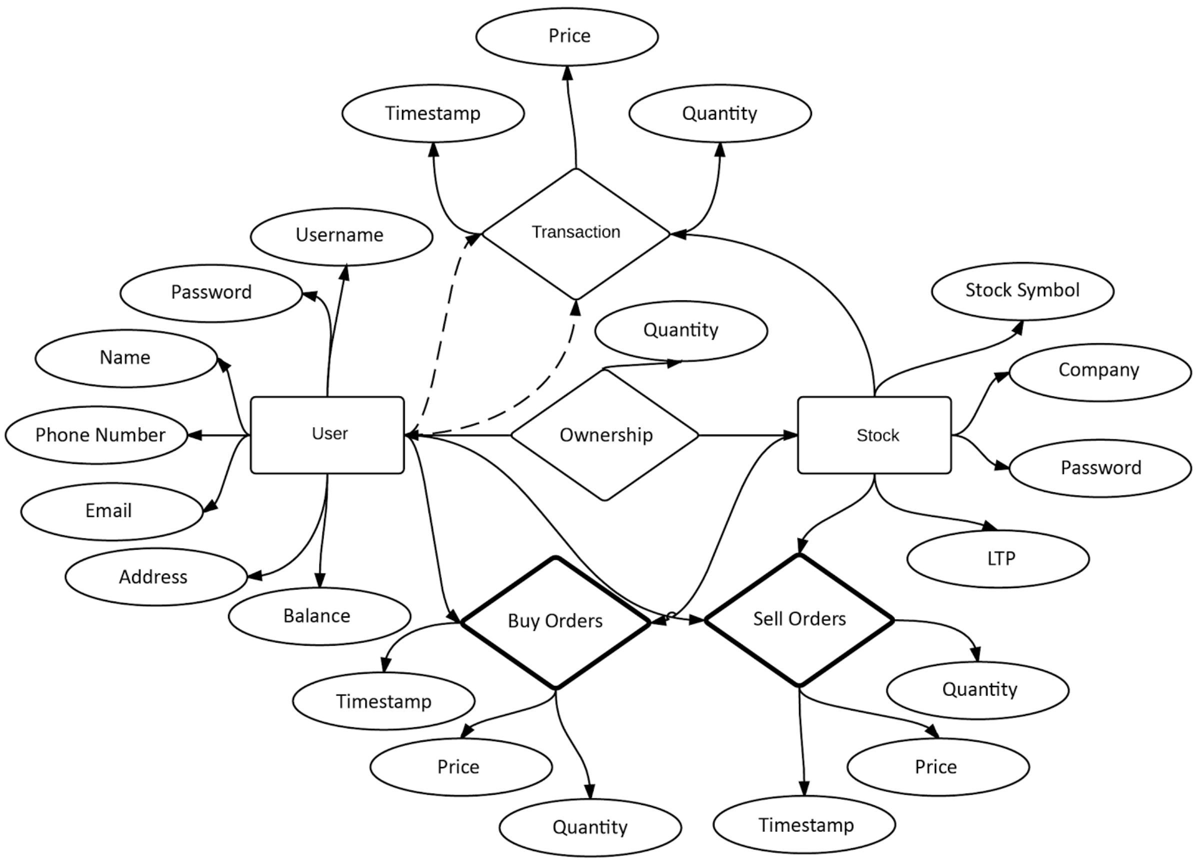
Data Modelling and Implementation Techniques

Abhinav Gupta (120050030)   Ainesh Pandey (14V051001)   Aman Gour (120050030)

Data Modelling

The database consists of 6 main tables. These tables are at the center of the database design of the Virtual Stock Market.
Tables : Users, Stocks, Ownership, Transactions, Sell Orders, Buy Orders



ER Diagram

Normalization in Database design

We used normalization to eliminate data redundancy and undesirable characteristics.
Normalization of our database helped us to serve two main purposes :

1. Eliminating redundant data
2. Ensuring data dependencies make sense i.e. data is logically stored.

We started out with 1-NF and then went higher in the hierarchy to remove the redundancy in data and to integrate the data well.
Current version of database design implements 3NF where we have removed the transitive functional dependency between the *Transaction* table and the *Users* and *Stocks* table.

Physical Database design

Tablespaces:

We divided the tables into **small tables** that have entries over 100s and the **large tables** that have over 1000's of entries. The small table consist of Stock, Users, Ownerships whereas the BuyOrder, SellOrder and Transactions are considered to be large tables because we are assuming that there will be lots of exchange of stocks over the stock exchange.

Indexing:

Problem : The bottleneck of this application is the number of transactions that takes place per second. The transactions that refers to the sell order table and the buy order table poses certain characteristic ways in which they access the data. In the sell orders all the data is accessed in the increasing order of price and secondarily sorted by the time at which the data arrives and similarly in the buy orders table.

Solution : We used indexing to quickly locate data without having to search every row in a database table every time a database table is accessed.

```sql
CREATE INDEX sell_index ON sellOrders (stocksymbol, askPrice, sellDateTime);
CREATE INDEX buy_index ON buyOrders (stocksymbol, bidPrice, buyDateTime);
```

We created these two indexes that provided rapid lookups and efficient access of ordered records in the buy orders and sell order table.

Implementation Details

Class :

1. Stock - As the database requires a frequent transfer of the stock details from the servlets to the jsp's, we designed a new class called stocks that provides the functionality of all the features and queries on the stock and provides a layer of abstraction during the data access related to stocks.

2. Admin - A class that controls the money flow in the virtual stock exchange. All the monetary information has to be extracted from and with the permission of the admin of the database and the transactions are implemented in this class. The purpose of this class is to keep a check and to prevent any malicious behaviors a broker or user might get involved in.

Servlets[1] :

1. **UserLogin** : Contains the backend implementation for user login and registration of the new users. UserLogin class uses the functionality provided by the admin class to open a new account and add money to it. This class also allows the user to edit its information (Profile).
   The JSP's that makes call to this servlet are :

   1. Home : Landing page of the app. This provides an interface that contains the information of the trends in the stock market to any user or company and also provides links to the login/register page for users and for company.

   2. Login : The interface for the user to login using the assigned username and password and also provides new users to register.

2. **Transaction :** As the name suggests this class deals with all the transaction and the orders, both the sell orders and the buy orders placed by the users. It also uses the functionality of the admin class to update the balance of the user.
   It uses *buy.jsp, sell.jsp, userhome.jsp* to complete the request made by user.

3. **Stock Info :** This servlet provide all the functionality to search through all the stocks and display results related to particular stocks.

[1]1. The servlets and classes are what we have decided so far and are subject to change to improve the efficiency of the application.
2. NSE Stock Database : http://www.nseindia.com/products/content/all_daily_reports.html

4. **Company Login :** Allows a company to login and register to the stock exchange. Any company that wants to trade over the stock exchange has to put some stocks in the stock exchange it can be either IPO or a list of already occurring stocks that it wants to trade.

5. **Create Stock :** This allows the company to create a new stock in the market and should refer to the admin class for several integrity constraints.

Data

1. **Stocks** : We created a python script that reads the stocks data from the NSE stock database and creates a sql file that can be used to insert the data in the stock exchange database.

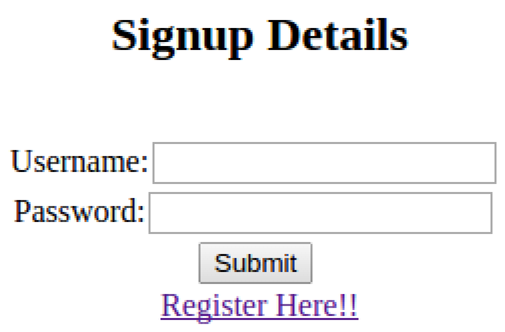
2. **Companies** : The companies that trade over the stock exchange are listed on the NSE. We plan to use this data, mould into the form needed using the scripts and will use in the database.

3. **Other** : Similar scripts are used for filling in the entries in the ownership table, users table and transaction table.

Interface Design

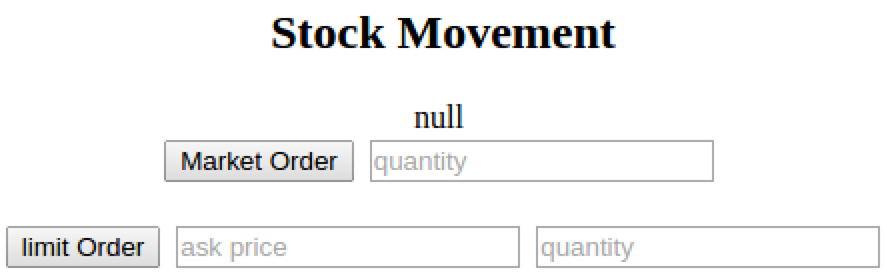
Currently we have used the jsp to provides a *bare-bone interface* for the basic happenings over the stock exchange and later we plan to use Twitter-Bootstrap to provide a fluid and intuitive interface.
Some snaps of current interface[2]

1. Login/ Register Page


Signup Details

Username:
Password:

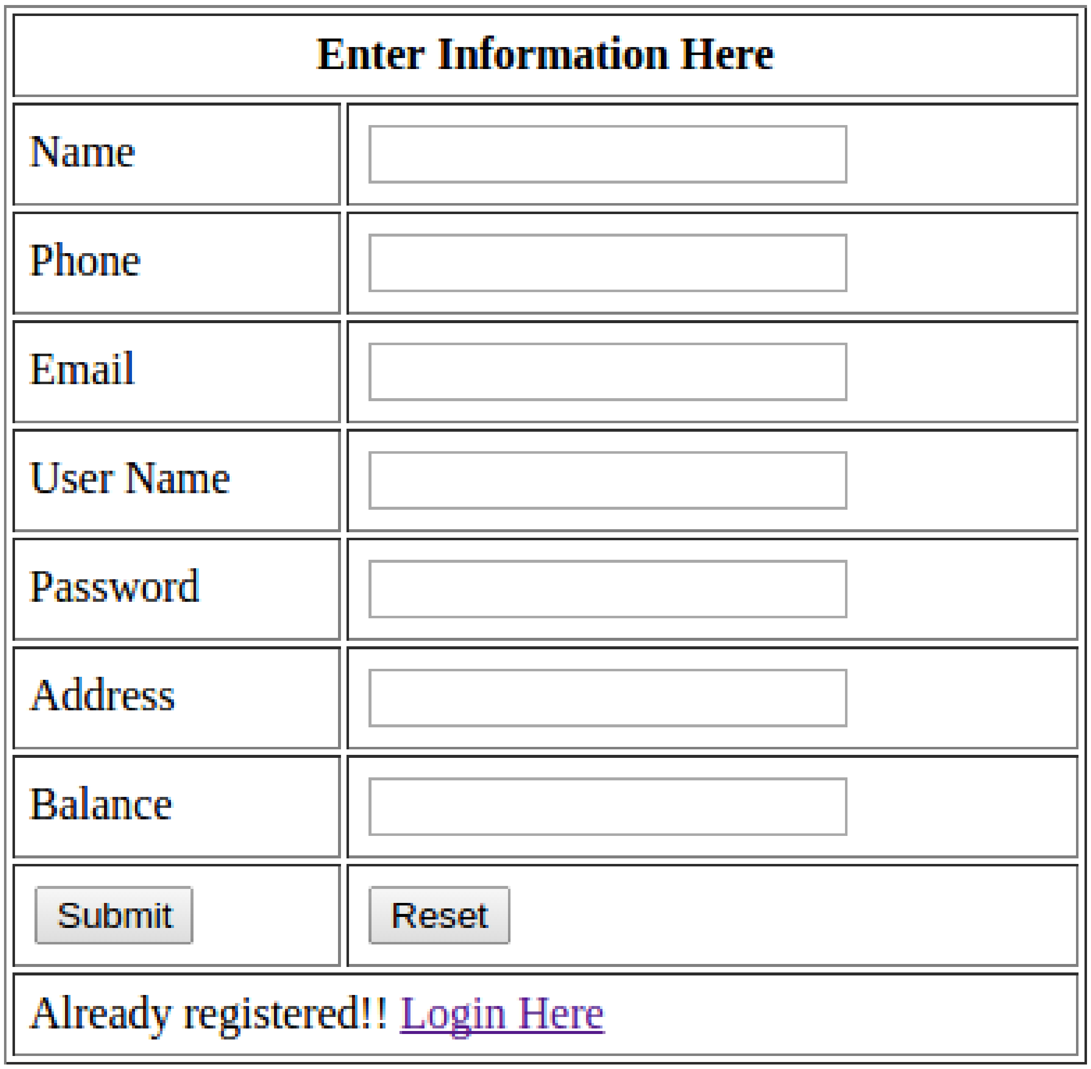
[2] We here show only few interfaces that we have designed because of space constraint



Stock Movement

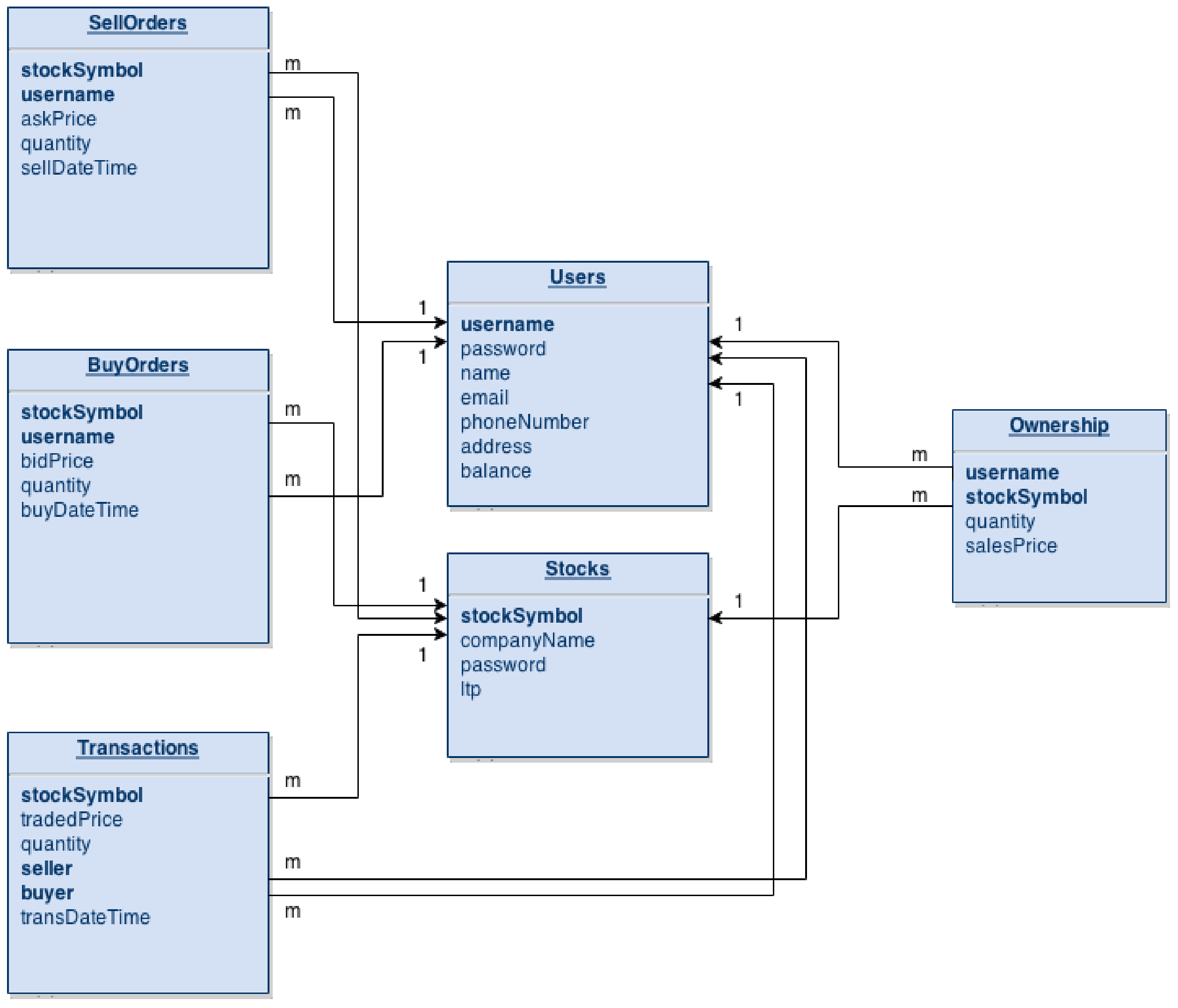
null

3. Edit Info Page



Enter Information Here	
Name	
Phone	
Email	
User Name	
Password	
Address	
Balance	
Submit	Reset
Already registered!! Login Here	

Database Schema



- The bold names indicates the primary key of the table
- And the arrows from one table to another marks the foreign key relationship in the direction of the arrow.