San Jose State University
College of Engineering
Computer Engineering Department
CMPE 202 – Software Systems Engineering

Security Trading System

Project Report

Team 5

Jing Zhang
Sudershan Malpani
Vishal Sharma

December 16, 2011

Table of Contents

1. Vision and Goals

The **Security Trading System** deals with the securities like mutual funds and stocks. This system brings the Companies and Investors together, to make the trading process functioning. The investors send the requests to buy/sell the securities, offered by the companies, and those requests got processed by the Exchange market system

The **Investor** sends the request to buy/sell the mutual funds and stocks; the request is processed by checking the account whether it is eligible for purchasing the order placed by the investor or not than only the order is placed in the exchange system. After the order is processed the result whether the order is fulfilled or not updates the investor's account and notifies him. One investor can have several accounts. The account consists of the Cash, stocks and mutual funds from different companies the investor is holding. A company buying the mutual funds of other companies will also be the investor that time.

The **company** provides IPO's if it is a new company, it contains listings of all the companies and if 2 or more companies are merging or splitting than the securities of those companies got updated.

In the **exchange system**, the process of placing the order by the investor in the respective queue of the stock takes place. The order given by the customer is matched with the sell/buy orders from different clients and gives the best match to the funds. If the order given by the investor is of mutual funds than the system investor will directly buy the mutual funds from the system. The duration in which the investor's order is valid is customizable and the order which the buyer is buying or the seller is selling could be fully or partially bought/sold in the exchange system.

2. Business Modeling

2.1 Process Summary Diagram

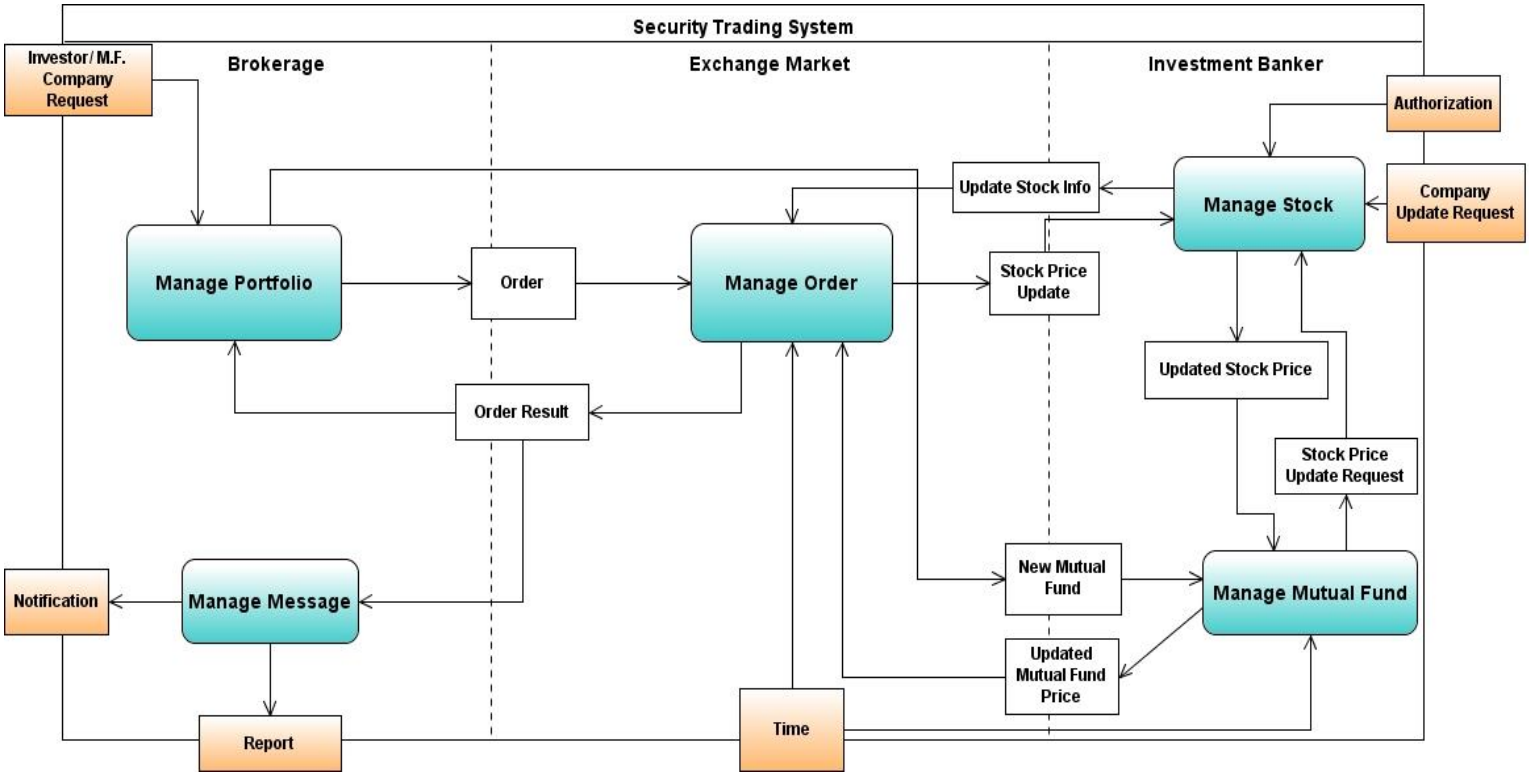


Figure 2.1

2.2 Process Decomposed Diagrams

2.2.1 Manage Portfolio

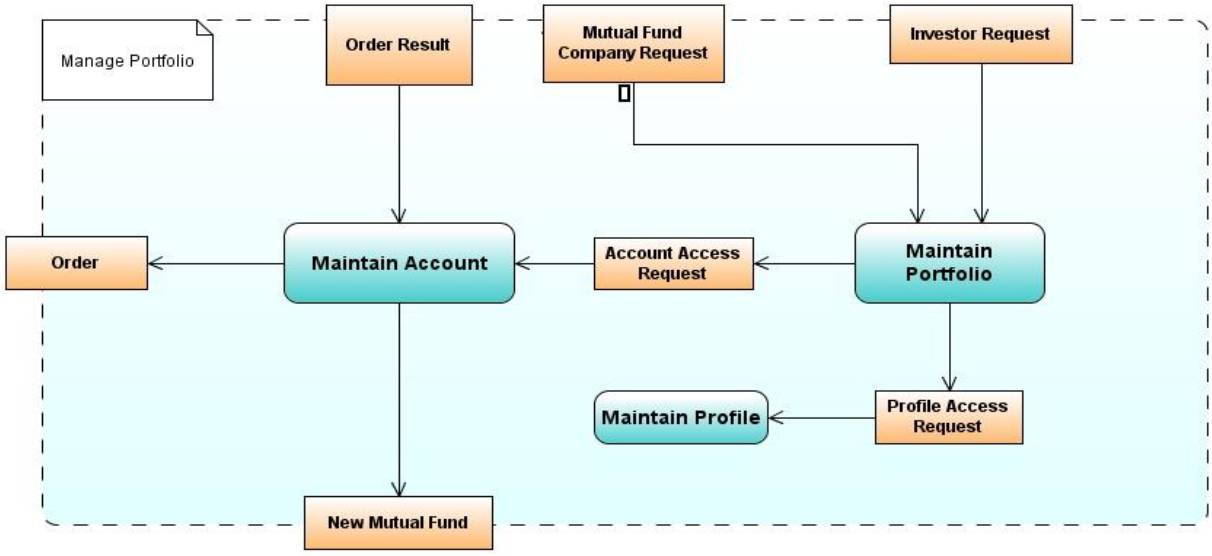


Figure 2.2 1

2.2.2 Manage Order

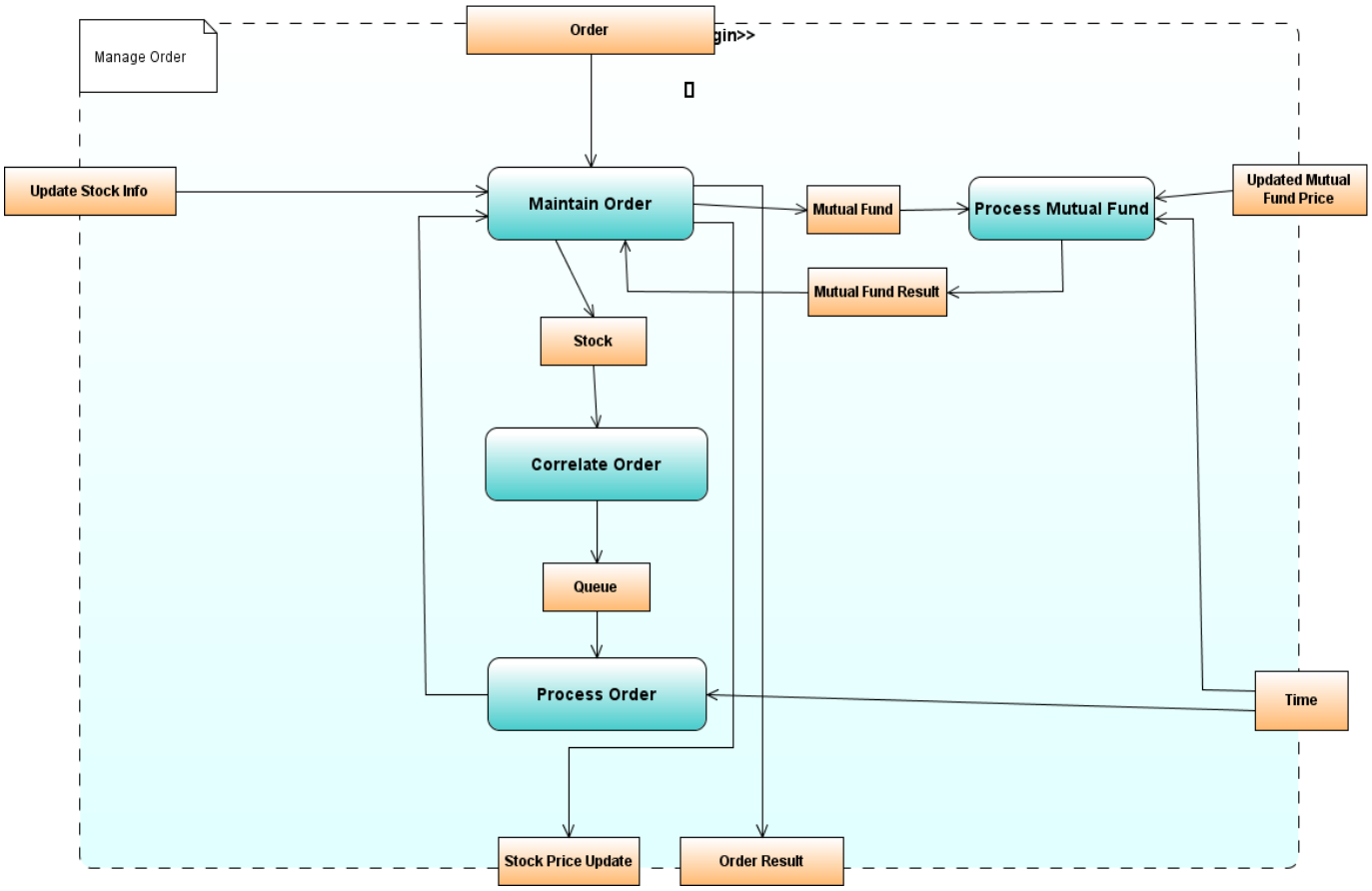


Figure 2.2 2

2.2.3 Manage Stock

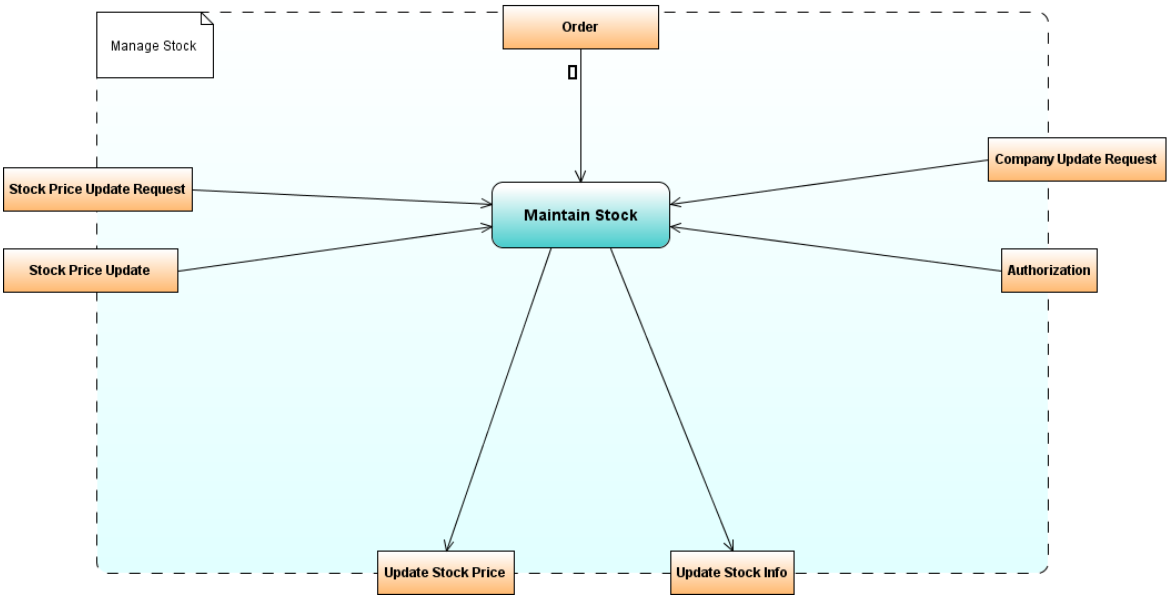


Figure 2.2 3

2.2.4 Manage Mutual Fund

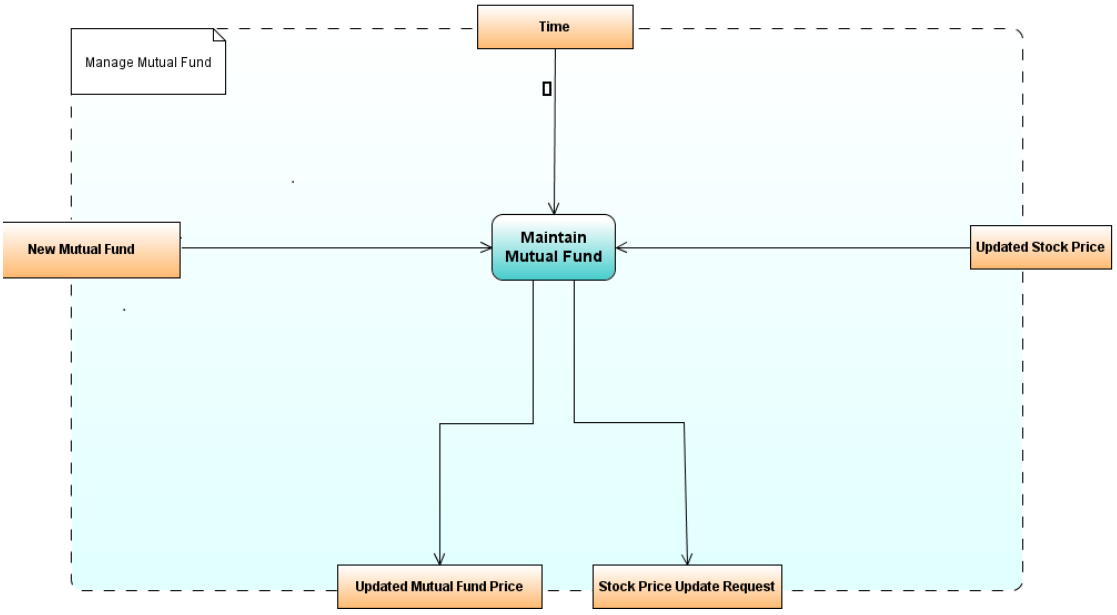


Figure 2.2 4

2.2.5 Manage Message

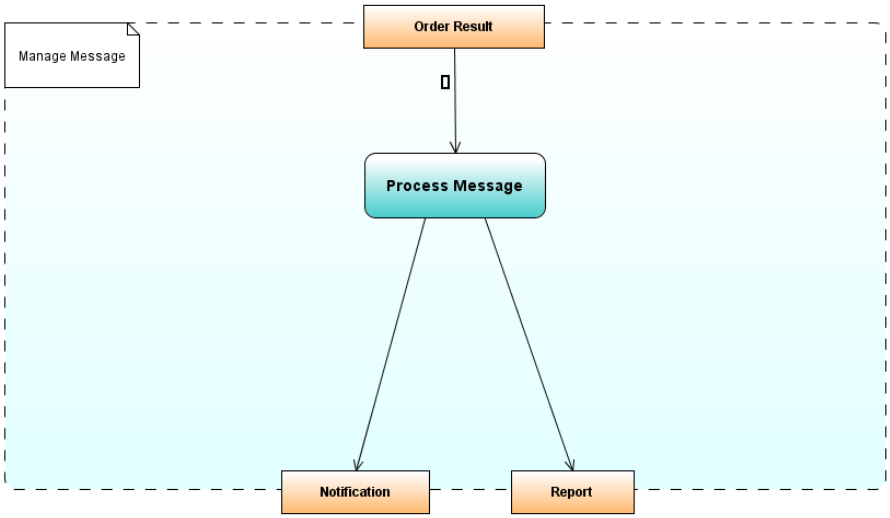


Figure 2.2 5

2.3 Domain Class Model

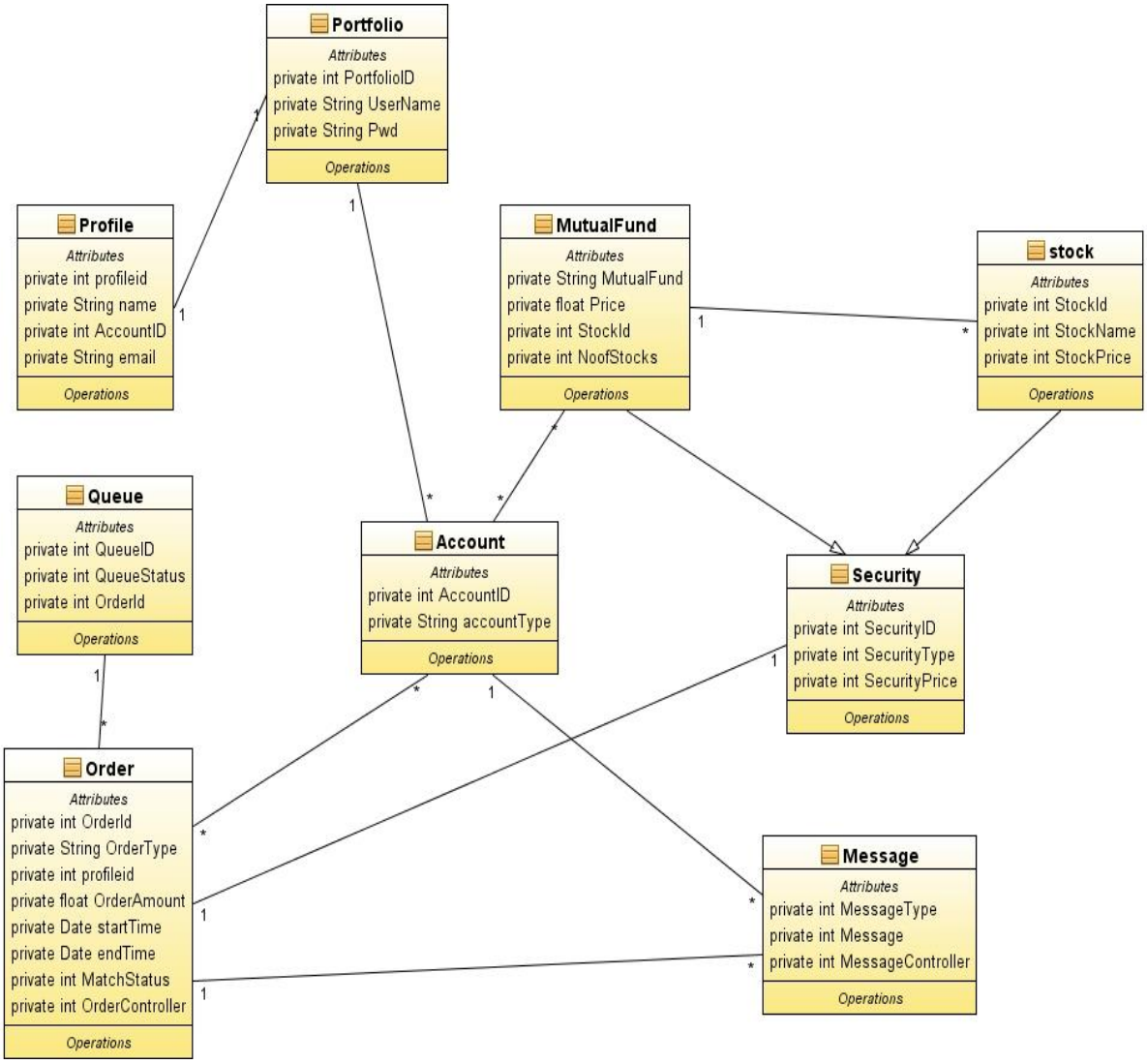


Figure 2.3

2.4 Domain State Model

2.4.1 Order State diagram

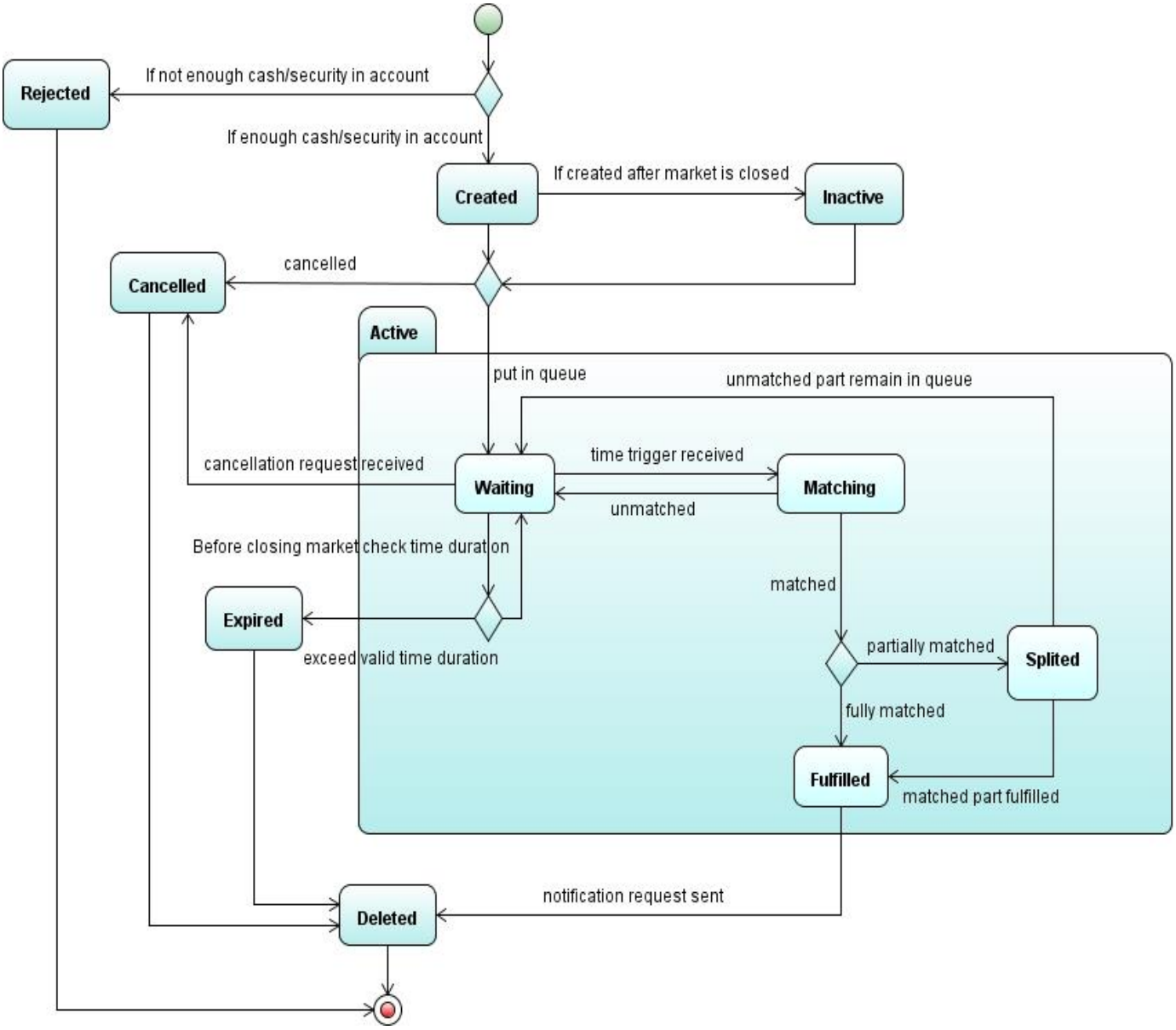


Figure 2.4.1

2.4.2 Order Queue State Diagram

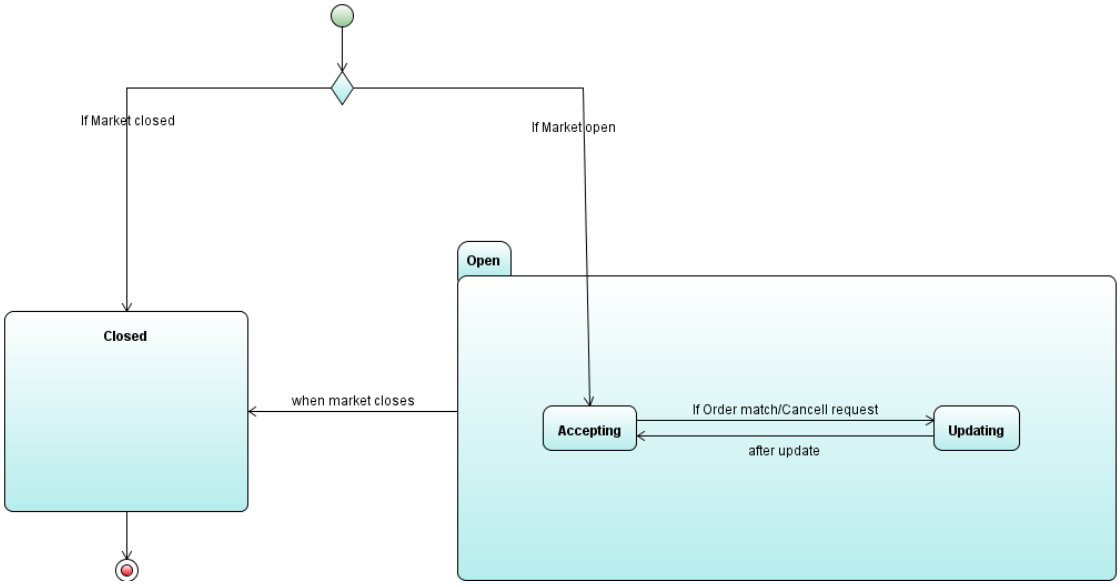


Figure 2.4.2

3. Requirement Elicitation

3.1 Problem Statement and Target Environment

In today's world, Security Trading or we can say buying and selling of Stocks and Mutual Funds has become very much rigid among the common people. It is important for them to have a system through which they can buy/sell their stocks and Mutual Fund as per there wish. They also want be aware of the minute to minute updates about the buy/sell orders they have placed. Moreover, the firms that want to become public/launch an IPO need a smoother way in which this act can be accomplished. Security Trading System (STS), provides a solution for the above stated things.

STS is a web oriented system which will provide an opportunity to the people to buy/sell their stocks/mutual funds in the exchange market. The system is responsible for finding a buyer for the seller and a seller for the buyer, if exists. The seller fixes the price and the time limit in the order and can update the same till the system has not found a buyer for it or before its time limit expires. An investor can file multiple buy/sell orders to the STS. The investor is notified for all the updates regarding his order.

The firms that wish to become public/launch an IPO can get it done by providing listings of their securities and prices for the same. The firm can also be a Mutual fund firm, that holds multiple stocks of multiple firms and sell them as mutual funds. Firms also have an option to merge/split their stocks as per their need. If a firm which is already public wishes to become private can withdraw from this system.

It provides a faster and an efficient way to manage ones securities (stocks and mutual funds).Also keeps the investor aware of the minute to minute updates. It makes it easy for the new firms to become public. We can say, it is a kind of one stop solution for both investors and the firms.

3.2 Functional Requirements

Req. ID	Description
R1.1 Basic Service	
R1.1.1	**System shall support multiple user type: investor, company.**
R1.1.2	**System shall support multiple types of actions: buy and sell security.**
R1.1.3	**System shall support listing security.**
R1.1.4	**System shall allow company to establish IPO.**
R1.1.5	**System shall allow user to create new portfolio including one profile and several accounts.**
R1.1.6	**System shall allow user to create order.**
R1.1.7	**System shall be able to process the order.**
R1.1.8	**System shall be able to create report/message as requested by user.**
R1.2 Exchange Support	
R1.2.1	**System shall be able to track all the exchanging process.**
R1.2.2	**System shall be able to generate message to user about order result.**
R1.3 Maintenance Support	
R1.3.1	**System shall be able to update order information**
R1.3.2	**System shall be able to retrieve order information**
R1.3.3	**System shall be able to delete order information**
R1.3.4	**System may allow user to update portfolio information**
R1.3.5	**System may allow user to retrieve portfolio information**
R1.3.6	**System may allow user to delete portfolio information**
R1.3.7	**System shall be able to update security information**
R1.3.8	**System shall be able to retrieve security information**
R1.3.9	**System shall be able to delete security information**

R1.4 User Support	
R1.4.1	**System shall be able to show user their order status**
R1.4.2	**System shall allow user to have the ability to update order after order is created.**
R1.4.3	**System shall be able to notify user for any update information.**
R1.5 Process Control	
R1.5.1	**System shall accept new order when user account has enough cash/security.**
R1.5.2	**System shall be able to notice the situation when order time stamp is invalid.**
R1.5.3	**System shall be able to split when order is partially matched.**

3.3 Non-Functional Requirements

Req. ID	Description
R2.1 Performance	
R2.1.1	**System shall be able to handle multiple user concurrently.**
R2.1.2	**System should have response time of three seconds or less**
R2.1.3	**System shall process a minimum of ten transactions per second.**
R2.2 Reliability	
R2.2.1	**System should be available for service when requested by end-users.**
R2.2.2	**System should have less failure rate than 1%.**
R2.3 Security	
R2.3.1	**System shall protect data from unauthorized access.**
R2.3.2	**System should ensure recovery of data in case of failure.**
R2.4 Usability	
R2.4.1	**System should have user-friendly interface.**
R2.4.2	**System should have well-constructed user manual and error handling.**
R2.4.3	**System should provide help facilities.**

R2.5 Compatibility	
R2.5.1	System shall be compatible with Java Platform.
R2.5.2	System shall be compatible with Windows XP/7 and Mac OS.

4. Requirement Analysis

4.1Use Case Diagrams

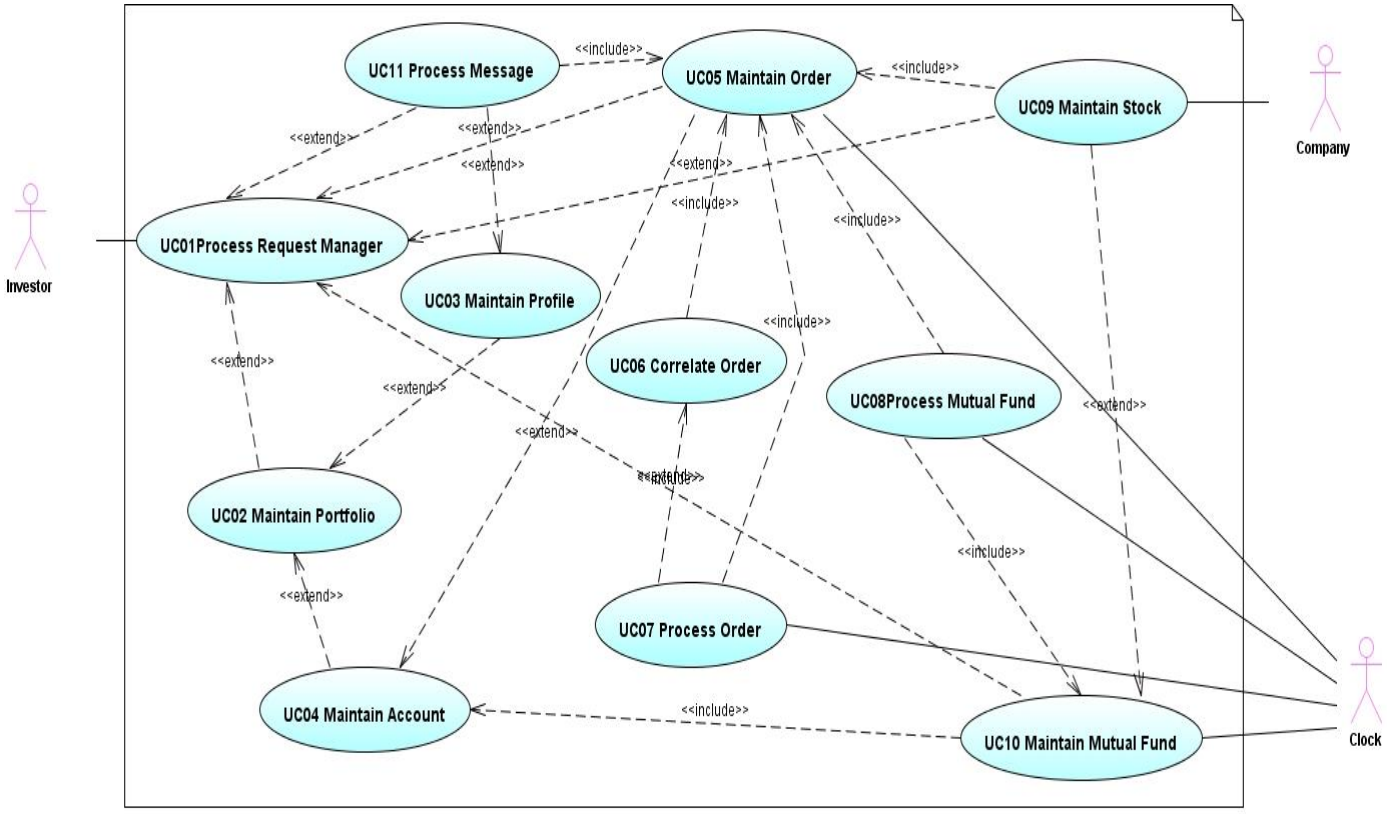


Figure 4 1

4.2Use case Descriptions

4.2.1 UC01 Process Request Manager

ID: UC01
Name: Maintain Portfolio
Type: Base
Actor: Investor
Description:
Send the controls to different use cases
Preconditions: None
Post conditions: None
Main Flow:

Investor	System
1. Send the desired functionality	
	2. Process the request type
	2.1 If type=Maintain portfolio <<extends>>UC02 Maintain Portfolio
	2.2 If type=Maintain Order <<extends>>UC02 Maintain Order
	2.3 If type=Maintain Message <<extends>>UC11 Maintain Message
	2.3 If type=Maintain Security <<extends>>UC08 Maintain Security

Alternative Flow: None
Special Requirements: None

4.2.2 UC02 Maintain Portfolio

ID: UC02
Name: Maintain Portfolio
Type: Reference
Actor: None
Description:
Maintain records for the Investor or Mutual Fund Company's profile
Preconditions: None
Post conditions: None
Main Flow:

System
1 Check request type.
1.1 If Request=create ,
1.1.1 Return Created Portfolio
1.2 If Request=Retrieve,
1.2.1 Return Retrieved Portfolio
1.3 If Request=Update ,
1.3.1Return Updated Portfolio

Alternative Flow: None
Special Requirements: None

4.2.3 UC03 Maintain Profile

ID: UC03
Name: Maintain Profile
Type: Reference
Actor:
None
Description:
Maintain records for the Investor's profile and also the orders that are placed by the investors. Also checks the validity of the orders.
Preconditions:
None
Post conditions: None
Main Flow:

System
1 Check request type.
1.1 If Request=create ,
1.1.1 Return Created Profile
1.2 If Request=Retrieve,
1.2.1 Return Retrieved Profile
1.3 If Request=Update ,
1.3.1Return Updated Profile

Alternative Flow: None
Special Requirements: None

4.2.4 UC04 Maintain Account

ID: UC04
Name: Maintain Account
Type:
Reference
Actors:
None
Description:
Maintains one or more accounts for the investors which include the cash accounts, mutual fund accounts and their stock accounts hold by them, It also checks whether the investor is eligible to buy the particular order placed by him.
Preconditions:
UC02 Maintain Profile
Post Conditions:
None
Main flow:

System

1. Check request type
1.1 If request is for Account
1.1.1 If request = Create, Create Account
1.1.2 If request = Retrieve, Retrieve Account
1.1.3 If request = Update, Update Account
<<Include>> UC05 Maintain Order
1.1.4 If request = Delete, Delete Account
1.2 If request is for Order
1.2.1 If request = Create, Create Order
1.2.2 If request = Retrieve, Retrieve Order
1.2.4 If request = Delete, Delete Order <<include>> UC05 Maintain Order

Alternative flow: None
Special Requirements: None

4.2.5 UC05 Maintain Order

ID: UC05
Name: Maintain Order
Type:
Reference, Base
Actor: Clock
Description:
Perform Maintenance of Order, Check Order's Time Stamp Validation When Received Order from Maintain Profile.
Preconditions:
UC04 Maintain Account
Post conditions:
None
Main Flow:

Clock	System
	1. Security Order Received
2. Send Time Trigger	
	2.1 Check Time Stamp
	2.1.1 If Time Stamp == Valid
	3. Check Order Status
	3.1 If Trigger==Create, Create Order
	3.1.1 Order Created
	3.1.2 <<Include>> UC06 Correlate Order
	3.2 If Trigger ==Retrieve, Retrieve Order
	3.2.1 Order Retrieved
	3.2.2 <<Include>> UC04 Maintain Account
	3.3 If Trigger ==Update, Update Order
	3.3.1 Order Updated
	3.3.2 <<Include>> UC06 Correlate Order
	3.4 If Trigger ==Delete, Delete Order
	3.4.1 Order Deleted

	3.4.2 <<Include>> UC04 Maintain Account
	3.4.3 <<Include>> UC06 Correlate Order

Alternative Flow: None
Special Requirements: None

4.2.6 UC06 Correlate Order

ID: UC06
Name: Correlate Order
Type: Reference
Actor: None
Description:
Correlate Orders to the specific queue.
Preconditions:
UC03 Maintain Order
Post conditions: None
Main Flow:

System
1.Trigger Received
2. Check Trigger Status
2.1 If Trigger ==Retrieve, Retrieve Order
2.1.1 Order Retrieved <<include>> UC05 Maintain Order
2.2 If Trigger ==Update, Update Queue
2.2.1 Queue Updated
2.3 If Trigger ==Delete, Delete Order
2.3.1 Order Deleted from the queue

Alternative Flow: None
Special Requirements: None

4.2.7 UC07 Process Order

ID: UC07
Name: Process Order
Type: Base
Actor:
Clock (Primary)
Description:
Perform matching of Stocks
Preconditions:
UC05 Maintain Order, UC06 Correlate Order
Post conditions: None
Main Flow:

Clock	**System**
1. Send Trigger to the System	
	2. Trigger received from clock
	3. Get the orders from buy and sell queues and perform matching

	3.1 If Order is matched, send match notification <<include>> UC05 Maintain Order
	3.2 If Order not match, continue in the loop to check next element in the queue

Alternative Flow: None
Special Requirements: None

4.2.8 UC08 Process Mutual Fund

ID: UC08
Name: Process Mutual Fund
Type: Base
Actor:
Clock (Primary)
Description:
If the investor requests to buy mutual funds
Preconditions:
UC04 Maintain Order, UC10 Maintain Mutual Fund
Post conditions:
None
Main Flow:

Clock	System
1. Trigger at market closing	
	2. Retrieve Mutual fund price from maintain mutual fund <<include>> UC10 Maintain Mutual Fund
	2.1 Process Mutual Fund
	2.2 Send result to Maintain Order <<include>> UC05 Maintain Order

4.2.9 UC09 Maintain Stock

ID: UC09
Name: Maintain Stock
Type: Base
Actor:
Company (Primary)
Commissioner (Offstage)
Description:
Company is adding its IPO to the exchange system and responsible for Splitting and Merging companies.
Preconditions:
None
Post conditions:
None
Main Flow:

Company	**System**
1. Provide request for merge, split or setting up new IPO to the	

Commissioner.	
	2. Authorization received from the Commissioner.
	3. Processes the request
	3.1 If request= new IPO, Manage Security for new company, Create Stocks and mutual funds update Maintain Order.
	3.2 If request= Split/Merge the company or Order Matched in Maintain Order, update the maintain Order or update from maintain order.

Alternative Flow: None
Special Requirements: None

4.2.10 UC10 Maintain Mutual Fund

ID: UC10
Name: Maintain Mutual Fund
Type: Reference + Base
Actor: Clock (Base Part-Primary)
Description: Perform Maintenance of Mutual Fund.
Preconditions: UC04 Maintain Account, UC09 Maintain Stock
Post conditions: None
Main Flow:

Clock	System
1. Send Trigger to the system	
	2. Check Request
	2.1 Create M.F. if Mutual fund company places request to create Mutual Fund
	2.1.1 M.F. Created
	2.2 Retrieve Stock quotes as time trigger at 4.00 come<<include>> Maintain Stock
	2.2.1 Stock Quotes Retrieved from Maintain Stocks

Alternative Flow: None
Special Requirements: None

4.2.11 UC11 Process Message

ID: UC11
Name: Process Message
Type:
Reference
Actor:
None
Description:
Send notifications and report to the user.
Preconditions:

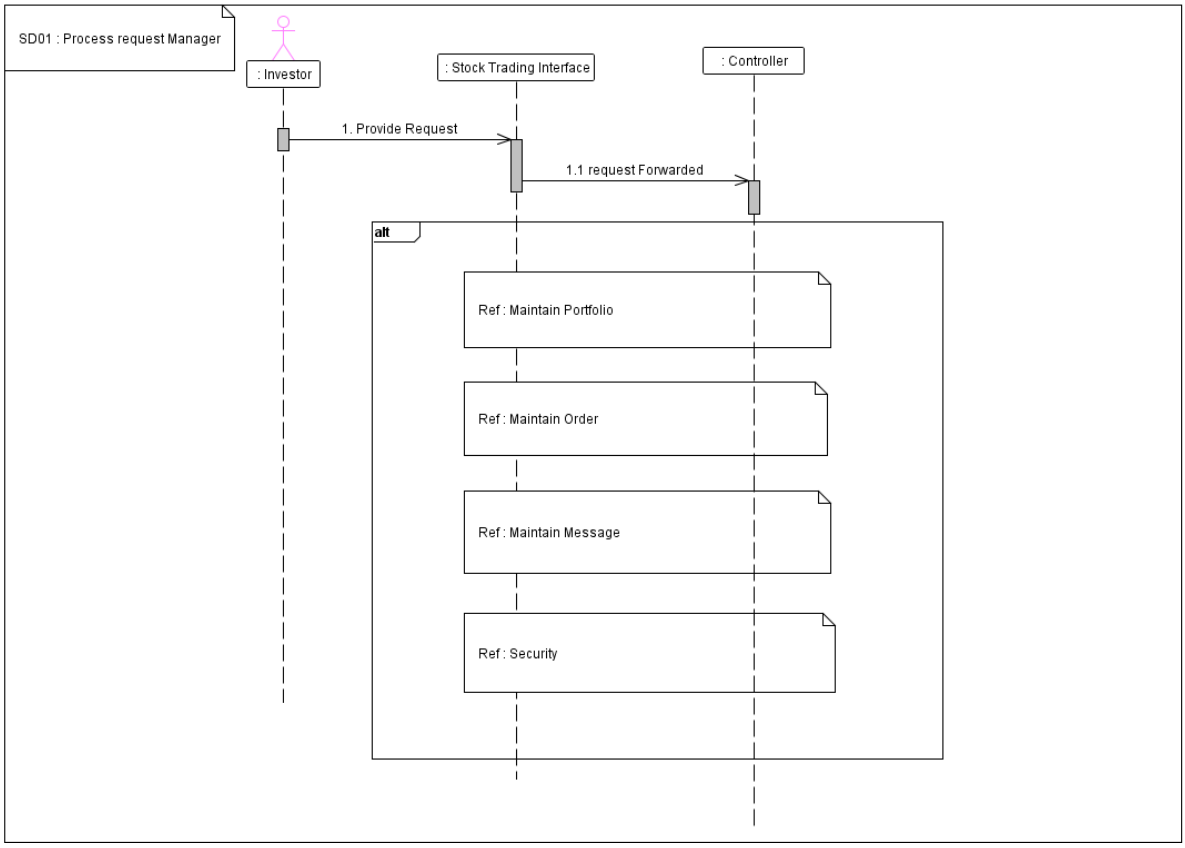
UC04 Maintain Order, UC03 Maintain Account
Post conditions:
None
Main Flow:

System
1. Request from System Received for message
2. Check request
2.1 If request is for generating reports or sending notifications than create message
2.1.1 Return report/notification created
2.2 If request is that there is any updations or changes in the system than retrieve the change information
2.2.1 Return notifications

Alternative Flow: None
Special Requirements: None

4.3Use Case Sequence Diagrams

4.3.1 SD01 Process Request Manager



4.3.2 SD02 Maintain Portfolio

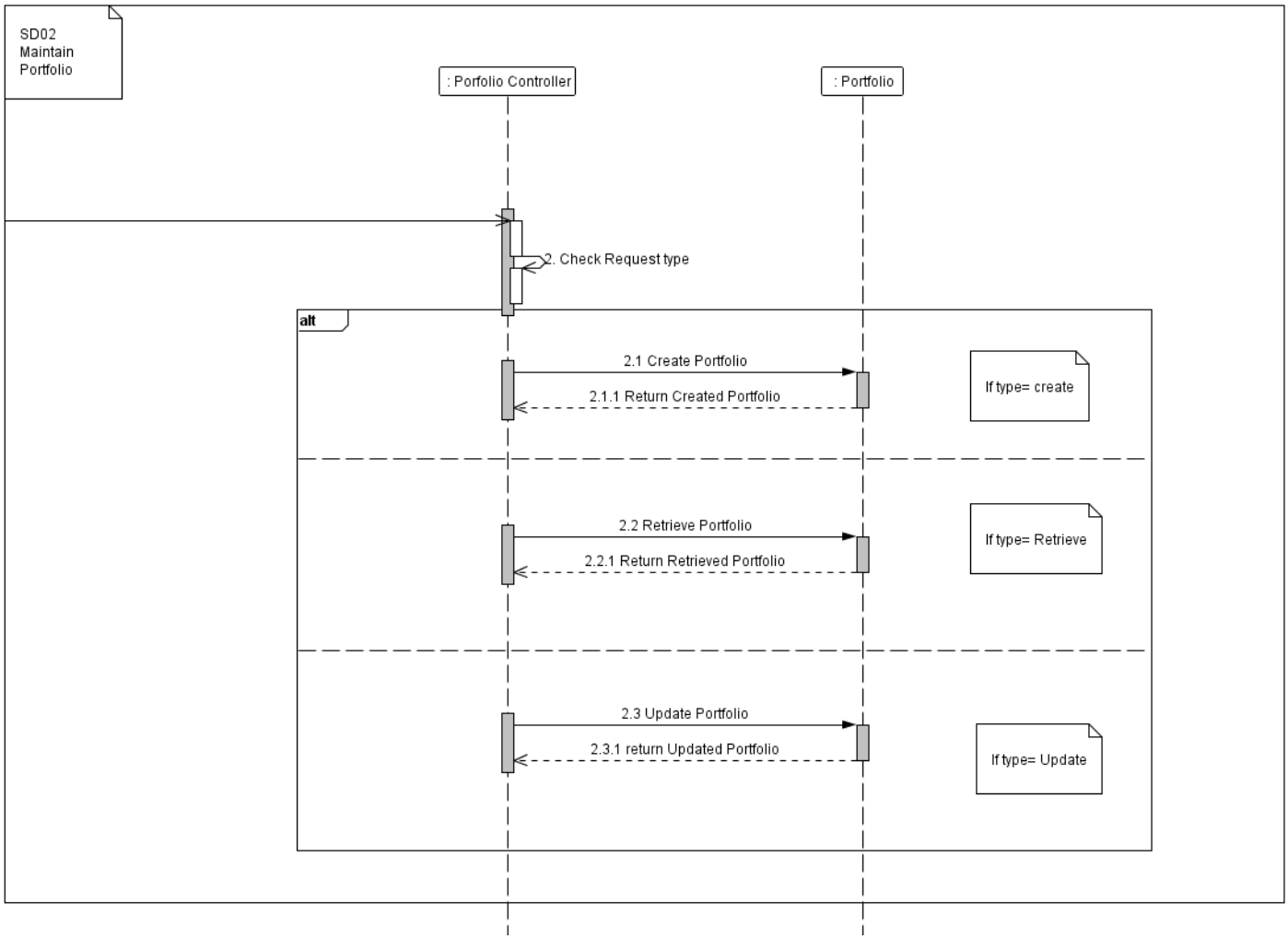


Figure 4.3.2

4.3.3 SD03 Maintain Profile

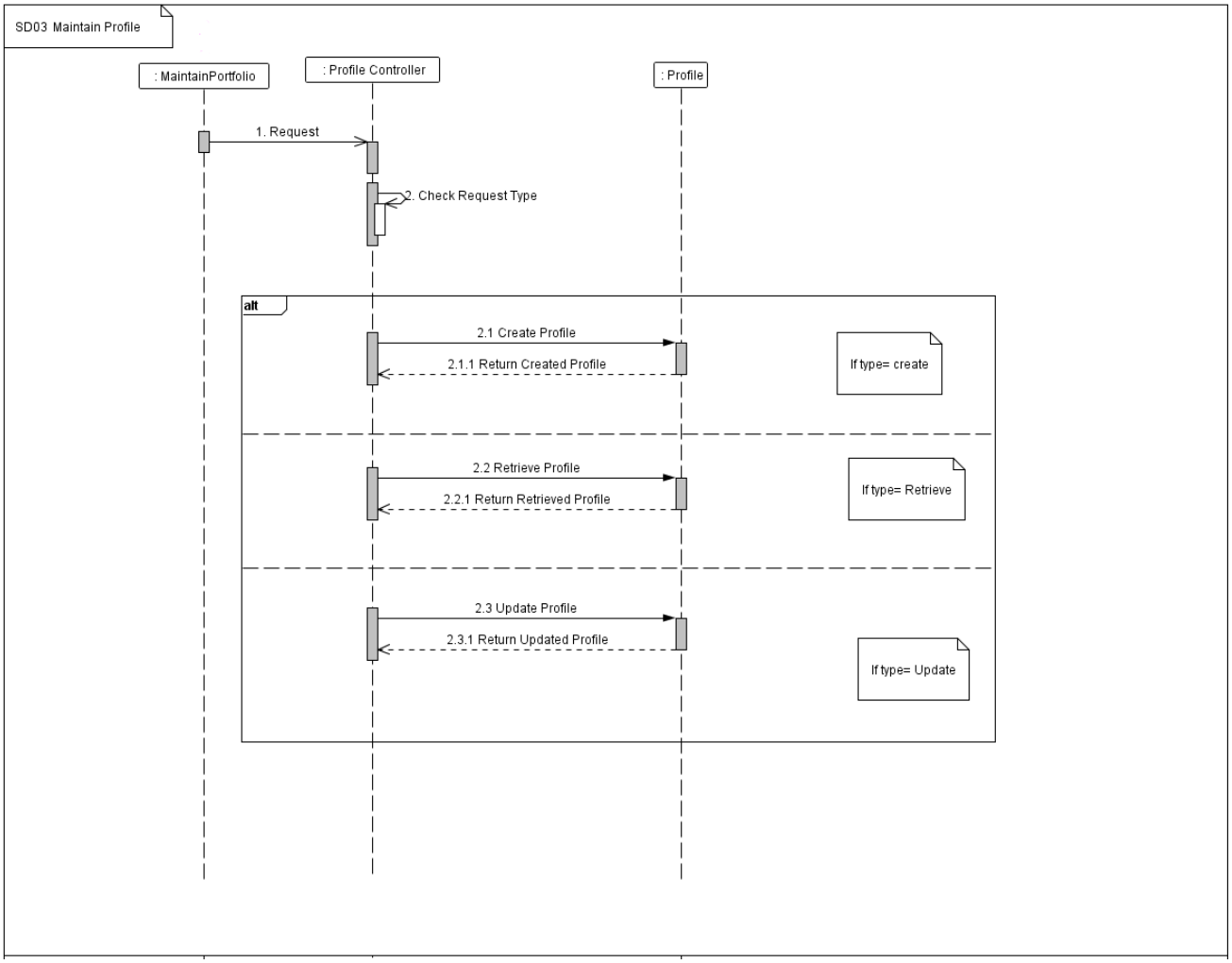


Figure 4.3.3

4.3.4 SD04 Maintain Account

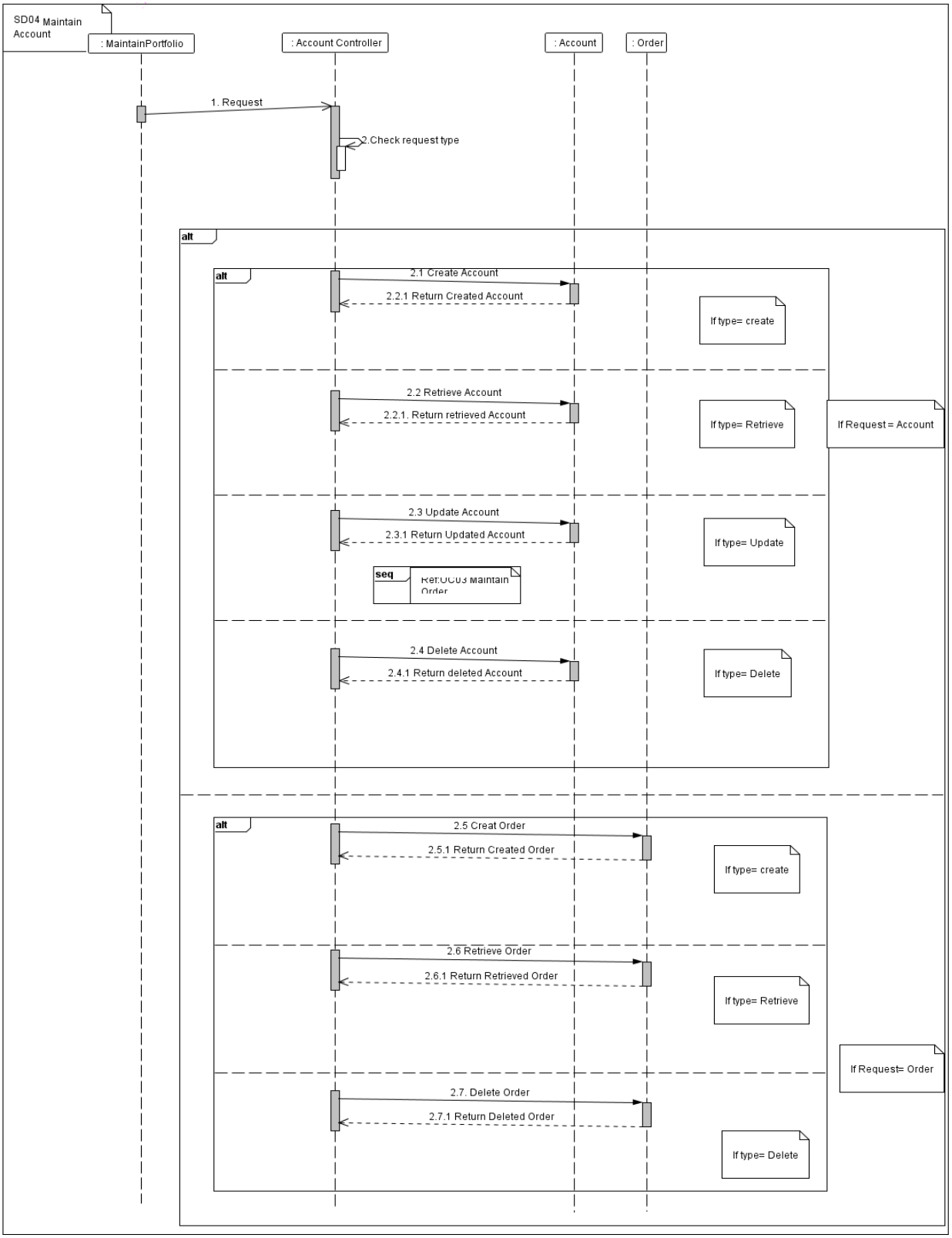


Figure 4.3.4

4.3.5 SD05 Maintain Order

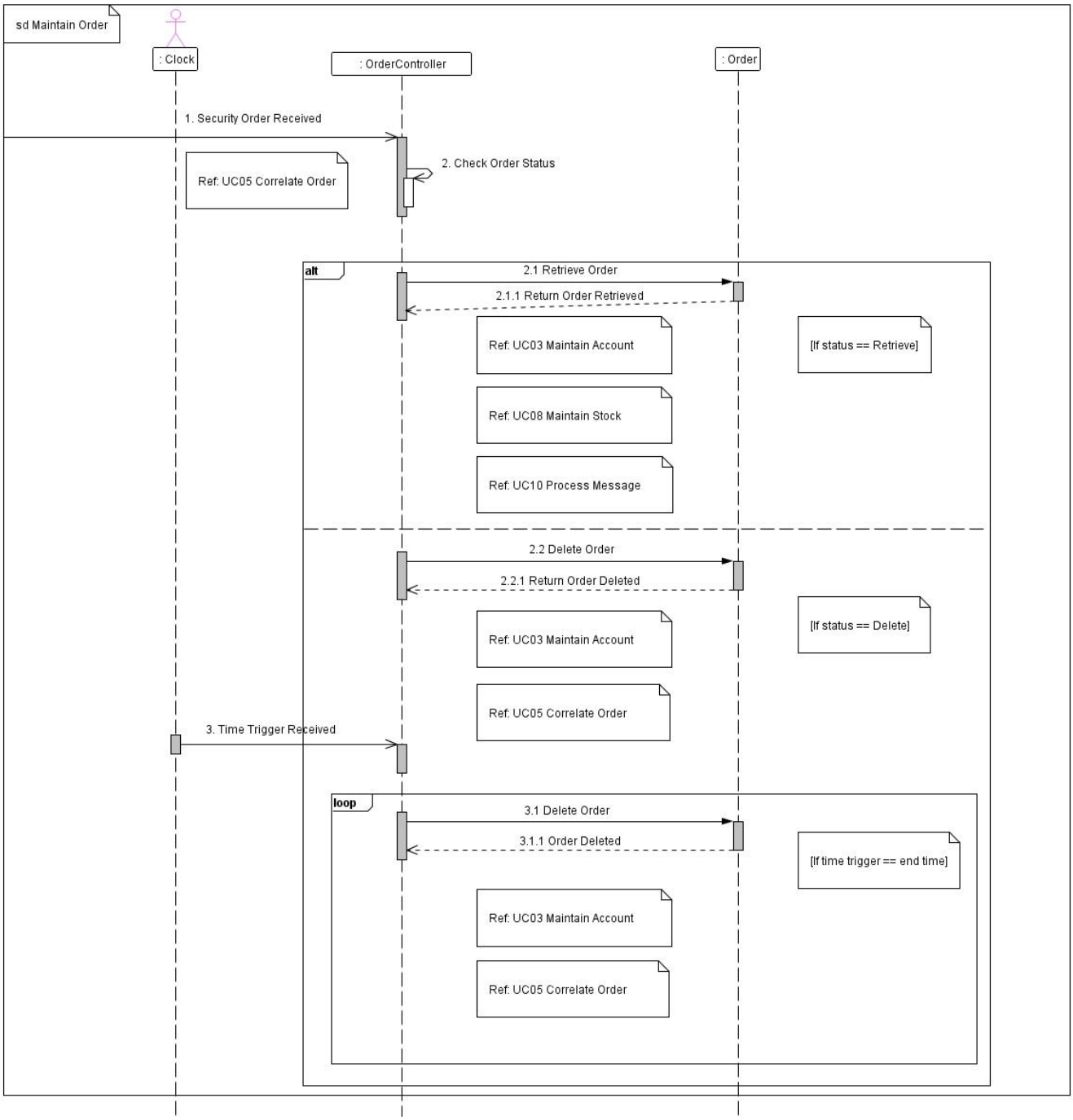


Figure 4.3.5

4.3.6 SD06 Correlate Order

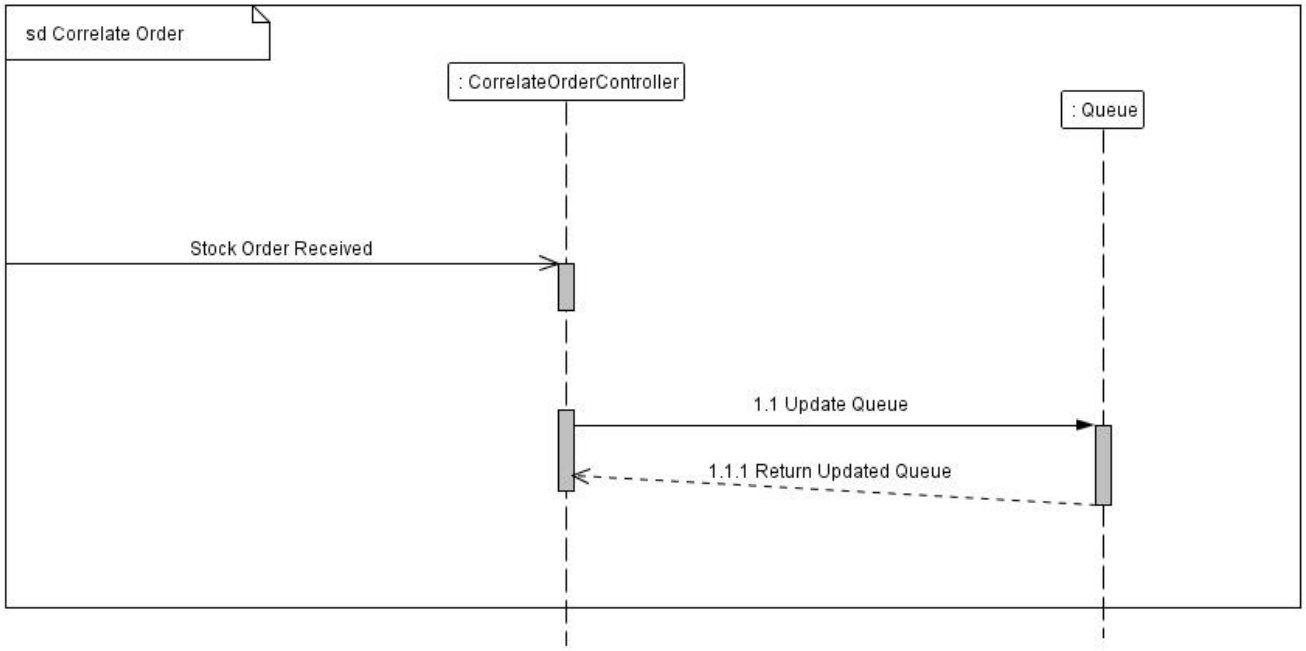


Figure 4.3.6

4.3.7 SD07 Process Order

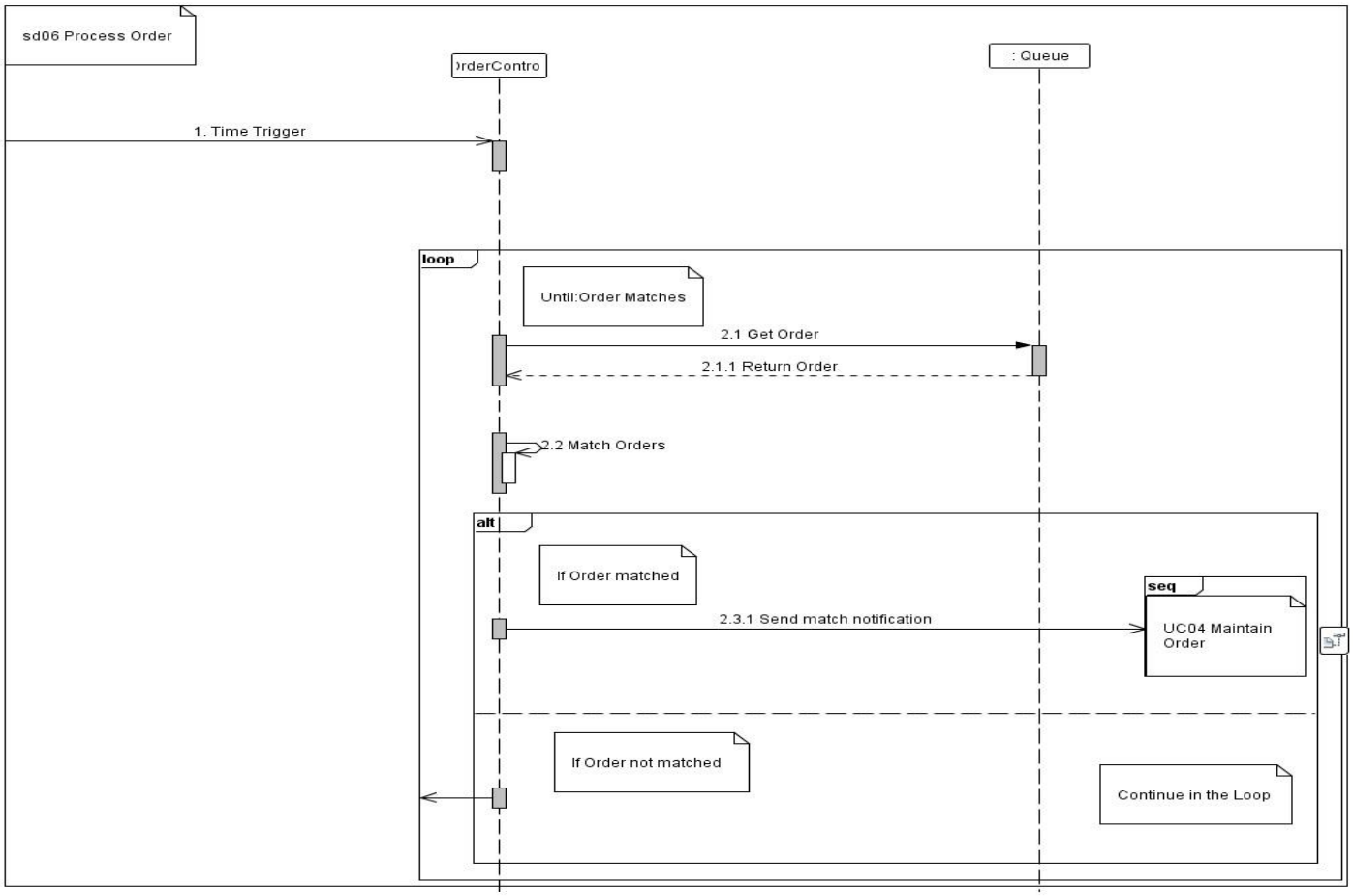


Figure 4.3.7

4.3.8 SD08 Process Mutual Fund

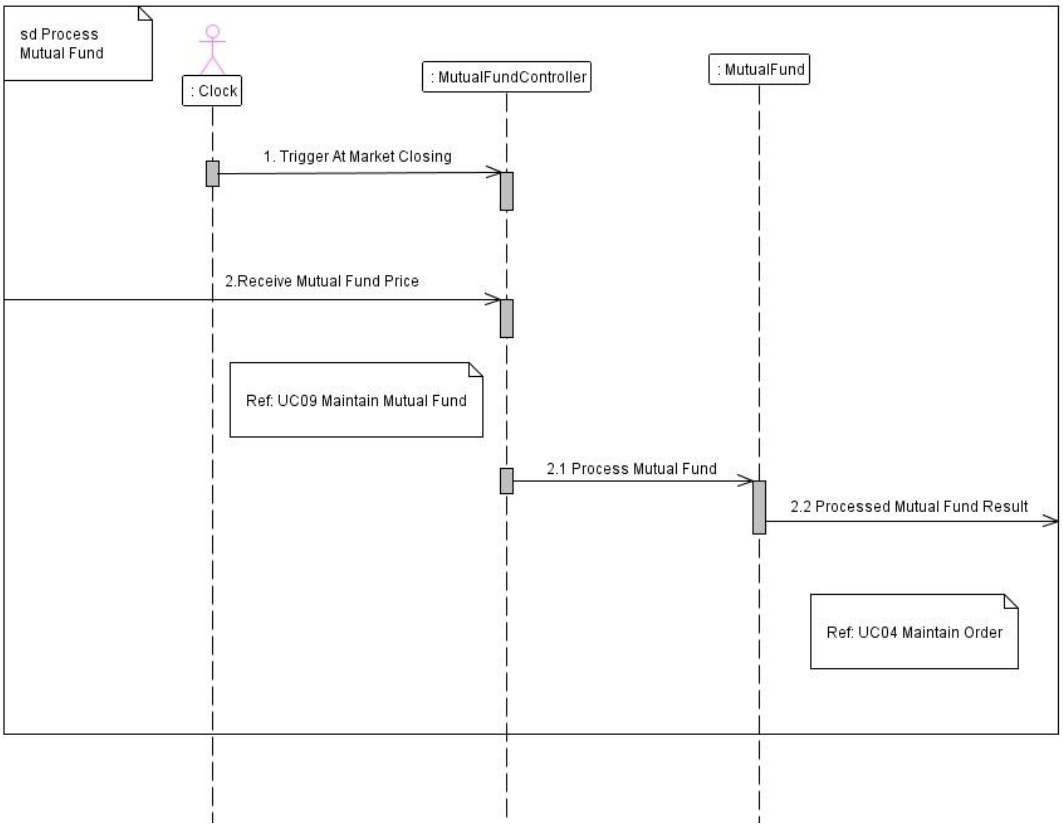


Figure 4.3.8

4.3.9 SD09 Maintain Stock

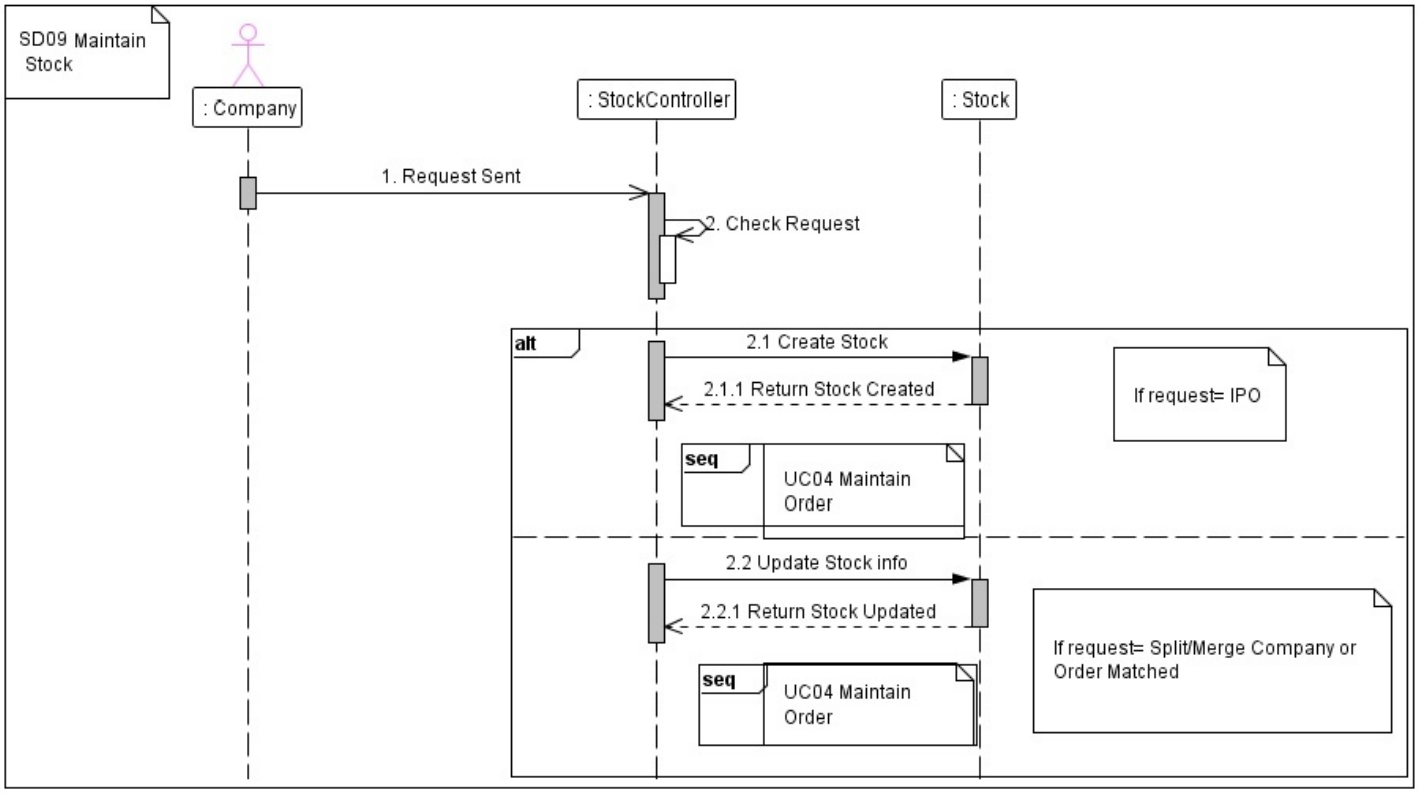


Figure 4.3.9

4.3.10 SD10 Maintain Mutual fund

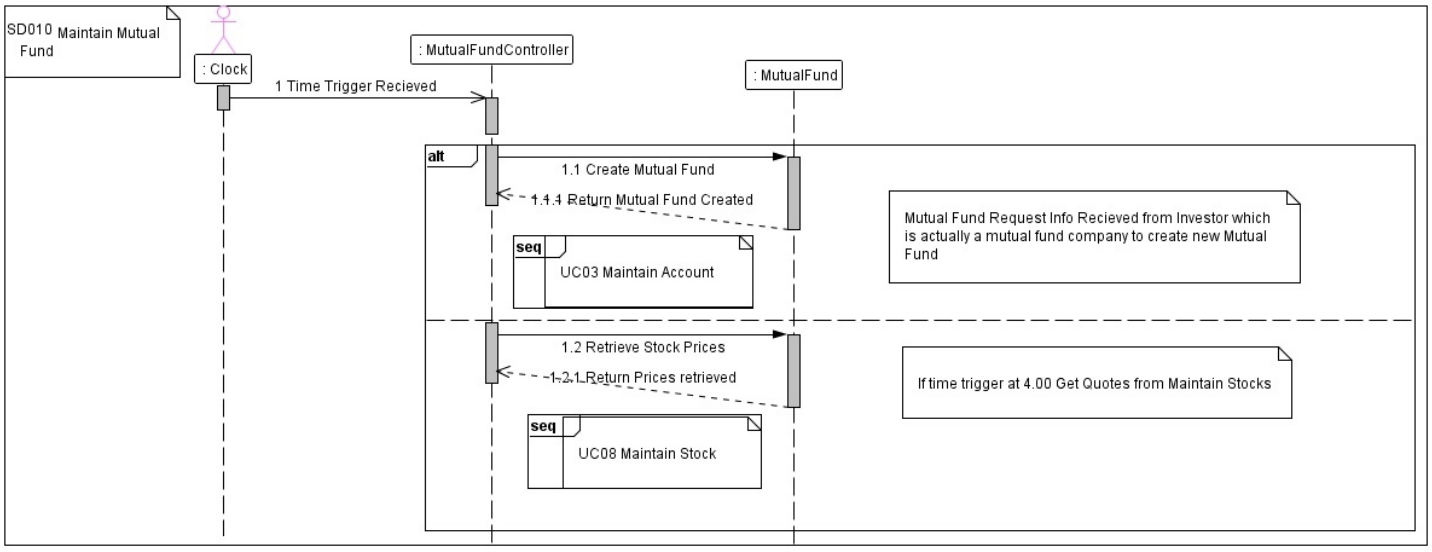


Figure 4.3.10

4.3.11 SD11 Process Message

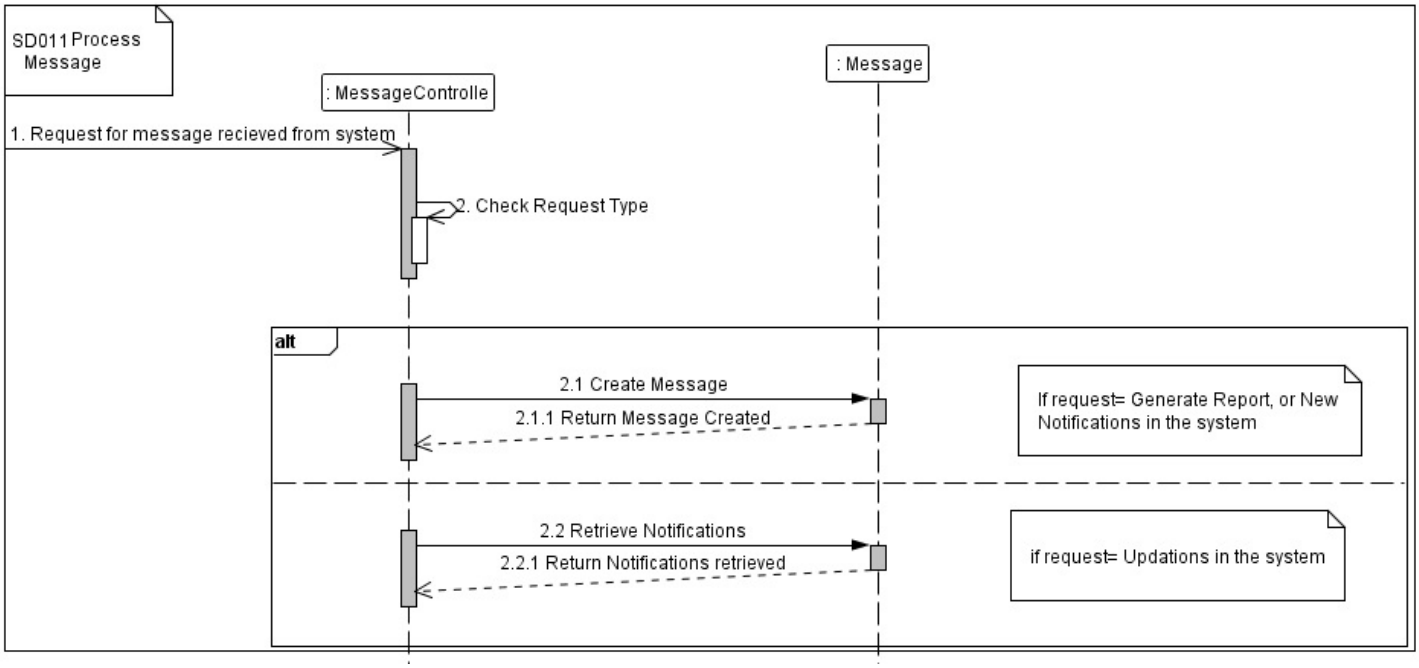


Figure 4.3.11

4.4 Requirements Traceability Matrix

Table 4.4

	Use Case										
	Process Request Manager	Maintain Portfolio	Maintain Profile	Maintain Account	Maintain Order	Correlate Order	Process Order	Process Mutual Fund	Maintain Mutual Fund	Maintain Stock	Maintain Message
	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10	UC11
R1.1.1	X	X	X	X							
R1.1.2	X				X	X	X	X			
R1.1.3									X	X	
R1.1.4										X	
R1.1.5		X	X	X							
R1.1.6				X	X						
R1.1.7							X	X			
R1.1.8											X
R1.2.1	X			X	X	X	X	X			
R1.2.2											X
R1.3.1				X	X						
R1.3.2				X	X						
R1.3.3				X	X						
R1.3.4		X	X	X							
R1.3.5		X	X	X							
R1.3.6		X	X	X							
R1.3.7									X	X	
R1.3.8									X	X	
R1.3.9									X	X	

R1.4.1	**X**			**X**							
R1.4.2				**X**	**X**	**X**					
R1.4.3											**X**
R1.5.1	**X**			**X**							
R1.5.2					**X**						
R1.5.3				**X**	**X**	**X**	**X**				
Iterations	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Workflows	**RADIT**	**RADIT**	**RADIT**	**RADIT**	**RADIT**	**RADIT**	**RADIT**	**RADIT**	**RADIT**	**RADIT**	**RADIT**

4.5 Use Case Interaction Overview Diagram

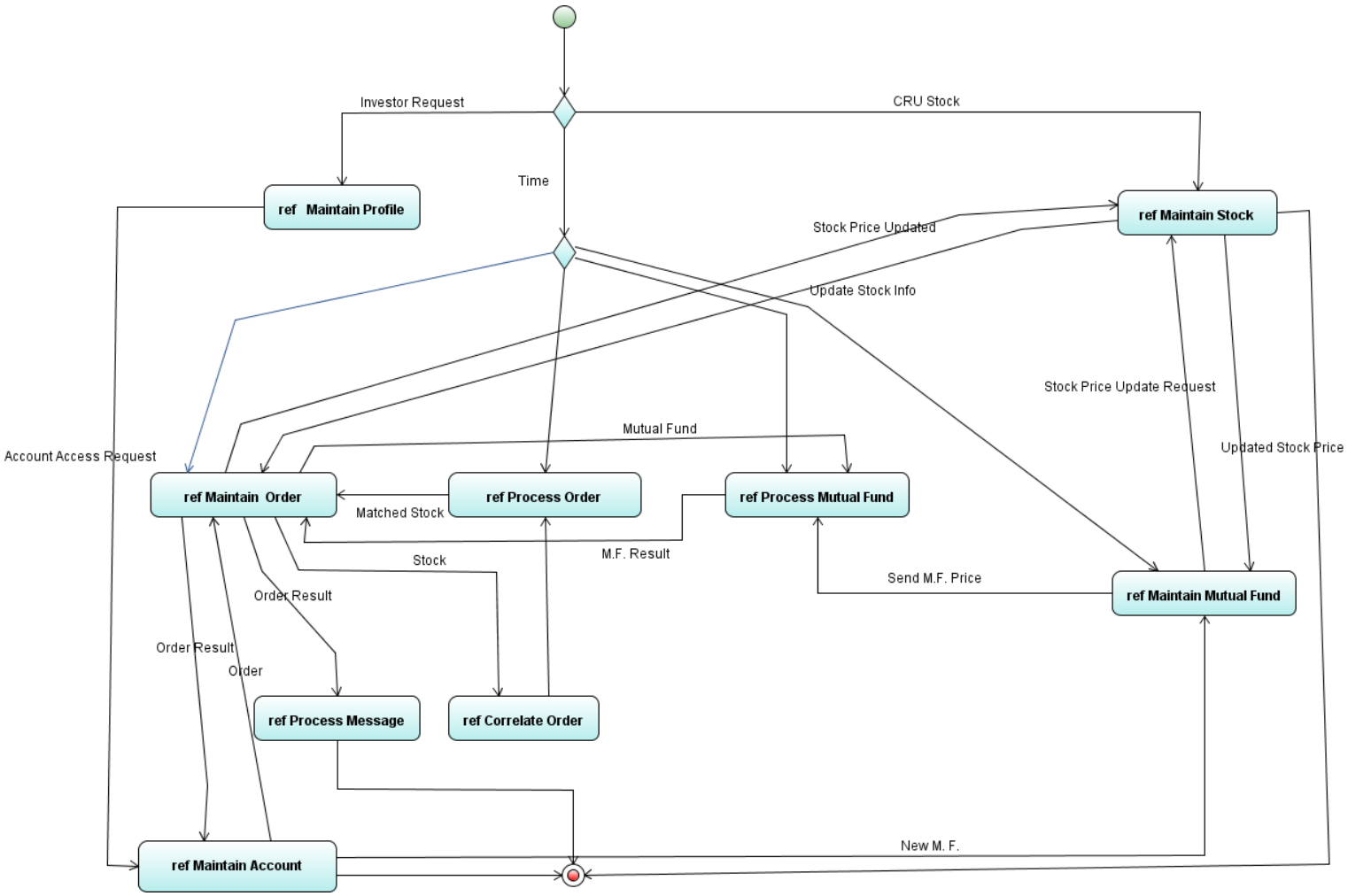


Figure 4.5

5. Design

5.1 Design Class Diagram

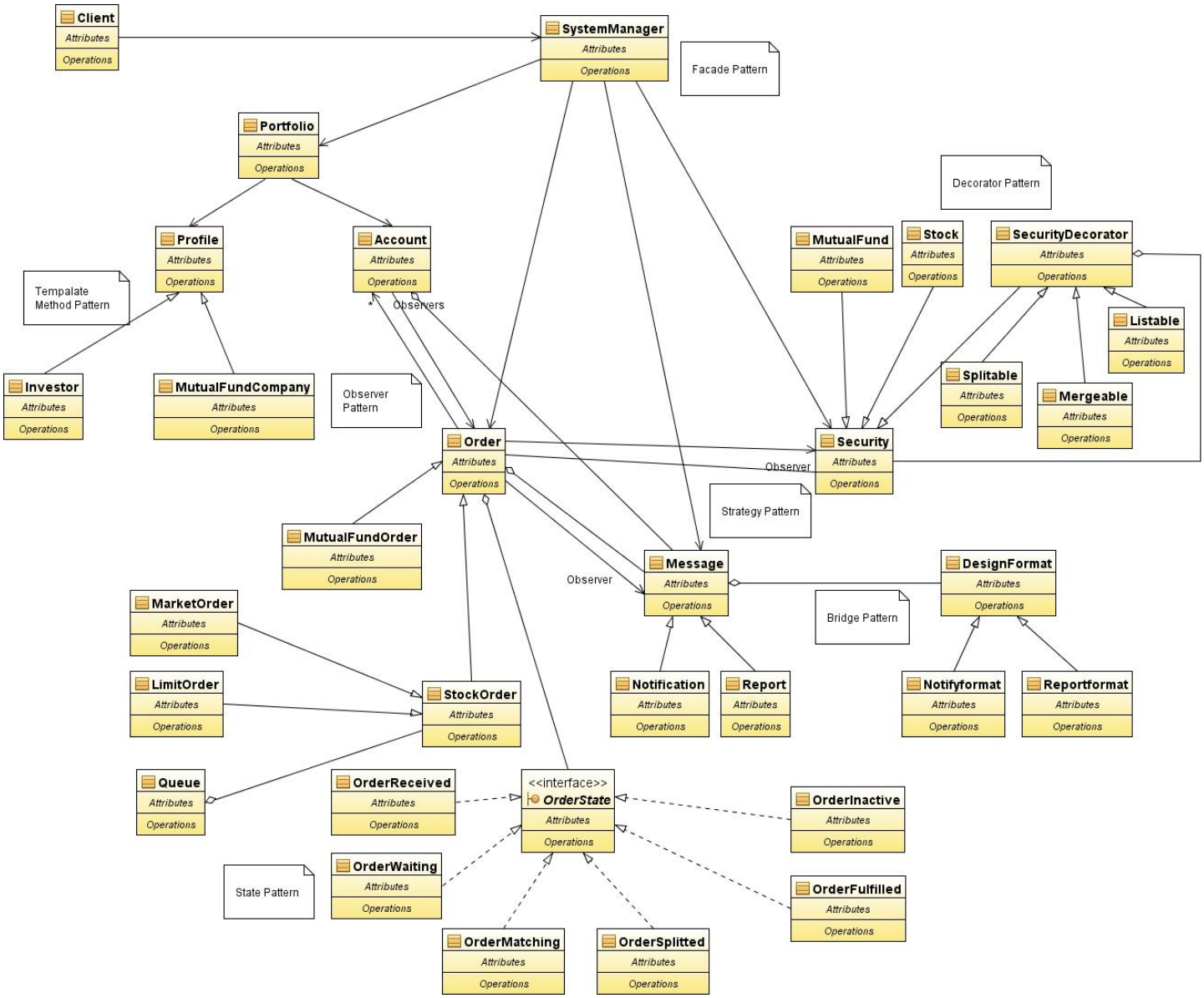


Figure 5 1

5.2 Design Class Description

SystemManager
Attributes: Portfolio p Message m Security sec Order ord
Operations: +addUser() +login() +getCredentials() +generateInvestormenu() +generateMutualFundCompanyMenu()

Portfolio
Attributes: -String name; -String password; -Long[] accountId;
Operations: +updateProfile() +createAccount()

Profile
Attributes: -String name; -String email; -String password; -String address; -Long[] accountId;
Operations: +updateProfile() +createAccount()

Account
Attributes: #brokrageName: String #email: String #accountId: long #stock: Stock[] #mutualFund: MutualFund[] #cash: double #order: Order[]
Operations: +updateAccount() +updateOrder() +createOrder()

BrokerageAccount
Attributes:
Operations: +updateOrder() +updateBalance()

MutualFundCompany
Attributes:
Operations: +createNewMutualFund()

Order
Attributes: #status: String #orderType: String #seller_Or_buyer: String #combinedOrderId(format as : profileName_accountName_orderId): String #profileId: long #accountId: long #priceSet: double #securityId_shareNum: Hashtable<Security,int>
Operations: +isExpired() +addObserver() +deleteObserver() +notifyAccount() +notifyMessenger() +notifySecurity()

MutualFundOrder
Attributes:
Operations: +setPrice() +getPrice()

QueueState
Attributes:
Operations: +receiveOrder() +processOrder() +eliminateOrder()

ClosedQueue
Attributes: -securityId: long - sellOrder: Order[] - buyOrder: Order[]
Operations: +receiveOrder() +processOrder() +eliminateOrder()

OpenQueue
Attributes: -securityId: long - sellOrder: Order[] - buyOrder: Order[]
Operations: +receiveOrder() +processOrder() +eliminateOrder()

Message
Attributes: -msgid: int #messagetype: String #message: String
Operations: +generate_message() #generate_notification() #generate_report() +send_message() # send_notification() # send_report()

Notification
Attributes:
Operations: +generate_message() +send_ message ()

Report
Attributes:
Operations: +generate_ message () +send_ message ()

DesignFormat
Attributes:
Operations: +generate_notification() +generate_report() + send_notification() + send_report()

NotifyFormat
Attributes:
Operations: +generate_notification() + send_notification()

ReportFormat
Attributes:
Operations: +generate_report() + send_report()

Security
Attributes: -SecurityID: long #Securityname: String #Price: double -Securitytype: String
Operations: -displaysecurity()

MutualFund
Attributes: -mutualfundname -mutualfundPrice
Operations: -displaymutualfund() -setPrice() -getPrice()

Stock
Attributes: -StockSymbol -StockPrice -numberofstocks -Pricechange
Operations: -displaystock() -setPrice() -getPrice()

SecurityDecorator
Attributes:
Operations: -displaySecurity()

Updatable
Attributes:
Operations: -mergecompany() -splitcompany()

Listable
Attributes:
Operations: -addnewStockcompany() -addnewMutualFund()

OrderState
Attributes:
Operations: #processing Order()

OrderWaiting
Attributes:
Operations: +processing Order()

OrderRecieved
Attributes:
Operations: +processing Order()

OrderFulfilled
Attributes:
Operations: +processing Order()

OrderInactive
Attributes:
Operations: +processing Order()

5.3 System Decomposition

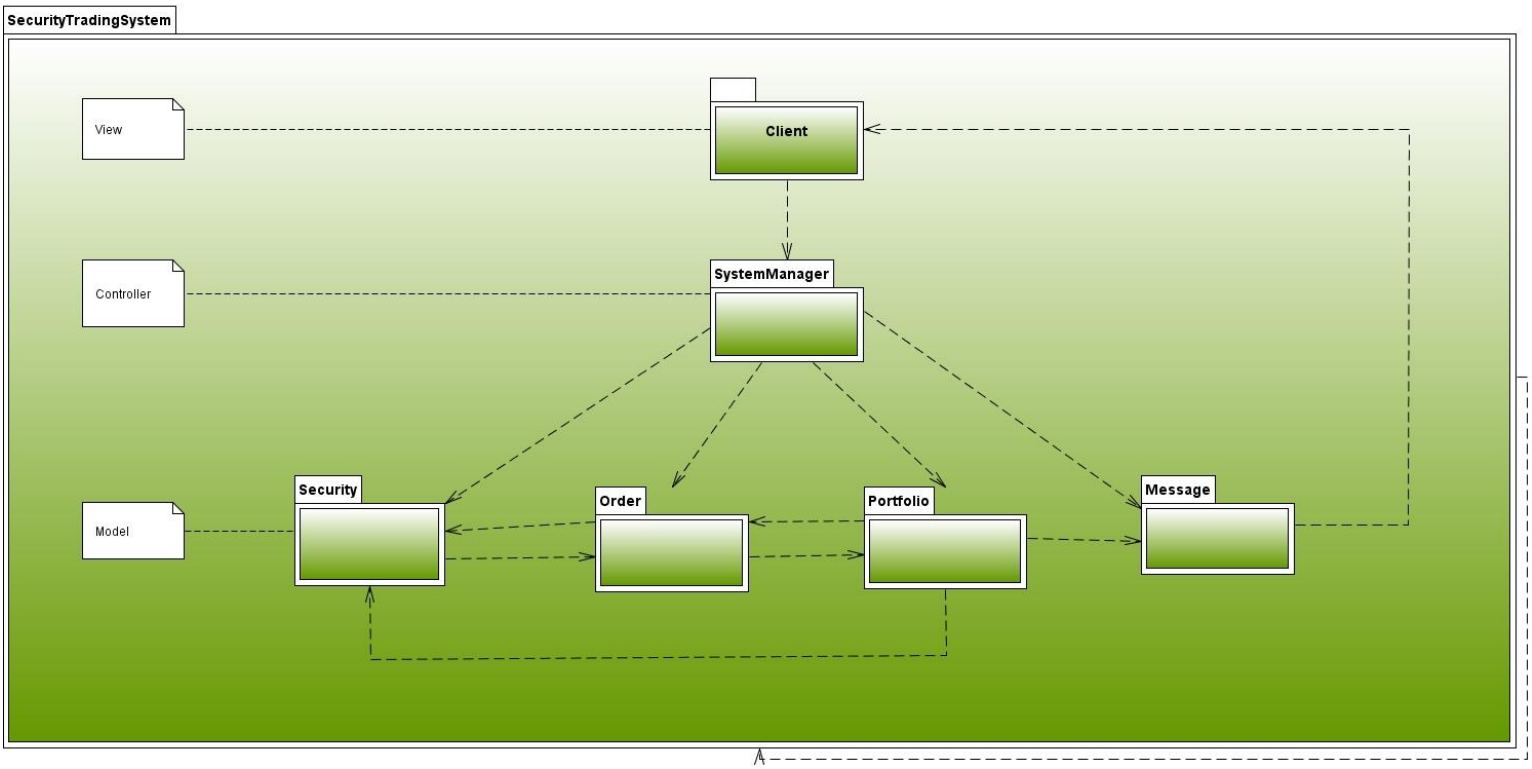


Figure 5.3

5.4 System Architecture

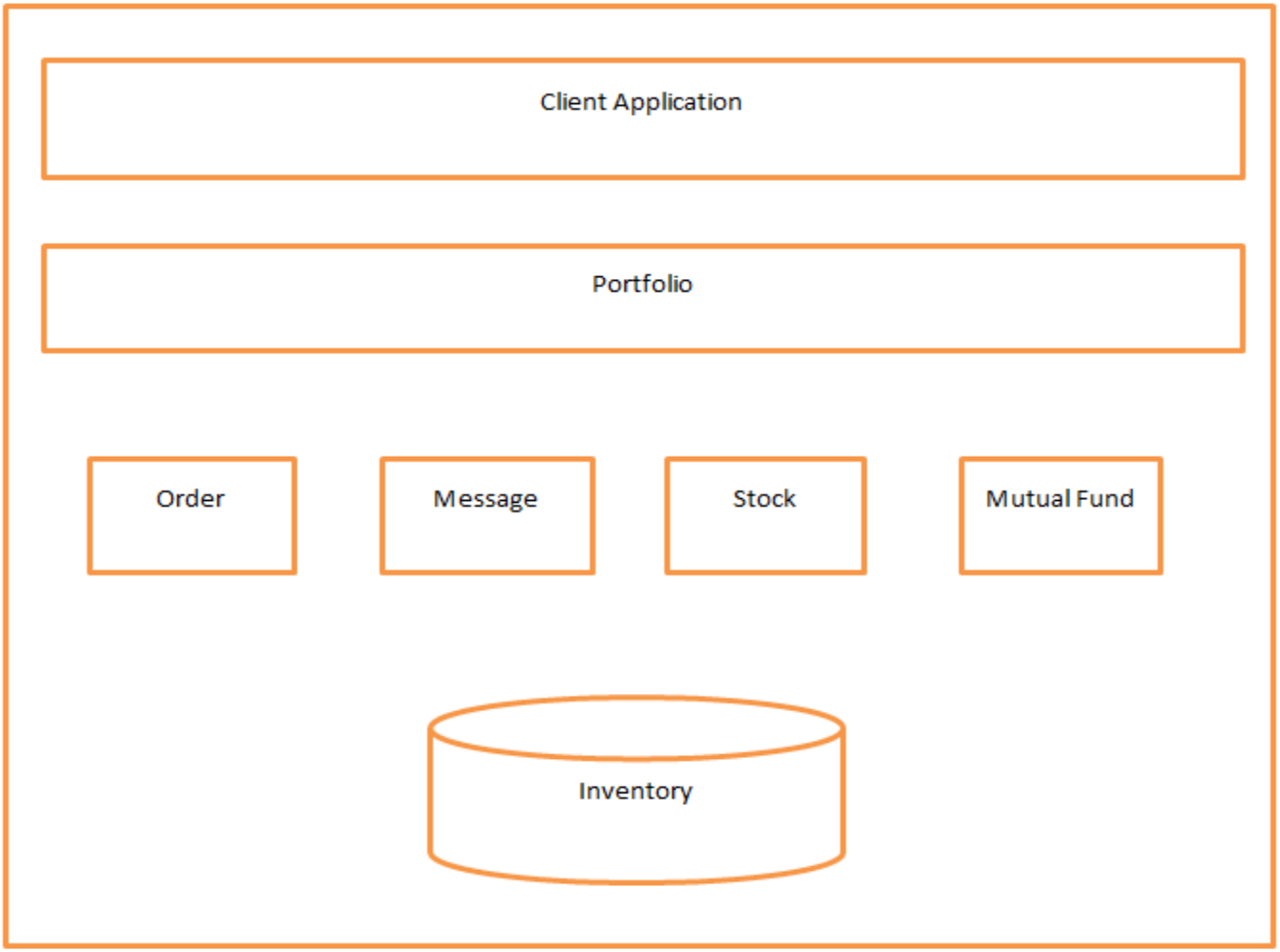


Figure 5.4

Architecture consists of four basic layers. The client interacts with the top level which consists of the Client Application. A client could be Mutual Fund Company, investor or stock company.

Depending on the type of client, Client Application interacts with the corresponding subsystem. If the client is Mutual Fund Company or investor, Client Application will interact with portfolio layer and portfolio will forward coming requests to corresponding subsystem which is on the third layer. If the client is stock company, Client Application will directly interact with stock subsystem and execute request such as merging stock, split stock and create IPO. Corresponded mutual fund will be affected as a result in Mutual Fund subsystem.

Subsystems in second layer and third layer are related, affected by each other but independent at the same time. All the request related to profile and account will be execute within Portfolio subsystem. All kinds of orders will be create within account subsystem under Portfolio and forwarded to Order subsystem to be processed. Market orders will be executed immediately whereas limited order needs to wait in queue for its turn to match. Matching will be done every second triggered by time and if match has been found, result will

be sent to Message, Stock and Portfolio for notification to client, updating stock price and update order status. So will be other kinds of order results. Mutual fund order will not be executed until 4pm every day, which is also triggered by time.

Whenever CRUD operations are performed on portfolio, order, message, stock and mutual fund. Data is updated in Inventory for future reference.

6. Application

6.1 Application Setup

The system begins populated with two investors from which one is the mutual fund company, each of the investor having two to three different accounts. The system is consisting of three stocks and three mutual funds which the investors are going to buy.

The tables below describe the data in the application:

Portfolio Table

PortfolioID	Name	AccID	InvType	OrderID
1	Diego	274652, 283947	1	123423, 343231
2	Fidelity OTC Portfolio K	938427, 329402, 829384	2	345234, 633111, 451113

Profile Table

ProfileID	Name	email	Phone
1	Diego	diego@xyz.com	9832931000
2	Fidelity OTC Portfolio K	fidelity@aaa.com	4082263849

Account Table

AccountID	AccountName	OrderID	Cash	Stock	MutualFund	ProfileID
274652	CMP1	343231	50000	(1,40) (3,20)		1
283947	CMP2	123423	10000	(1,100)	(3,100)	1
329402	CMP2	633111	5000		(2,200)	2
829384	CMP1	345234	30000	(2,400)	(3,50) (1,40)	2
938427	CMP1	451113	46000	(2,50)	(1,300)	2

Order Table

OrderID	AccountID	OrderType	Sell_or_Buy	TimeDuration
123423	283947	Limit	Sell	12.10.2011
343231	274652	Market	Buy	
345234	829384	Market	Buy	
451113	938427	Limit	Sell	12.04.2011
633111	329402	Limit	Sell	11.30.2011

Security Table

SecurityID	SecuritySymbol	SecurityType
1	AAPL	2
2	ATHAX.O	1

3	DEEVX.O	1
4	FOCKX.O	1
5	HP	2
6	YHOO	2

Stock table

StockID	StockSymbol	Price	Type
1	AAPL	376.72	2
2	YHOO	14.77	2
3	HP	54.02	2

Mutual Fund Table

MutualFundID	MutualFundName	Symbol	Price	Type
1	American Century Heritage A	ATHAX.O	18.72	1
2	Delaware Smid CAP Growth C	DEEVX.O	18.51	1
3	Fidelity OTC Portfolio K	FOCKX.O	54.28	1

6.2Application Interaction

The following outputs will be generated after the application is executed:

********SCREENSHOTS*******

Create User and Login

```
        Welcome to Stock Trading System

1. Login
2. View Security listings
3. Create new User
4. Exit
5. Change Clock
3
*************Enter your details*************
********************************************
1. Enter your desired user name:
dallas
2. Enter desired password:
tycoon

    WELCOME TO STOCK TRADING SYSTEM!
****************************************
***************Main Menu****************
****************************************
please choose your role:
1. Investor
2. Mutual Fund Company
3. Stock Company
4. Go back to main menu
5. Exit
1
....please enter your username:
dallas
Enter your password:
tycoon
Welcome user!! uid: 3
```

Adding new Mutual Fund

```
    WELCOME TO STOCK TRADING SYSTEM!
**************************************
***************Main Menu***************
**************************************
please choose your role:
1. Investor
2. Mutual Fund Company
3. Stock Company
4. Go back to main menu
5. Exit
2
******************************************
********Mutual Fund Company Menu*********
******************************************
please choose your role:
1. Add a new mutual Fund
2. Show all listings
3. Go back to main menu
1
Please enter the new mutual fund name:
VJSOX
and number of mutual funds:
234
What Stocks are contained in the mutual fund, Enter id's:

EnterID:
1
Adding:
1. Stock: AAPL  Price: 376.72  Number of Stocks: 500000

Enter Number of stocks:
200
```

```
mutual fund avg price: 376.72
Do you want to add more Stocks in mutualfund(y/n)?
y

EnterID:
2
Adding:
2. Stock: YHOO  Price: 14.77  Number of Stocks: 600000

Enter Number of stocks:
34
mutual fund avg price: 324.12897435897435
Do you want to add more Stocks in mutualfund(y/n)?
n
Mutual Fund Created from function
****************************************
********Mutual Fund Company Menu*********
****************************************
please choose your role:
1. Add a new mutual Fund
2. Show all listings
3. Go back to main menu
2
```

Adding new Stock

```
****************************************
**********Stock Company Menu***********
****************************************
please choose your role:
1. Add a IPO
2. Show all listings
3. Go back to main menu
1
Please enter the new stock name:
farenhiet
and its current price:
34.42
and number of stocks:
400000
Stock Created from function
----------------------------------------
```

Showing Listings

```
****************************************
********Mutual Fund Company Menu*********
****************************************
please choose your role:
1. Add a new mutual Fund
2. Show all listings
3. Go back to main menu
at least one Q is empty right now,wait for order to be inserted....
2
        *************Stock listings*********

1. Stock: AAPL  Price: 376.72  Number of Stocks: 500000
2. Stock: YHOO  Price: 14.77  Number of Stocks: 600000
3. Stock: farenhiet  Price: 34.42  Number of Stocks: 400000

        *************Mutual Fund listings*********

1. Mutual Fund: American Century Heritage A  Price: 256.07   Number of mutual funds: 150
2. Mutual Fund: Delaware Smid CAP Growth C  Price: 169.8914285714286   Number of mutual funds: 350
3. Mutual Fund: VJSOX  Price: 376.72   Number of mutual funds: 234
```

Listing Accounts

```
Welcome user!! uid: 2
****************************************
     ********Investor Menu*********
****************************************
Enter your Choice:
1. List your Accounts
2. Create An order
3. View your order status
4. Exit
1
acc size:2
Account no 0:1
User ID : 1
e-mail : x@y.c
Account name: Acc1
Cash in Account: 500000.0


Account no 1:2
User ID : 2
e-mail : t@w.c
Account name: Acc2
Cash in Account: 600000.0
```