | |
| **Fr** | **Sa** | **Su** | **Mo** | **Tu** | **We** | | **Th** | **Fr** | **Sa** | **Su** | **Mo** | **Tu** | **We** | | | **Tu** | | **We** | |
| 27  Report | 28 | 29 | 30  Report | 1  Report | 2  Report | | 3  Report | 4  Report | 5 | 6 | 7  Report | 8  Report | 9  Report | | | 10  Report | | 11  End of the project | |

|  |
| --- |
| **Sprint’s development processes** |

* **The table below will illustrate the process of how each sprint was developed.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint 1** | | | |
| * Sprint goal |  | | |
| * Sprint backlog |  | | |
| * Sprint plan |  | | |
| * Sprint review meeting |  | | |
| * Sprint retrospective |  | | |
| **Sprint 2** | | **Sprint 3** | |
| * Sprint goal |  | * Sprint goal |  |
| * Sprint backlog |  | * Sprint backlog |  |
| * Sprint plan |  | * Sprint plan |  |
| * Sprint review meeting |  | * Sprint review meeting |  |
| * Sprint retrospective |  | * Sprint retrospective |  |

**More Details:**

1. **Sprint 1:**

By looking at the requirements document, it was obvious that without the implementation of security class, it is not possible to move to the other requirements. Therefore, the team decided to firstly implement the Security class in sprint 1.

To be able to create the best structured class that would be able to hold the features of the software evolution and maintain its functionality in future maintenance, the best solution was to create a class security that holds the attributes of an security which is obviously, the name, the symbol, the sector and the industry of a particular security, and a class that will be the responsible aspect of these securities in general and manage them even in the future, taking in mind all cases and changes that may occur. For example, if a security does not exist in feature or if a new security is presented in the broker trade system or any other future modifications in the trading system.

**Sprint 2:**

After creating the Security class, it was logical to move on to the next requirements to make improvements to the system.

The securities trading system has used its internal data representation to print out any information that are required by the users, for example if a trader ask to list all clients the system will used the clients file to print out the saved data directly which is not very user friendly

furthermore, clients can trade by asking any price and their orders are executed by the system regardless of the current market value.

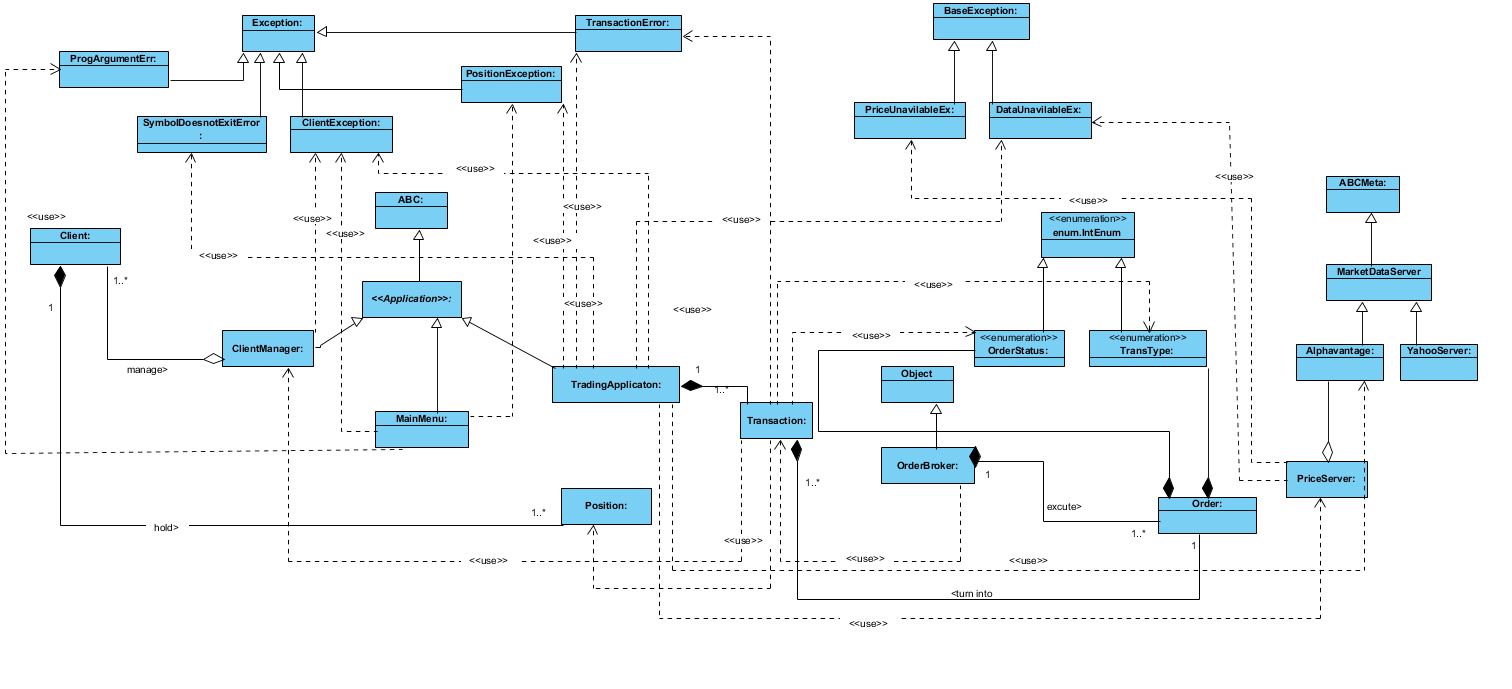
By considering the impact of these faults on the system including safety and security implications the team has implemented a more realistic

order execution.

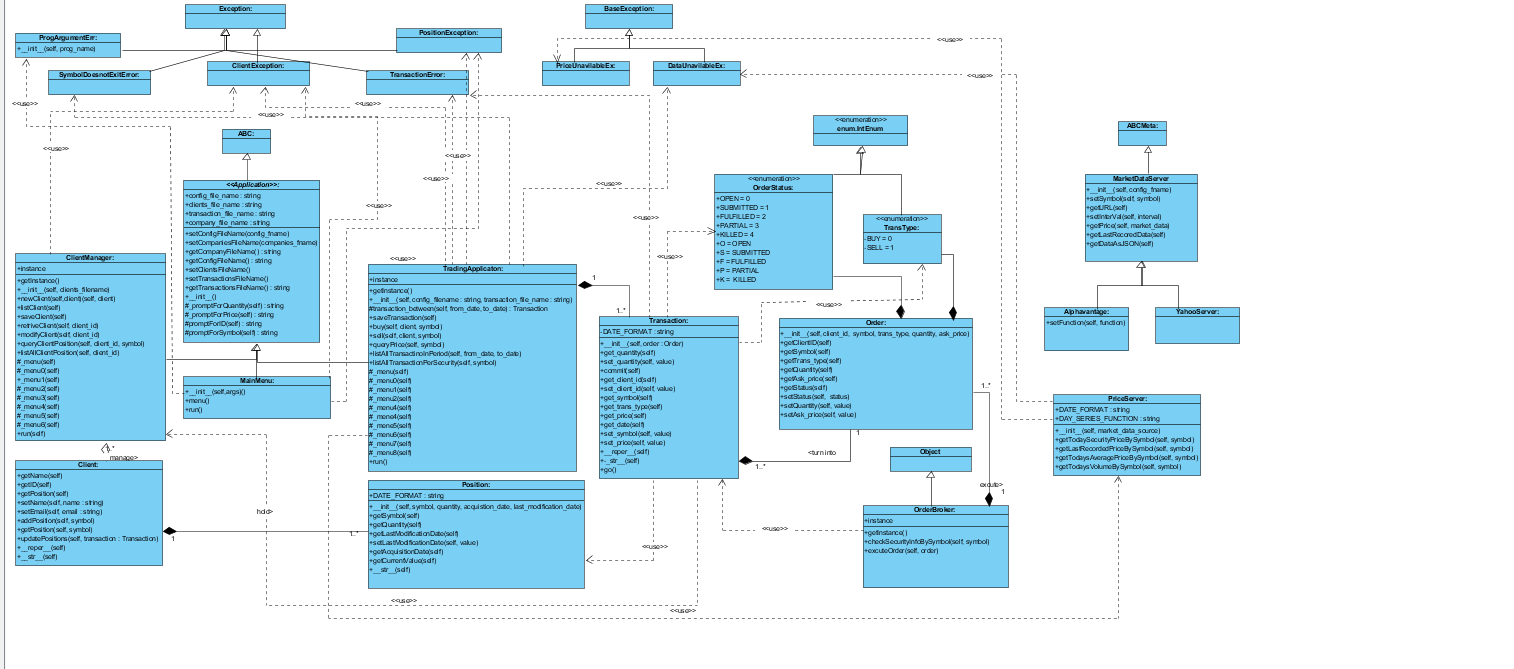
Also, in order improve the system output the team has determined the type of maintenance as improvements to the system to make it more user-friendly.

Class Digram (BEFORE MODIFICATION)

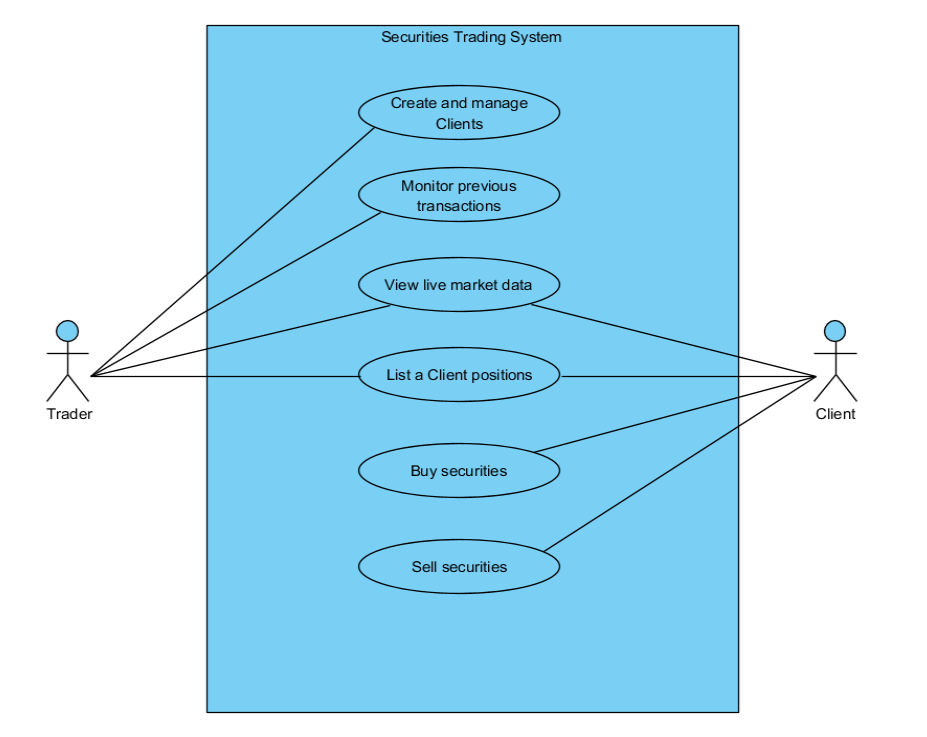
|  |
| --- |
| **System Design** |



Class Digram (BEFORE MODIFICATION)



Use Case Digram (BEFORE MODIFICATION)



A close up of a map

Description generated with high confidence

Architecture Diagram (AFTER MODIFICATION)

**Market Data**

Market Data Server

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a social media post

Description generated with very high confidence

Class Diagram (AFTER MODIFICATION)

**TRADES**

**UI**

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

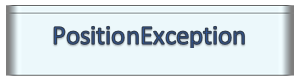
A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

**SERVER**

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with high confidenceA screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with high confidence

A screenshot of a cell phone

Description generated with high confidence

A picture containing screenshot

Description generated with very high confidence

**EXCEPTIONS**

Use Case Diagram (AFTER MODIFICATION)

**Security Trading Application**

A picture containing text, map

Description generated with very high confidence

|  |
| --- |
| **Appendices** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sprint 1** | | | | | |  | Day 1 | Day 2 | Day 3 | Day 4 | | Date | 2018-04-11 | 2018-04-12 | 2018-04-13 | 2018-04-16 | | Time | 10:00:00 | 10:30:00 | 10:15:00 | 10:00:00 | | Attendance | 3/3 | 3/3 | 3/3 | 3/3 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sprint 2** | | | | | |  | Day 1 | Day 2 | Day 3 | Day 4 | | Date | 2018-04-17 | 2018-04-18 | 2018-04-19 | 2018-04-20 | | Time | 10:00:00 | 10:30:00 | 10:00:00 | 10:00:00 | | Attendance | 3/3 | 3/3 | 3/3 | 3/3 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sprint 3** | | | | | |  | Day 1 | Day 2 | Day 3 | Day 3 | | Date | 2018-04-23 | 2018-04-24 | 2018-04-25 | 2018-04-26 | | Time | 11:00:00 | 10:30:00 | 10:00:00 | 10:00:00 | | Attendance | 3/3 | 3/3 | 3/3 | 3/3 | |

|  |
| --- |
| **User’s Guide** |

|  |
| --- |
| **Solution implementation** |

* **Implementing a Security and a Security Manager Class:**

The system was representing a security in term of ticker symbol only, and did not have a class referring to security, neither it had any information about the securities except of the symbol of a security. The user did not get any additional information of a certain security as it was unable to do complex operations. In additional to that the system looked dishevelled and unorganised.

And in term of software evolution and maintenance, this case is a serious issue that would prevent any future maintenance on the software. Therefore, to be able to tackle it, the lecturer required a security class to represent the security in the broker trades system, that carry the necessary and essential attributes and methods.

Although the security class itself would tackle the main issue that is holding the software for future evolutions and maintenance, however, the team was thinking for further development in this direction and extra features that would extend the ability of the software for any future changes that may occur. Therefore, the best solution was to create a new class, named SecurityManager, that is responsible for the security and may be able to create in the future a new menu in the main menu of the application and will permit traders staff to add the new securities that have been added to the trade system or remove any securities that is not anymore in the trade system and modify any changes that might occur.

The security class holds four attributes representing the security’s name, symbol, sector and industry that will be initialised inside the class’s constructor, beside to the getter’s and setter’s methods. Whereas the security manager class will read the stock companies’ file and pass it to the Security class’s constructor. Moreover, the security manager class would have other methods related to the security such as,

1. retrieveSecSymbol method, that is retrieving the symbol of a certain security by typing the company’s name the user wants to know about.
2. getCurrentMarketValue method, that is retrieving the price of a particular security based on the symbol a user will type on the system.
3. listSecurities method, that will list all existing securities

in addition to other methods that allows a user to check for additional information of a symbol, name, industry or sector of a particular security.

* **Improve the System Output:**

The securities trading system enables the intended users to display some information such as transaction details, list all clients...etc. However, this system prints this information via its internal data representation, which is not efficient way as the users should have knowledge of how the data is represented.

So, based on the request of improve the system output to make it more user-friendly, and after the modification has analysed for its impact on the system, the team has decided to use a string format and Pandas library to fulfil this requirement.

Firstly, the function **listClients(self)** which is provided by the class **Client Manager** in **ui package** has modified to make it display the clients details in more clearly and friendly way.

Pandas library has employed here as it is providing high-performance, easy-to-use data structures and data analysis tools, columns can be inserted and deleted from data structures for size mutability and intelligent label-based slicing, fancy indexing, and sub setting of large data sets.

The function **pandas.read\_csv()** call to read the clients file and display the information in columns which is easily specify their names with the argument **"names"** while reading the csv file to be understandable to the users:

**clientDataset= pd.read\_csv(self.clients\_file\_name,skiprows=1,delimiter=":",header=None,**

**names= ["ID","Name","Email","Transaction Details"],index\_col= "ID",na\_filter=False)**

**print(clientDataset).**

Secondly, the function **\_\_str\_\_(self)** which is provided by the class **Transaction** in **trades** package has modified using the string format to make it perform the transactions details in such a user-friendly way:

**def \_\_str\_\_(self):**

**return "%s|%5d|%5d|%-5s|%.2f|%5d" % (self.date, self.clientID, self.trans\_type, self.symbol, self.price, self.quantity).**

This function is called by other functions in the class **Trading Application** in **ui package** such as **lisAllTransactions,**

**listTransactionsInPeriod ...,** in order to display different information.

* **Implement a more realistic order execution:**

In this version of system all orders are executed successfully by the 'Order Broker' class without considering the current market value. As result to this unrealistic methodology the clients can submit an order to buy or sell securities with an ask price much lower or higher than the market price. So, based on the request of implement a more realistic order execution, and after the modification has analysed for its impact on the system, the team has implemented a simple execution strategy which is constrained by the current market value.

To make the sell and buy order more realistic, the functions **sell** and **buy** which are provided by the class **Trading Application** in **ui** packag have modified using the if statement to check that the ask price given by the client with the current market value that can be known by call the function getCurrentMarketValue(symbol) that is in the class Security.

In sell order the new variable **max\_price** has declared to calculate the maximum ask price dependened on the quantity of securites that have asked by the clients i.e**. max\_price = quantity \* current\_value** . While the maximum ask price is less than or equal to the current market value, the oreder will be executed (and a transaction is recorder). Otherwise the order will be killed.

On contrast, in buy order the new variable **min\_price** has declared to calculate the minimum ask price dependened on the quantity of securites that have asked by the clients i.e. **min\_price = quantity \* current\_value**. If the minimum ask price is grater than or equal to the current market value, the oreder will be executed (and a transaction is recorder). Otherwise the order will be killed.

In addition, the new version of system helps the clients to ask an appropriate price by print the the maximum ask price or minimum ask price before asking the client about it, so the clients can determine the price unrandomly:

In **sell** function: print("You asked to sell %d stocks of %s, which is now trading at %s, maximum price is %s" % (quantity, symbol, current\_value, max\_price)).

In **buy** function: Print ("You asked to buy %d stocks of %s, which is now trading at %s, minimum price will be %s" % (quantity, symbol,current\_value, min\_price))

|  |  |
| --- | --- |
| BEFORE:  List All Clients:      Buy Order:    Sell Order:    List Transactions For a Client: | AFTER:  List All Clients:    Buy Order:    Sell Order:    List Transactions For a Client: |

|  |  |
| --- | --- |
| List Transaction In Date:    List All Transactions: | List Transaction In Date:    All Transactions: |

|  |
| --- |
| **Conclusions and Reflections** |

By completed this assignment each member of the team has gainded a good experience in how to applay the software evolution and software maintenance on an existing software system "The securities trading system".

Since these processes can be very sensitive to even small changes, the team has took a responsibility to understand and analisied how the internal system works by read the code seriously and run the system before make any change on it. Then a new release of the system has planned, and all proposed change (fault repair, new functionality, and adaptation) are considered.

The main challenge was how to embedded the new class Security to support any evolution of the system on the future as this process involves change to its design. This was a good opportunity to learn that the motivated of the change should not merely be technically but it must contribute to the objectives of the system.

After that, the team has a great practice in how to use the the system's documentation created in the design step to code the new modules.

|  |
| --- |
| **References** |

* [2] V. T. Rajlich and K. H. Bennett. 2000. A stage model for the software life cycle. IEEE Computer, July,2-8.
* Chapter 9 – software evolution. Retrieved from <http://csis.pace.edu/~marchese/CS389/L9/Ch9_summary.pdf>
* Models of Software Evolution. (july,1990). Retrieved from http://www.dtic.mil/dtic/tr/fulltext/u2/a227328.pdf
* [1] Lehman, M.M., and L. Belady. Program Evolution: Processes of Software Change. Academic Press, New York, 1985
* UML Activity Diagram Examples. (2009-2017). Retrieved from [www.uml-diagrams.org/activity-diagrams-examples.html](http://www.uml-diagrams.org/activity-diagrams-examples.html)
* The maintenance processes. Retrieved from <http://rti.etf.bg.ac.rs/rti/ms1es/Literatura/Grubb_Takang-Software_Maintenance_Ch5.pdf>
* Priyadaeshi Tripathy and kshirasagar Naik (2015) “software evolution and maintenance: A Practitioner’s Approach”, John Wiley & Sons, Inc.
* Sommerville, I. (2015) “Software Engineering” 10th edition; Addison Wesley; ISBN 1292096136