EQUITY TRADING APPLICATION

A PROJECT REPORT

Internship Project: Sapient Global Markets, Unitech Infospace, Tower A Building 2, Sector-21 Gurgaon India

Submitted in partial fulfillment for the award of the degree of

B.TECH

in

Information Technology

By

Mavez Singh Dabas (10BIT0245)

Under the Guidance of

Prof. Neelu Khare

School of Information Technology & Engineering

MAY 2014

DECLARATION BY THE CANDIDATE

I hereby declare that the project report entitled **“EQUITY TRADING APPLICATION”** submitted by me to School of Information Technology & Engineering, Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the degree of **B.Tech (Information Technology)** is a record of bonafide final project work carried out by me under the guidance of **Prof. Neelu Khare**. I further declare that the work reported in this final project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore   Mavez Singh Dabas

Date: 08/05/2014

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
VELLORE ■ CHENNAI
www.vit.ac.in

School of Information Technology & Engineering [SITE]

CERTIFICATE

This is to certify that the final year project report entitled **"EQUITY TRADING APPLICATION"** submitted by **Mavez Singh Dabas (10BIT0245)** to School of Information Technology & Engineering, Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the degree of **B.Tech (Information Technology)** is a record of bonafide project work carried out by him under my guidance. The major project fulfills the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Prof. Neelu Khare

Associate Professor

Internal Guide

The project work is satisfactory / unsatisfactory

Internal Examiner External Examiner

Approved by

Program Chair Dean

Sapient

May 9, 2014

INTERIM CERTIFICATE

This is to certify that **Mr. Mavez Singh Dabas**, a student of B.Tech Information Technology 8th Semester, **Vellore Institute Of Technology** is undergoing internship at SAPIENT, **Gurgaon** , from **January 7, 2014 to July 7, 2014.**

If you have any query, please send a request at employmentverification2@sapient.com.

Yours sincerely,

Kamal Karwal

Digitally signed by Kamal Karwal DN: c=IN, o=Sapient Consulting Limited, ou=ITES, postalCode=[illegible], st=Haryana, [illegible] cn=Kamal Karwal Date: [illegible]

Kamal Kumar Karwal
Director, People Success
Dated: May 9, 2014

"This letter is based on information contained in our records and is solely for the purpose of confirming such information and does not bind Sapient in any way on behalf of the Sapient person mentioned herein, or otherwise. By issuing this letter, Sapient is not making any recommendations or providing any guarantee and, therefore, shall not be held liable in any way."

Sapient Consulting Limited (Unit-1)
Building No.-2, Tower-A,Ground&First Floor
Building No-5,Ground&First Floor
Building No-1,Tower-B,Ground,1,2,3,4,5 Floor
Gurgaon Infospace Ltd,SEZ
Sector-21,Village Dundahera
Gurgaon-122016,Haryana

TEL +91 [124] 499 6000
FAX +91 [124] 499 6001

sapient.com

Sapient Consulting Limited (Unit-3)
Building No-8, Tower-B,Ground to 9 Floor
Gurgaon Infospace Ltd,SEZ
Sector-21,Village Dundahera
Gurgaon-122016,Haryana

TEL +91 [124] 672 4000
FAX +91 [124] 672 6001

sapient.com

Registered Office: 104, Ashoka Estate, Barakhamba Road, New Delhi-110001, India

ACKNOWLEDGEMENT

I wish to express our heartfelt gratitude to **Dr. G. Vishwanathan** Chancellor, VIT University, Vellore for providing facilities for the semester project.

I am highly grateful to our Vice Presidents **Shri. Sankar Vishwanathan**, , **Shri. Sekar Vishwanathan**, **Shri. G.V. Selvam** and **Dr. V. Raju** Vice Chancellor, for providing necessary resources.

My sincere gratitude to **Dr. V.Sankaran**, Dean, School of Information Technology and Engineering [SITE], for giving us the opportunity to undertake the project.

I wish to express my sincere gratitude to **Prof. Ranichandra C.,** Programme Chair of B.Tech IT, School of Information Technology and Engineering for providing me an opportunity to do my project work.

I would like to express my sincere gratitude and thanks to my internal guide **Prof. Neelu Khare**, Associate Professor, School of Information Technology and Engineering whose esteemed guidance and immense support encouraged to complete the project successfully.

Also we would like to thank **Prof. Rama Prabha K P** for their guidance throughout with their outstanding knowledge and consistent support.

This acknowledgement would be incomplete without expressing my whole hearted thanks to my family and friends who motivated me during the course of the work.

Place : Vellore **Mavez Singh Dabas**

Date : 08/05/2014

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Our client XYZ has asked our team to produce an Equity Trading Application. A Desktop Application with three different Actors Portfolio Manager, Execution Trader and Broker. Portfolio Manager makes the Investment decision with other people investment. Execution Trader Buys and Sells financial instruments such as stocks, bonds, commodities and derivatives. Broker Buys and sells shares and other securities through market makers or Agency Only Firms on behalf of investors. In order to work on the actors our client application can create a Portfolio with Portfolio Manager. Portfolio Manager can create multiple trade orders with different exchanges and securities among several industries. Portfolio Manager sends the trade for execution to the Execution Trader. Trader combines various orders accordingly to certain constraints to make a block. Execution Trader is responsible to send the block for execution to the Broker. Broker executes the orders and sends back the executed blocks to the Execution Trader and Portfolio Manager who can view the executed status of the orders sent by them.

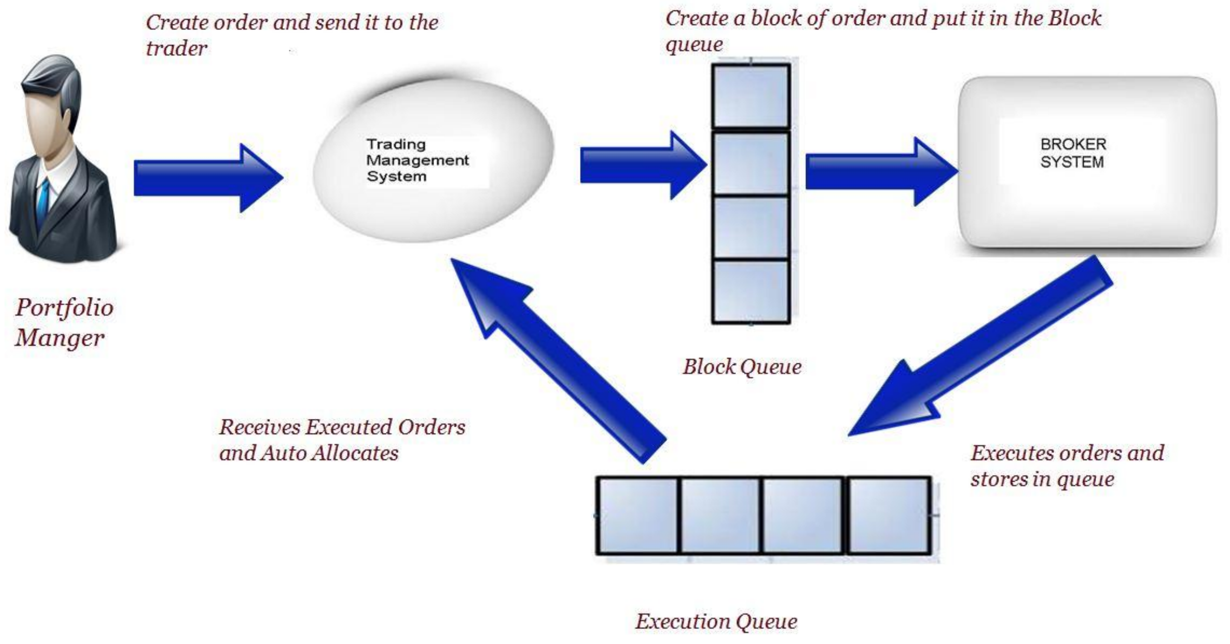

Create order and send it to the trader
Create a block of order and put it in the Block queue
Trading Management System
BROKER SYSTEM
Portfolio Manger
Block Queue
Receives Executed Orders and Auto Allocates
Executes orders and stores in queue
Execution Queue

Fig 1

1. INTRODUCTION

1.1 EQUITY TRADING

Equity Trading is about buying and selling of company stock shares. Shares in large publicly traded companies are bought and sold through major stock exchanges, such as the New York Stock Exchange, London Stock Exchange or Bombay Stock Exchange. Stock shares in smaller public companies are bought and sold in over-the-counter (OTC) markets.
Equity trading can be performed by the owner of the shares, or with the help of an agent authorized to do trade on behalf of the share's owner.

The Application is broadly classified in 3 modules:

Portfolio Manager: It will create new orders under a portfolio and then send these orders to the Execution Trader for trading.

Execution Trader: Execution Trader will receive the orders from PM and divide them into BLOCKS based on some rules and then send it to the Broker for execution.

Broker System: This is a background service which would execute the blocks send by Execution Trader and on the basis of auto allocation the Block is executed either partially or completely.

1.2 AIM

The main aim of this project is to develop and understand the basic functionality of trading application platform and to know about how the trading is done.

1.3 OBJECTIVE

After doing this project, we were able to understand and grasp the trading rules and regulations and we got a real time desktop application to manage how trading of equities, derivatives are done.

1.4 PROBLEM STATEMENT

The main problem with existing Trading Application is hidden costs and deceptive advertising associated with online trading.

Another major problem with existing Application is delayed and varied execution speeds in executing the trade.

1.5 MOTIVATION

The Application was assigned to the whole batch as a MOCK Project which was to be done in the assigned time frame. MOCK Project is a mandatory project to implement all the learning in the training period.

1.6 CHALLENGES FACED

- Code-Integration was a challenge
- Applying SVN Tool in collaboration of the code, keeping it updated.
  SVN is a Sapient tool which is used in application development for the code integration.
- Reaching Consensus among the different tacks of the application.

1.7 STATEMENT OF ASSUMPTION

To work on the Application the teams were given with different Use cases. The Use cases which were used as the conditions and requirements for the Application.

2. LITERATURE SURVEY

To resolve the existing problems we have used WCF service for the Broker Part of execution. Using the Broker as a desktop service we plan to simulate a number of executions in parallel and enhance the interoperability.

Whenever the orders come to Execution Trader, the orders having the similar parameters will first go into a PROPOSED BLOCK where the user can either create a new block or add to some of the already existing blocks available for him.

These orders must have come from the Portfolio Manager which the manager has sent for execution.

2.1 SYSTEM PLANNING

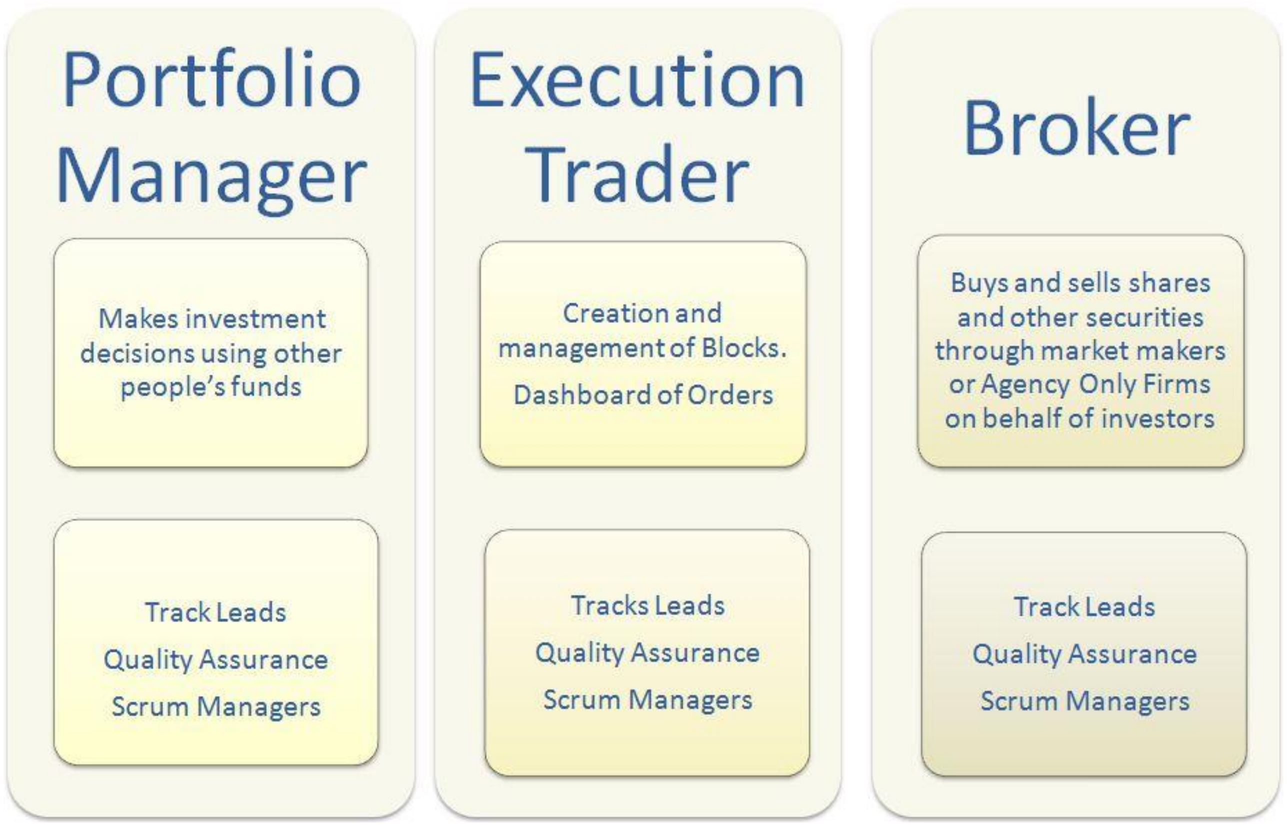

Portfolio Manager
Makes investment decisions using other people's funds
Track Leads
Quality Assurance
Scrum Managers
Execution Trader
Creation and management of Blocks.
Dashboard of Orders
Tracks Leads
Quality Assurance
Scrum Managers
Broker
Buys and sells shares and other securities through market makers or Agency Only Firms on behalf of investors
Track Leads
Quality Assurance
Scrum Managers

Fig 2

The whole team was divided into three tracks as mentioned Portfolio Manager, Execution Trader and Broker.
Each member of track worked as Track Leads, Quality Assurance and Scrum Managers.

2.2 DESCRIPTION

2.2.1 Portfolio Manager and Execution Trader:

Presentation layer consists of the user interface. Interfaces of one each for Portfolio Manager and Execution Trader. Portfolio Manager can create, view, edit new orders, cancel orders, amend the open orders, and send the orders for execution and view positions.
Execution trader can create, view and propose blocks, edit and delete existing blocks, remove orders from block and send blocks for execution.

2.2.2 Broker:

The Application for the broker is a background process. Broker is activated as a WCF service in the Execution Trader Application. Broker receives the order send for execution from the execution trader matches the price and executes the order block. The system receives data from Execution trader to execute via WCF service and after the execution of the trade is carried out the status is send to the Execution Trader and Portfolio Manager.

2.3 HARDWARE AND SOFTWARE REQUIREMENT

1. Development Tool (Visual Studio - 2010)
2. SQL Server Express
3. Presentation Layer – (WPF)
4. Middle Ware (WCF)
5. Miscellaneous (MVVM, EF, LINQ)
6. Quality Assurance (Manual)
7. Unit Testing and Integration Testing (N-Unit)
8. Defect Tracking (Result Space)
9. Change Management (SVN)

3. SYSTEM DESIGN

Equity Trading is the buying and selling of company stock shares. Shares in large publicly traded companies are bought and sold through one of the major stock exchanges, such as the New York Stock Exchange Bombay Stock Exchange which serve as managed auctions for stock trades. Stock shares in smaller public companies are bought and sold in over-the-counter (OTC) markets.

Equity trading can be performed by the owner of the shares, or by an agent authorized to buy and sell on behalf of the share's owner.

The whole Application is broadly classified in 3 modules Portfolio Manager, Execution Trader and Broker.

3.1 MODULES

3.1.1 Portfolio Manager It will create new orders under a portfolio and then send these orders to the Execution Trader for trading.

- View Current Positions
- Filter the positions according to constraints
- View Current Orders
- Create Orders
- Amend Orders
- Send Orders to Trader for Execution

3.1.2 Execution Trader The main function of Execution Trader is to receive the orders from Portfolio Manager and divide them into BLOCKS based on some rules and then send it to the Broker for execution.

- View Blocks
- Create Block
- Add orders in existing blocks
- Remove orders from existing block
- Cancel a block
- Send block for execution
- Receives executions from Broker and processes the executions.

3.1.3 Broker This is a background process which would be continuously running and execute the blocks send by ET and on the basis of auto allocation the Block is executed either partially or completely.

- Maintain configuration of various securities
- Receive blocks from Execution trader
- Performs execution process
- Auto Allocates orders after receiving them from Execution Trader

3.2 ARCHITECTURE DIAGRAMS

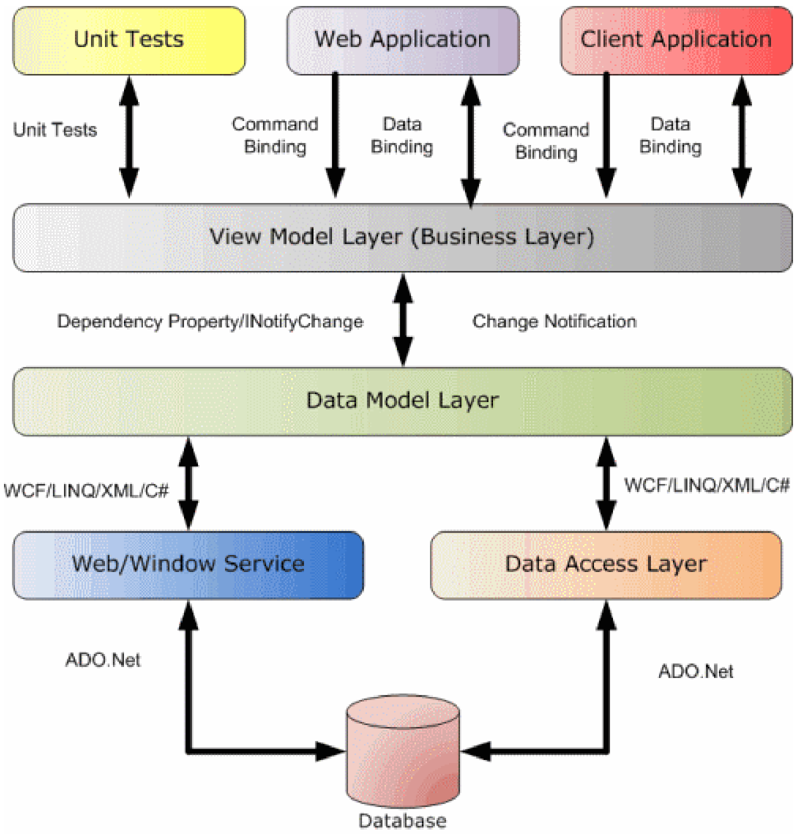


Fig 3

3.2.1 USECASE

1. Portfolio Manager and Execution Trader

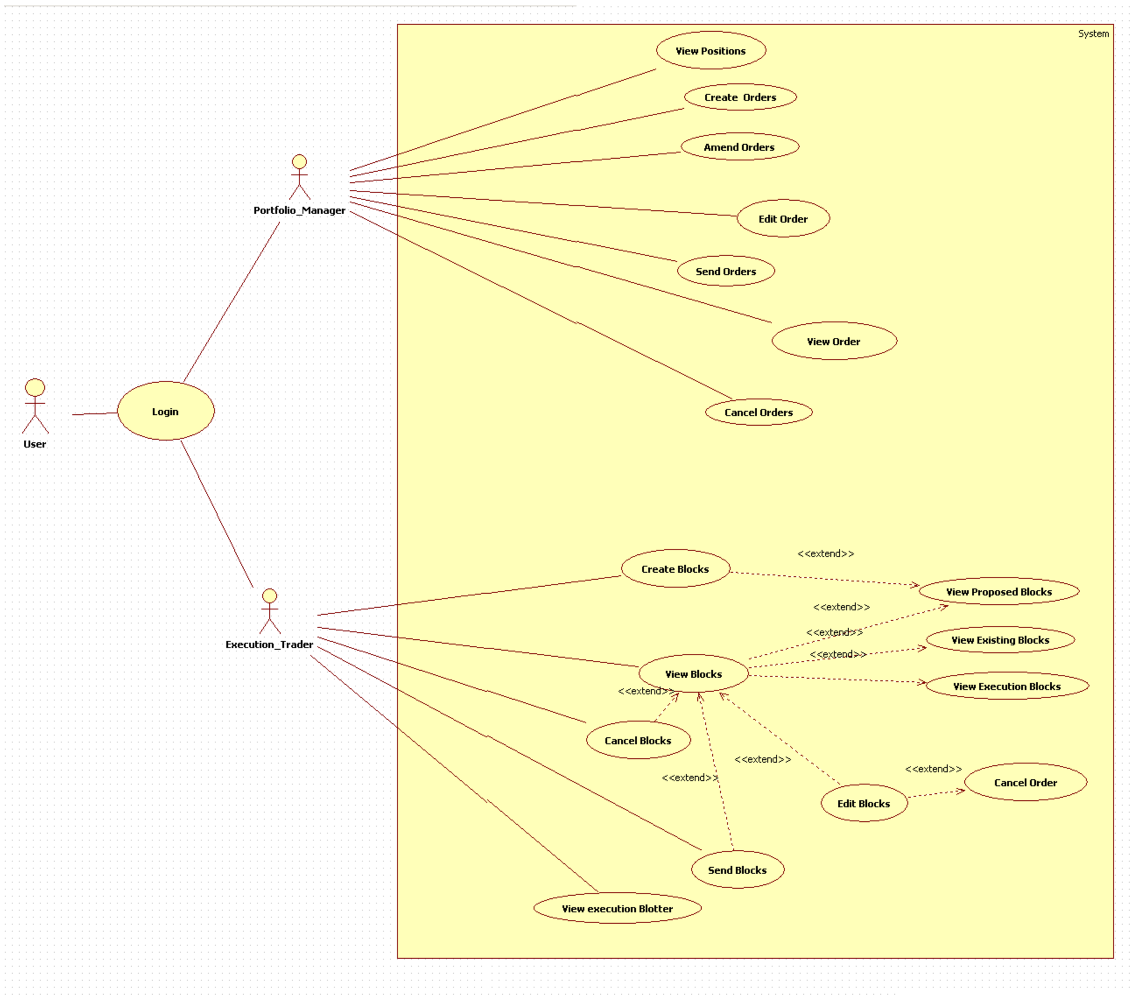


Fig 4

3.2.2 SEQUENCE DIAGRAM

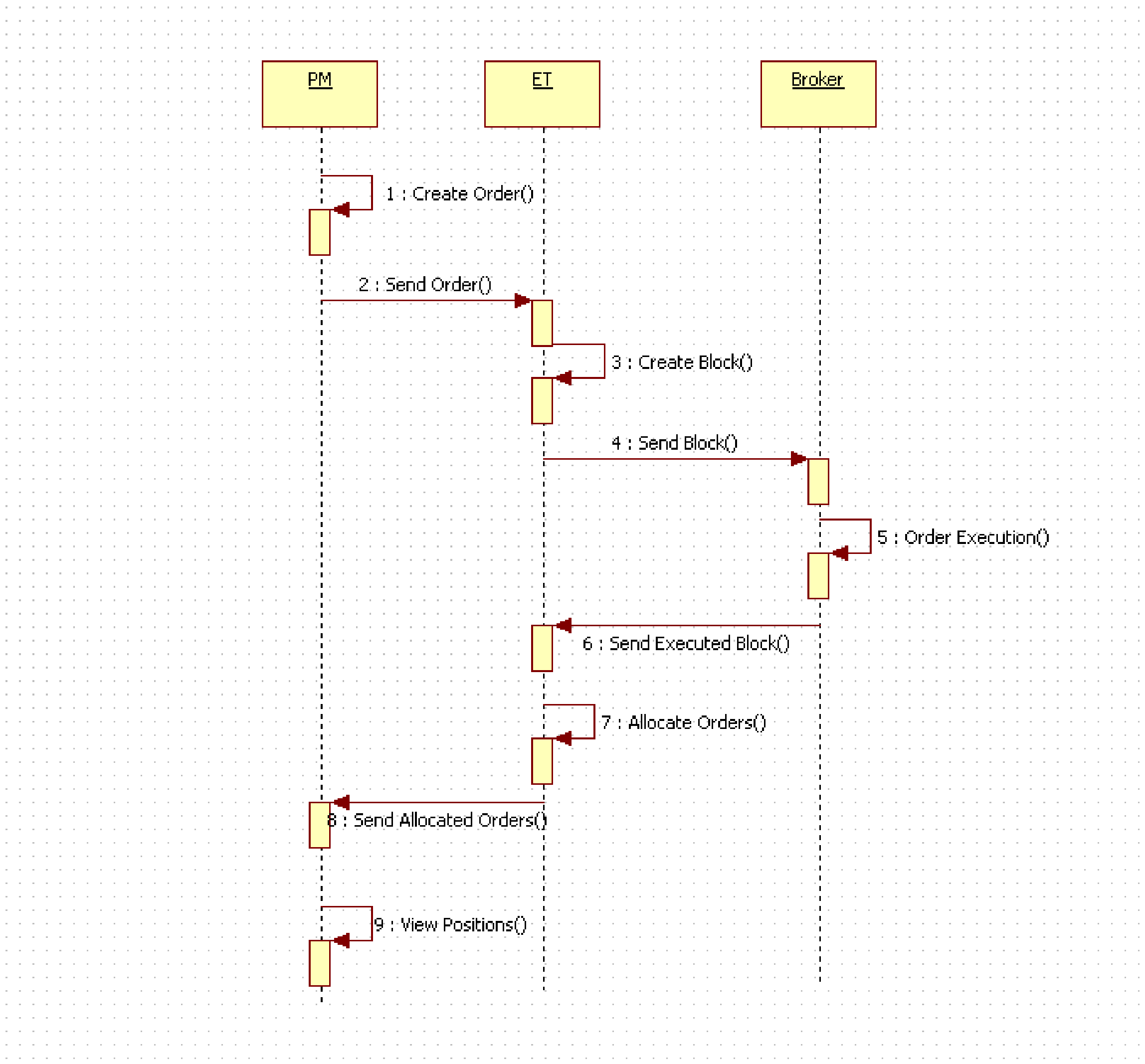


Fig 5

3.2.3 DATABASE DESCRIPTION

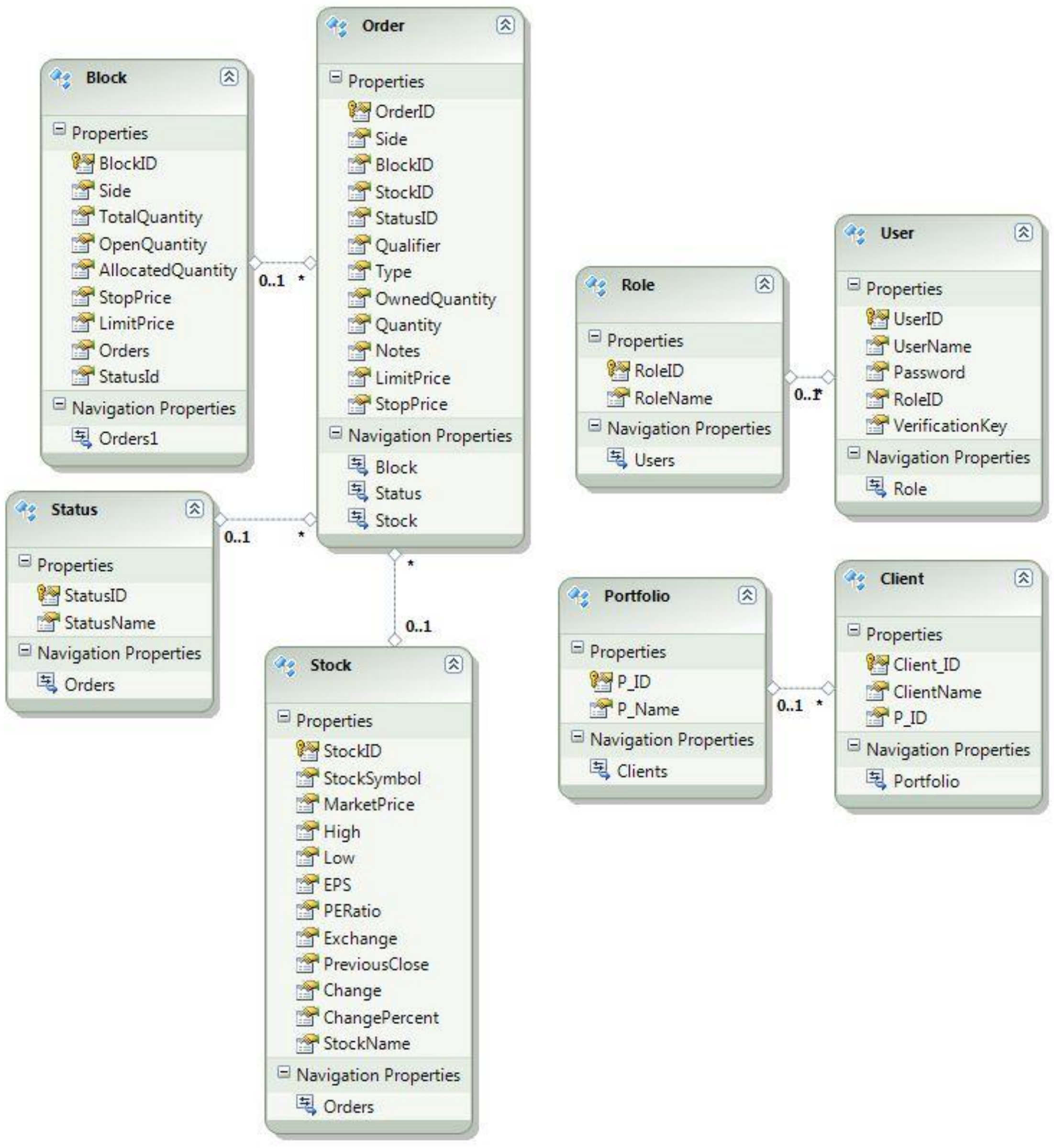


Fig 6

3.3 TECHNOLOGY USED

This application uses various technological specifications. The application is build on Microsoft Visual Studio 2010. The backbone of the project is the EF Entity Framework. Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write. The Entity Framework enables developers to work with data in the form of domain-specific objects and properties, such as customers and customer addresses, without having to concern themselves with the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data, and can create and maintain data-oriented applications with less code than in traditional applications.
The Database of the application was made using the POCO classes. The Entity Framework enables you to use custom data classes together with your data model without making any modifications to the data classes themselves. This means that you can use "plain-old" CLR objects (POCO), such as existing domain objects, with your data model.

Making the code more manageable we have used MVVM approach. MVVM enables the developer to differentiate between the functional and the business layer. The model-view-viewmodel pattern attempts to gain both the advantages of separation of functional development as well as leveraging the advantages of data bindings and the framework by binding data as far back (meaning as close to the pure application model) as possible while using the binder, view model, and any business layer’s inherent data checking features to validate any incoming data. The result is that the model and framework drive as much of the operations as possible, eliminating or minimizing application logic which directly manipulates the code behind.

Elements of the MVVM pattern include:

Model: The model refers to either (a) a domain model which represents the real state content (an object-oriented approach), or (b) the data access layer that represents that content (a data-centric approach).

View: The view refers to all elements displayed by the GUI such as buttons, labels, and other controls.

View model: The view model is a “model of the view” meaning it is an abstraction of the view that also serves in mediating between the view and the model which is the target of the view data bindings. It could be seen as a specialized aspect of what would be a controller that acts as a converter that changes model information into view information and passes commands from the view into the model. The view model exposes public properties, commands, and abstractions. The view model has been likened to a conceptual state of the data as opposed to the real state of the data in the model.

The Graphical User Interface of the application for Portfolio Manager and the Execution Trader were developed using the WPF (Windows Presentation Foundation). Windows Presentation Foundation (or WPF) is a graphical subsystem for rendering user interfaces in Windows-based applications by Microsoft. WPF attempts to provide a consistent programming model for building applications and separates the user interface from business logic. WPF employs XAML, an XML-based language, to define and link various interface elements. WPF applications can also be deployed as standalone desktop programs, or hosted as an embedded object in a website. WPF aims to unify a number of common user interface elements, such as 2D/3D rendering, fixed and adaptive documents, runtime animation, and pre-rendered media. These elements can then be linked and manipulated based on various events, user interactions, and data bindings.

The Broker of the application has been hosted as a WCF Service. Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint.

4. METHODOLOGY

4.1 AGILE METHODOLOGY

In the project we have used Agile Methodology which is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen tight iterations throughout the development cycle.

We have used Agile Methodology because of certain reasons:

- Customer satisfaction by rapid delivery of useful software.
- Welcome changing requirements, even late in development.
- Working software is delivered frequently (weeks rather than months).
- Working software is the principal measure of progress.
- Sustainable development, able to maintain a constant pace.
- Close daily cooperation between business people and developers.
- Face-to-face conversation is the best form of communication (co-location).
- Projects are built around motivated individuals, who should be trusted.
- Continuous attention to technical excellence and good design.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- Regular adaptation to changing circumstances.

Sapient Approach :-

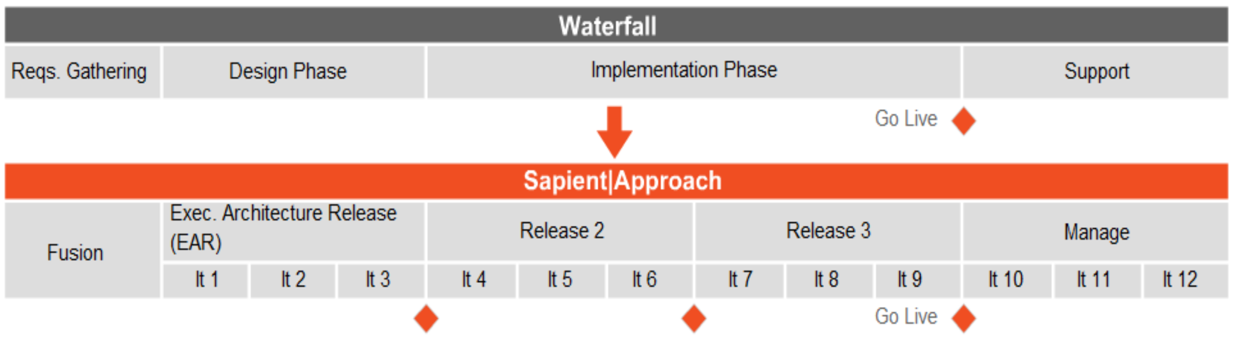
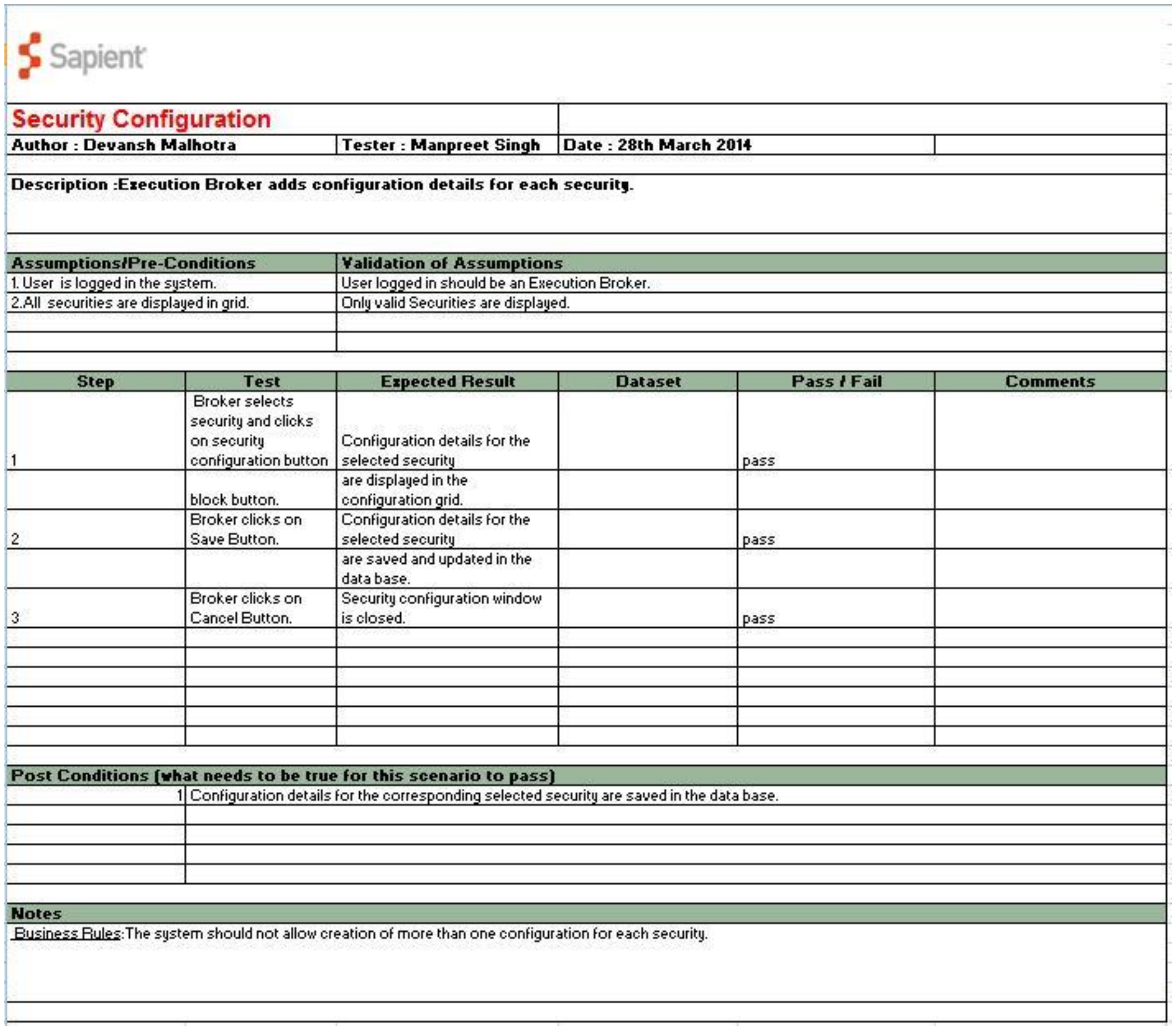


Fig 7

4.2 UNIT TESTING

Test Cases were jotted down and Manual Functional Testing was performed



Sapient

Security Configuration

Author : Devansh Malhotra	**Tester : Manpreet Singh**	**Date : 28th March 2014**	

Description :Execution Broker adds configuration details for each security.

Assumptions/Pre-Conditions	**Validation of Assumptions**
1. User is logged in the system.	User logged in should be an Execution Broker.
2.All securities are displayed in grid.	Only valid Securities are displayed.

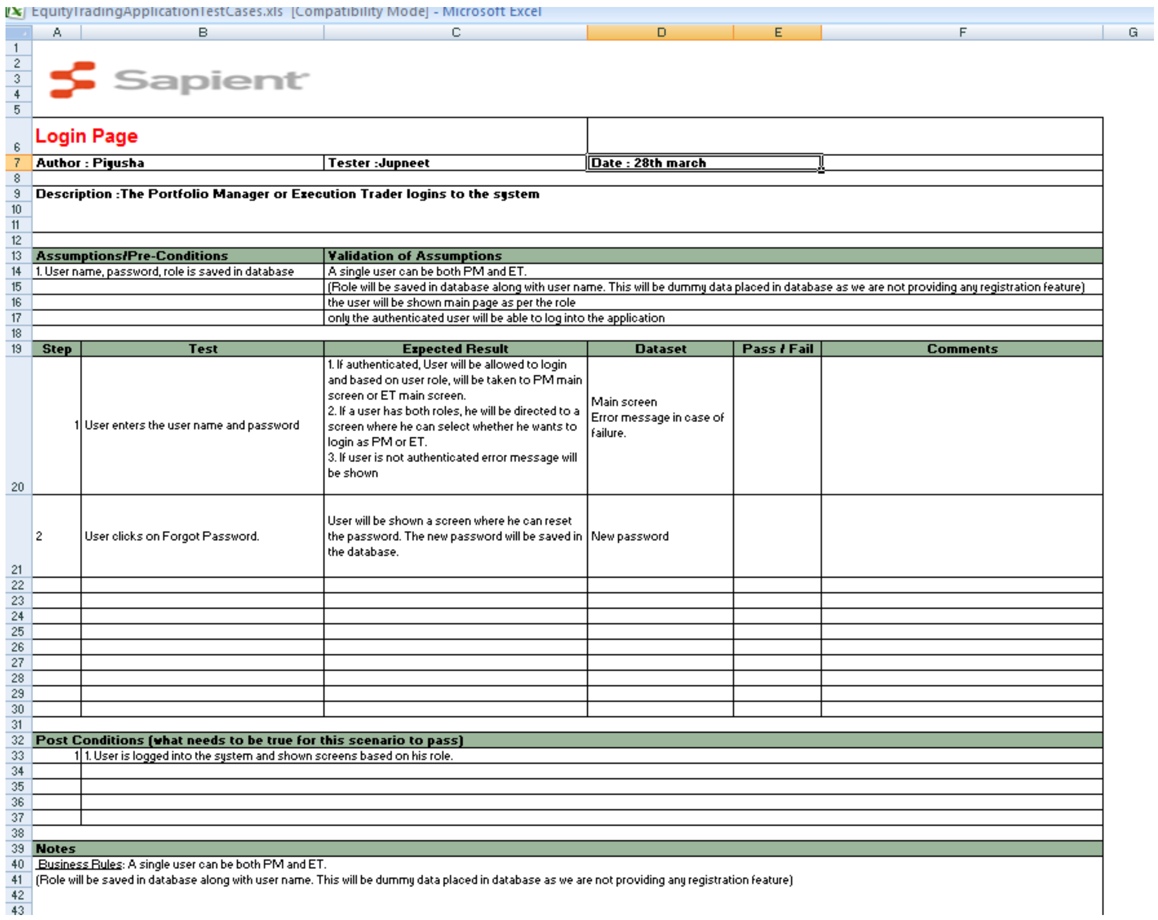
Step	**Test**	**Expected Result**	**Dataset**	**Pass / Fail**	**Comments**
1	Broker selects security and clicks on security configuration button	Configuration details for the selected security		pass	
	block button.	are displayed in the configuration grid.			
2	Broker clicks on Save Button.	Configuration details for the selected security		pass	
		are saved and updated in the data base.			
3	Broker clicks on Cancel Button.	Security configuration window is closed.		pass	

Post Conditions (what needs to be true for this scenario to pass)

1	Configuration details for the corresponding selected security are saved in the data base.

Notes

Business Rules:The system should not allow creation of more than one configuration for each security.



EquityTradingApplicationTestCases.xls [Compatibility Mode] - Microsoft Excel

Sapient

Login Page

Author : Piyusha	**Tester :Jupneet**	**Date : 28th march**

Description :The Portfolio Manager or Execution Trader logins to the system

Assumptions/Pre-Conditions	**Validation of Assumptions**
1. User name, password, role is saved in database	A single user can be both PM and ET.
	(Role will be saved in database along with user name. This will be dummy data placed in database as we are not providing any registration feature)
	the user will be shown main page as per the role
	only the authenticated user will be able to log into the application

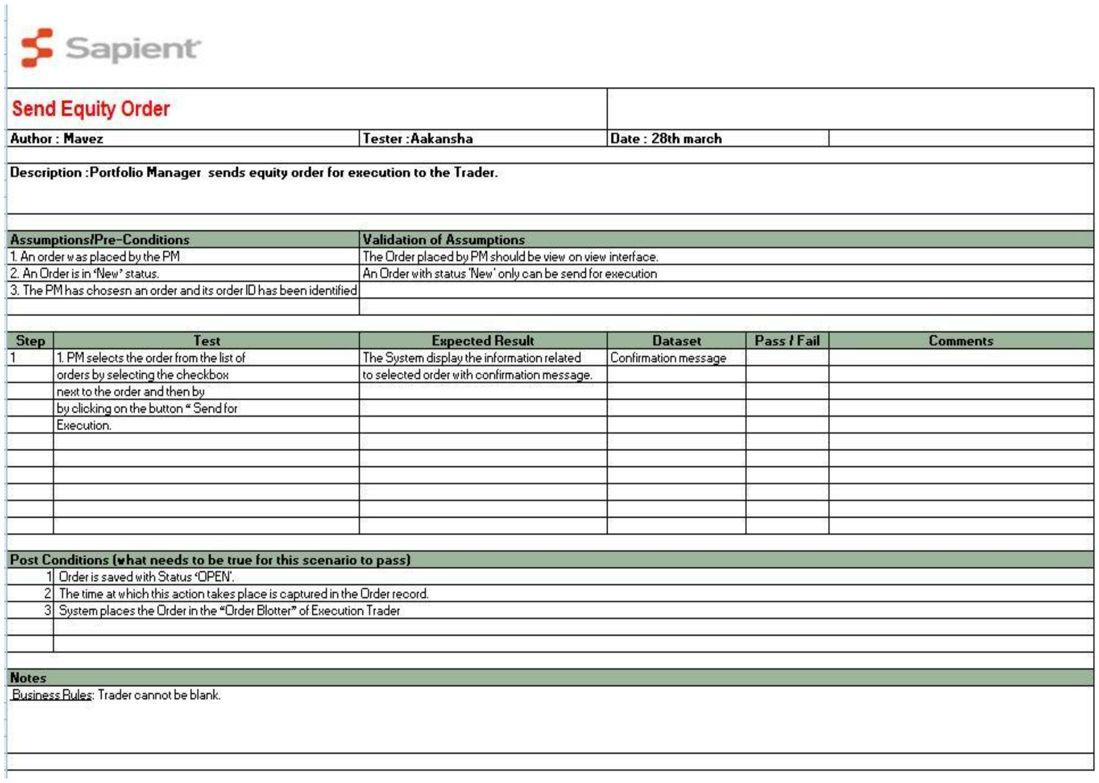
Step	Test	Expected Result	Dataset	Pass / Fail	Comments
1	User enters the user name and password	1. If authenticated, User will be allowed to login and based on user role, will be taken to PM main screen or ET main screen. 2. If a user has both roles, he will be directed to a screen where he can select whether he wants to login as PM or ET. 3. If user is not authenticated error message will be shown	Main screen Error message in case of failure.		
2	User clicks on Forgot Password.	User will be shown a screen where he can reset the password. The new password will be saved in the database.	New password		

Post Conditions (what needs to be true for this scenario to pass)

1	1. User is logged into the system and shown screens based on his role.

Notes

Business Rules: A single user can be both PM and ET.
(Role will be saved in database along with user name. This will be dummy data placed in database as we are not providing any registration feature)



Sapient

Send Equity Order

Author : Mavez	**Tester :Aakansha**	**Date : 28th march**	

Description :Portfolio Manager sends equity order for execution to the Trader.

Assumptions/Pre-Conditions	**Validation of Assumptions**
1. An order was placed by the PM	The Order placed by PM should be view on view interface.
2. An Order is in 'New' status.	An Order with status 'New' only can be send for execution
3. The PM has chosesn an order and its order ID has been identified	

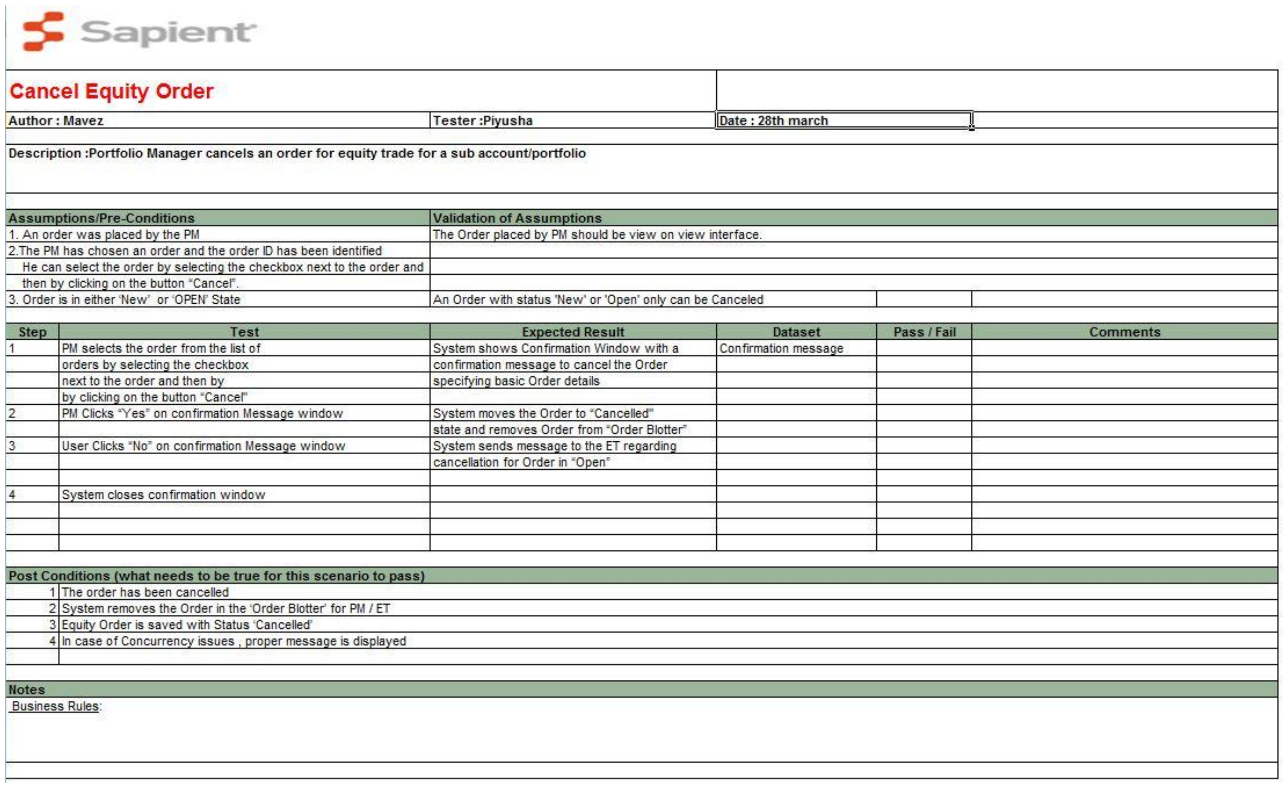
Step	**Test**	**Expected Result**	**Dataset**	**Pass / Fail**	**Comments**
1	1. PM selects the order from the list of	The System display the information related	Confirmation message		
	orders by selecting the checkbox	to selected order with confirmation message.			
	next to the order and then by				
	by clicking on the button " Send for				
	Execution.				

Post Conditions (what needs to be true for this scenario to pass)

1	Order is saved with Status 'OPEN'.
2	The time at which this action takes place is captured in the Order record.
3	System places the Order in the "Order Blotter" of Execution Trader

Notes

Business Rules: Trader cannot be blank.



Sapient

Cancel Equity Order

Author : Mavez	**Tester :Piyusha**	**Date : 28th march**	

Description :Portfolio Manager cancels an order for equity trade for a sub account/portfolio

Assumptions/Pre-Conditions	**Validation of Assumptions**
1. An order was placed by the PM	The Order placed by PM should be view on view interface.
2.The PM has chosen an order and the order ID has been identified	
He can select the order by selecting the checkbox next to the order and	
then by clicking on the button "Cancel".	
3. Order is in either 'New' or 'OPEN' State	An Order with status 'New' or 'Open' only can be Canceled

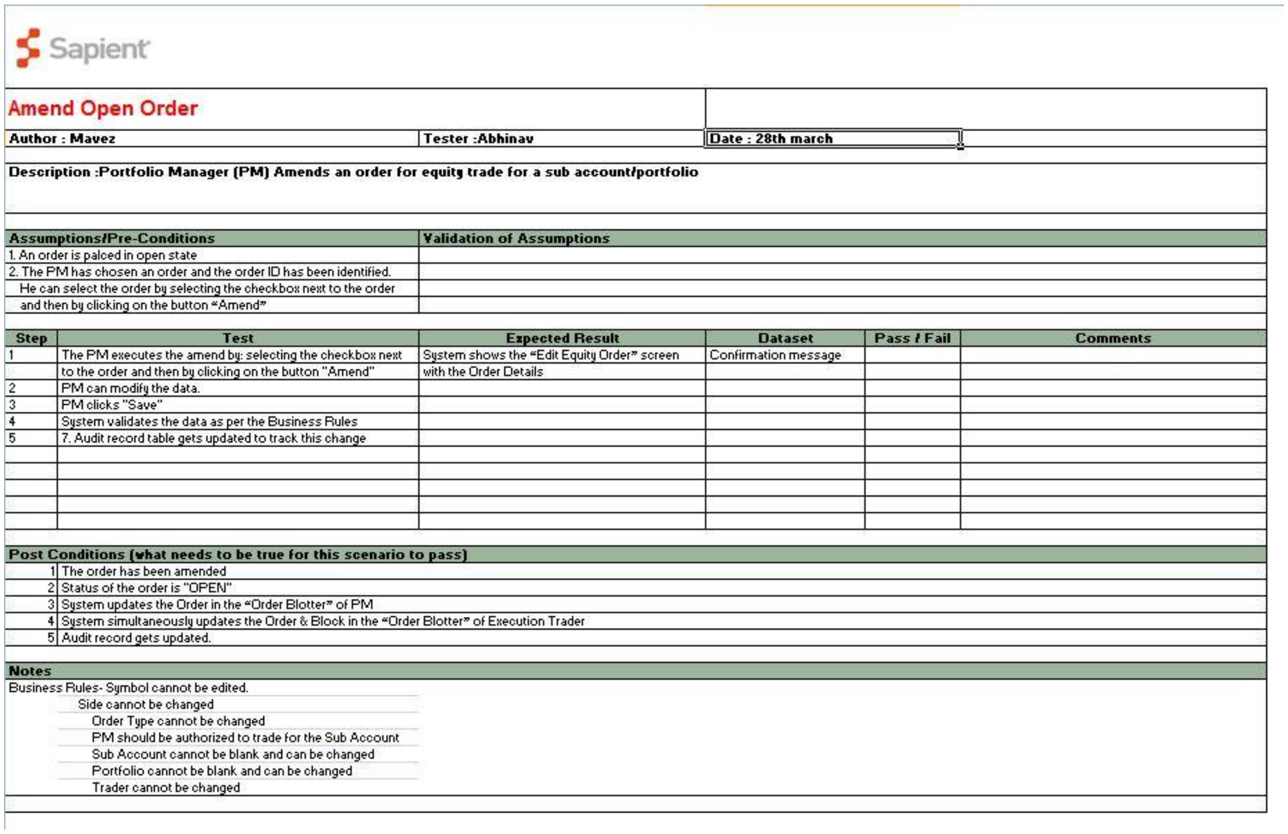
Step	**Test**	**Expected Result**	**Dataset**	**Pass / Fail**	**Comments**
1	PM selects the order from the list of	System shows Confirmation Window with a	Confirmation message		
	orders by selecting the checkbox	confirmation message to cancel the Order			
	next to the order and then by	specifying basic Order details			
	by clicking on the button "Cancel"				
2	PM Clicks "Yes" on confirmation Message window	System moves the Order to "Cancelled"			
		state and removes Order from "Order Blotter"			
3	User Clicks "No" on confirmation Message window	System sends message to the ET regarding			
		cancellation for Order in "Open"			
4	System closes confirmation window				

Post Conditions (what needs to be true for this scenario to pass)

1	The order has been cancelled
2	System removes the Order in the 'Order Blotter' for PM / ET
3	Equity Order is saved with Status 'Cancelled'
4	In case of Concurrency issues , proper message is displayed

Notes

Business Rules:



Sapient

Amend Open Order

Author : Mavez	**Tester :Abhinav**	**Date : 28th march**

Description :Portfolio Manager (PM) Amends an order for equity trade for a sub account/portfolio

Assumptions/Pre-Conditions	**Validation of Assumptions**
1. An order is palced in open state	
2. The PM has chosen an order and the order ID has been identified.	
He can select the order by selecting the checkbox next to the order	
and then by clicking on the button "Amend"	

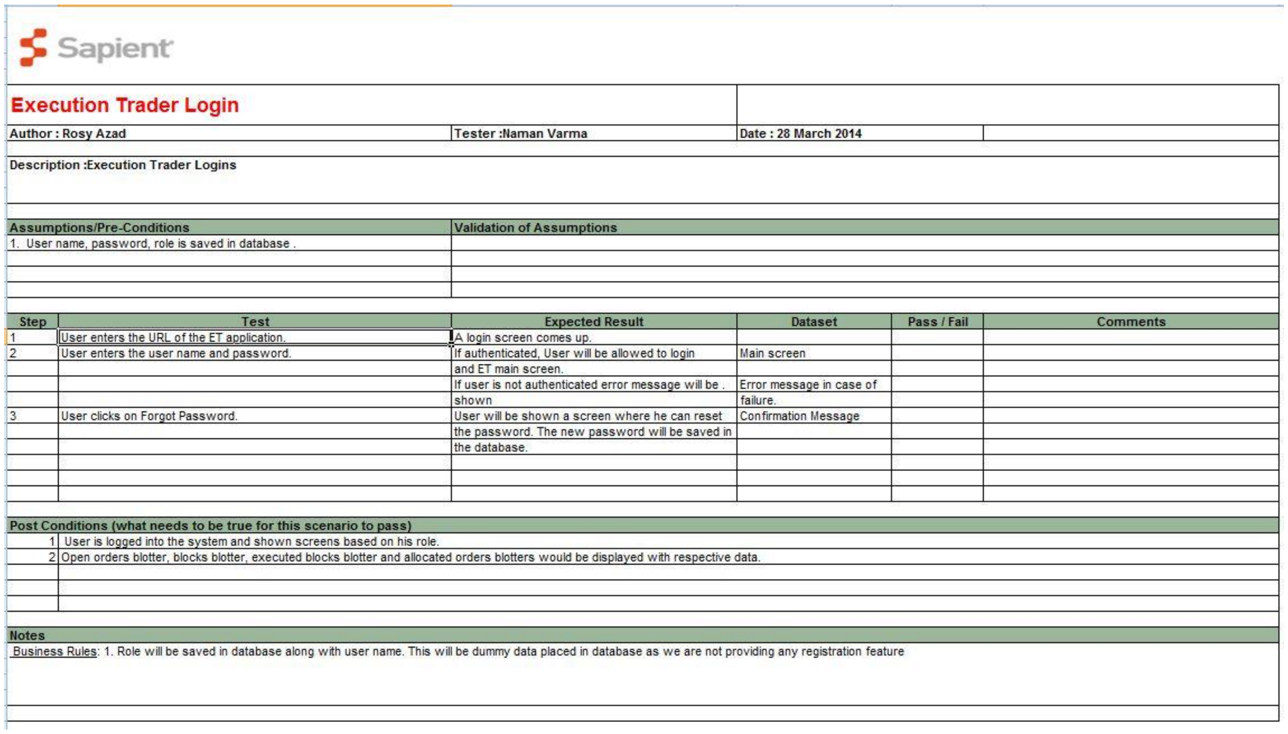
Step	**Test**	**Expected Result**	**Dataset**	**Pass / Fail**	**Comments**
1	The PM executes the amend by: selecting the checkbox next	System shows the "Edit Equity Order" screen	Confirmation message		
	to the order and then by clicking on the button "Amend"	with the Order Details			
2	PM can modify the data.				
3	PM clicks "Save"				
4	System validates the data as per the Business Rules				
5	7. Audit record table gets updated to track this change				

Post Conditions (what needs to be true for this scenario to pass)

1	The order has been amended
2	Status of the order is "OPEN"
3	System updates the Order in the "Order Blotter" of PM
4	System simultaneously updates the Order & Block in the "Order Blotter" of Execution Trader
5	Audit record gets updated.

Notes

Business Rules- Symbol cannot be edited.
Side cannot be changed
Order Type cannot be changed
PM should be authorized to trade for the Sub Account
Sub Account cannot be blank and can be changed
Portfolio cannot be blank and can be changed
Trader cannot be changed



Sapient

Execution Trader Login

Author : Rosy Azad	**Tester :Naman Varma**	**Date : 28 March 2014**

Description :Execution Trader Logins

Assumptions/Pre-Conditions	**Validation of Assumptions**
1. User name, password, role is saved in database .	

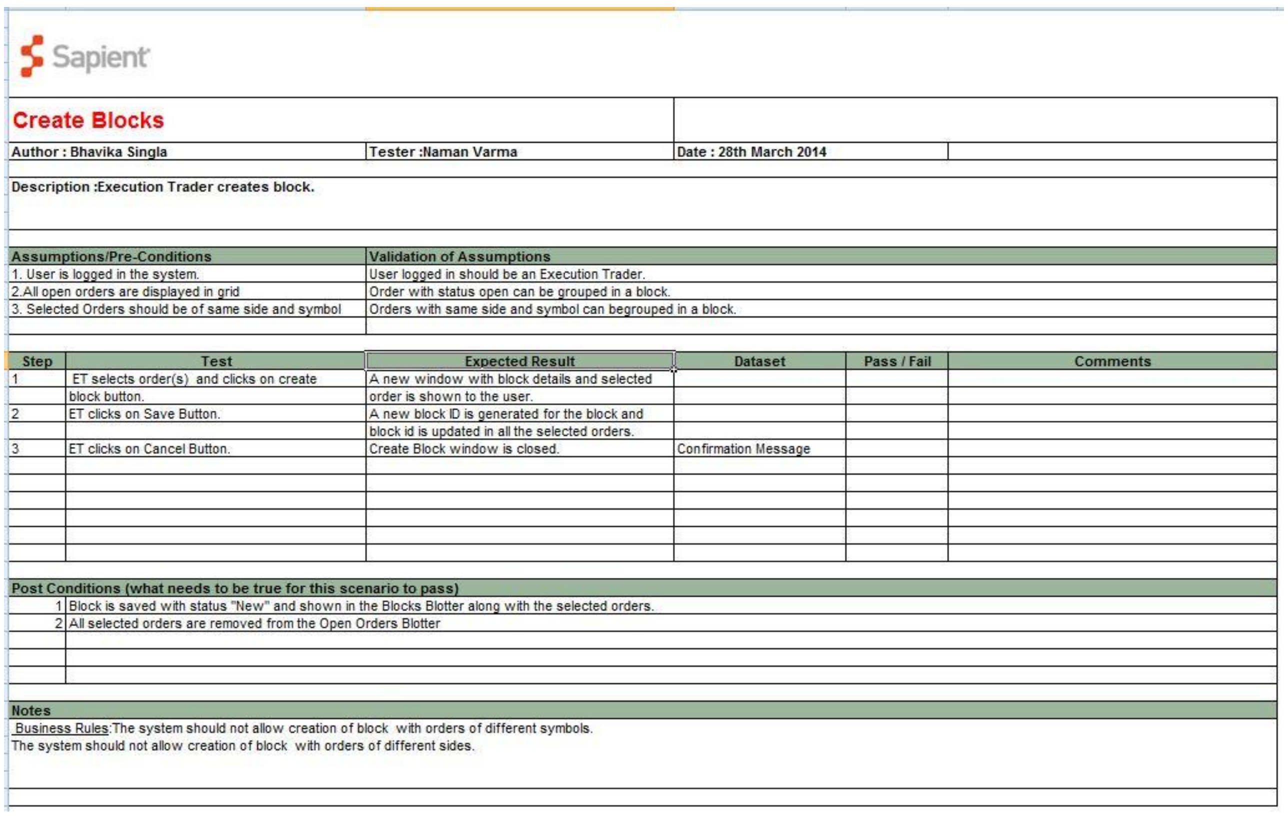
Step	**Test**	**Expected Result**	**Dataset**	**Pass / Fail**	**Comments**
1	User enters the URL of the ET application.	A login screen comes up.			
2	User enters the user name and password.	If authenticated, User will be allowed to login	Main screen		
		and ET main screen.			
		If user is not authenticated error message will be .	Error message in case of		
		shown	failure.		
3	User clicks on Forgot Password.	User will be shown a screen where he can reset	Confirmation Message		
		the password. The new password will be saved in			
		the database.			

Post Conditions (what needs to be true for this scenario to pass)

1	User is logged into the system and shown screens based on his role.
2	Open orders blotter, blocks blotter, executed blocks blotter and allocated orders blotters would be displayed with respective data.

Notes

Business Rules: 1. Role will be saved in database along with user name. This will be dummy data placed in database as we are not providing any registration feature



Sapient

Create Blocks

Author : Bhavika Singla	**Tester :Naman Varma**	**Date : 28th March 2014**	

Description :Execution Trader creates block.

Assumptions/Pre-Conditions	**Validation of Assumptions**
1. User is logged in the system.	User logged in should be an Execution Trader.
2.All open orders are displayed in grid	Order with status open can be grouped in a block.
3. Selected Orders should be of same side and symbol	Orders with same side and symbol can begrouped in a block.

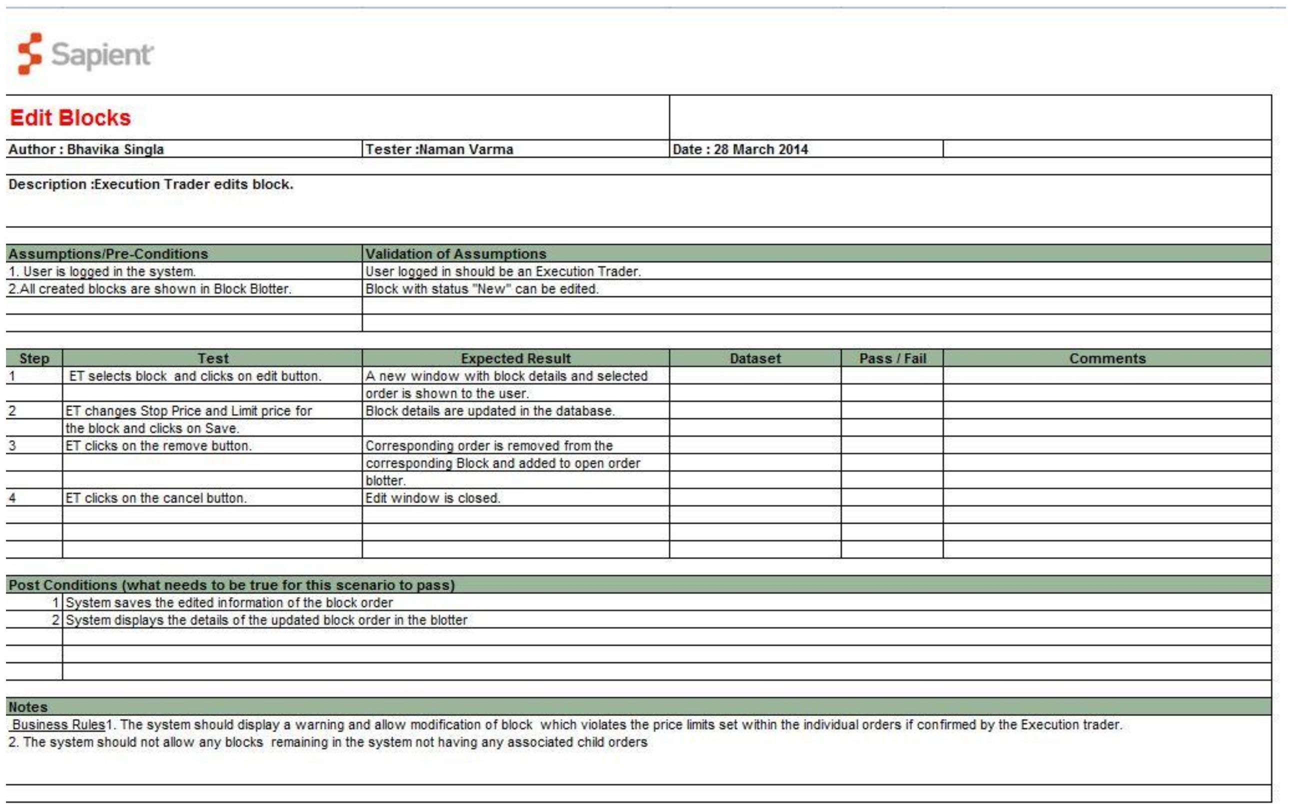
Step	Test	Expected Result	Dataset	Pass / Fail	Comments
1	ET selects order(s) and clicks on create	A new window with block details and selected			
	block button.	order is shown to the user.			
2	ET clicks on Save Button.	A new block ID is generated for the block and			
		block id is updated in all the selected orders.			
3	ET clicks on Cancel Button.	Create Block window is closed.	Confirmation Message		

Post Conditions (what needs to be true for this scenario to pass)

1	Block is saved with status "New" and shown in the Blocks Blotter along with the selected orders.
2	All selected orders are removed from the Open Orders Blotter

Notes

Business Rules:The system should not allow creation of block with orders of different symbols.
The system should not allow creation of block with orders of different sides.



Sapient

Edit Blocks

Author : Bhavika Singla	**Tester :Naman Varma**	**Date : 28 March 2014**	

Description :Execution Trader edits block.

Assumptions/Pre-Conditions	**Validation of Assumptions**
1. User is logged in the system.	User logged in should be an Execution Trader.
2.All created blocks are shown in Block Blotter.	Block with status "New" can be edited.

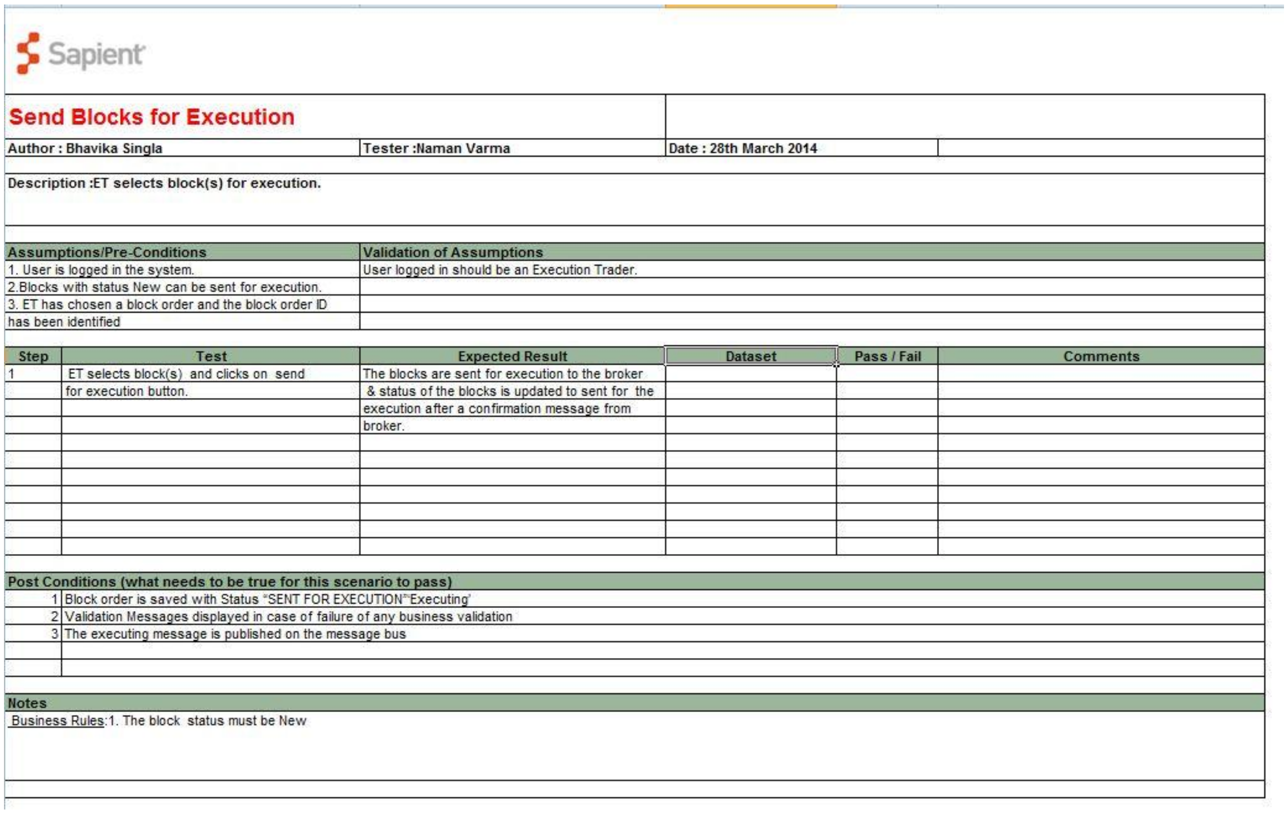
Step	Test	Expected Result	Dataset	Pass / Fail	Comments
1	ET selects block and clicks on edit button.	A new window with block details and selected			
		order is shown to the user.			
2	ET changes Stop Price and Limit price for	Block details are updated in the database.			
	the block and clicks on Save.				
3	ET clicks on the remove button.	Corresponding order is removed from the			
		corresponding Block and added to open order			
		blotter.			
4	ET clicks on the cancel button.	Edit window is closed.			

Post Conditions (what needs to be true for this scenario to pass)

1	System saves the edited information of the block order
2	System displays the details of the updated block order in the blotter

Notes

Business Rules1. The system should display a warning and allow modification of block which violates the price limits set within the individual orders if confirmed by the Execution trader.
2. The system should not allow any blocks remaining in the system not having any associated child orders



Sapient

Send Blocks for Execution

Author : Bhavika Singla	Tester :Naman Varma	Date : 28th March 2014

Description :ET selects block(s) for execution.

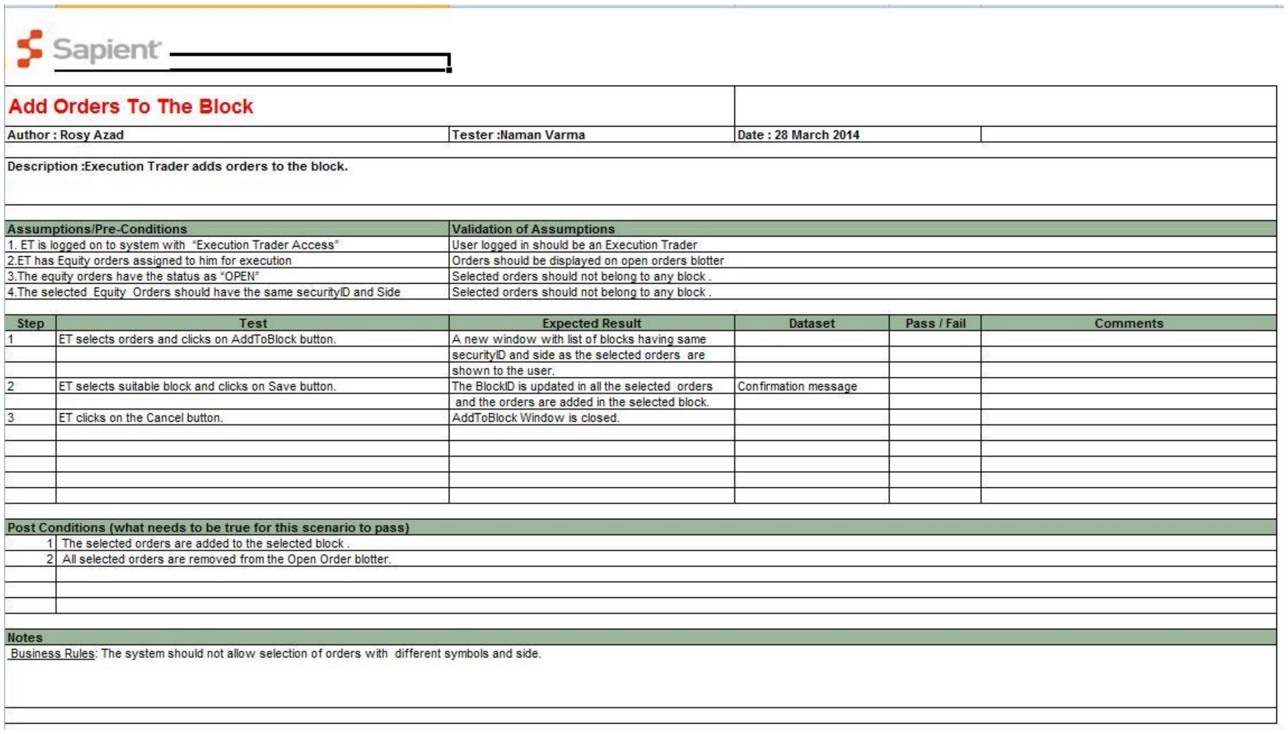
Assumptions/Pre-Conditions	Validation of Assumptions
1. User is logged in the system.	User logged in should be an Execution Trader.
2.Blocks with status New can be sent for execution.	
3. ET has chosen a block order and the block order ID	
has been identified	

Step	Test	Expected Result	Dataset	Pass / Fail	Comments
1	ET selects block(s) and clicks on send	The blocks are sent for execution to the broker			
	for execution button.	& status of the blocks is updated to sent for the			
		execution after a confirmation message from			
		broker.			

Post Conditions (what needs to be true for this scenario to pass)	
1	Block order is saved with Status "SENT FOR EXECUTION"'Executing'
2	Validation Messages displayed in case of failure of any business validation
3	The executing message is published on the message bus

Notes

Business Rules:1. The block status must be New



Sapient

Add Orders To The Block

Author : Rosy Azad	Tester :Naman Varma	Date : 28 March 2014

Description :Execution Trader adds orders to the block.

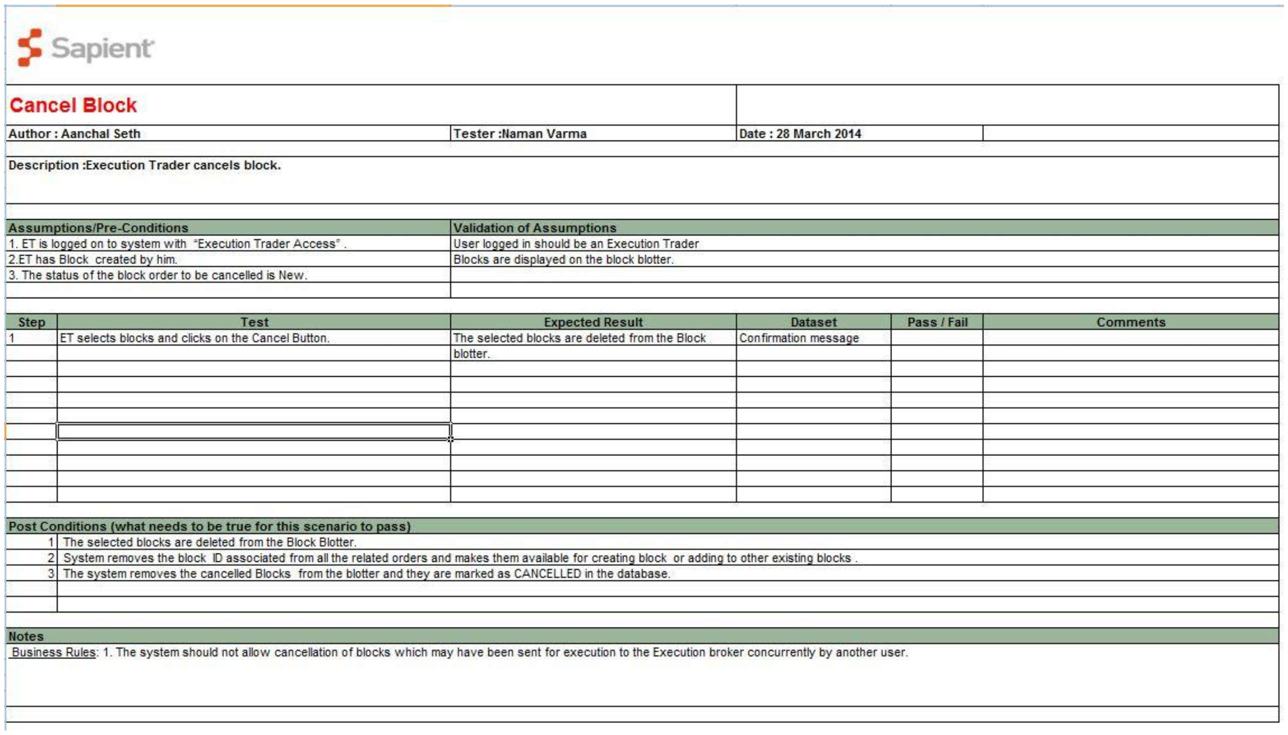
Assumptions/Pre-Conditions	Validation of Assumptions
1. ET is logged on to system with "Execution Trader Access"	User logged in should be an Execution Trader
2.ET has Equity orders assigned to him for execution	Orders should be displayed on open orders blotter
3.The equity orders have the status as "OPEN"	Selected orders should not belong to any block .
4.The selected Equity Orders should have the same securityID and Side	Selected orders should not belong to any block .

Step	Test	Expected Result	Dataset	Pass / Fail	Comments
1	ET selects orders and clicks on AddToBlock button.	A new window with list of blocks having same			
		securityID and side as the selected orders are			
		shown to the user.			
2	ET selects suitable block and clicks on Save button.	The BlockID is updated in all the selected orders	Confirmation message		
		and the orders are added in the selected block.			
3	ET clicks on the Cancel button.	AddToBlock Window is closed.			

Post Conditions (what needs to be true for this scenario to pass)	
1	The selected orders are added to the selected block .
2	All selected orders are removed from the Open Order blotter.

Notes

Business Rules: The system should not allow selection of orders with different symbols and side.



Cancel Block

Author : Aanchal Seth	Tester :Naman Varma	Date : 28 March 2014

Description :Execution Trader cancels block.

Assumptions/Pre-Conditions	Validation of Assumptions
1. ET is logged on to system with "Execution Trader Access" .	User logged in should be an Execution Trader
2.ET has Block created by him.	Blocks are displayed on the block blotter.
3. The status of the block order to be cancelled is New.	

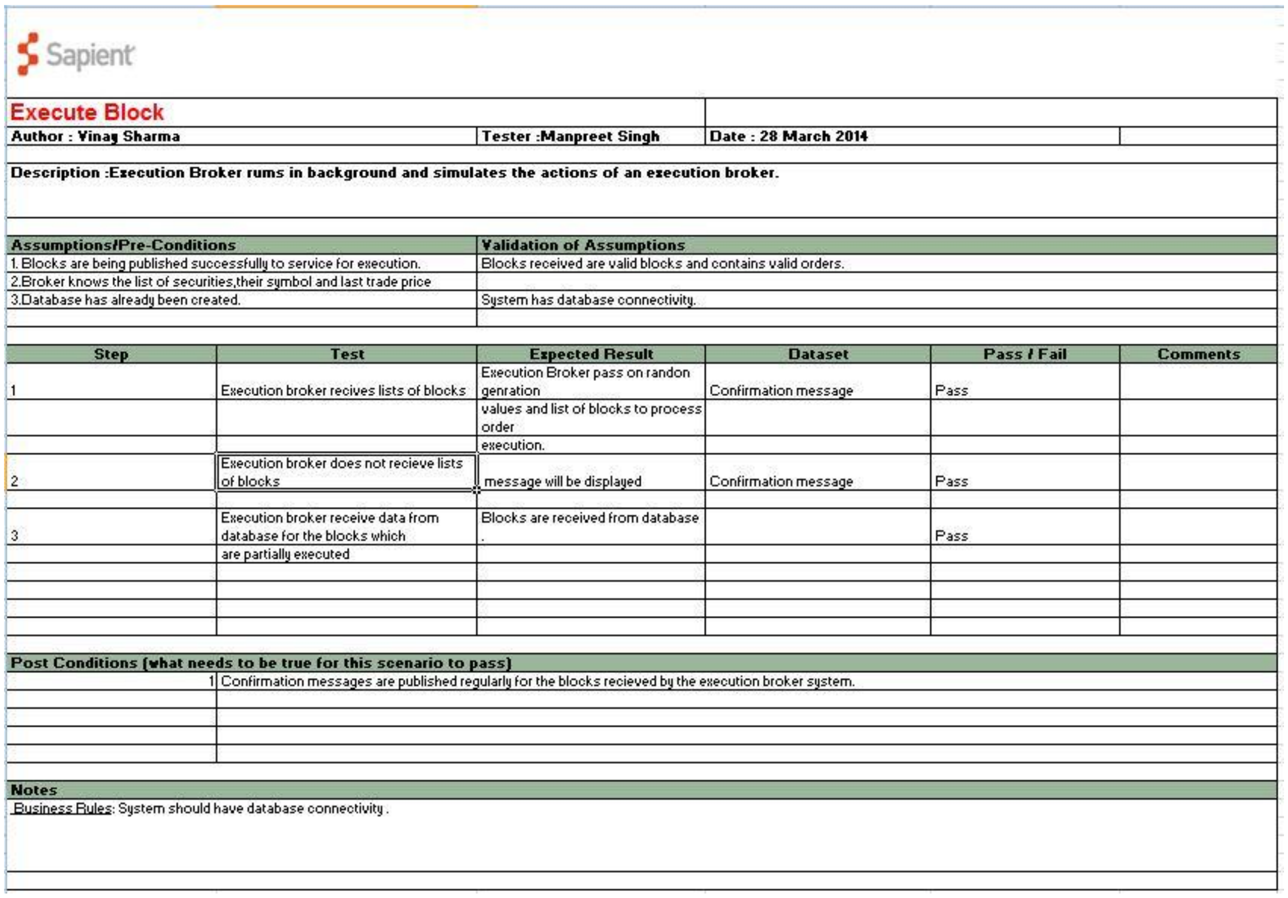
Step	Test	Expected Result	Dataset	Pass / Fail	Comments
1	ET selects blocks and clicks on the Cancel Button.	The selected blocks are deleted from the Block blotter.	Confirmation message		

Post Conditions (what needs to be true for this scenario to pass)

1	The selected blocks are deleted from the Block Blotter.
2	System removes the block ID associated from all the related orders and makes them available for creating block or adding to other existing blocks .
3	The system removes the cancelled Blocks from the blotter and they are marked as CANCELLED in the database.

Notes

Business Rules: 1. The system should not allow cancellation of blocks which may have been sent for execution to the Execution broker concurrently by another user.



Execute Block

Author : Vinay Sharma	Tester :Manpreet Singh	Date : 28 March 2014

Description :Execution Broker rums in background and simulates the actions of an execution broker.

Assumptions/Pre-Conditions	Validation of Assumptions
1. Blocks are being published successfully to service for execution.	Blocks received are valid blocks and contains valid orders.
2.Broker knows the list of securities,their symbol and last trade price	
3.Database has already been created.	System has database connectivity.

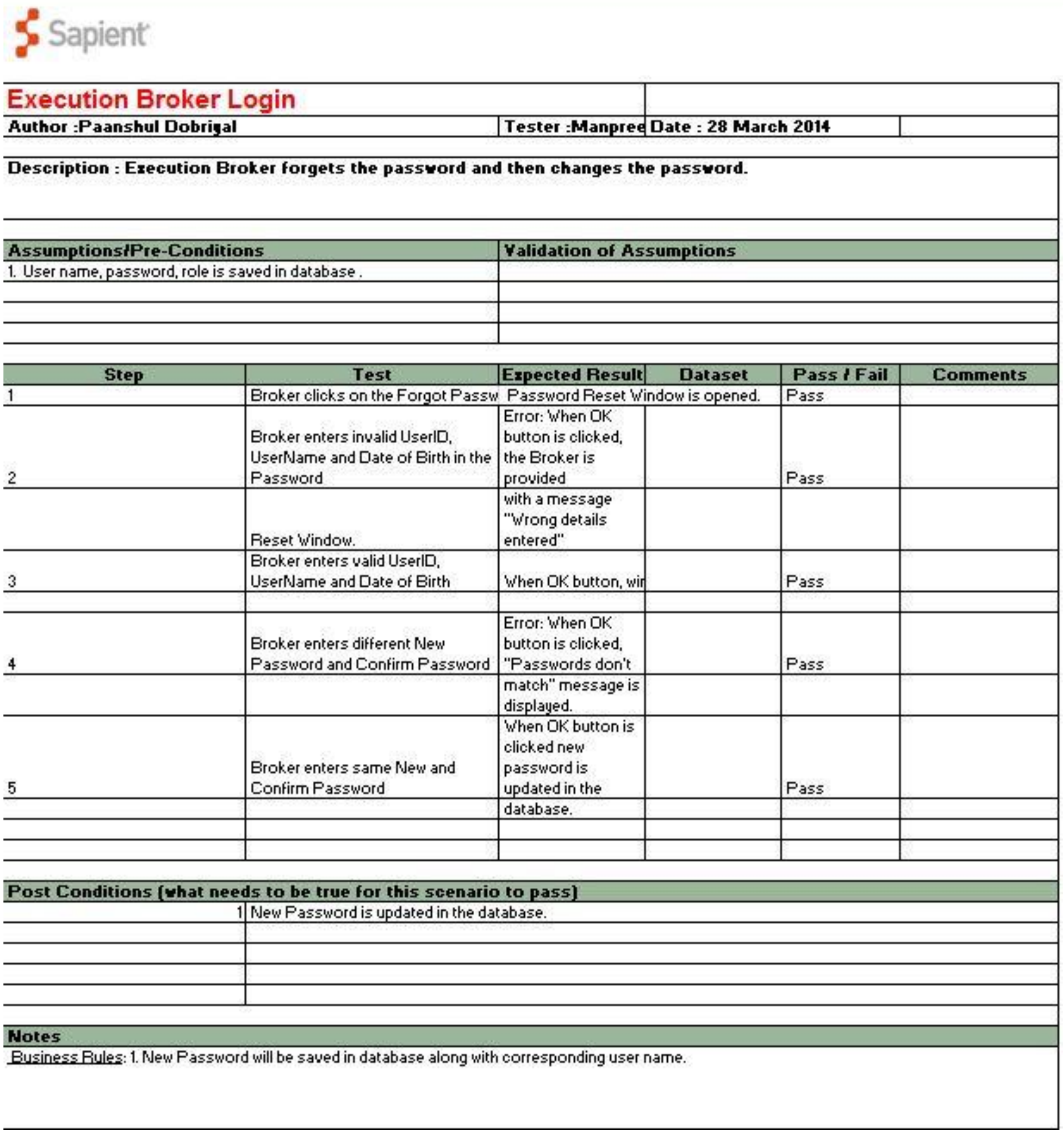
Step	Test	Expected Result	Dataset	Pass / Fail	Comments
1	Execution broker recives lists of blocks	Execution Broker pass on randon genration	Confirmation message	Pass	
		values and list of blocks to process order			
		execution.			
2	Execution broker does not recieve lists of blocks	message will be displayed	Confirmation message	Pass	
3	Execution broker receive data from database for the blocks which are partially executed	Blocks are received from database .		Pass	

Post Conditions (what needs to be true for this scenario to pass)

1	Confirmation messages are published regularly for the blocks recieved by the execution broker system.

Notes

Business Rules: System should have database connectivity .



Sapient

Execution Broker Login

Author :Paanshul Dobrigal	**Tester :Manpree**	**Date : 28 March 2014**

Description : Execution Broker forgets the password and then changes the password.

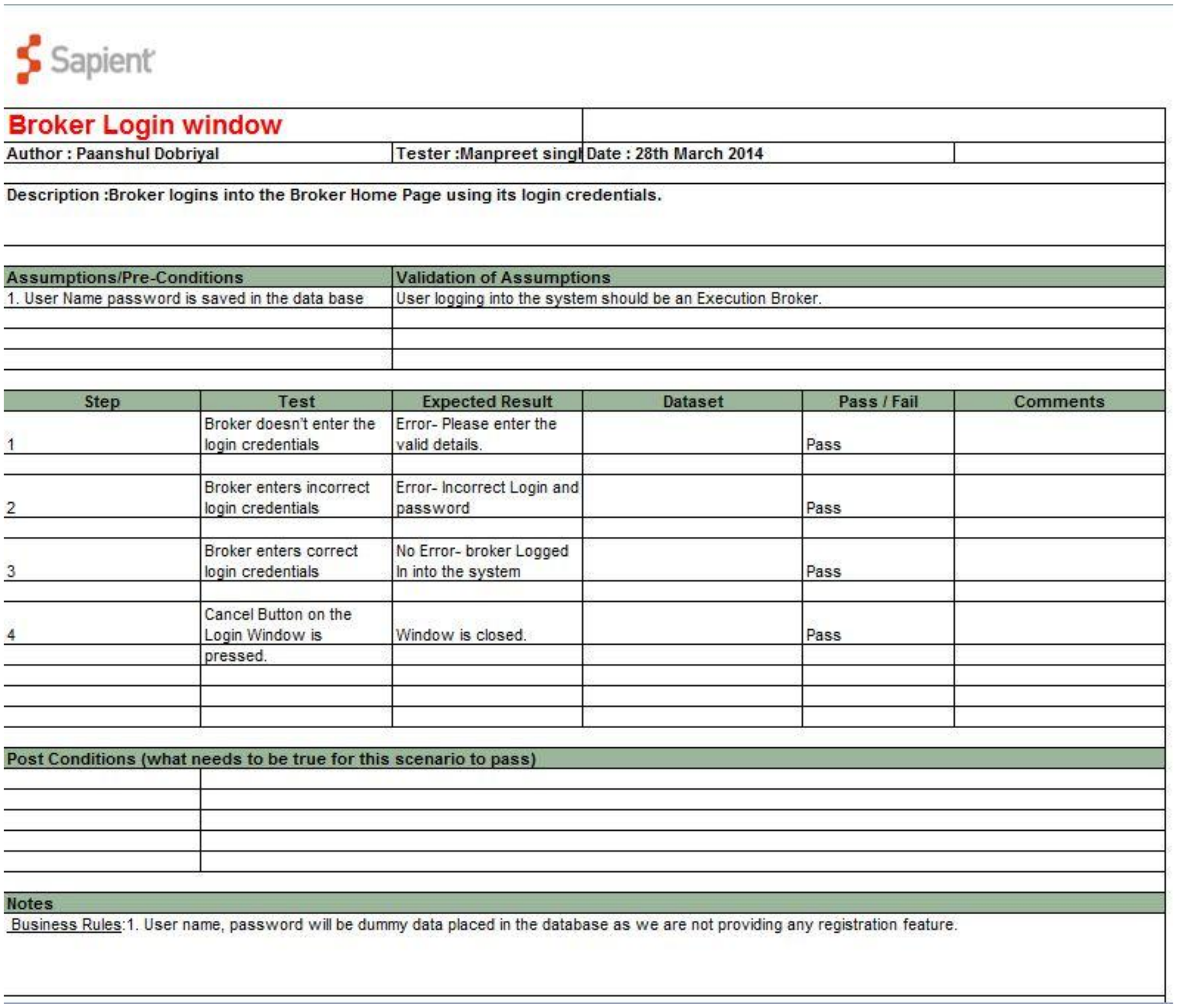
Assumptions/Pre-Conditions	**Validation of Assumptions**
1. User name, password, role is saved in database .	

Step	**Test**	**Expected Result**	**Dataset**	**Pass / Fail**	**Comments**
1	Broker clicks on the Forgot Passw	Password Reset Window is opened.		Pass	
2	Broker enters invalid UserID, UserName and Date of Birth in the Password	Error: When OK button is clicked, the Broker is provided		Pass	
	Reset Window.	with a message "Wrong details entered"			
3	Broker enters valid UserID, UserName and Date of Birth	When OK button, wir		Pass	
4	Broker enters different New Password and Confirm Password	Error: When OK button is clicked, "Passwords don't		Pass	
		match" message is displayed.			
5	Broker enters same New and Confirm Password	When OK button is clicked new password is updated in the		Pass	
		database.			

Post Conditions (what needs to be true for this scenario to pass)	
1	New Password is updated in the database.

Notes

Business Rules: 1. New Password will be saved in database along with corresponding user name.



Broker Login window

Author : Paanshul Dobriyal	**Tester :Manpreet singl**	**Date : 28th March 2014**

Description :Broker logins into the Broker Home Page using its login credentials.

Assumptions/Pre-Conditions	Validation of Assumptions
1. User Name password is saved in the data base	User logging into the system should be an Execution Broker.

Step	Test	Expected Result	Dataset	Pass / Fail	Comments
1	Broker doesn't enter the login credentials	Error- Please enter the valid details.		Pass	
2	Broker enters incorrect login credentials	Error- Incorrect Login and password		Pass	
3	Broker enters correct login credentials	No Error- broker Logged In into the system		Pass	
4	Cancel Button on the Login Window is pressed.	Window is closed.		Pass	

Post Conditions (what needs to be true for this scenario to pass)

Notes

Business Rules:1. User name, password will be dummy data placed in the database as we are not providing any registration feature.

5. RESULTS & DISCUSSIONS

1 Got acquainted with Trading and Risk Management Operations.

2. Basics and Working of Financial Markets

3. Working in Real Time Simulators to understand the paradigm of Market

- MARK IT
- Moneybhai.com

4. Working in an collaborative environment and facilitating multiple skills

- Effective communication
- Conflict Management
- Efficient Delegation
- Soft skills
- Facilitation Skills
- Presentation Skills

5. Working with different technology and development methodology.

6. Application that could work in real time scenario starting from the analysis of market to order a trade till the execution of the order.

7. Screen Shots

7.1 Login

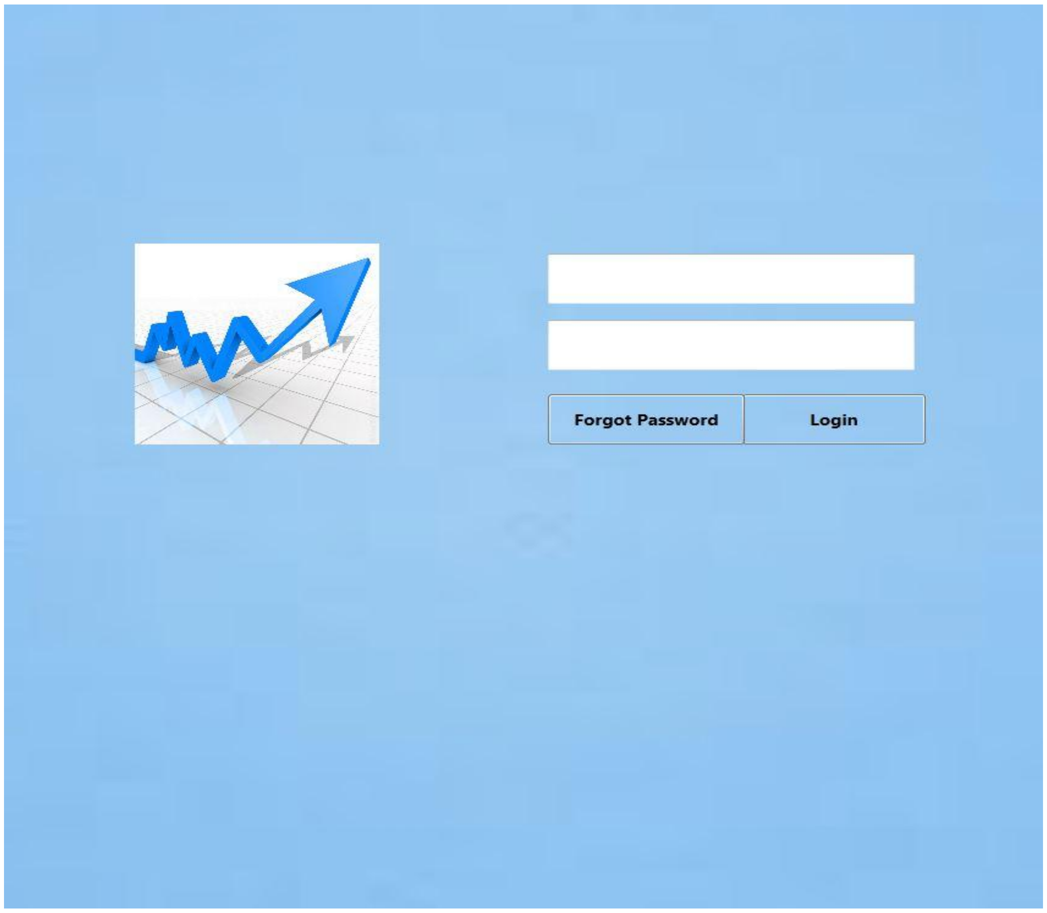
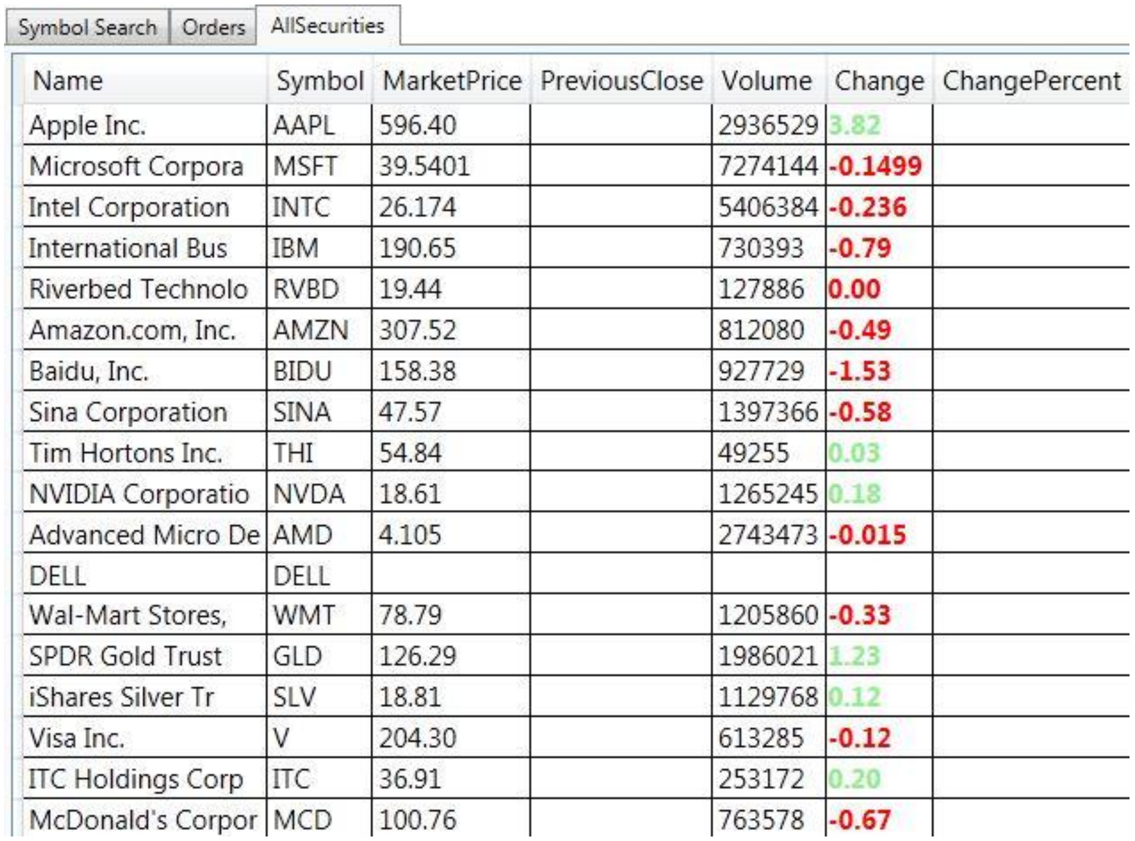


Fig 8

7.2 Portfolio Page

Fig 9

7.3 Securities

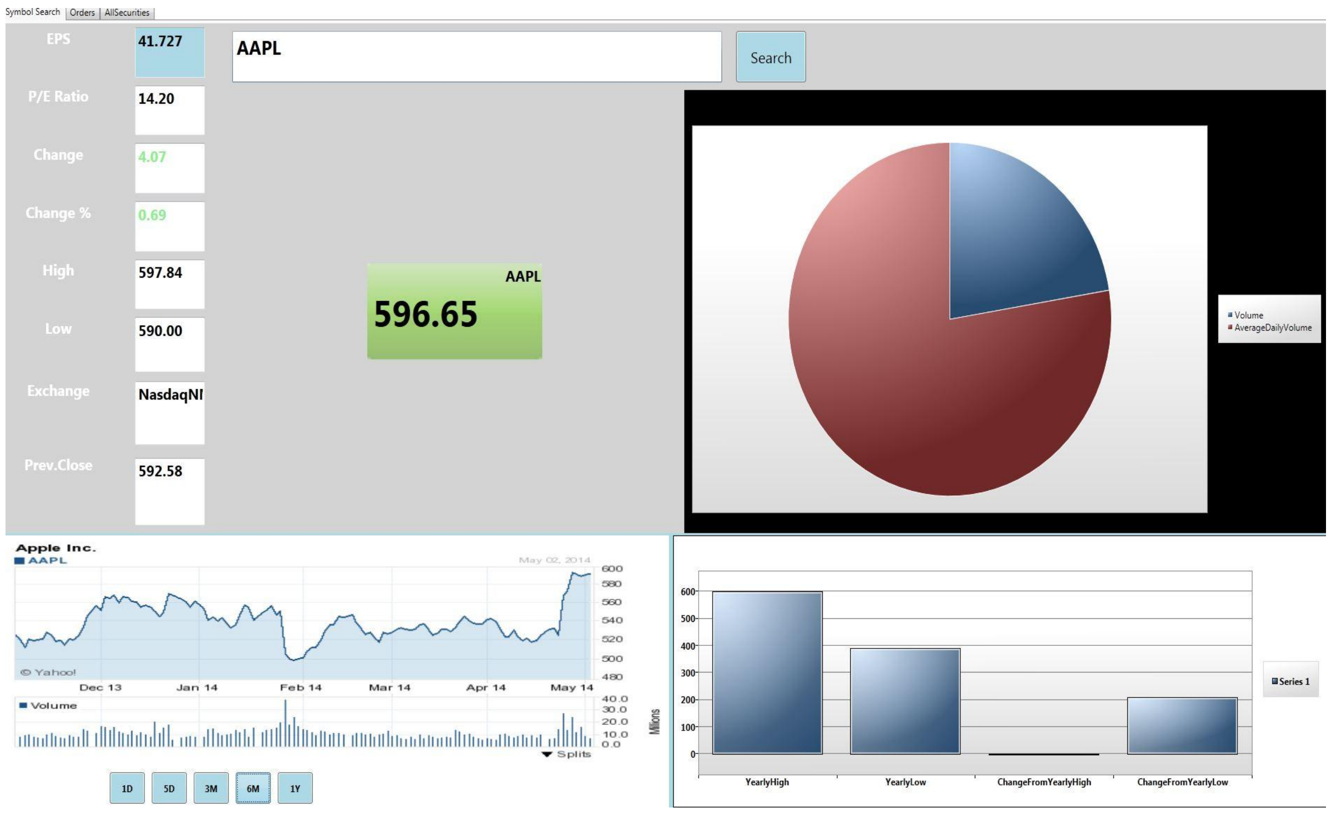


Symbol Search | Orders | AllSecurities

Name	Symbol	MarketPrice	PreviousClose	Volume	Change	ChangePercent
Apple Inc.	AAPL	596.40		2936529	3.82	
Microsoft Corpora	MSFT	39.5401		7274144	-0.1499	
Intel Corporation	INTC	26.174		5406384	-0.236	
International Bus	IBM	190.65		730393	-0.79	
Riverbed Technolo	RVBD	19.44		127886	0.00	
Amazon.com, Inc.	AMZN	307.52		812080	-0.49	
Baidu, Inc.	BIDU	158.38		927729	-1.53	
Sina Corporation	SINA	47.57		1397366	-0.58	
Tim Hortons Inc.	THI	54.84		49255	0.03	
NVIDIA Corporatio	NVDA	18.61		1265245	0.18	
Advanced Micro De	AMD	4.105		2743473	-0.015	
DELL	DELL					
Wal-Mart Stores,	WMT	78.79		1205860	-0.33	
SPDR Gold Trust	GLD	126.29		1986021	1.23	
iShares Silver Tr	SLV	18.81		1129768	0.12	
Visa Inc.	V	204.30		613285	-0.12	
ITC Holdings Corp	ITC	36.91		253172	0.20	
McDonald's Corpor	MCD	100.76		763578	-0.67	

Fig 10

7.4 Main Window



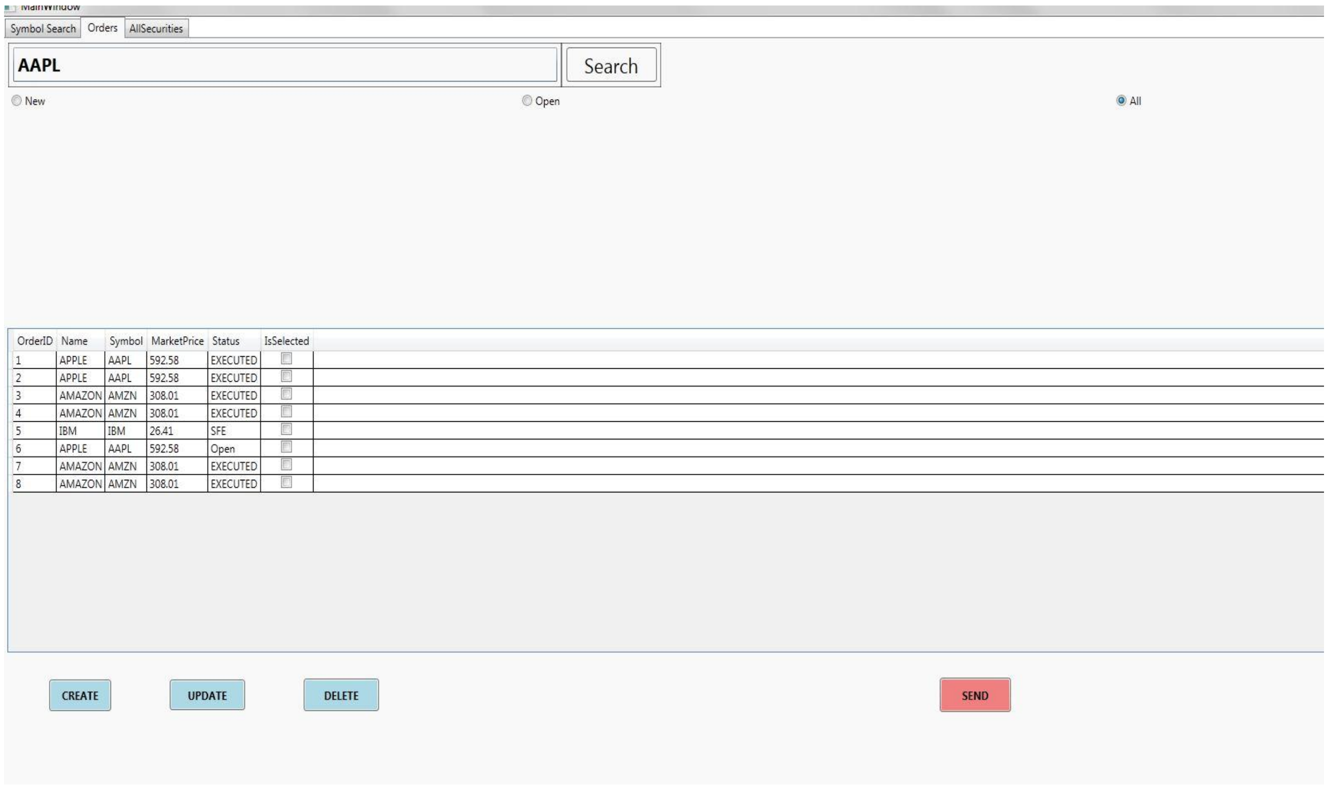
EPS	41.727
P/E Ratio	14.20
Change	4.07
Change %	0.69
High	597.84
Low	590.00
Exchange	NasdaqNI
Prev.Close	592.58

AAPL

AAPL
596.65

Fig 11

7.5 Order List



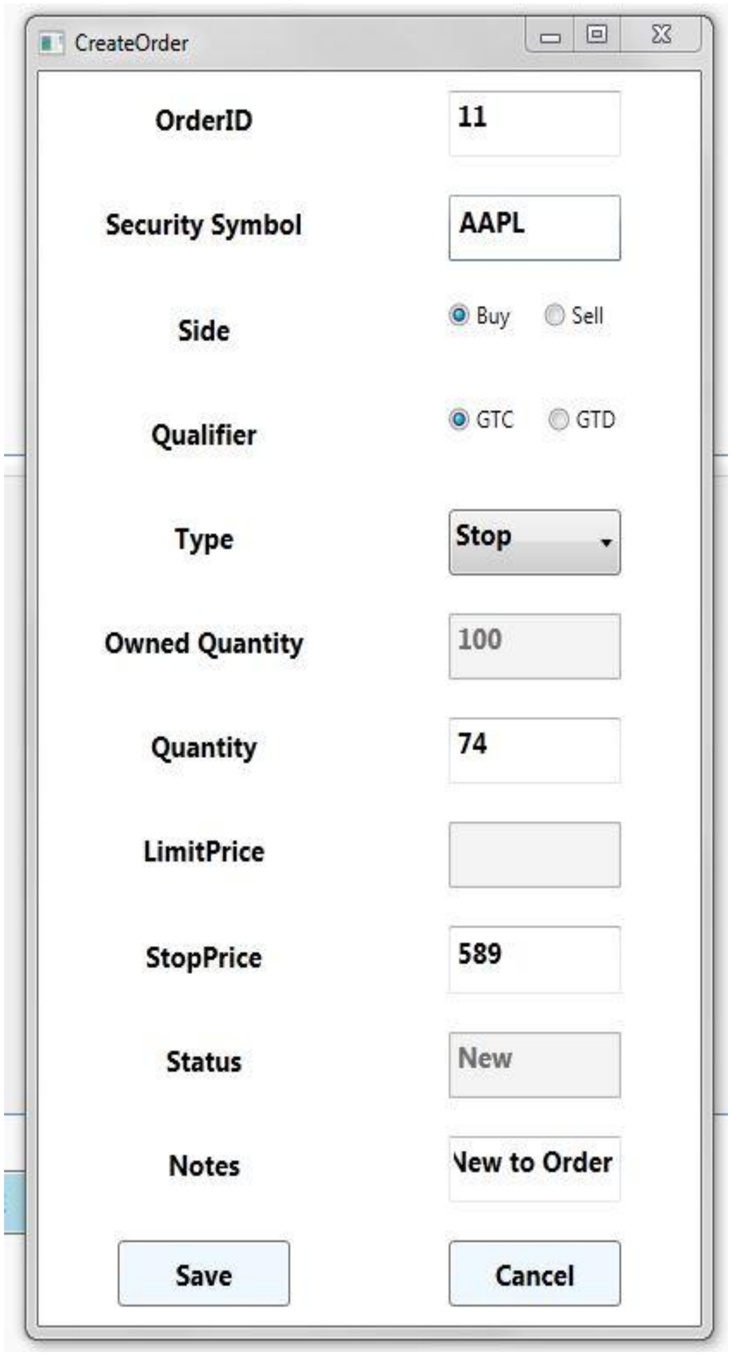
AAPL

New [ ] Open [ ] [✓ All]

OrderID	Name	Symbol	MarketPrice	Status	IsSelected
1	APPLE	AAPL	592.58	EXECUTED	☐
2	APPLE	AAPL	592.58	EXECUTED	☐
3	AMAZON	AMZN	308.01	EXECUTED	☐
4	AMAZON	AMZN	308.01	EXECUTED	☐
5	IBM	IBM	26.41	SFE	☐
6	APPLE	AAPL	592.58	Open	☐
7	AMAZON	AMZN	308.01	EXECUTED	☐
8	AMAZON	AMZN	308.01	EXECUTED	☐

Fig 12

7.6 Create Order

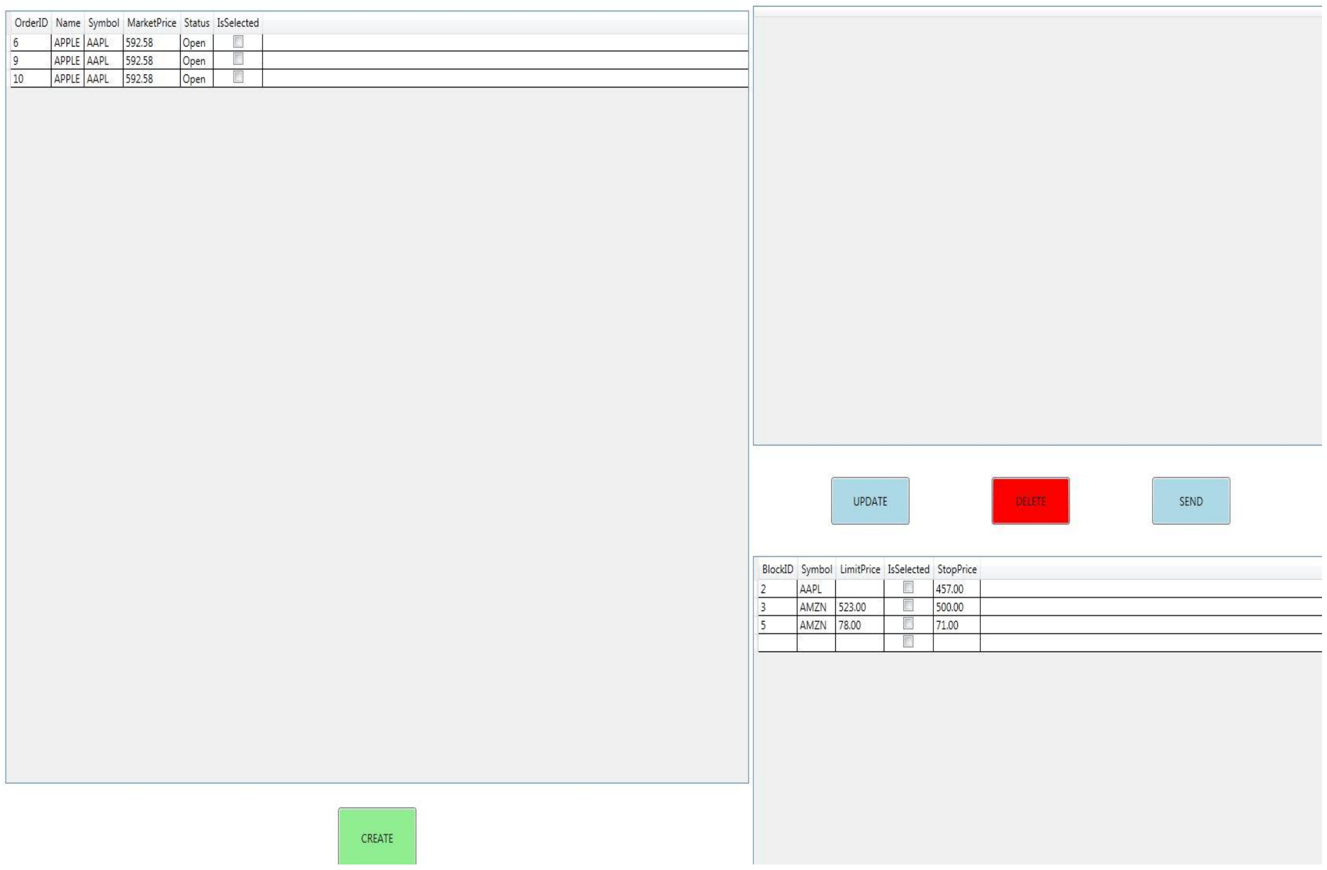


CreateOrder

OrderID	11
Security Symbol	AAPL
Side	[✓ Buy] Sell
Qualifier	[✓ GTC] GTD
Type	Stop
Owned Quantity	100
Quantity	74
LimitPrice	
StopPrice	589
Status	New
Notes	Vew to Order

Fig 13

7.7 Execution Trader



OrderID	Name	Symbol	MarketPrice	Status	IsSelected
6	APPLE	AAPL	592.58	Open	☐
9	APPLE	AAPL	592.58	Open	☐
10	APPLE	AAPL	592.58	Open	☐

BlockID	Symbol	LimitPrice	IsSelected	StopPrice
2	AAPL		☐	457.00
3	AMZN	523.00	☐	500.00
5	AMZN	78.00	☐	71.00
			☐	

Fig 14

6. CONCLUSION AND FUTURE WORK

1. In this Application the Portfolio Manager can successfully edit, amend or cancel the orders placed by him based on the required criteria.

2. Portfolio Manager and send them successfully to the Execution Trader.

3. The Portfolio Manager can also view the status of the orders placed by him after successful execution in the Application.

4. The new orders which are sent by Portfolio Manager will first go inside the Proposed Block and from there Execution trader can create a new Block or can add to the Existing Block.

5. The Execution Trader can view the existing blocks which are already been created.

6. The Execution can successfully edit, cancel and send the Blocks to the Broker System for execution.

7. The Broker System will take care of Block Execution and send back the executed block to the Execution Trader which in turn will send the status of the executed blocks to the Portfolio Manager.

7. APPENDICES

7.1 CODE MVVM

- Model Order

```
namespace PortfolioManager.Models
{
  public  class OrderModel
    {
        public int OrderID { get; set; }

        public int StockID { get; set; }

        public string Side { get; set; }

        public int StatusID { get; set; }

        public int BlockID { get; set; }

        public string Qualifier { get; set; }

        public EnumOrderType OrderType { get; set; }

        public int OwnedQuantity { get; set; }

        public int Quantity { get; set; }

        public string Notes { get; set; }

        public bool IsSelected { get; set; }
  }
}
```

- View Main Window

```
<Window x:Class="PortfolioManager.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:local="clr-namespace:PortfolioManager.Helpers"
        xmlns:converter="clr-namespace:PortfolioManager.Converters"
        xmlns:auto="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Input.Toolkit"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:chart="clr-
namespace:System.Windows.Controls.DataVisualization.Charting;assembly=System.Windows.Controls.Dat
aVisualization.Toolkit"
        Title="MainWindow" WindowState="Maximized" ResizeMode="NoResize" mc:Ignorable="d"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" Height="945" Width="1594">

    <Window.Resources>
        <converter:BoolToVisibilityConverter
x:Key="BoolToVisibility"></converter:BoolToVisibilityConverter>
        <converter:PositiveNegativeIndicatorConverter
x:Key="PositveNegative"></converter:PositiveNegativeIndicatorConverter>
        <converter:PositiveNegativeAllSecurities
x:Key="AllSecuritiesConverter"></converter:PositiveNegativeAllSecurities>

        <Style TargetType="DataGridRow">
            <Setter Property="IsSelected" Value="{Binding IsSelected,Mode=TwoWay}" />
        </Style>


    </Window.Resources>
    <Grid>
        <TabControl>
            <TabItem Header="Symbol Search">
                <Grid Name="Grid" ShowGridLines="False"  Background="LightGray">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="3.5*"/>
                        <RowDefinition Height="1.9*"/>

                    </Grid.RowDefinitions>

                    <Grid Name="topGrid" ShowGridLines="False" Grid.Row="0">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="0.2*"/>
                            <ColumnDefinition/>
                        </Grid.ColumnDefinitions>

                        <Grid Name="profitabilityGrid" ShowGridLines="False" Grid.Column="0">
                            <Grid.RowDefinitions>
                                <RowDefinition Height="0.7*"/>
                                <RowDefinition Height="0.7*" />
                                <RowDefinition Height="0.7*"/>
                                <RowDefinition Height="0.7*" />
                                <RowDefinition Height="0.7*"/>
                                <RowDefinition Height="0.75*" />
                                <RowDefinition Height="0.9*" />
                                <RowDefinition  />
                            </Grid.RowDefinitions>

                            <Grid.ColumnDefinitions>
                                <ColumnDefinition/>
                                <ColumnDefinition/>
                            </Grid.ColumnDefinitions>

                            <Label Name="epsLabel" Content="EPS" HorizontalAlignment="Center"
FontWeight="Bold" Foreground="White" FontSize="20" Grid.Row="0" Grid.Column="0" ></Label>
                            <Label Name="peRatioLabel" Content="P/E Ratio" HorizontalAlignment="Center"
FontWeight="Bold" Foreground="White" FontSize="20" Grid.Row="1" Grid.Column="0" ></Label>
                            <Label Name="change" Content="Change" HorizontalAlignment="Center"
FontWeight="Bold" Foreground="White" FontSize="20" Grid.Row="2" Grid.Column="0" ></Label>
                            <Label Name="changePercent" Content="Change %" HorizontalAlignment="Center"
FontWeight="Bold" Foreground="White" FontSize="20" Grid.Row="3" Grid.Column="0" ></Label>
                            <Label Name="high" Content="High" HorizontalAlignment="Center" FontWeight="Bold"
Foreground="White" FontSize="20" Grid.Row="4" Grid.Column="0" ></Label>
                            <Label Name="low" Content="Low" HorizontalAlignment="Center" FontWeight="Bold"
Foreground="White" FontSize="20" Grid.Row="5" Grid.Column="0" ></Label>
                            <Label Name="stockExchange" Content="Exchange" HorizontalAlignment="Center"
FontWeight="Bold" Foreground="White" FontSize="20" Grid.Row="6" Grid.Column="0" ></Label>
                            <Label Name="previousClose" Content="Prev.Close" HorizontalAlignment="Center"
FontWeight="Bold" Foreground="White" FontSize="20" Grid.Row="7" Grid.Column="0" ></Label>

                            <TextBox Name="epsLabelValue" Text="{Binding EPS}" HorizontalAlignment="Center"
FontWeight="Bold" Width="100" Foreground="Black" FontSize="20" Grid.Row="0" Grid.Column="1"
Margin="5" Background="LightBlue"></TextBox>
                            <TextBox Name="peRatioLabelValue" Text="{Binding PERatio}"
HorizontalAlignment="Center" FontWeight="Bold" Width="100" Foreground="Black" FontSize="20"
Grid.Row="1" Grid.Column="1" Margin="5" ></TextBox>
                            <TextBox Name="changeLabelValue" Text="{Binding Change}"
HorizontalAlignment="Center" FontWeight="Bold" Width="100" Foreground="{Binding Path=Change,
Converter={StaticResource PositveNegative}}" FontSize="20" Grid.Row="2" Grid.Column="1"
Margin="5" ></TextBox>
                            <TextBox Name="changePercentLabelValue" Text="{Binding ChangePercent}"
HorizontalAlignment="Center" FontWeight="Bold" Width="100" Foreground="{Binding
Path=ChangePercent, Converter={StaticResource PositveNegative}}" FontSize="20" Grid.Row="3"
Grid.Column="1" Margin="5"></TextBox>
                            <TextBox Name="highLabelValue" Text="{Binding High}"
HorizontalAlignment="Center" FontWeight="Bold" Width="100" Foreground="Black" FontSize="20"
Grid.Row="4" Grid.Column="1" Margin="5"></TextBox>
                            <TextBox Name="lowPercentLabelValue" Text="{Binding Low}"
HorizontalAlignment="Center" FontWeight="Bold" Width="100" Foreground="Black" FontSize="20"
Grid.Row="5" Grid.Column="1" Margin="5" ></TextBox>
```

```
                    <TextBox Name="stockExchangeLabelValue" Text="{Binding StockExchange}"
HorizontalAlignment="Center" FontWeight="Bold" Width="100" Foreground="Black" FontSize="20"
Grid.Row="6" Grid.Column="1" Margin="7" ></TextBox>
                    <TextBox Name="previousCloseLabelValue" Text="{Binding PreviousClose}"
HorizontalAlignment="Center" FontWeight="Bold" Width="100" Foreground="Black" FontSize="20"
Grid.Row="7" Grid.Column="1" Margin="10" ></TextBox>

                </Grid>

                <Grid Name="mainGrid" ShowGridLines="False" Grid.Column="1">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="0.15*"/>
                        <RowDefinition/>
                    </Grid.RowDefinitions>

                    <StackPanel Orientation="Horizontal" Grid.Row="0" Name="searchPanel">

                        <auto:AutoCompleteBox Name="tBoxSearch" Width="700" Text="{Binding
SearchText,Mode=TwoWay}" ItemsSource="{Binding Symbols}" FontSize="25" Margin="10"
FontWeight="Bold"></auto:AutoCompleteBox>

                        <Button Name="btnSearch" Width="100" Margin="10" FontSize="20"
Background="LightBlue" Command="{Binding SearchCommand}"  Content="Search"></Button>
                    </StackPanel>




                    <Grid Grid.Row="1">

                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="0.7*"/>
                            <ColumnDefinition/>
                        </Grid.ColumnDefinitions>

                        <Grid Width="250" Height="125">
                            <Grid.Background>
                                <ImageBrush ImageSource="/Resources/glossyBack.png"
Opacity="0.5"></ImageBrush>
                            </Grid.Background>

                            <Grid.ColumnDefinitions>
                                <ColumnDefinition/>
                                <ColumnDefinition Width="0.5*"/>
                            </Grid.ColumnDefinitions>

                            <Label Name="currentPrice" Content="{Binding CurrentPrice}" Grid.Column="0"
FontSize="48" FontWeight="Bold" Margin="2" HorizontalAlignment="Center"
VerticalAlignment="Center" ></Label>
```

```
                                <TextBlock Name="symbol" VerticalAlignment="Top" Grid.Column="1"
Margin="2" FontSize="20" FontWeight="Bold" Text="{Binding SymbolSearched}"
HorizontalAlignment="Right">
                                </TextBlock>

                            </Grid>

                            <Grid Grid.Column="1" ShowGridLines="False">
                                <chart:Chart Canvas.Top="80"  Canvas.Left="10" Name="mcChart"
Background="Black">
                                    <chart:Chart.Series>
                                        <chart:PieSeries Title="Experience" ItemsSource="{Binding ListForPieChart}"
                                         IndependentValueBinding="{Binding Path=Name}"
                                         DependentValueBinding="{Binding Path=Value}">
                                        </chart:PieSeries>
                                    </chart:Chart.Series>
                                </chart:Chart>
                            </Grid>
                        </Grid>

                    </Grid>

                </Grid>

                <Grid Name="graphAndChartGrid" ShowGridLines="False" Grid.Row="1">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition/>
                        <ColumnDefinition/>
                    </Grid.ColumnDefinitions>
                    <Border BorderBrush="LightBlue" BorderThickness="3">
                        <Grid Name="graphGrid" Background="White">
                            <Grid.RowDefinitions>
                                <RowDefinition Height="6*"/>
                                <RowDefinition/>
                            </Grid.RowDefinitions>
                            <Grid Name="graph" Background="{Binding BrushGraph}" Grid.Row="0"
Margin="10">

                            </Grid>

                            <StackPanel Name="durationButtons" Orientation="Horizontal" Grid.Row="1">
                                <Button Name="oneDay" Content="1D" Command="{Binding OneDayCommand}"
Width="50" Margin="150,5,5,5" FontWeight="Bold" Background="LightBlue"></Button>
                                <Button Name="fiveDays" Content="5D" Command="{Binding
FiveDayCommand}"   Width="50" Margin="5" FontWeight="Bold" Background="LightBlue"></Button>
                                <Button Name="threeMonths" Content="3M" Command="{Binding
ThreeMonthCommand}" Width="50" Margin="5" FontWeight="Bold"
Background="LightBlue"></Button>
                                <Button Name="sixMonths" Content="6M" Command="{Binding
SixMonthCommand}" Width="50" Margin="5" FontWeight="Bold" Background="LightBlue"></Button>
```

```
                              <Button Name="oneYear" Content="1Y" Command="{Binding
OneYearCommand}" Width="50" Margin="5" FontWeight="Bold" Background="LightBlue"></Button>
                          </StackPanel>
                      </Grid>
                  </Border>
                  <Border BorderBrush="LightBlue" BorderThickness="3" Grid.Column="1">
                      <Grid Name="chart" >

                          <chart:Chart Name="stockChart"
                                 Background="White" Foreground="Black" FontWeight="Bold">
                              <chart:Chart.Series>
                                  <chart:ColumnSeries  Foreground="Black" Background="Black"
                                         ItemsSource="{Binding ListForCharts}"
                                         IndependentValueBinding="{Binding Path=NameOfStock}"
                                         DependentValueBinding="{Binding Path=NetPnL}">
                                  </chart:ColumnSeries>
                              </chart:Chart.Series>
                          </chart:Chart>

                      </Grid>
                  </Border>
              </Grid>


          </Grid>
      </TabItem>

      <TabItem Header="Orders">
          <UniformGrid>
              <Grid>
                  <Grid.RowDefinitions>
                      <RowDefinition Height="50" />
                      <RowDefinition Height="30"/>
                      <RowDefinition Height="300"/>
                      <RowDefinition/>
                  </Grid.RowDefinitions>

                  <StackPanel Name="sPanel" Orientation="Horizontal" Margin="1">
                      <Border BorderBrush="Black" BorderThickness="0.5" >

                          <auto:AutoCompleteBox Name="searchBox" Text="{Binding
SearchTextOrder,Mode=TwoWay}" ItemsSource="{Binding Symbols}" FontSize="20"
FontWeight="Bold" Width="650" Margin="5" Background="Transparent"></auto:AutoCompleteBox>

                      </Border>
                      <Border BorderBrush="Black" BorderThickness="0.5">
                          <Button Name="btSearch" Width="108" Command="{Binding
SearchCommandOrder}" Background="Transparent" Margin="5" >
                              <Button.Triggers>
                                  <EventTrigger RoutedEvent="Button.Click">
                                      <BeginStoryboard>
```

```
                                   <Storyboard>
                                       <DoubleAnimation
                        Storyboard.TargetName="btnSearch"
                        Storyboard.TargetProperty="Opacity"
                        From="1.0" To="0.0" Duration="0:0:1"
                        AutoReverse="True"/>
                                   </Storyboard>
                               </BeginStoryboard>
                           </EventTrigger>
                        </Button.Triggers>
                        <StackPanel Orientation="Horizontal">
                           <TextBlock FontSize="22" >Search</TextBlock>
                        </StackPanel>
                     </Button>
                  </Border>
               </StackPanel>


               <StackPanel Name="radioButtons" Orientation="Horizontal" Grid.Row="1">
                  <RadioButton Name="rBtnNew" Content="New" Margin="5" Width="600"
HorizontalAlignment="Left" Command="{Binding NewCommand}"/>

                  <RadioButton Name="rbtnOpen" Content="Open" Margin="5" Width="700"
HorizontalAlignment="Center" Command="{Binding OpenCommand}"/>

                  <RadioButton Name="rbtnAll" Content="All" Margin="5" Width="100"
HorizontalAlignment="Right" Command="{Binding AllCommand}"/>

               </StackPanel>

               <DataGrid Name="ordersGrid" Grid.Row="2"
MouseLeftButtonDown="dataGrid_MouseLeftButtonDown" ItemsSource="{Binding ListToDisplay}"
SelectedItem="{Binding SelectedItem,Mode=TwoWay}" IsReadOnly="True" Height="349"
Margin="0,0,0,141" Grid.RowSpan="2">
                  <DataGrid.Resources>
                     <Style TargetType="DataGridRow">
                        <Setter Property="IsSelected"
                             Value="{Binding IsSelected}" />
                     </Style>
                  </DataGrid.Resources>
               </DataGrid>

               <Grid Name="buttonsGrid" ShowGridLines="True" Grid.Row="3">
                  <StackPanel Orientation="Horizontal">
                     <Button Name="btnCreate" Content="CREATE" Height="34" Width="74"
Margin="50,20,20,20" FontWeight="Bold" Background="LightBlue" Command="{Binding
CreateCommand}"></Button>
                     <Button Name="btnUpdate" Content="UPDATE" Height="33" Width="90"
Margin="50,20,20,20" FontWeight="Bold" Background="LightBlue"  Command="{Binding
UpdateCommand}"></Button>
```

```
                <Button Name="btnDelete" Content="DELETE" Height="35" Width="90"
Margin="50,20,20,20" FontWeight="Bold"  Background="LightBlue"  Command="{Binding
DeleteCommand}"></Button>
                <Button Name="btnSend" Content="SEND" Height="37" Width="85"
Margin="650,20,20,20" FontWeight="Bold"  Background="LightCoral" Command="{Binding
SendCommand}" ></Button>
              </StackPanel>
            </Grid>
          </Grid>
        </UniformGrid>

      </TabItem>

      <TabItem Name="ALLSecurities" Header="AllSecurities">

        <DataGrid Name="All" FontSize="15" IsReadOnly="True" ItemsSource="{Binding
AllSecurities}" AutoGenerateColumns="False">
          <DataGrid.Columns>
            <DataGridTextColumn Header="Name" Binding="{Binding Name,Mode=TwoWay}"
Foreground="Black"  />
            <DataGridTextColumn Header="Symbol" Binding="{Binding Symbol,Mode=TwoWay}"
Foreground="Black" />
            <DataGridTextColumn Header="MarketPrice" Binding="{Binding
MarketPrice,Mode=TwoWay}" Foreground="Black" />
            <DataGridTextColumn Header="PreviousClose" Binding="{Binding
PreviousClose,Mode=TwoWay}" Foreground="Black" />
            <DataGridTextColumn Header="Volume" Binding="{Binding Volume,Mode=TwoWay}"
Foreground="Black" />
            <DataGridTextColumn Header="Change" Binding="{Binding Change,Mode=TwoWay}"
FontWeight="Bold">
              <DataGridTextColumn.ElementStyle>
                <Style TargetType="{x:Type TextBlock}">
                  <Setter Property="Foreground" Value="{Binding
Path=Change,Converter={StaticResource AllSecuritiesConverter}}">

                  </Setter>
                </Style>
              </DataGridTextColumn.ElementStyle>

            </DataGridTextColumn>
            <DataGridTextColumn Header="ChangePercent" Binding="{Binding
ChangePercent,Mode=TwoWay}" Foreground="Black" />
          </DataGrid.Columns>
        </DataGrid>

      </TabItem>

    </TabControl>
  </Grid>
</Window>
```

- Update Order View Model

```
namespace PortfolioManager.ViewModels
{
    public class UpdateOrderViewModel : ViewModelBase
    {
        PortfolioDAL dalObject;

        private bool buy;
        public bool Buy
        {
            get { return buy; }
            set
            {
                buy = value;
                RaisePropertyChanged("Buy");
            }
        }

        private bool sell;
        public bool Sell
        {
            get { return sell; }
            set
            {
                sell = value;
                RaisePropertyChanged("Sell");

            }
        }

        private bool gtc;
        public bool GTC
        {
            get { return gtc; }
            set
            {
                gtc = value;
                RaisePropertyChanged("GTC");
            }
        }

        private bool gtd;
        public bool GTD
        {
            get { return gtd; }
            set
            {
                gtd = value;
                RaisePropertyChanged("GTD");
            }
        }

        private bool limitEnabled;
        public bool LimitEnabled
        {
            get { return limitEnabled; }
            set
            {
                limitEnabled = value;
                RaisePropertyChanged("LimitEnabled");
            }
        }

        private bool stopEnabled;
        public bool StopEnabled
        {
            get { return stopEnabled; }
            set
            {
                stopEnabled = value;
                RaisePropertyChanged("StopEnabled");
            }
        }

        public UpdateOrderViewModel()
        {
            dalObject = new PortfolioDAL();
            OrderId = dalObject.GetMaxOrderID().ToString();
            Symbols = dalObject.GetAllSymbols();
            Buy = true;
            GTC = true;
            TypeOfOrders = new List<string>() {
            "Market","Stop","Limit","StopLimit"
            };
            Status = "New";
            LimitEnabled = false;
            StopEnabled = false;
            symbolSelected = "AAPL";
        }

        private string orderId;
        public string OrderId
        {
            get { return orderId; }
            set
            {
                orderId = value;
                RaisePropertyChanged("OrderId");
            }
        }
```

```
private List<string> typeOfOrders;
public List<string> TypeOfOrders
{
   get { return typeOfOrders; }
   set { typeOfOrders = value; }
}

private string ownedQuantity;
public string OwnedQuantity
{
   get { return ownedQuantity; }
   set
   {
      ownedQuantity = value;
      RaisePropertyChanged("OwnedQuantity");
   }
}

private string quantity;
public string Quantity
{
   get { return quantity; }
   set
   {
      quantity = value;
      RaisePropertyChanged("Quantity");
   }
}

private string limitPrice;
public string LimitPrice
{
   get { return limitPrice; }
   set
   {
      limitPrice = value;
      RaisePropertyChanged("LimitPrice");
   }
}

private string stopPrice;
public string StopPrice
{
   get { return stopPrice; }
   set
   {
      stopPrice = value;
      RaisePropertyChanged("StopPrice");
   }
}
```

```
private string status;
public string Status
{
    get { return status; }
    set
    {
        status = value;
        RaisePropertyChanged("Status");
    }
}

private string notes;
public string Notes
{
    get { return notes; }
    set
    {
        notes = value;
        RaisePropertyChanged("Notes");
    }
}

private List<string> symbols;
public List<string> Symbols
{
    get { return symbols; }
    set { symbols = value; }
}

private string selectedType;

public string SelectedType
{
    get { return selectedType; }
    set
    {
        selectedType = value;
        RaisePropertyChanged("SelectedValueType");
        SelectedOrderType();
    }
}

private string symbolSelected;
public string SymbolSelected
{
    get { return symbolSelected; }
    set
    {
        symbolSelected = value;
        RaisePropertyChanged("SymbolSelected");
    }
}

private ICommand buyCommand;
public ICommand BuyCommand
{
    get
    {
        if (buyCommand == null)
            buyCommand = new RelayCommand(p => BuySelected());

        return buyCommand;
    }

}

private ICommand sellCommand;
public ICommand SellCommand
{
    get
    {
        if (sellCommand == null)
            sellCommand = new RelayCommand(p => SellSelected());

        return sellCommand;
    }

}

private ICommand saveCommand;
public ICommand SaveCommand
{
    get
    {
        if (saveCommand == null)
            saveCommand = new RelayCommand(p => Save());
        return saveCommand;
    }

}

private ICommand gtcCommand;
public ICommand GtcCommand
{
    get
    {
        if (gtcCommand == null)
            gtcCommand = new RelayCommand(p => GtcSelected());

        return gtcCommand;
    }

}

private ICommand gtdCommand;
public ICommand GtdCommand
{
    get
    {
        if (gtdCommand == null)
            gtdCommand = new RelayCommand(p => GtdSelected());

        return gtdCommand;
    }

}

private void BuySelected()
{
    Buy = true;
}

private void SellSelected()
{
    Sell = true;
}

private void GtcSelected()
{
    GTC = true;
}

private void GtdSelected()
{
    GTD = true;
}

private void SelectedOrderType()
{
    if (SelectedType == "StopLimit" && Status == "New")
    {
        LimitEnabled = true;
        StopEnabled = true;
    }
    else if (SelectedType == "Limit" && Status=="New")
    {
        LimitEnabled = true;
    }
    else if (SelectedType == "Stop" && Status=="New")
    {
        StopEnabled = true;
```

```
            }

        }

        private void Save()
        {
            int orderId = int.Parse((OrderId));
            Order order = dalObject.GetOrderByOrderId(orderId);
            order.Quantity = int.Parse(Quantity);
            dalObject.UpdateOrder(order);
        }
    }
}
```

- Main Window ViewModel

```
namespace PortfolioManager.ViewModels
{
    enum RadioState { New, Open, All, Search }

    public class MainWindowViewModel : ViewModelBase
    {
        private PortfolioDAL dalObject;
        IModelDialogService dialogService;
        Random r;
        RadioState state = RadioState.All;

        private readonly DispatcherTimer timer = new DispatcherTimer(DispatcherPriority.Background);
        public ObservableCollection<Quote> Quotes { get; set; }

        //List of orders from DAL
        private ObservableCollection<OrderModel> listFromDAL;
        public ObservableCollection<OrderModel> ListFromDAL
        {
            get { return listFromDAL; }
            set { listFromDAL = value; }
        }

        private GridFields  selectedItem;
        public GridFields  SelectedItem
        {
            get { return selectedItem; }
            set { selectedItem = value;
            RaisePropertyChanged("SelectedItem");
            }
        }

        private ObservableCollection<AllData> allSecurities;
        public ObservableCollection<AllData> AllSecurities
        {
            get { return allSecurities; }
    set { allSecurities = value; }
}

private List<string> symbols;
public List<string> Symbols
{
    get { return symbols; }
    set { symbols = value; }
}

private ObservableCollection<PieChartData> listForPieChart;
public ObservableCollection<PieChartData> ListForPieChart
{
    get { return listForPieChart; }
    set { listForPieChart = value; }
}

private string high;
public string High
{
    get { return high; }
    set
    {
        high = value;
        RaisePropertyChanged("High");

    }
}

private decimal? currentPrice;
public decimal? CurrentPrice
{
    get { return currentPrice; }
    set
    {
        currentPrice = value;
        RaisePropertyChanged("CurrentPrice");
    }
}

private string low;
public string Low
{
    get { return low; }
    set
    {
        low = value;
        RaisePropertyChanged("Low");
    }

}
```

```
private string change;
public string Change
{
    get { return change; }
    set
    {
        change = value;
        RaisePropertyChanged("Change");
    }
}

private string changePercent;
public string ChangePercent
{
    get { return changePercent; }
    set
    {
        changePercent = value;
        RaisePropertyChanged("ChangePercent");
    }
}

private string stockExchange;
public string StockExchange
{
    get { return stockExchange; }
    set
    {
        stockExchange = value;
        RaisePropertyChanged("StockExchange");
    }
}

private string previousClose;
public string PreviousClose
{
    get { return previousClose; }
    set
    {
        previousClose = value;
        RaisePropertyChanged("PreviousClose");
    }
}

private string symbolSearched;
public string SymbolSearched
{
    get { return symbolSearched; }
    set
    {
        symbolSearched = value;
        RaisePropertyChanged("SymbolSearched");
    }
}

private string searchText = "AAPL";
public string SearchText
{
    get { return searchText; }
    set
    {
        searchText = value;
        RaisePropertyChanged("SearchText");
    }
}

private string searchTextOrder = "AAPL";
public string SearchTextOrder
{
    get { return searchTextOrder; }
    set
    {
        searchTextOrder = value;
        RaisePropertyChanged("SearchTextOrder");
    }
}

private string pERatio;
public string PERatio
{
    get
    {
        return pERatio;
    }

    set
    {
        pERatio = value;
        RaisePropertyChanged("PERatio");

    }
}

private string ePS;

public string EPS
{
    get { return ePS; }
    set
    {
        ePS = value;
        RaisePropertyChanged("EPS");
    }
}

private bool flag;
public bool Flag
{
    get { return flag; }
    set
    {
        flag = value;
        RaisePropertyChanged("Flag");
    }
}

private ObservableCollection<GridFields> listToDisplay;
public ObservableCollection<GridFields> ListToDisplay
{
    get { return listToDisplay; }
    set { listToDisplay = value; }
}

private ImageBrush brushGraph;
public ImageBrush BrushGraph
{
    get { return brushGraph; }
    set { brushGraph = value; }
}

private ObservableCollection<ChartDetails> listForCharts;
public ObservableCollection<ChartDetails> ListForCharts
{
    get { return listForCharts; }
    set { listForCharts = value; }
}

private bool visibilityOfChart;
public bool VisibilityOfChart
{
    get { return visibilityOfChart; }
    set
    {
        visibilityOfChart = value;
        RaisePropertyChanged("VisibilityOfChart");
    }
}

private Visibility visibilityOfLabelPanel;
public Visibility VisibilityOfLabelPanel
{
    get { return visibilityOfLabelPanel; }
    set
    {
        visibilityOfLabelPanel = value;
        RaisePropertyChanged("VisibilityOfLabelPanel");
    }
}

private bool enableChartButton;
public bool EnableChartButton
{
    get { return enableChartButton; }
    set
    {
        enableChartButton = value;
        RaisePropertyChanged("EnableChartButton");
    }
}

//Dummy data

//Commands

private ICommand allCommand;
public ICommand AllCommand
{
    get
    {
        if (allCommand == null)
            allCommand = new RelayCommand(p => All());

        return allCommand;
    }
}

private ICommand newCommand;
public ICommand NewCommand
{
    get
    {
        if (newCommand == null)
            newCommand = new RelayCommand(p => New());

        return newCommand;
    }
}

private ICommand openCommand;
public ICommand OpenCommand
{
    get
    {
        if (openCommand == null)
            openCommand = new RelayCommand(p => Open());

        return openCommand;
    }
}

private ICommand sendCommand;
public ICommand SendCommand
{
    get
    {
        if (sendCommand == null)
            sendCommand = new RelayCommand(p => SendOrder());

        return sendCommand;
    }
}

private ICommand createCommand;
public ICommand CreateCommand
{
    get
    {
        if (createCommand == null)
            createCommand = new RelayCommand(p => CreateOrder());

        return createCommand;
    }
}

private ICommand updateCommand;
public ICommand UpdateCommand
{
    get
    {
        if (updateCommand == null)
            updateCommand = new RelayCommand(p => UpdateOrder());

        return updateCommand;
    }
}

private ICommand deleteCommand;
public ICommand DeleteCommand
{
    get
    {
        if (deleteCommand == null)
            deleteCommand = new RelayCommand(p => DeleteOrder());
        return deleteCommand;
    }
}

private ICommand oneDayCommand;
public ICommand OneDayCommand
{
    get
    {
        if (oneDayCommand == null)
            oneDayCommand = new RelayCommand(p => OneDayGraph());
        return oneDayCommand;
    }

}

private ICommand fiveDayCommand;
public ICommand FiveDayCommand
{
    get
    {
        if (fiveDayCommand == null)
            fiveDayCommand = new RelayCommand(p => FiveDayGraph());
        return fiveDayCommand;
    }

}

private ICommand threeMonthCommand;
public ICommand ThreeMonthCommand
{
    get
    {
        if (threeMonthCommand == null)
            threeMonthCommand = new RelayCommand(p => ThreeMonthGraph());
        return threeMonthCommand;
    }

}

private ICommand sixMonthCommand;
public ICommand SixMonthCommand
{
    get
    {
        if (sixMonthCommand == null)
            sixMonthCommand = new RelayCommand(p => SixMonthGraph());
        return sixMonthCommand;
    }

}
```

```
private ICommand oneYearCommand;
public ICommand OneYearCommand
{
    get
    {
        if (oneYearCommand == null)
            oneYearCommand = new RelayCommand(p => OneYearGraph());
        return oneYearCommand;
    }

}

private ICommand searchCommand;
public ICommand SearchCommand
{
    get
    {
        if (searchCommand == null)
            searchCommand = new RelayCommand(p => Search());

        return searchCommand;
    }
}

private ICommand searchCommandOrder;
public ICommand SearchCommandOrder
{
    get
    {
        if (searchCommandOrder == null)
            searchCommandOrder = new RelayCommand(p => SearchOrder());

        return searchCommandOrder;
    }
}

private void Search()
{
    listForCharts.Clear();

    if (SearchText != null && SearchText != "")
    {
        SymbolSearched = SearchText;

        EnableChartButton = true;

        Image image = new Image();
        if (SearchText != null)
        {
            //confirm if it's a symbol or not
```

```
                image.Source = new BitmapImage(new Uri("http://chart.finance.yahoo.com/c/3m/" +
SearchText));
                brushGraph.ImageSource = image.Source;
            }
            EPS = Quotes.First(q => q.Symbol == SearchText).EarningsShare.ToString();
            PERatio = Quotes.First(q => q.Symbol == SearchText).PeRatio.ToString();
            High = Quotes.First(q => q.Symbol == SearchText).DailyHigh.ToString();
            Low = Quotes.First(q => q.Symbol == SearchText).DailyLow.ToString();
            StockExchange = Quotes.First(q => q.Symbol == SearchText).StockExchange.ToString();
            PreviousClose = Quotes.First(q => q.Symbol == SearchText).PreviousClose.ToString();
            Change = Quotes.First(q => q.Symbol == SearchText).Change.ToString();
            ChangePercent = Quotes.First(q => q.Symbol == SearchText).ChangeInPercent.ToString();

            //PieChartData data = new PieChartData();
            //data.Name= "Volume";
            //data.Value = Quotes.First(q => q.Symbol == SearchText).Volume;

            List<PieChartData> pieChart = new List<PieChartData>()
           {
             new PieChartData(){ Name= "Volume", Value=Quotes.First(q => q.Symbol ==
SearchText).Volume},
             new PieChartData(){ Name= "AverageDailyVolume", Value=Quotes.First(q => q.Symbol ==
SearchText).AverageDailyVolume}
           };

            ListForPieChart.Clear();

            foreach (var item in pieChart)
            {
                ListForPieChart.Add(item);
            }

            CurrentPrice = Quotes.First(q => q.Symbol == SearchText).PreviousClose + Quotes.First(q =>
q.Symbol == SearchText).Change;

            ListForCharts.Add(new ChartDetails() { NameOfStock = "YearlyHigh", NetPnL = Quotes.First(q
=> q.Symbol == SearchText).YearlyHigh });
            ListForCharts.Add(new ChartDetails() { NameOfStock = "YearlyLow", NetPnL = Quotes.First(q
=> q.Symbol == SearchText).YearlyLow });
            ListForCharts.Add(new ChartDetails() { NameOfStock = "ChangeFromYearlyHigh", NetPnL =
Quotes.First(q => q.Symbol == SearchText).ChangeFromYearHigh });
            ListForCharts.Add(new ChartDetails() { NameOfStock = "ChangeFromYearlyLow", NetPnL =
Quotes.First(q => q.Symbol == SearchText).ChangeFromYearLow });
          }

        }

        private void SearchOrder()
        {
            state = RadioState.Search;
```

```
    listToDisplay.Clear();
    List<Order> list = dalObject.GetAllOrders();

    foreach (var item in list)
    {
        Stock stock = dalObject.GetStockById((int)item.StockID);

        string stockName = stock.StockName;
        string stockSymbol = stock.StockSymbol;
        decimal? marketPrice = stock.MarketPrice;

        string statusName = dalObject.GetStatusByStatusId((int)item.StatusID);

        if (stockSymbol == SearchTextOrder)
        {
            listToDisplay.Add(new GridFields()
            {
                OrderID = item.OrderID,
                Name = stockName,
                Symbol = stockSymbol,
                MarketPrice = marketPrice,
                Status = statusName
            });
        }
    }

}

private void OneYearGraph()
{
    Image image = new Image();
    image.Source = new BitmapImage(new Uri("http://chart.finance.yahoo.com/c/1y/" + SearchText));
    brushGraph.ImageSource = image.Source;
}

private void SixMonthGraph()
{
    Image image = new Image();
    image.Source = new BitmapImage(new Uri("http://chart.finance.yahoo.com/c/6m/" + SearchText));
    brushGraph.ImageSource = image.Source;
}

private void ThreeMonthGraph()
{
    Image image = new Image();
    image.Source = new BitmapImage(new Uri("http://ichart.finance.yahoo.com/c/3m/" + SearchText));
    brushGraph.ImageSource = image.Source;
}

private void FiveDayGraph()
{
```

```
            Image image = new Image();
            image.Source = new BitmapImage(new Uri("http://ichart.finance.yahoo.com/w?s=" + SearchText));
            brushGraph.ImageSource = image.Source;
        }

        private void OneDayGraph()
        {
            Image image = new Image();
            image.Source = new BitmapImage(new Uri("http://ichart.yahoo.com/t?s=" + SearchText));
            brushGraph.ImageSource = image.Source;
        }

        public MainWindowViewModel()
        {
            System.Media.SoundPlayer player = new
System.Media.SoundPlayer("G:\\Resume\\DOCUMENTS\\DOCUMENTS\\REASON\\RAHANUMAamp.
wav");

            string connectionString =
"Server=storefileinfo.netne.net;Database=a3026020_JCdata;Uid:a3026020_sql1091;web1091;";


            // player.Load();
            //player.Play();
            //System.Media.SystemSounds.Beep.Play();

            ListForPieChart = new ObservableCollection<PieChartData>();
            listToDisplay = new ObservableCollection<GridFields>();
            Quotes = new ObservableCollection<Quote>();
            dialogService = new ModelDialogService();

            //Some example tickers
            Quotes.Add(new Quote("AAPL"));
            Quotes.Add(new Quote("MSFT"));
            Quotes.Add(new Quote("INTC"));
            Quotes.Add(new Quote("IBM"));
            Quotes.Add(new Quote("RVBD"));
            Quotes.Add(new Quote("AMZN"));
            Quotes.Add(new Quote("BIDU"));
            Quotes.Add(new Quote("SINA"));
            Quotes.Add(new Quote("THI"));
            Quotes.Add(new Quote("NVDA"));
            Quotes.Add(new Quote("AMD"));
            Quotes.Add(new Quote("DELL"));
            Quotes.Add(new Quote("WMT"));
            Quotes.Add(new Quote("GLD"));
            Quotes.Add(new Quote("SLV"));
            Quotes.Add(new Quote("V"));
            Quotes.Add(new Quote("ITC"));
            Quotes.Add(new Quote("MCD"));
            //getting the data from yahoo finance..!
            YahooStockEngine.Fetch(Quotes);

            //poll every 'x' seconds
            timer.Interval = new TimeSpan(0, 0, 60);
            timer.Tick += (o, e) => YahooStockEngine.Fetch(Quotes);
            timer.Tick += new EventHandler(timer_Tick);

            timer.Start();

            brushGraph = new ImageBrush();
            allSecurities = new ObservableCollection<AllData>();
            listForCharts = new ObservableCollection<ChartDetails>();


            dalObject = new PortfolioDAL();
            r = new Random();

            symbols = new List<string>();
            symbols = dalObject.GetAllSymbols();


            foreach (var item in Quotes)
            {
                allSecurities.Add(new AllData() {
                Name=item.Name,
                Symbol = item.Symbol,
                PreviousClose = item.PreviousClose,
                MarketPrice = item.PreviousClose+item.Change,
                Volume = item.Volume,
                Change = item.Change,
                ChangePercent = item.ChangePercent,
                });
            }

        }

        void timer_Tick(object sender, EventArgs e)
        {
            listToDisplay.Clear();
            allSecurities.Clear();
            //Stock s = new Stock();
            //s.StockID = 1;
            //s.StockSymbol = "AAPL";
            //s.StockName = "Apple";
            //s.MarketPrice = Quotes.First(q => q.Symbol == s.StockSymbol).Change + Quotes.First(q =>
q.Symbol == s.StockSymbol).PreviousClose+r.Next(-3,3);
            //dalObject.UpdateStock(s);

            List<Stock> stocks = dalObject.GetAllStocks();
            foreach (var item in stocks)
```

```
{
    Stock stock = dalObject.GetStockById((int)item.StockID);
   // Quote q = Quotes.First(quote => quote.Symbol == stock.StockSymbol);
    //stock.MarketPrice = q.Change + q.PreviousClose;

    //Add other thing if required..
    dalObject.UpdateStock(stock);
}


List<Order> list = dalObject.GetAllOrders();


foreach (var item in list)
{
    Stock stock = dalObject.GetStockById((int)item.StockID);

    string stockName = stock.StockName;
    string stockSymbol = stock.StockSymbol;
    decimal? marketPrice = stock.MarketPrice;

    string statusName = dalObject.GetStatusByStatusId((int)item.StatusID);

    switch (state)
    {
        case RadioState.New:
            if (dalObject.GetStatusByStatusId((int)item.StatusID) == "New")
            {
                listToDisplay.Add(new GridFields()
                {
                    OrderID = item.OrderID,
                    Name = stockName,
                    Symbol = stockSymbol,
                    MarketPrice = marketPrice,
                    Status = statusName
                });
            }
            break;

        case RadioState.Open:
            if (dalObject.GetStatusByStatusId((int)item.StatusID) == "Open")
            {
                listToDisplay.Add(new GridFields()
                {
                    OrderID = item.OrderID,
                    Name = stockName,
                    Symbol = stockSymbol,
                    MarketPrice = marketPrice,
                    Status = statusName
                });
            }
            break;

        case RadioState.All:

            listToDisplay.Add(new GridFields()
            {
                OrderID = item.OrderID,
                Name = stockName,
                Symbol = stockSymbol,
                MarketPrice = marketPrice,
                Status = statusName
            });

            break;

        case RadioState.Search:

            if (stockSymbol == SearchTextOrder)
            {
                listToDisplay.Add(new GridFields()
                {
                    OrderID = item.OrderID,
                    Name = stockName,
                    Symbol = stockSymbol,
                    MarketPrice = marketPrice,
                    Status = statusName
                });
            }

            break;

        default:
            break;

    }
}

allSecurities.Clear();
foreach (var item in Quotes)
{
    allSecurities.Add(new AllData()
    {
        Name = item.Name,
        Symbol = item.Symbol,
        MarketPrice = item.PreviousClose + item.Change,
        Volume = item.Volume,
        Change = item.Change,
        ChangePercent = item.ChangePercent

    });
}
```

```
    Search();

}

private void CreateOrder()
{
    dialogService.ShowDialog<CreateOrderViewModel>(ViewType.CreateOrderView, null, null);
    Refresh();
}

private void UpdateOrder()
{
    Order order = dalObject.GetOrderByOrderId(SelectedItem.OrderID);
    UpdateOrderViewModel vModel = new UpdateOrderViewModel();

    vModel.OrderId = order.OrderID.ToString();

    if (order.Side == "BUY")
        vModel.Buy = true;
    else
        vModel.Sell = true;
    vModel.OwnedQuantity = order.OwnedQuantity.ToString();
    vModel.Status = dalObject.GetStatusByStatusId((int)order.StatusID);

    vModel.SelectedType = order.Type;
    vModel.Quantity = order.Quantity.ToString();
    vModel.Notes = order.Notes;

    if (vModel.SelectedType == "StopLimit")
    {
        vModel.LimitEnabled = true;
        vModel.StopEnabled = true;
    }
    else if (vModel.SelectedType == "Limit")
    {
        vModel.LimitEnabled = true;
    }
    else if (vModel.SelectedType == "Stop")
    {
        vModel.StopEnabled = true;
    }

    if (order.Qualifier == "GTC")
        vModel.GTC = true;
    else
        vModel.GTD = true;

    dialogService.ShowDialog<UpdateOrderViewModel>(ViewType.UpdateOrderView, vModel, null);
    Refresh();
```

```
}

private void DeleteOrder()
{
   foreach (var item in listToDisplay)
   {
      if (item.IsSelected == true)
         dalObject.DeleteOrder(dalObject.GetOrderByOrderId(item.OrderID));
   }

   Refresh();
}

private void SendOrder()
{
   int count = 0;

   foreach (var item in listToDisplay)
   {
      if (item.IsSelected == true && item.Status == "New")
      {
      count++;
      Order order = dalObject.GetOrderByOrderId(item.OrderID);
      order.StatusID = 2;
      dalObject.UpdateOrder(order);
      }
   }
   MessageBox.Show(count.ToString());
   Refresh();
}

private void New()
{
   state = RadioState.New;

   listToDisplay.Clear();
   List<Order> list = dalObject.GetAllOrders();

   foreach (var item in list)
   {
      Stock stock = dalObject.GetStockById((int)item.StockID);

      string stockName = stock.StockName;
      string stockSymbol = stock.StockSymbol;
      decimal? marketPrice = stock.MarketPrice;

      string statusName = dalObject.GetStatusByStatusId((int)item.StatusID);

      if (dalObject.GetStatusByStatusId((int)item.StatusID) == "New")
      {
         listToDisplay.Add(new GridFields()
            {
                OrderID = item.OrderID,
                Name = stockName,
                Symbol = stockSymbol,
                MarketPrice = marketPrice,
                Status = statusName
            });
        }
    }
}

private void Open()
{
    listToDisplay.Clear();
    state = RadioState.Open;

    List<Order> list = dalObject.GetAllOrders();

    foreach (var item in list)
    {
        Stock stock = dalObject.GetStockById((int)item.StockID);

        string stockName = stock.StockName;
        string stockSymbol = stock.StockSymbol;
        decimal? marketPrice = stock.MarketPrice;

        string statusName = dalObject.GetStatusByStatusId((int)item.StatusID);

        if (dalObject.GetStatusByStatusId((int)item.StatusID) == "Open")
        {
            listToDisplay.Add(new GridFields()
            {
                OrderID = item.OrderID,
                Name = stockName,
                Symbol = stockSymbol,
                MarketPrice = marketPrice,
                Status = statusName
            });
        }
    }

}

private void Refresh()
{

    listToDisplay.Clear();
    List<Order> list = dalObject.GetAllOrders();


    foreach (var item in list)
```

```
{
   Stock stock = dalObject.GetStockById((int)item.StockID);

   string stockName = stock.StockName;
   string stockSymbol = stock.StockSymbol;
   decimal? marketPrice = stock.MarketPrice;

   string statusName = dalObject.GetStatusByStatusId((int)item.StatusID);

   switch (state)
   {
      case RadioState.New:
         if (dalObject.GetStatusByStatusId((int)item.StatusID) == "New")
         {
            listToDisplay.Add(new GridFields()
            {
               OrderID = item.OrderID,
               Name = stockName,
               Symbol = stockSymbol,
               MarketPrice = marketPrice,
               Status = statusName
            });
         }
         break;

      case RadioState.Open:
         if (dalObject.GetStatusByStatusId((int)item.StatusID) == "Open")
         {
            listToDisplay.Add(new GridFields()
            {
               OrderID = item.OrderID,
               Name = stockName,
               Symbol = stockSymbol,
               MarketPrice = marketPrice,
               Status = statusName
            });
         }
         break;

      case RadioState.All:

         listToDisplay.Add(new GridFields()
         {
            OrderID = item.OrderID,
            Name = stockName,
            Symbol = stockSymbol,
            MarketPrice = marketPrice,
            Status = statusName
         });

         break;
```

```
            case RadioState.Search:

                if (stockSymbol == SearchTextOrder)
                {
                    listToDisplay.Add(new GridFields()
                    {
                        OrderID = item.OrderID,
                        Name = stockName,
                        Symbol = stockSymbol,
                        MarketPrice = marketPrice,
                        Status = statusName
                    });
                }

                break;

            default:
                break;

        }
    }
}

private void All()
{
    listToDisplay.Clear();
    state = RadioState.All;

    List<Order> list = dalObject.GetAllOrders();

    foreach (var item in list)
    {
        Stock stock = dalObject.GetStockById((int)item.StockID);

        string stockName = stock.StockName;
        string stockSymbol = stock.StockSymbol;
        decimal? marketPrice = stock.MarketPrice;

        string statusName = dalObject.GetStatusByStatusId((int)item.StatusID);

        listToDisplay.Add(new GridFields()
            {
                OrderID = item.OrderID,
                Name = stockName,
                Symbol = stockSymbol,
                MarketPrice = marketPrice,
                Status = statusName
            });

    }
```

```
        }
    }
}
```

- Relay Command

```
namespace EquityTradingApplication.Commands
{
    class RelayCommand:ICommand
    {

        readonly Predicate<object> _canExecute;
        readonly Action<object> _execute;

        public RelayCommand(Action<object> execute) : this(null, execute) { }

        public RelayCommand(Predicate<object> canExecute, Action<object> execute)
        {
            if (execute == null)
                throw new ArgumentNullException("execute");

            _canExecute = canExecute;
            _execute = execute;
        }

        public bool CanExecute(object parameter)
        {
            return _canExecute == null ? true : _canExecute(parameter);
        }

        public event EventHandler CanExecuteChanged
        {
            add { CommandManager.RequerySuggested += value; }
            remove { CommandManager.RequerySuggested -= value; }
        }

        public void Execute(object parameter)
        {
            _execute(parameter);
        }
    }
}
```

8. REFERENCES

1. STACKOVERFLOW: http://stackoverflow.com/
2. MVNREPOSITORY: www.mvnrepository.com
3. Mark IT: https://www.markitacademy.com/
4. Money Bhai: http://moneybhai.moneycontrol.com/
5. DUN & BRAD STREET: http://www.dnb.co.in/
6. http://en.wikipedia.org/wiki/Windows_Presentation_Foundation
7. http://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx
8. http://en.wikipedia.org/wiki/Model_View_ViewModel
9. http://en.wikipedia.org/wiki/Entity_Framework
10. http://msdn.microsoft.com/en-us/library/vstudio/dd456853%28v=vs.100%29.aspx