Orderbook Architecture Document

1. Introduction

1.1. Purpose

This document provides a high level overview and explanation of an orderbook application. Contained within are sections that cover in detail the uses, implementation, and reasonings behind every component of the application.

1.2. Definitions, Acronyms, and Abbreviations

Term	Definition
Order Book	List of orders that show interest of buyers to sell or buy a financial instrument (stock)
Order	A purchase or sale inquiry for a stock
Buy Order	An inquiry to purchase a specific stock at a specific price
Sell Order	An inquiry to sell a specific stock at a specific price
Order ID	ID number for a given order
Size	The volume of stock for a given order
Side	Whether an order is a buy or sell order
Time (order)	The time at which an order is placed
Active	If the order listing is still available to execute
Offer Price	The sale or buy price of a share of a stock order
Symbol	Associated ticker symbol for a given stock
Transaction	The fulfillment of a given buy order and sell order
Buy Order ID	The Order ID associated with the buy order of a transaction

Sell Order ID	The Order ID associated with the sell order of a transaction
Final Time	The time at which a transaction is completed
Final Price	The price at which the transaction was executed
Amount	The volume of shares sold/bought in the transaction
Final Symbol	The ticker symbol for stock bought/sold in the transaction
Match	When the buy order price >= sell order price
Delete Order	Remove a buy or sell order from the orderbook
Trade History	List of all transaction that have occurred in the past

2. Architectural Goals and Constraints

2.1. Setup

The application is currently created to be hosted locally on a single machine. While running it will send and retrieve information from a MySQL database stored on the same machine (Database design is covered in a later section). This program can be adjusted to work with databases on different machines, however this was not the main focus for this version of the application.

2.2. Reliability

The application has been vigorously tested to ensure that all logic is sound and any errors that may arise have been accounted for and will be dealt with by the application should they arise.

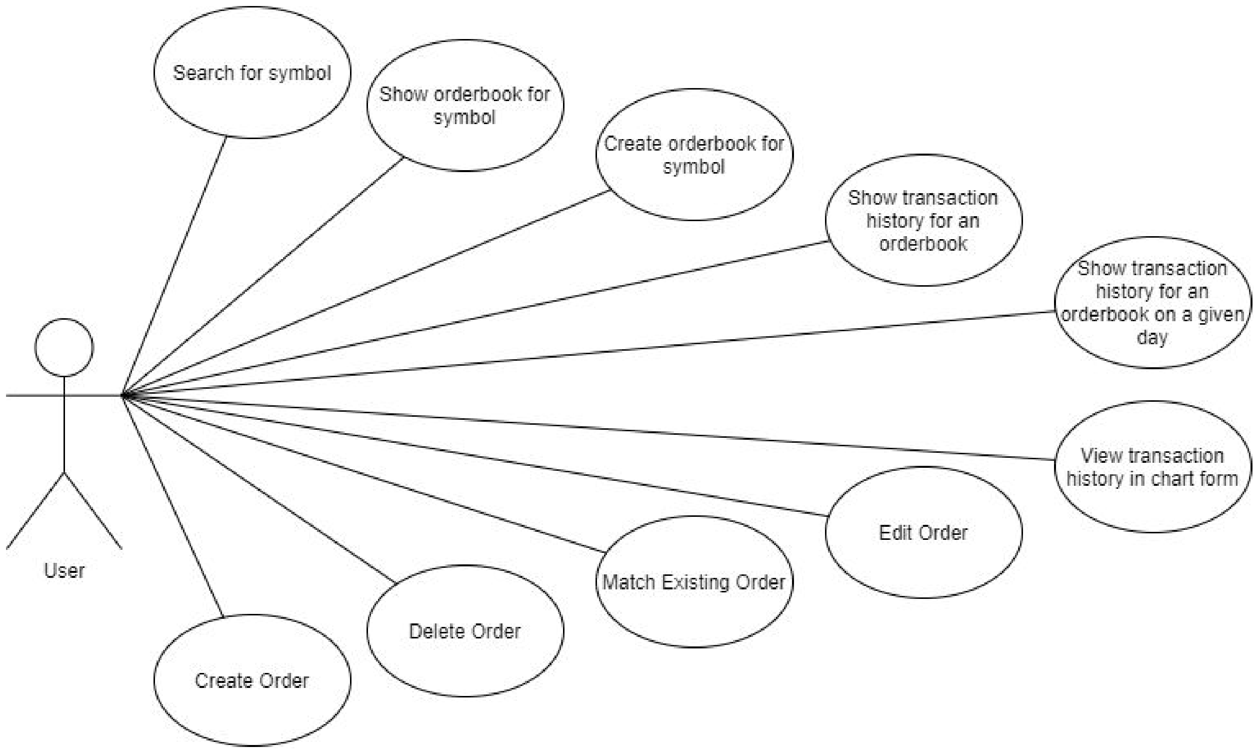
2.3. Development tools

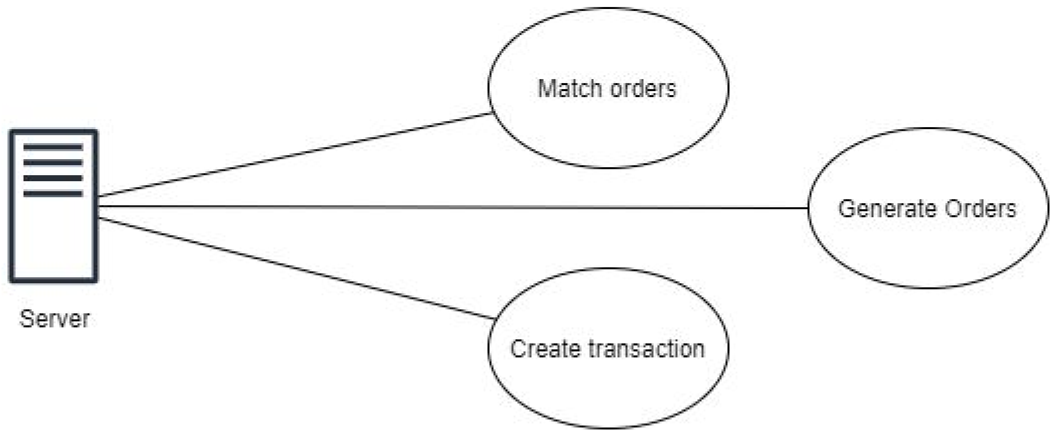
This application was created using the following tools:

Database: MySQL
Diagrams: Draw.io
Programming: NetBeans IDE, Visual Studio Code

3. Use-Case View

3.1. Use case diagrams


Search for symbol
Show orderbook for symbol
Create orderbook for symbol
Show transaction history for an orderbook
Show transaction history for an orderbook on a given day
View transaction history in chart form
Edit Order
Match Existing Order
Delete Order
Create Order
User



3.2. Use case realizations

3.2.1. User specific use cases

3.2.1.1. View transaction history for an orderbook from navigation bar

Use case name	View transaction history for an orderbook from navigation bar
Scenario	User wants to view all transaction history for an orderbook for a specific symbol using the navigation bar.
Triggering Event	Clicking symbol name from “View Trade History” dropdown menu in navigation bar.
Brief Description	On click, a drop down menu will appear listing all symbols currently tracked by the application. From this list, a user needs to choose a symbol to be redirected to an orderbook for said symbol.
Actors	User
Related use cases	View transaction history for an orderbook from home page View transaction history for an orderbook on a given day
Preconditions	None
Post conditions	None
Flow of events	User loads any webpage for the program. User chooses “View History” from the navigation bar. User chooses a symbol from the drop down list. Application requests all transactions from the connected database. Application adds transactions to the webpage via model. Application redirects users to the webpage. Webpage populates with the given model.
Exception conditions	If the symbol cannot be found, the user will be redirected to an error page asking them if they would like to create an orderbook for said symbol.

3.2.1.2. View transaction history for an orderbook from home page

Use case name	View transaction history for an orderbook from home page

Scenario	User wants to view all transaction history for an orderbook for a specific symbol from the home page
Triggering Event	Clicking “View Trade History” button in symbol activity form
Brief Description	On click, the user will be redirected to the trade history for a selected symbol’s orderbook.
Actors	User
Related use cases	View transaction history for an orderbook from navigation bar View transaction history for an orderbook on a given day
Preconditions	None
Post conditions	None
Flow of events	User loads the home page User chooses a symbol from the drop down menu in “Symbol Activity”. User chooses “View Trade History” directly below the symbol drop down menu. Application requests all transactions from the connected database. Application adds transactions to the webpage via model. Application redirects users to the webpage. Webpage populates with the given model.
Exception conditions	If the symbol cannot be found, the user will be redirected to an error page asking them if they would like to create an orderbook for said symbol.

3.2.1.3. View transaction history for an orderbook on a given day

Use case name	View transaction history for an orderbook on a given day
Scenario	User wants all transactions that occurred on a specific day
Triggering Event	User clicks “Find for Date” after entering a date
Brief Description	On click, the user will be given all transactions that occurred on a given day, sorted by earliest first.

Actors	User
Related use cases	View transaction history for an orderbook from navigation bar View transaction history for an orderbook from home page
Preconditions	None
Post conditions	None
Flow of events	User navigates to the trade history for an orderbook. User enters a date. User clicks on the “Find for Date” button. Application requests all transactions that occurred on the given date. Application adds a new list of transactions to model. Webpage repopulates itself using the new list.
Exception conditions	None

3.2.1.4. View transaction history in chart form

Use case name	View transaction history in chart form
Scenario	User wants a visual representation of all transactions currently listed.
Triggering Event	User clicks “View Chart” button in trade history
Brief Description	On click, the user will be shown a graph containing all transactions currently listed. This graph will have its x value be the dates of the transactions and its y value be the price of the transactions.
Actors	User
Related use cases	None
Preconditions	None
Post conditions	None
Flow of events	User navigates to the trade history page for an orderbook.

	Optional: User enters a date to get a list of all transactions for the given date User clicks “View Chart”. Webpage will then display all transactions currently listed in the form of a graph.
Exception conditions	None

3.2.1.5. Create orderbook for symbol from home page

Use case name	Create orderbook for symbol from home page
Scenario	User wants to create an orderbook for a new symbol from the home page form
Triggering Event	User clicks “Submit” after entering the symbol they want to create
Brief Description	On click, the application will either redirect the user to an existing orderbook if the symbol entered already exists or to the newly created orderbook page.
Actors	User
Related use cases	Create orderbook for symbol from symbol not found webpage
Preconditions	Valid text in the pattern of 1-45 alpha characters
Post conditions	A new orderbook will be created, assuming the symbol entered did not already exist
Flow of events	User navigates to the home page. User enters a symbol for a wanted orderbook. User clicks “Submit”. If the orderbook already exists, the user will be redirected to it. If the orderbook did not exist, it will be created before being redirected to it.
Exception conditions	If no text or invalid text is entered, the form will tell the user and the form will not run.

3.2.1.6. Create orderbook for symbol from symbol not found error page

Use case name	Create orderbook for symbol from symbol not found error page
Scenario	User creates an orderbook for a symbol after being redirected to the symbol not found error page
Triggering Event	User clicks “Yes!, Let’s start trading {symbol}”.
Brief Description	On click, the application will create a new orderbook for the displayed symbol.
Actors	User
Related use cases	None
Preconditions	User must be at the symbol not found error page for entering a symbol that does not exist.
Post conditions	A new orderbook for the wanted symbol will be created
Flow of events	User is redirected to the error page after the application cannot find the wanted symbol. User clicks “Yes! Let’s start trading {symbol}.”. Application will create a new orderbook. Application redirects user to the orderbook.
Exception conditions	None

3.2.1.7. Search for symbol

Use case name	Search for symbol
Scenario	User uses the search bar to find a symbol and its related orderbook
Triggering Event	User clicks “Search a Symbol” after entering the symbol they want to search for.

Brief Description	On click, the user will be redirected to one of three locations. 1. Home page if no symbol was entered 2. The orderbook for the symbol entered 3. Error page for no symbol found
Actors	User
Related use cases	None
Preconditions	None
Post conditions	None
Flow of events	User enters the symbol they want to find the orderbook for. User clicks “Search for Symbol”. Application will search for orderbook using given symbol. If found, user will be redirected to orderbook. If given symbol is empty, user will be redirected to home page If symbol could not be found, user will be redirected to a symbol not found webpage.
Exception conditions	None

3.2.1.8. Show orderbook for a stock symbol from navigation bar

Use case name	Show orderbook for a stock symbol from navigation bar
Scenario	User wants the orderbook for a specific stock using the navigation bar
Triggering Event	User chooses a symbol from the “View an Orderbook” drop down menu in the navigation bar.
Brief Description	Clicking on the button “View Orderbook” will open a drop down list containing all symbols currently tracked in the application.
Actors	User
Related use cases	Show orderbook for a stock symbol from home page
Preconditions	None

Post conditions	None
Flow of events	User clicks “View an Orderbook” from the navigation bar. User chooses a symbol from the opened drop down menu. Application gets all orders for said symbol and adds them to the model. Application redirects user to a webpage to display orders. Webpage populates with all orders from the model.
Exception conditions	None.

3.2.1.9. Show orderbook for a stock symbol from home page

Use case name	Show orderbook for a stock symbol from navigation bar
Scenario	User wants the orderbook for a specific stock using the navigation bar
Triggering Event	User clicks “View Orderbook” in symbol activity form
Brief Description	On click, the user will be redirected to the orderbook for the selected symbol.
Actors	User
Related use cases	Show orderbook for a stock symbol from navigation bar
Preconditions	None
Post conditions	None
Flow of events	User chooses the symbol from the drop down menu located in the symbol activity form. User chooses the “View Orderbook” button directly below the chosen symbol. Application gets all orders for said symbol and adds them to the model. Application redirects user to a webpage to display orders. Webpage populates with all orders from the model.
Exception conditions	None.

3.2.1.10. Match existing order

Use case name	Match existing order
Scenario	User matches top buy/sell order
Triggering Event	User clicks “Match” button next to an order
Brief Description	On click, an order will be generated that will match the chosen order (price, amount, etc.) with the exception of the order type (this will be the opposite). Once created, a transaction is conducted using the two orders, fulfilling both and creating/updating the relevant entries in the database for storage.
Actors	User
Related use cases	Make transaction
Preconditions	At least one order must exist and the given order must be at the top of the list for their type (highest buy price or lowest sell price).
Post conditions	Orders should be fulfilled and a transaction connecting the two should be created. All three will be updated/saved to the database.
Flow of events	User navigates to an orderbook that contains at least one order. User chooses the top buy/sell order they want to fulfill. An opposite order is generated based on the order chosen. Orders are used to create a transaction, which is stored in the database along with the updated(fulfilled) orders.
Exception conditions	If no orders exist, this operation will not be available to the user.

3.2.1.11. Create Order

Use case name	Create Order
Scenario	User creates a new buy/sell order for a given orderbook

Triggering Event	User clicks “Add Buy Order” after filling out necessary information
Brief Description	On click, user will given a prompt asking them to specify the price and amount of the order. Clicking “Add Buy Order” will create the order.
Actors	User
Related use cases	Edit Order
Preconditions	Price and amount must be specified
Post conditions	None
Flow of events	User navigates to an orderbook. User chooses to create either a sell or buy order. User specifies the price and amount for the order. User clicks “Add Buy Order”.
Exception conditions	Not entering a price/amount will cause a prompt warning the user that the field is required.

3.2.1.12. Edit Order

Use case name	Edit Order
Scenario	User wants to edit an existing order
Triggering Event	User clicks “Edit” next to an existing order
Brief Description	On click, user will be redirected to a webpage to edit the amount and size of a given order.
Actors	User
Related use cases	Create Order
Preconditions	Order must exist and be considered active
Post conditions	An updated order
Flow of events	User navigates to an order with active orders. User clicks “Edit” next to an existing order.

	Application directs the user to a webpage populated with information about the selected order (price, size, and symbol).
Exception conditions	None

3.2.1.13. Delete order

Use case name	Delete order
Scenario	User chooses to delete an order from an orderbook for a given symbol
Triggering Event	User chooses delete for an order
Brief Description	On click, the selected order will either be deleted completely from the database or will be considered inactive (size will be set to zero) so it will not be displayed. This is done only if the selected order has been previously used for a transaction(s).
Actors	User
Related use cases	None
Preconditions	A order must exist
Post conditions	An order considered inactive by the application (size will be 0) or the order will be deleted.
Flow of events	User navigates to an orderbook that contains at least one order. User clicks “Delete” for an order. Order is checked against the database to see if any transactions exist that reference said order. If such a transaction exists, the order’s size is set to zero to be considered inactive. Otherwise it will be deleted completely from the database.
Exception conditions	If no orders exist, this operation will not be available to the user.

3.2.2. Automated use cases

3.2.2.1. Generate Orders

Use case name	Generate Orders
Scenario	Application auto-generates orders assuming there's a need for new orders.
Triggering Event	Timed - Every 30 seconds
Brief Description	Once the timer goes off, the amount of buy and sell orders in every orderbook is checked to see if they are less then the amount limit specified in the business logic.
Actors	Application
Related use cases	Create order
Preconditions	None
Post conditions	Randomly generated buy/sell orders are created
Flow of events	Timer goes off. For every orderbook, the amount of buy orders and sell orders are checked. If the buy orders are less than the specified limit, a predefined amount will be generated. If the sell orders are less than the specified limit, a predefined amount will be generated. Else, it will move on to the next symbol or end the function call.
Exception conditions	None

3.2.2.2. Match orders

Use case name	Match Orders
Scenario	Two orders are compared to see if a transaction is possible.
Triggering Event	Timed - Every 30 seconds

Brief Description	Once the timer goes off, for every orderbook the buy and sell orders are compared to see if any transaction is possible. If so, a transaction between the two will occur.
Actors	Application
Related use cases	Create transaction
Preconditions	Two orders capable of a transaction
Post conditions	Updated orders and a new transaction.
Flow of events	Timer goes off. A list of all orderbooks currently available is created. For said orderbooks, their buy and sell orders are compared to see if any transactions are possible. If possible they are sent to create a transaction.
Exception conditions	Orders are not compatible, in which case no transaction is made.

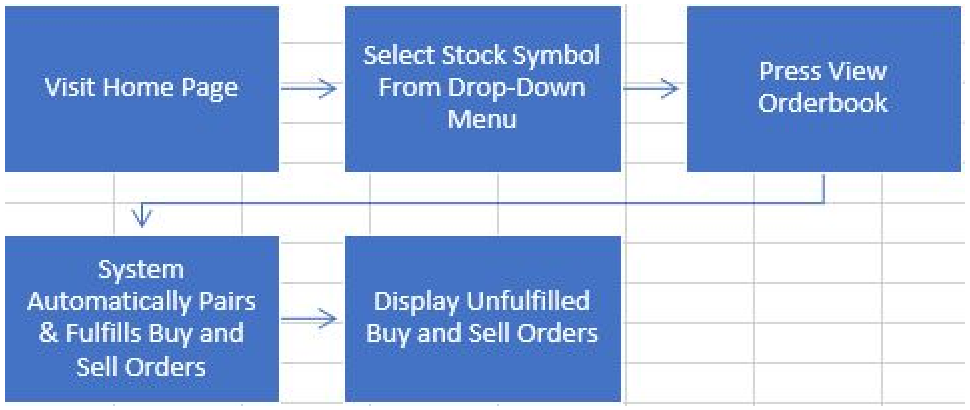
3.2.2.3. Create transaction

Use case name	Create transaction
Scenario	Two orders are used to create a transaction, updating the orders to show the proper values as well.
Triggering Event	Called by “Match Orders” and “Match Existing Order”. See related for more information.
Brief Description	Two orders are used to create a new transaction object. This transaction will contain the price, amount of stocks traded, symbol of the stocks, and date of creation. This will be recorded and the orders used will have their values in storage updated to reflect the change in amount.
Actors	Application
Related use cases	Match Orders, Match Existing Order
Preconditions	Two orders that are compatible.
Post conditions	Two updated orders and a newly created transaction.

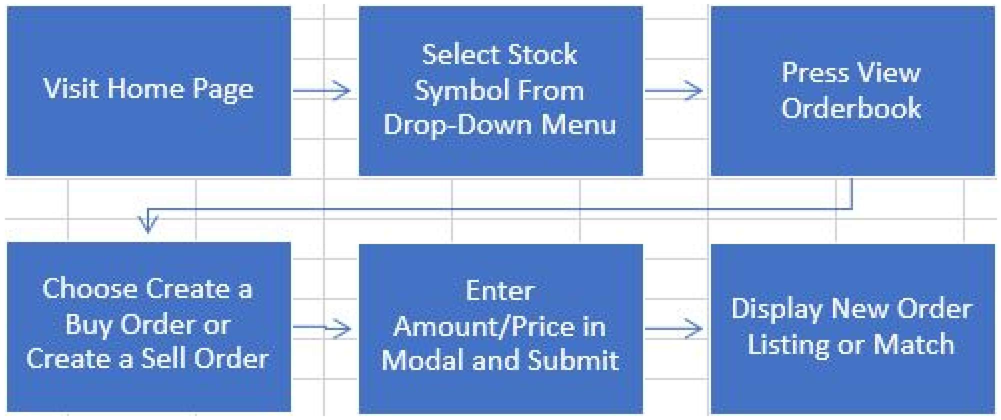
Flow of events	Two orders are sent in. The necessary data for the transaction is pulled from the orders. The symbol will match the two orders. The price will be set to equal the buy order price. The amount will be set to the lowest amount value of either order. Orders will be updated to reflect their new amount values. Transaction will be recorded.
Exception conditions	None.

4. Activity Diagrams

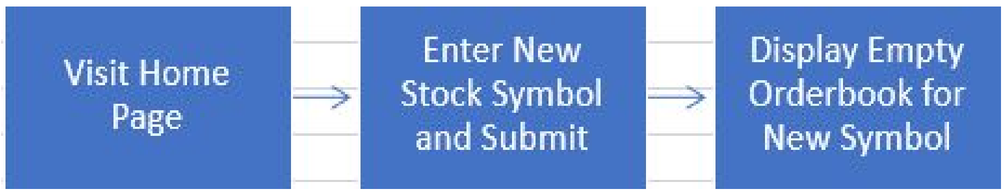
4.1. Automatic Order Matching/Runtime processes


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
System Automatically Pairs & Fulfills Buy and Sell Orders
Display Unfulfilled Buy and Sell Orders

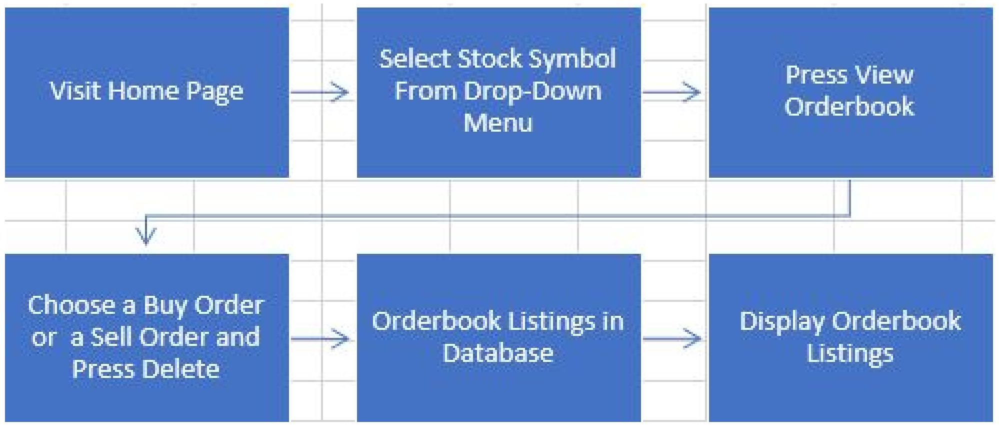
4.2. Adding an Order


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
Choose Create a Buy Order or Create a Sell Order
Enter Amount/Price in Modal and Submit
Display New Order Listing or Match

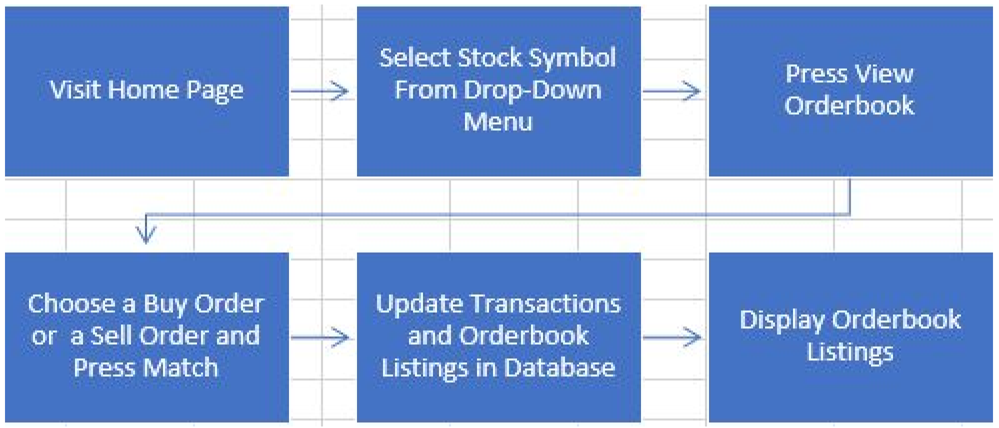
4.3 Create New Orderbook


Visit Home Page
Enter New Stock Symbol and Submit
Display Empty Orderbook for New Symbol

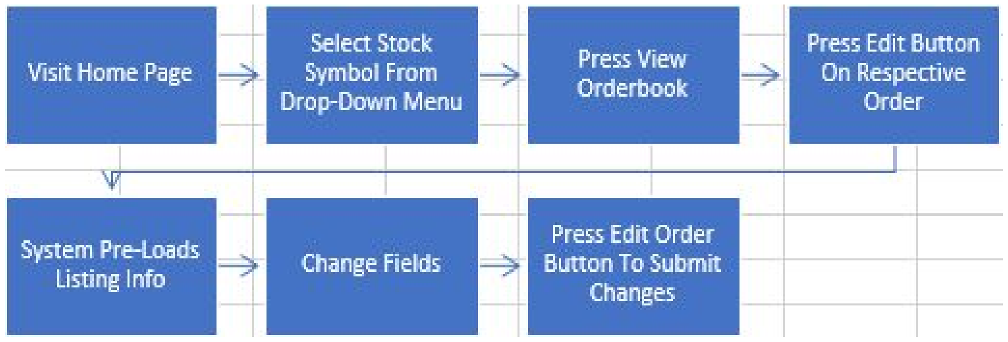
4.4 Deleting an Order


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
Choose a Buy Order or a Sell Order and Press Delete
Orderbook Listings in Database
Display Orderbook Listings

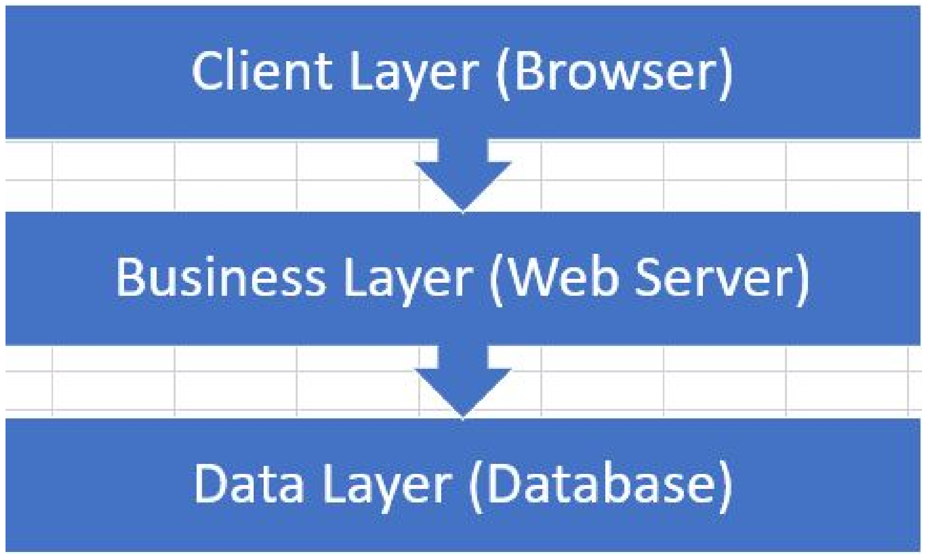
4.5 Manual Order Matching


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
Choose a Buy Order or a Sell Order and Press Match
Update Transactions and Orderbook Listings in Database
Display Orderbook Listings

4.6 Edit An Existing Order


Visit Home Page
Select Stock Symbol From Drop-Down Menu
Press View Orderbook
Press Edit Button On Respective Order
System Pre-Loads Listing Info
Change Fields
Press Edit Order Button To Submit Changes

5. Implementation View



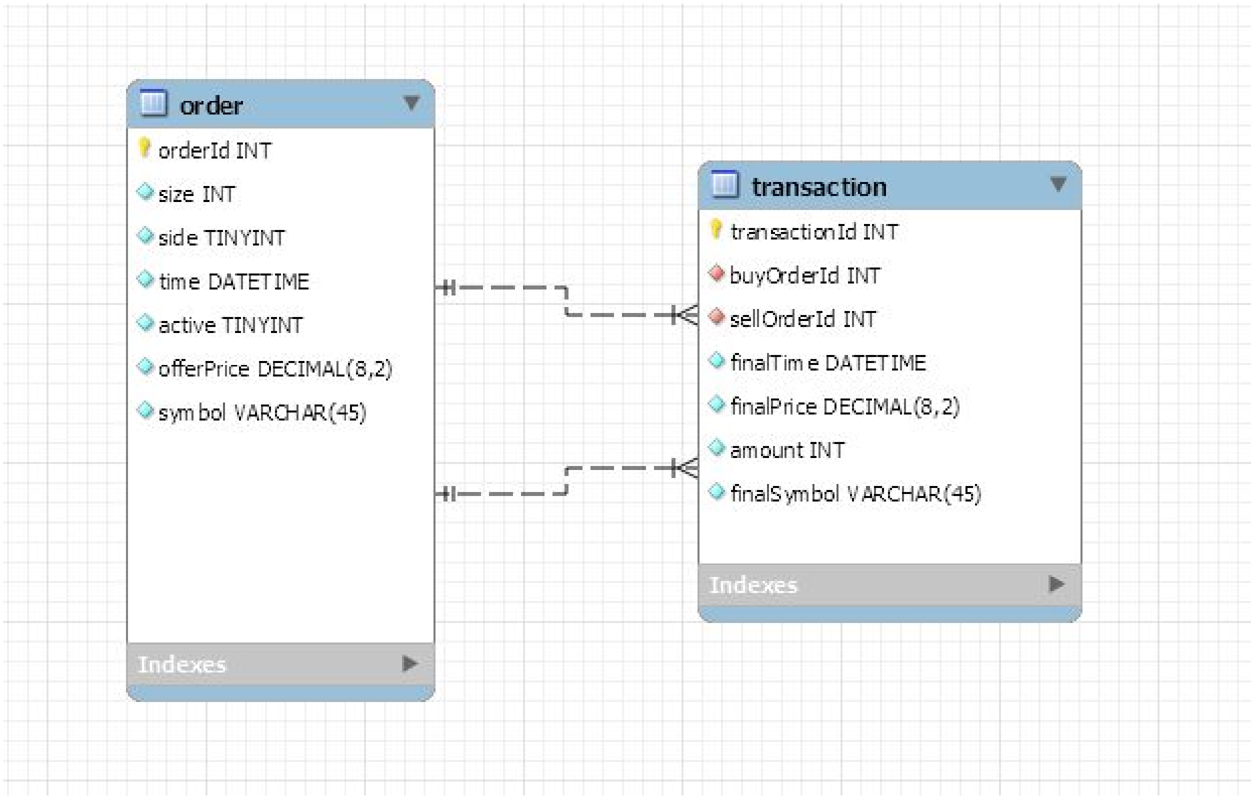
The application was created in the style of MVC. The reasoning of this was to control what each layer could access to avoid unnecessary/unexpected changes to the data and to other aspects of the application.

View layer (client layer) is the user interface (web pages). The actions the user takes in these web pages is handled by controller classes. These classes in turn call functions from the model classes to handle data and business logic. Doing this has the added benefit of reducing complexity by reducing the sizes of each function.

The model classes are separated into two layers, the business layer and data layer. The business layer contains all business logic and connects the view layer (controller) with the data layer. The data layer in turn connects the business layer with the connected database and is directly responsible for the saving and retrieving of data from a database.

6. Data View

6.1. ER diagram



6.2. MySQL implementation

The database was designed in MySQL. The main tables are:

1. Order
2. Transaction

Hibernate framework is used to automate the relations between the two tables. Classes were also created for each table in the database, making it possible for the application to work with the database’s data.