E-City Stocks
Design Document
For Software Engineering

Version 1.2

Revision History

Date	Version	Description	Author
8/11/08	1.0	Draft	Edwin Guzman
12/11/08	1.2	Draft	Edwin Guzman
14/11/08	1.2	Final	TEAM

E-City Stock Founders

Name	Role
Edwin Guzman	Project Manager, Software Developer
Stanis Laus Billy	Software Developer
Jose Diaz	Software Designer, Software Developer

Table of Contents

Design Document

1. Introduction

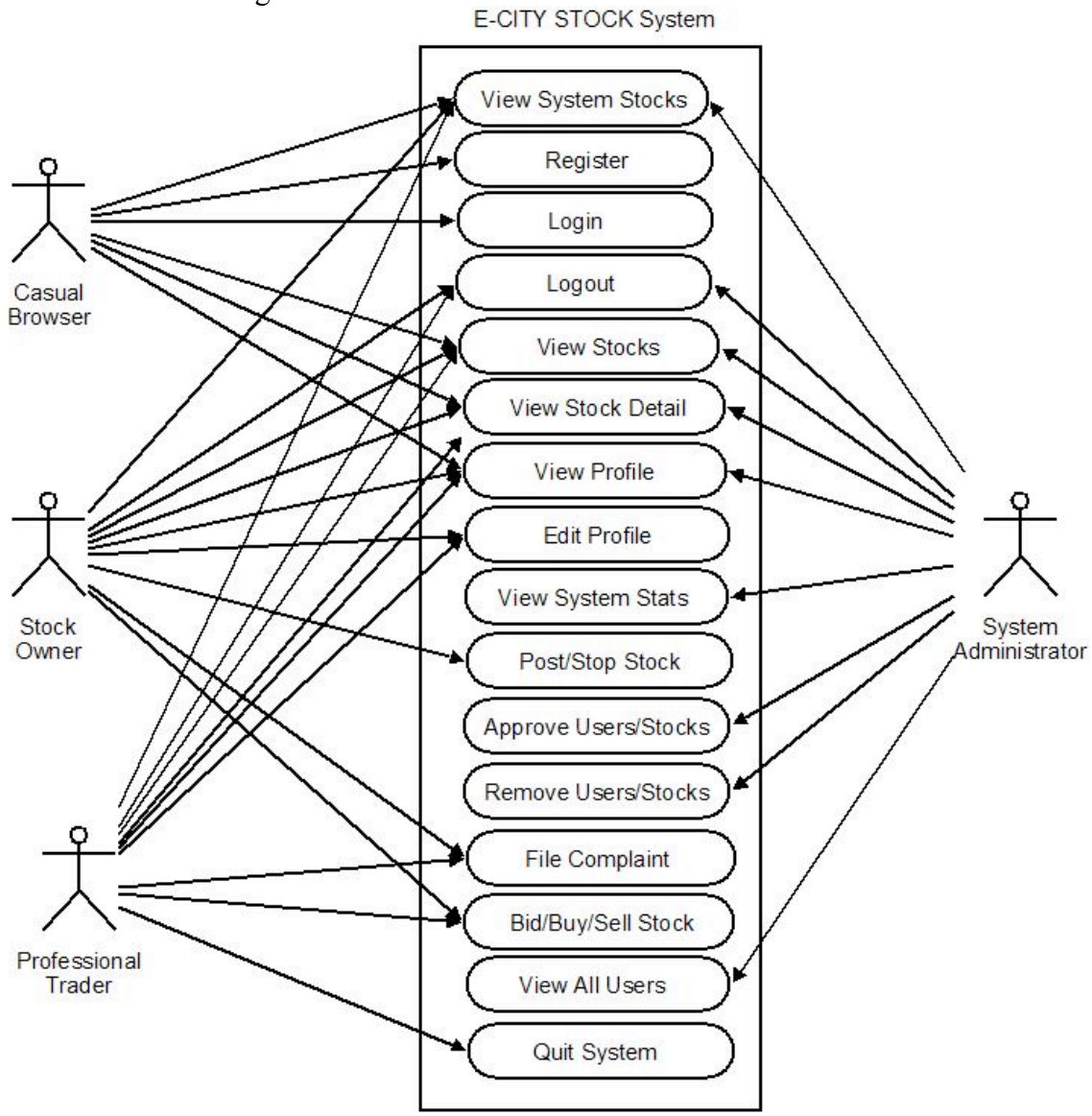
1.1 Overall Picture

E-City Stocks (ECS) is intended to be a system where users can deal and manage their stocks. Users, depending on their system status, have different actions (functions) that they can do regarding stocks and other users. E-City Stocks models real world stock marketing and user-to-stock and user-to-user interactions. This software is intended to run on a web browser, preferably Internet Explorer 6.0 or higher, that supports standard HTML, CSS 2.0, JavaScript, and XMLHTTPRequest.

This document is meant to show features of E-City Stocks. Not only will this document help and guide developers but it will also serve as a legal document for the prospective client. This Design Document explores in detail the structure and design of E-City Stocks through diagrams and intricate pseudo-code of the classes, models, and views in our system.

2. Overall Description

2.1 a. Use-Case Diagram



b. Use-Case Descriptions

View System Stock: Presents the user with a list of companies and brief information of their stocks.

Register: Allows a CB to apply to the system.

Login: Allows a CB to input his/her username and password and then have access to the system as a SO or a PT.

Logout: Allows a user in the system to exit from the system.

View Stocks: This presents a more detail list of all the stocks in a company.

View Stock Detail: This presents a detail description of each stock including high and low bidding and selling prices, as well as price history and owner.

View Profile: The user is presented with a Stock Owner, Professional Trader, or his own profile. If a user is viewing his own profile, different functionalities will be available to that user depending on his user type.

Edit Profile: A Stock Owner or Professional Trader can edit his/her own information that appears on the profile.

View System Statistics: This presents the SA with all current data of the systems including all users and all stock.

Post New Stock: Stock Owner can request a new stock to be posted. System Administrator reviews the request and either approves or rejects the request. If approved, a new stock will be posted under the ownership of the Stock Owner.

Stop Stock: Stock Owner can request to stop one of his stock. The System Administrator then approves or rejects the request. If approved, the stock will no longer be in the system.

Approve New User: Consists of a list of users who have register into the ECS system and are awaiting approval from the System Administrator.

Remove User: Consists of a list of ECS users who have been flagged, have run out of money, or have quit the system and await removal by the System Administrator.

Approve New Stock: Consists of a list of new stocks that are awaiting approval into the ECS system from the System Administrator.

Remove Stock: Consists of a list of stocks that have been stopped by their respective Stock Owner and await removal by the System Administrator.

File Complaint: Users can make complaints against other users depending on their user types. Stock Owners can only file complaints against Professional Traders. Professional Traders can file complaints against both Stock Owners and other Professional Traders. Once complaints are filed, they're queued in a list that will be reviewed by the System Administrator. If a user has at least 3 complaint tags, their account may be frozen.

Bid: Stock Owners and Professional Traders are allowed to place bids on stocks.

Buy: Stock Owners and Professional Traders are allowed to buy stocks.

Sell: Stock Owners and Professional Traders can sell their own stock but a Professional Trader cannot sell his stock if he does not have enough money.

View All Users: This displays all the users in the system for the SA to check on.

Quit System: A Professional Trader can quit the system, thus deleting his account. Once the Professional Trader's quit system request has been queued, the System Administrator reviews the request and either approves or rejects the request. If approved, the Professional Trader's account will be deleted, along with all of his assets recorded in the system.

2.2 Collaboration Diagrams

A. View System Stocks

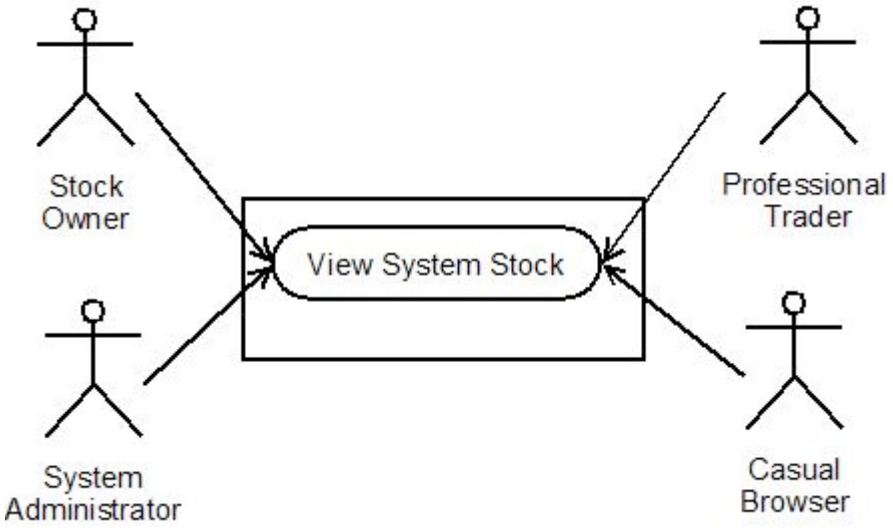


FIGURE A.1
View System Stocks Use-Case Diagram

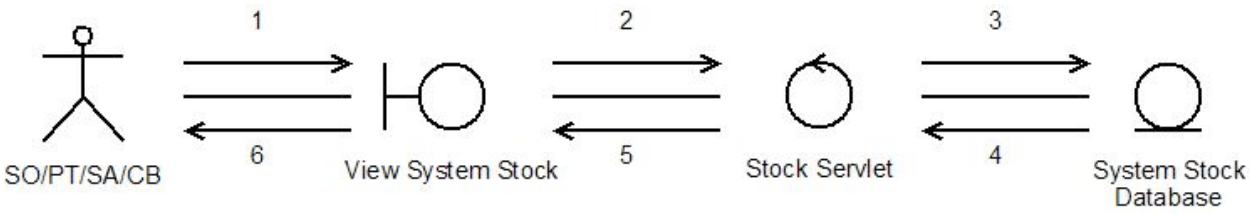


FIGURE A.2
View System Stock Collaboration Diagram

1. Any user clicks on the “View System Stocks” link.
2. Information is passed to the Stock Servlet.
3. Data is retrieve from the System Stock Database.
4. The database returns the data to the Stock Servlet.
5. The data is passed to the View System Stock interface and is shown to the user.

B. Register

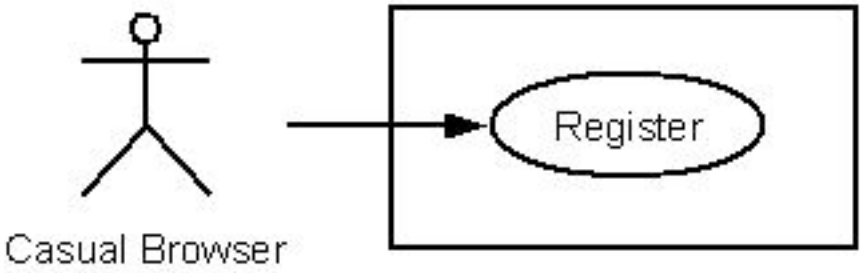


FIGURE B.1
Register Use-Case Diagram

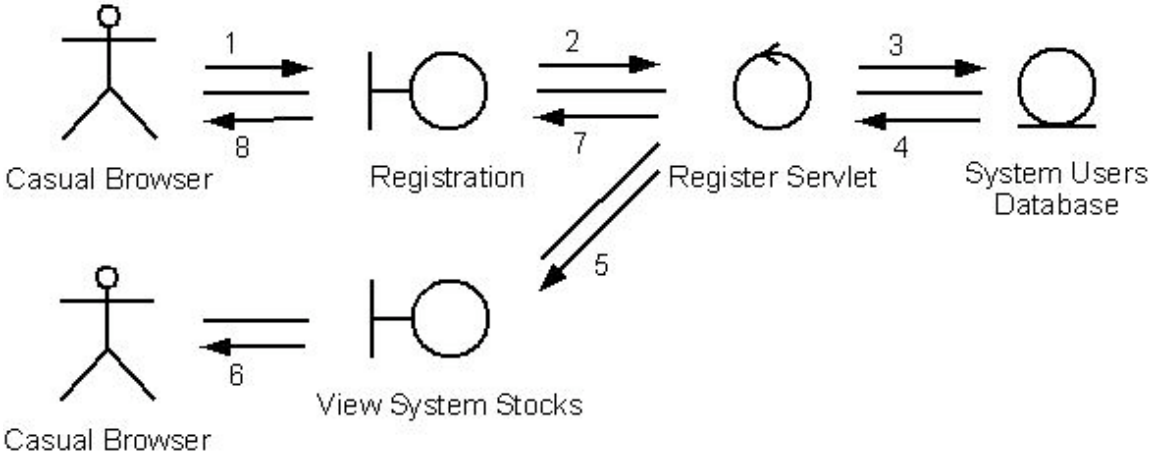


FIGURE B.2
Register Collaboration Diagram

1. The Casual Browser enters information and clicks “Submit”.
2. The information is processed in the Register Servlet.
3. The information is passed to the System Users Database and a new user is created.
4. The information is passed back to the Register Servlet.
5. The information is then passed to View System Stocks interface.
6. The Casual Browser is told that he has to wait for approval from the System Administrator.

Exceptional Case:

1. The Casual Browser enters information and clicks “Submit”.
2. The information is processed in the Register Servlet.
7. The information given has a mistake and is returned to the Registration interface.
8. The Casual Browser is told that his information is incorrect and that he has to fix his errors.

C. Login

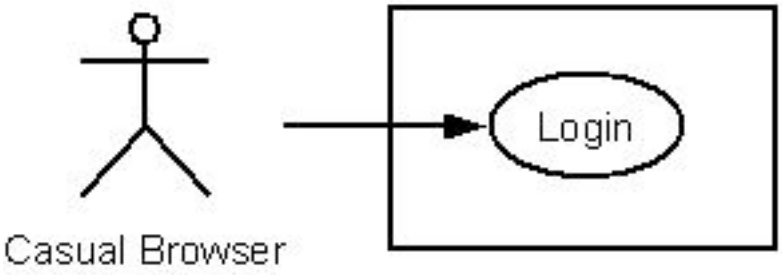


FIGURE C.1
Login Use-Case Diagram

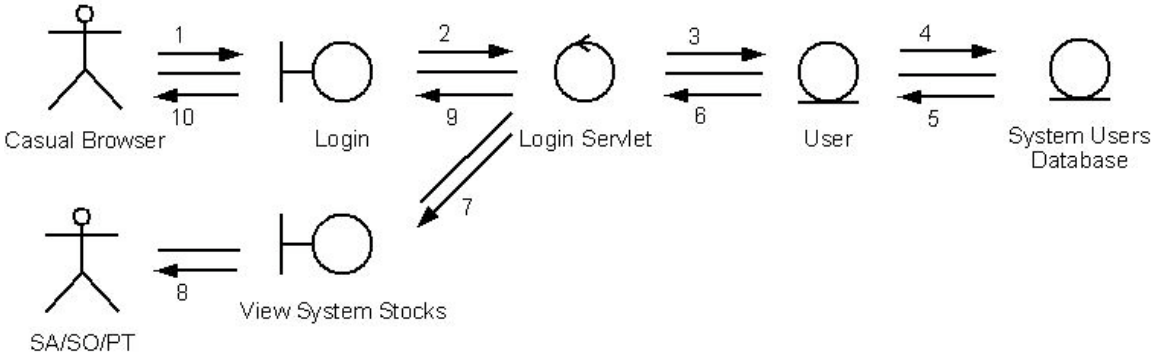


FIGURE C.2
Login Collaboration Diagram

1. The Casual Browser enters username and password and clicks “Login”.
2. The information is processed in the Login Servlet.
3. The information is then passed to the User class.
4. The information is passed to the System Users Database.
5. The information is correct and is then passed back to the User class.
6. The information is passed to the Login Servlet.
7. The information is then passed to the corresponding homepage (View System Stocks interface) of the user.
8. The corresponding homepage is shown to each user.

Exceptional Case:

1. The Casual Browser enters username and password on Login interface.
2. The information is processed in the Login Servlet.
3. The information is then passed to the User class.
4. The information is passed to the System Users Database.
5. The information is wrong and is then passed back to the User class.
6. The information is passed to the Login Servlet.
9. The information is passed back to the Login interface.
10. The Casual Browser is told that the information he submitted is wrong.

D. Logout

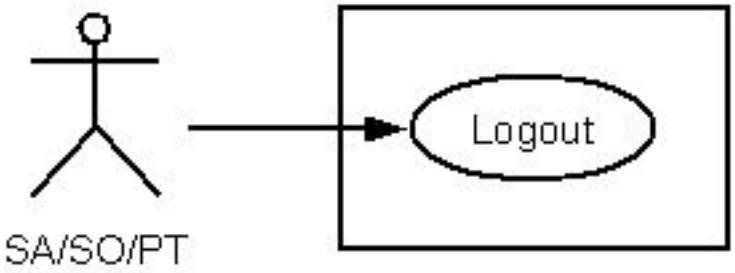


FIGURE D.1
Logout Use-Case Diagram

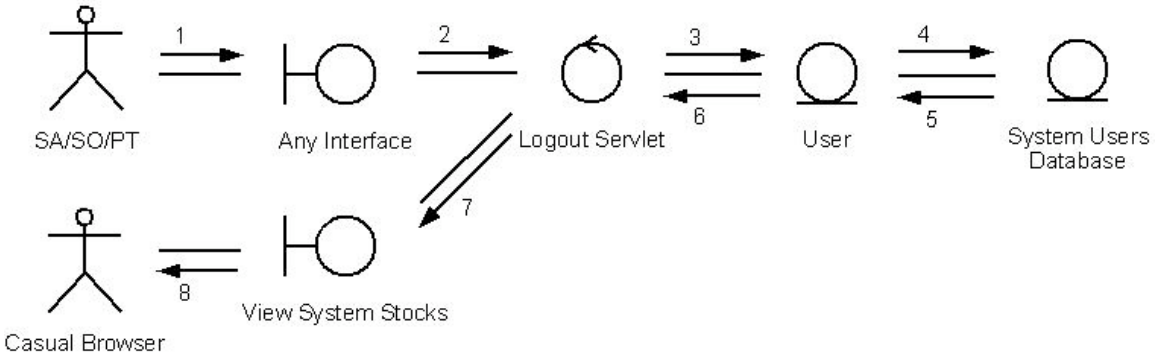


FIGURE D.2
Logout Collaboration Diagram

1. SA, SO, or PT click on logout on any interface.
2. The information is processed in the Logout Servlet.
3. The information is then passed to the User class where the user is then logged out.
4. The information is passed to the System Users Database.
5. The Database returns the information of the user's logged out status.
6. The information is processed by the Logout Servlet.
7. The logged out status is given to the homepage (View System Stocks) interface.
8. The homepage tells the user that they logged out successfully.

E. View Stocks

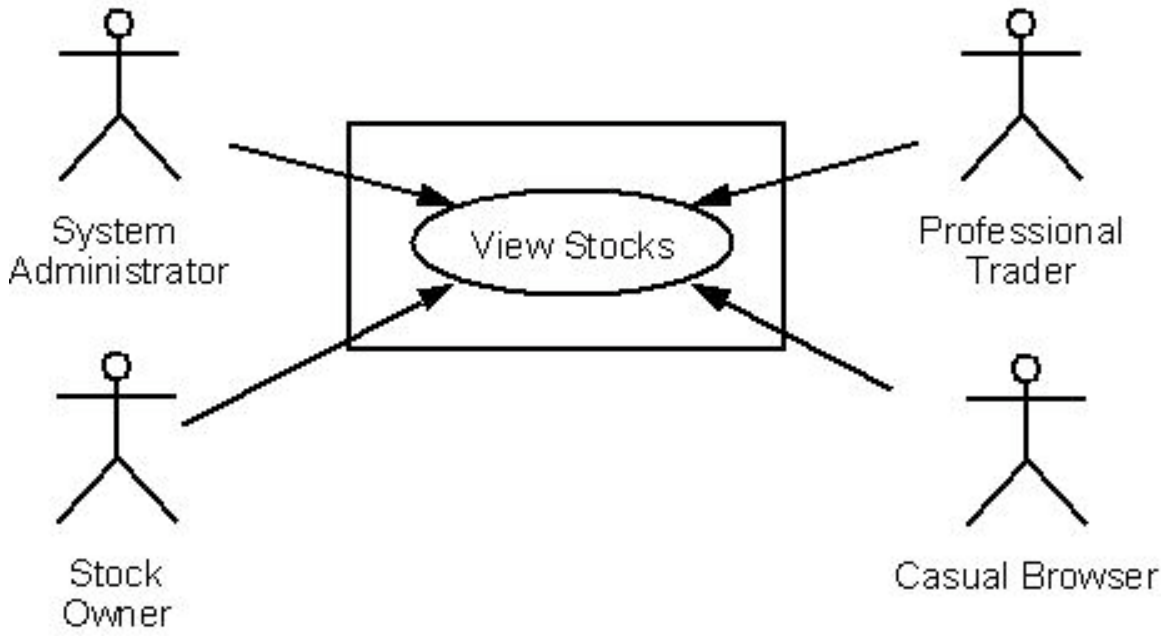


FIGURE E.1
View Stocks Use-Case Diagram

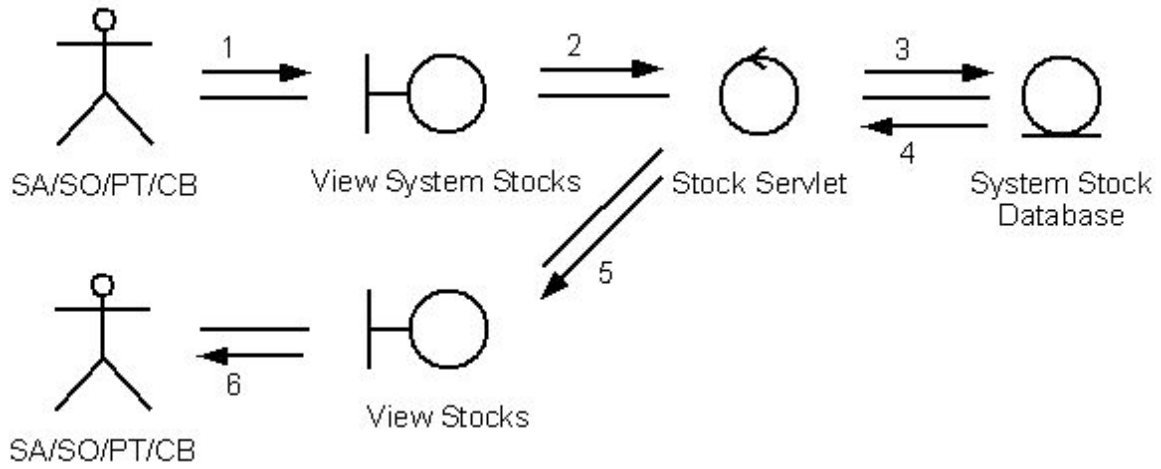


FIGURE E.2
View Stocks Collaboration Diagram

1. Any user clicks on the stock on the View System Stocks page.
2. The stock clicked on is passed to the Stock Servlet.
3. The stock is then passed to the System Stock Database.
4. The stock is retrieved and passed back to the Stock Servlet.
5. The selected stock passes on to the View Stock interface.
6. The stock is shown to the user.

F. View Stock Detail

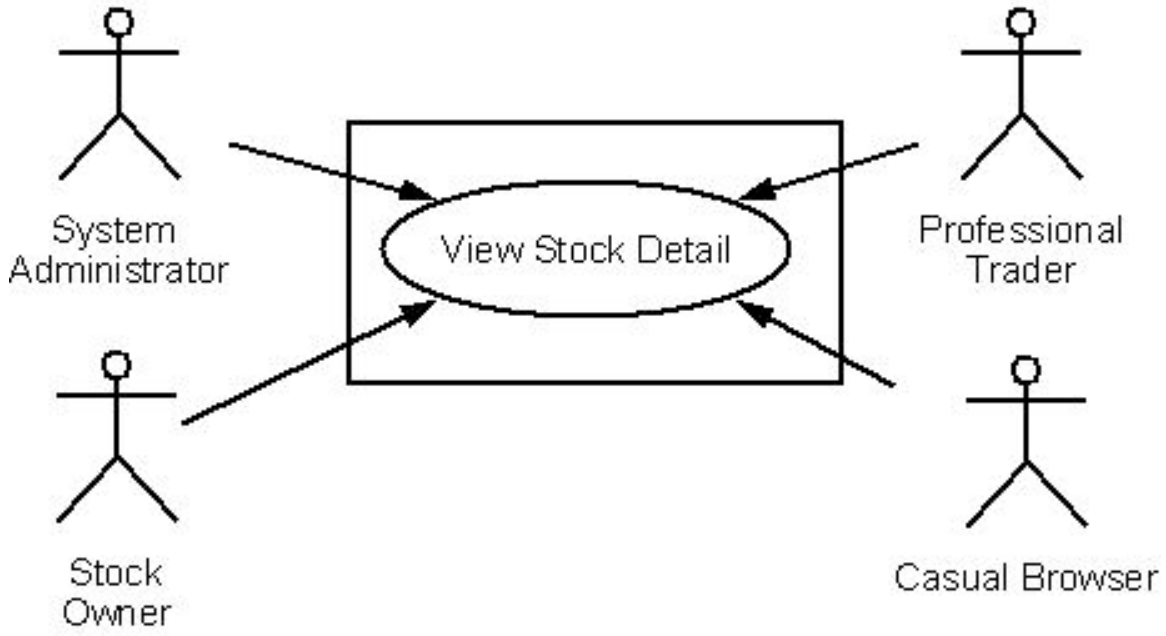


FIGURE F.1
View Stock Detail Use-Case Diagram

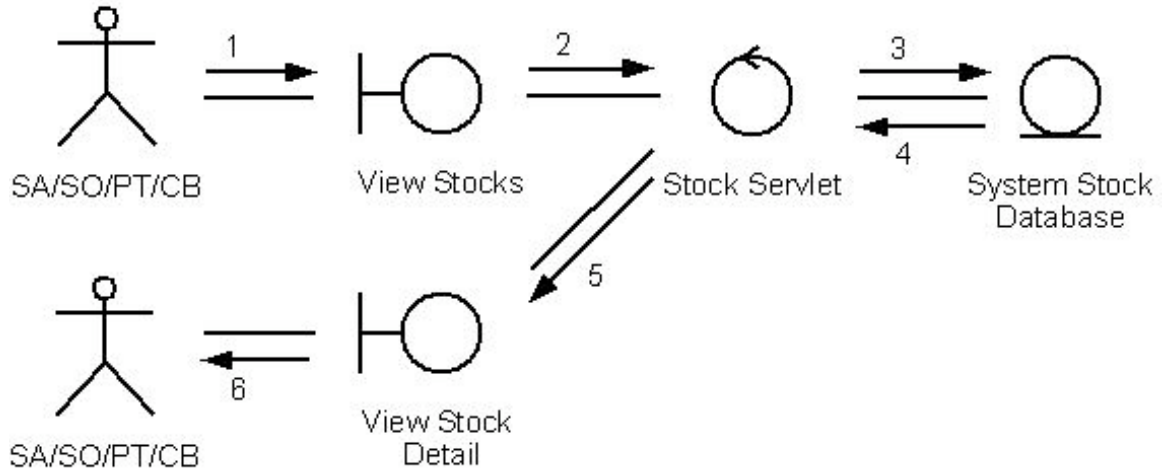


FIGURE F.2
View Stock Detail Collaboration Diagram

1. Any use clicks on the “View Stock Detail” on the View Stocks interface.
2. The information selected is passed to the Stock Servlet.
3. The information for the selected stock is gathered.
4. The database returns the information of the selected stock.
5. The information is passed to the View Stock Detail interface.
6. The selected stock is shown to the user.

G. View Profile

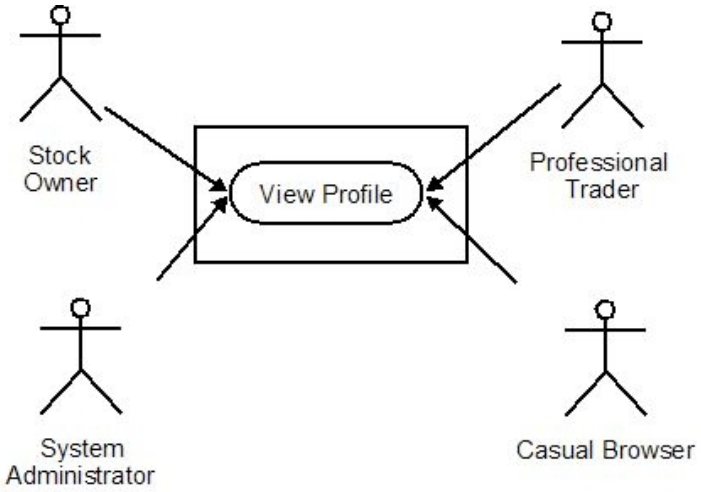


FIGURE G.1
View Profile Use-Case Diagram

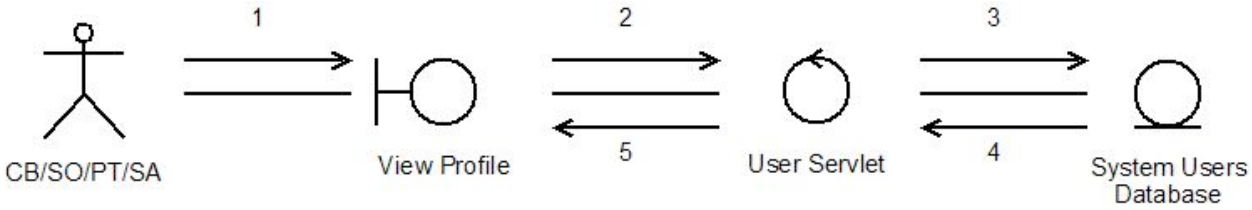


FIGURE G.2
View Profile Collaboration Diagram

1. Any user clicks on the “View Profile” link.
2. Pass information to the User Servlet.
3. The Information is passed to the System Users Database.
4. Pass information back to the User Servlet.
5. The information is passed to the View Profile interface which is displayed to the user.

H. Edit Profile

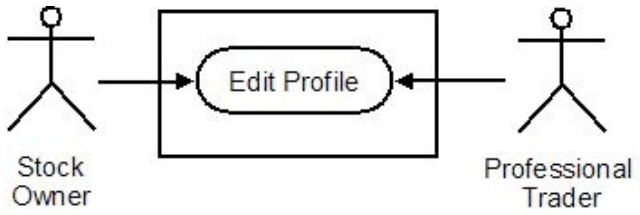


FIGURE H.1
Edit Profile Use-Case Diagram

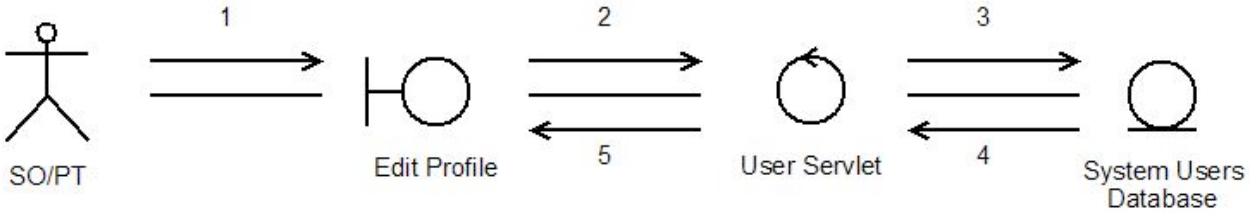


FIGURE H.2
Edit Profile Collaboration Diagram

1. SO/PT visits his/her own profile page and they click on “Edit”.
2. Pass information to the User Servlet.
3. Pass information to System Users Database.
4. Pass information back to the User Servlet.
5. Pass information to Edit Profile interface displaying a message to the user that profile has been updated successfully.

I. View System Statistics

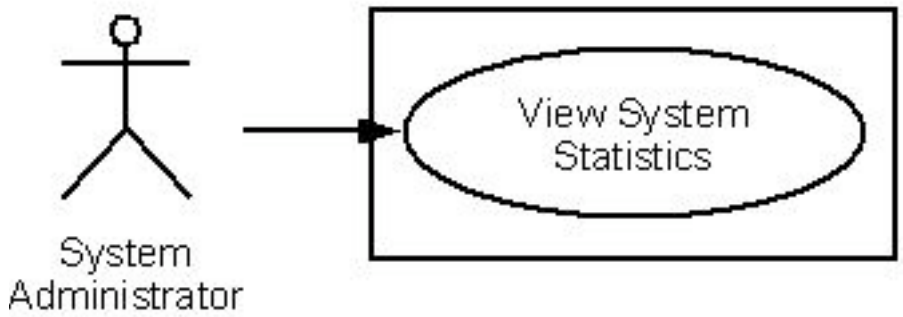


FIGURE I.1
View System Statistics Use-Case Diagram

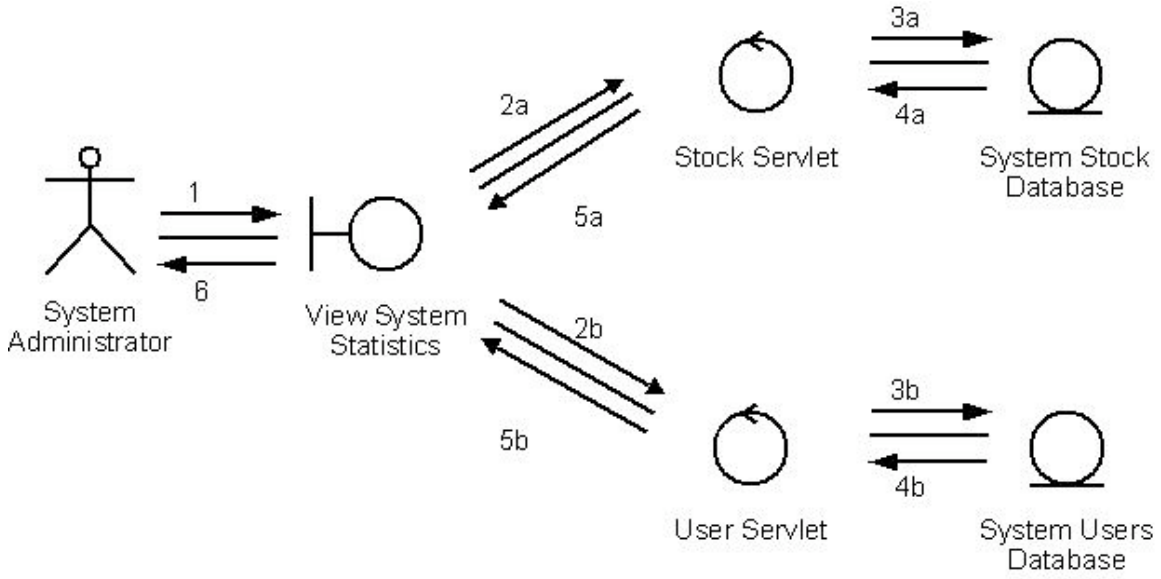


FIGURE I.2
View System Statistics Collaboration Diagram

1. The System Administrator clicks on “View System Statistics”.
2a. The information is passed to the Stock Servlet.
3a. The information is then passed to the System Stock Database.
4a. The retrieved information is passed back to the Stock Servlet.
5a. The information is then passed to the View Statistics interface.
2b. The information is passed to the User Servlet.
3b. The information is then passed to the System Users Database.
4b. The retrieved information is passed back to the User Servlet.
5b. The information is then passed to the View Statistics interface.
6. The system statistics are now shown to the System Administrator.

J. Post New Stock

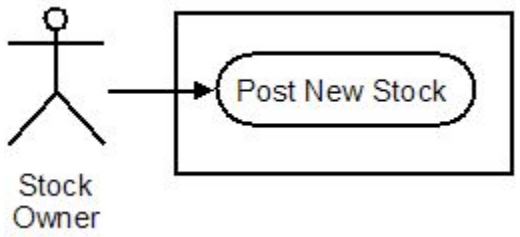


FIGURE J.1
Post New Stock Use-Case Diagram

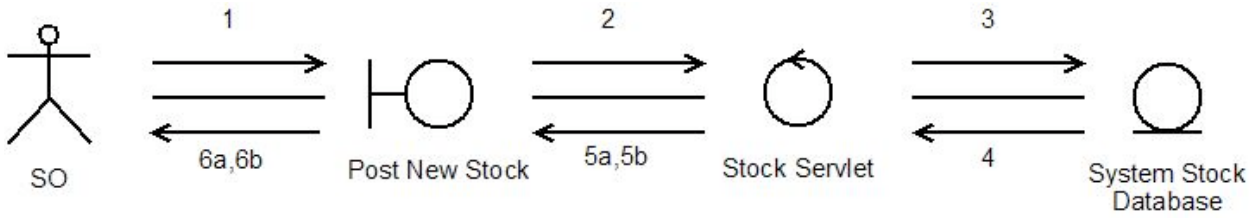


FIGURE J.2
Post New Stock Collaboration Diagram

1. SO clicks on the “Post New Stock” link.
2. Pass information to the Stock Servlet.
3. Pass information to the System Stock database.
4. Information is passed back to the Stock Servlet.
5a. The information is given to the Post New Stock interface.
6a. The interface tells the user that they must await approval from the System Administrator.

Exceptional case:
5b. If the user doesn’t specify an initial price and/or negative or 0 number of stocks, the Servlet generates an error and passes the error information to the GUI.
6b. An error is displayed to the user.

K. Stop Stock

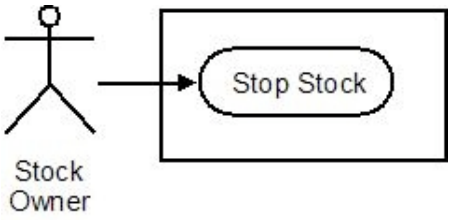


FIGURE K.1
Stop Stock Use-Case Diagram

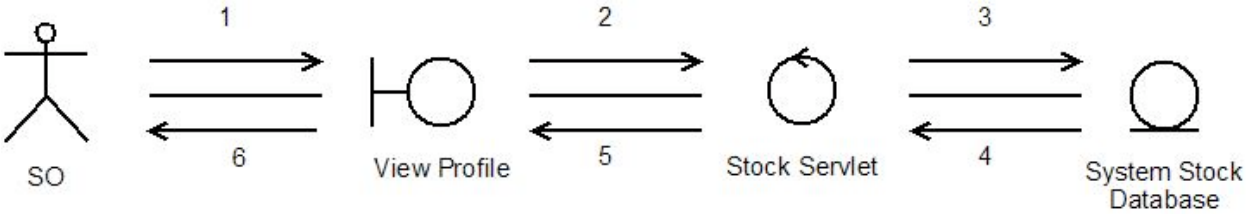


FIGURE K.2
Stop Stock Collaboration Diagram

1. SO visits his own profile – a list of the stocks he owns is displayed along with an option to stop individual stocks.
2. Pass information to the Stock Servlet.
3. Pass information to System Stock Database, flagging the stock.
4. Pass information back to Stock Servlet as confirmation.
5. Pass information to View Profile – for SO – interface.
6. Display a message to the Stock Owner that his stock has been queued. If SA has approved the stop request, display a message that his stock has been stopped.

L. Approve New User

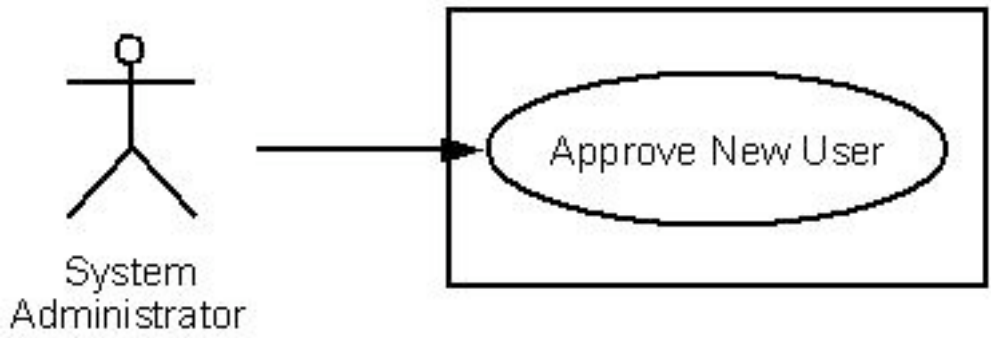


FIGURE L.1
Approve New User Use-Case Diagram

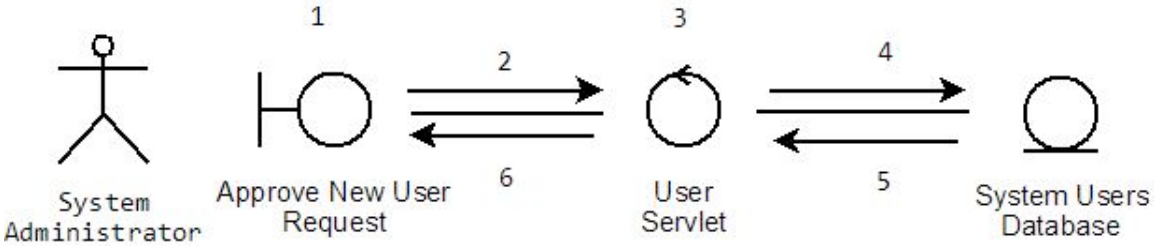


FIGURE L.2
Approve New User Collaboration Diagram

1. User Interface displays all the new users awaiting approval from SA. SA can select an individual or multiple user requests.
2. Transmit the list of selected user to the server.
3. The User Servlet parses the user list.
4. Insert approval for selected users into the System Users Database.
5. Retrieve any pending user approval in the system.
6. Returns an updated list of pending user approval.

M. Remove User

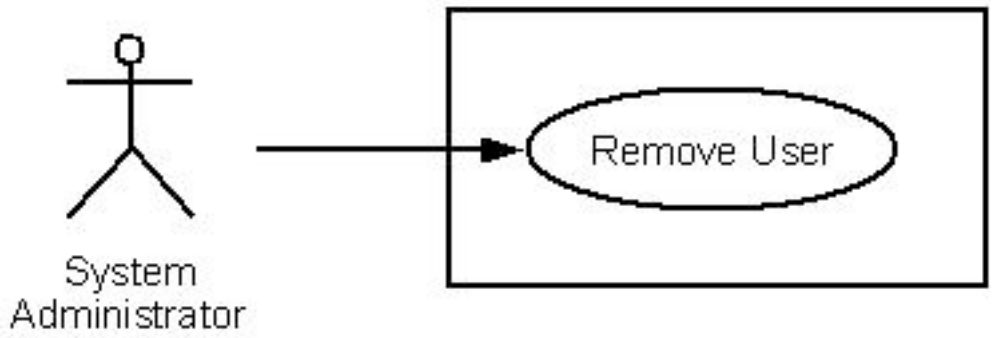


FIGURE M.1
Remove User Use-Case Diagram

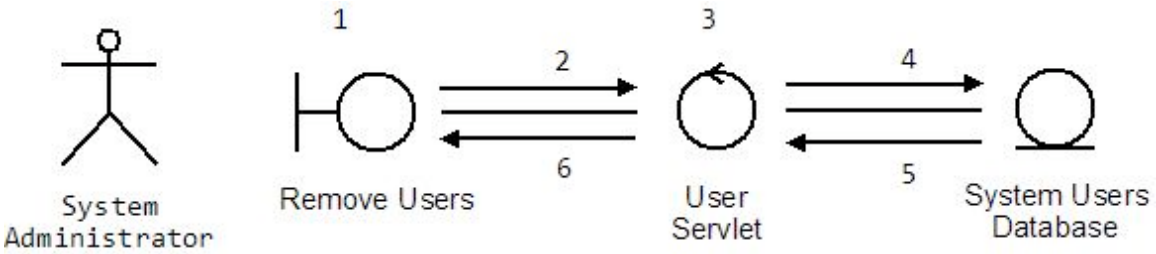


FIGURE M.2
Remove User Collaboration Diagram

1. User Interface displays all the users to be removed from the system by the SA. SA can select individual or multiple users from the list.
2. Transmit the list of selected user to the server.
3. The User Servlet parses the user list.
4. Delete selected users from the System Users Database.
5. Retrieve any pending user removal in the system.
6. Returns an updated list of pending user removal.

N. Approve New Stock

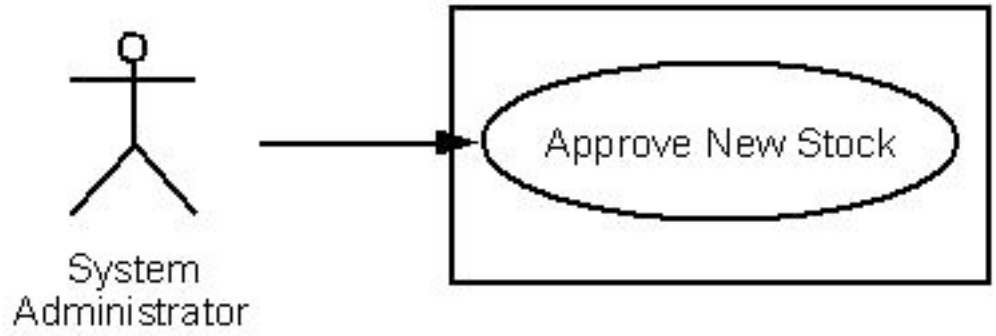


FIGURE N.1
Approve New Stock Use-Case Diagram

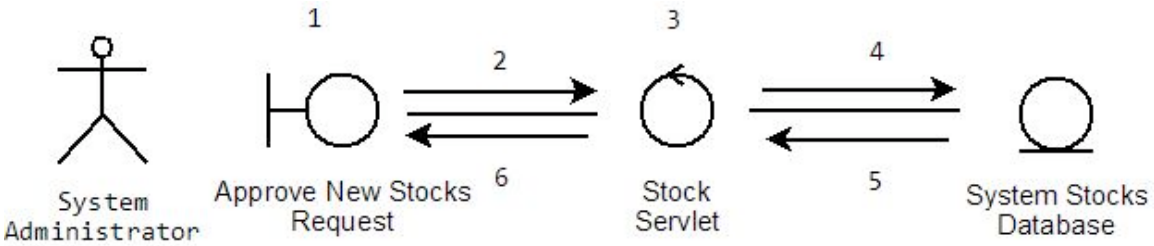


FIGURE N.2
Approve New Stock Collaboration Diagram

1. User Interface displays all the new stocks awaiting approval from SA. SA can select an individual or multiple stock requests.
2. Transmit the list of selected stocks to the server.
3. The Stock Servlet parses the stock list.
4. Insert approval for selected stocks into the System Stocks Database.
5. Retrieve any pending stock approval in the system.
6. Returns an updated list of pending stock approval.

O. Remove Stock

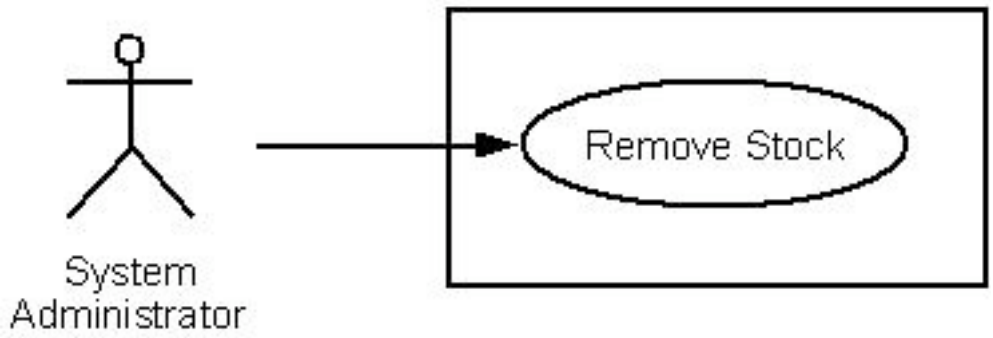


Figure O.1
Remove Stock Use-Case Diagram

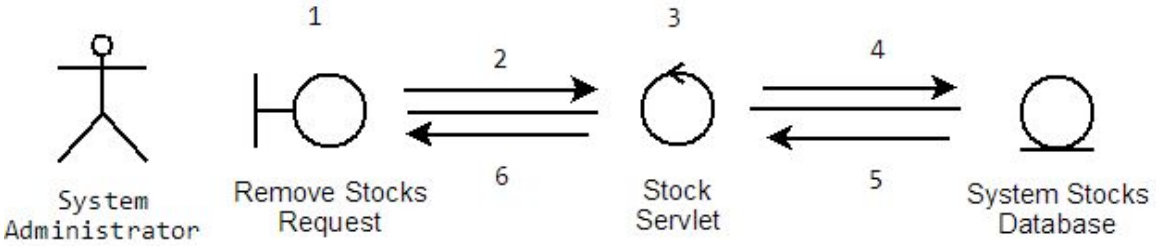


FIGURE O.2
Remove Stock Collaboration Diagram

1. User Interface displays all the stocks that have been stopped by their SO and are awaiting removal from the system by the SA. SA can select individual or multiple stocks from the list.
2. Transmit the list of selected stocks to the server.
3. The Stock Servlet parses the stock list.
4. Delete selected stocks from the System Stocks Database.
5. Retrieve any pending stock removal in the system.
6. Returns an updated list of pending stock removal.

P. File Complaint

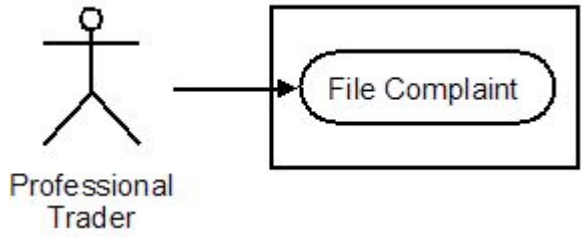


FIGURE P.1
PT File Complaint Use-Case Diagram

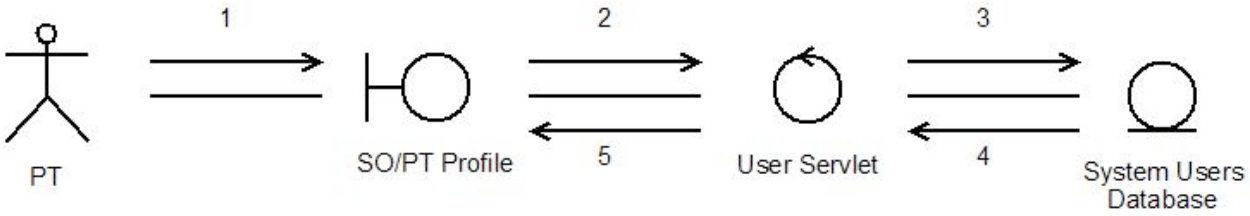


FIGURE P.2
PT File Complaint Collaboration Diagram

1. PT visits a SO/PT profile page other than his own.
2. PT clicks on “File Complaint” button. Information is passed to the User Servlet.
3. Pass information to the database, flagging the user complained against.
4. Pass information back to User Servlet.
5. Display message on GUI for successful complaint.

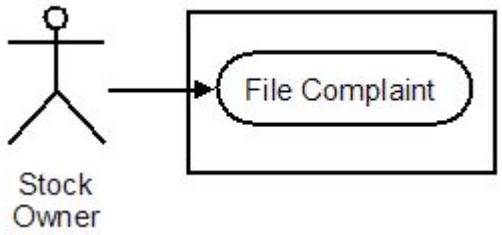


FIGURE P.3
SO File Complaint Use-Case Diagram

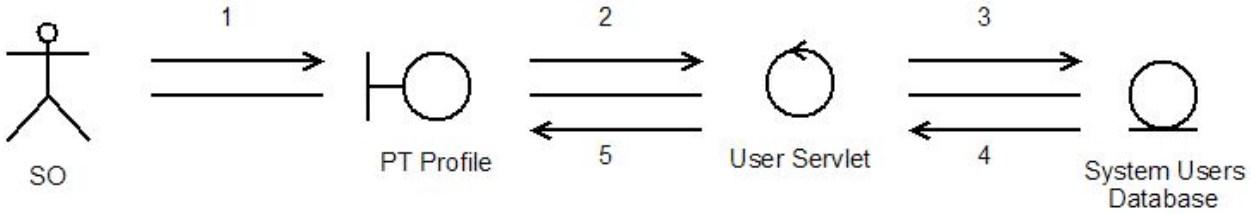


FIGURE P.4
SO File Complaint Collaboration Diagram

1. SP visits a PT profile page.
2. SO/PT clicks on “File Complaint” button. Information is passed to the User Servlet.
3. Pass information to the database, flagging the user complained against.
4. Pass information back to User Servlet.
5. Display message on GUI for successful complaint.

Q. Bid

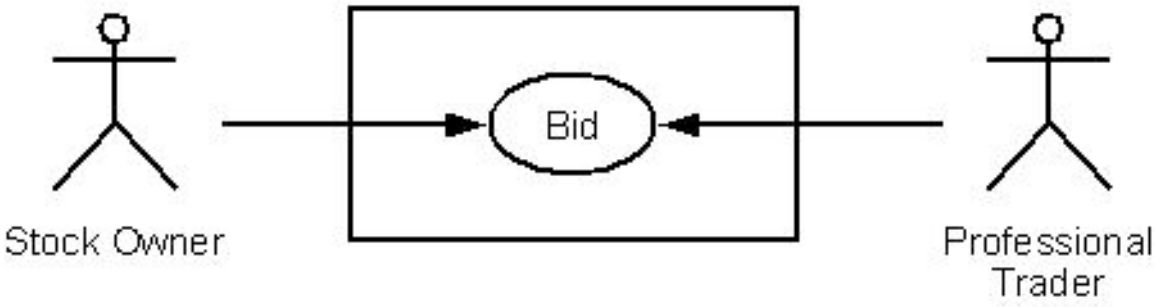


FIGURE Q.1
Bid Use-Case Diagram

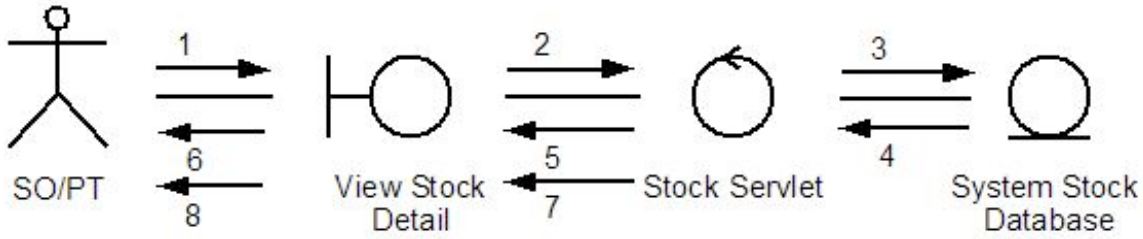


FIGURE Q.2
Bid Collaboration Diagram

1. Stock Owner or Professional Trader click on “Bid”.
2. The stock that is being bid on is passed to the Stock Servlet.
3. The stock is processed as having a bid and is passed to the System Stock Database.
4. The stock is returned to Stock Servlet.
5. The data is passed back to the View Stock Detail interface.
6. The user now sees the bid he placed on the stock.

Exceptional Case:

7. The information passed to the Stock Servlet is wrong. The user placed a bid lower than the current bid. An error is passed back.
8. The View Stock Detail interface warns the SO or PT that they must place a higher bid on the stock.

R. Buy

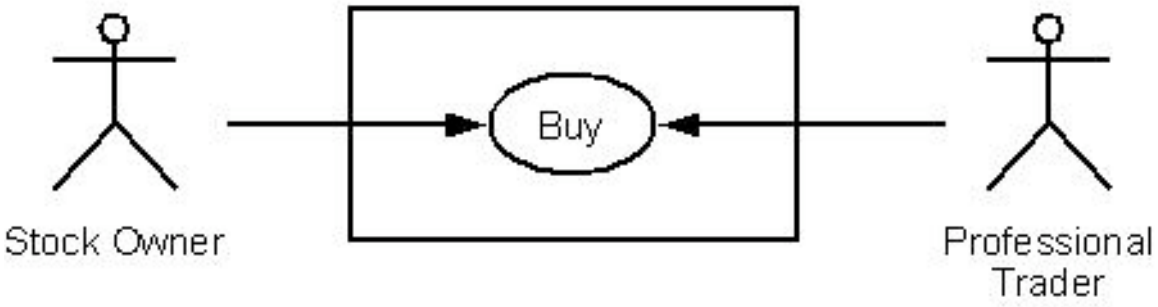


FIGURE R.1
Buy Use-Case Diagram

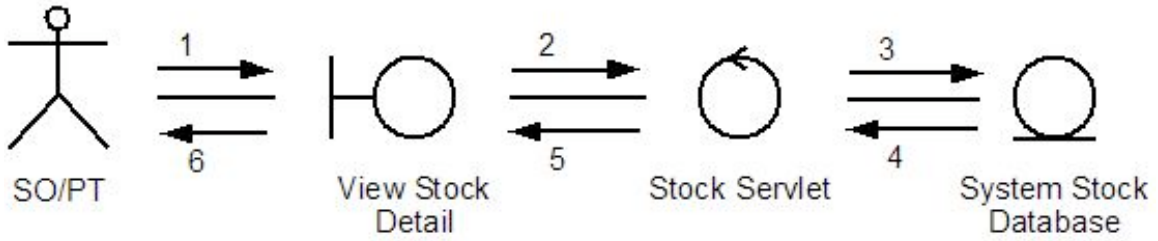


FIGURE R.2
Buy Collaboration Diagram

1. Stock Owner or Professional Trader clicks on “Buy”.
2. The stock that is being bought is passed to the Stock Servlet.
3. The stock is processed as being bought and is passed to the System Stock Database.
4. The stock is returned to Stock Servlet.
5. The data is passed back to the View Stock interface.
6. The user waits until the owner approves.

S. Sell

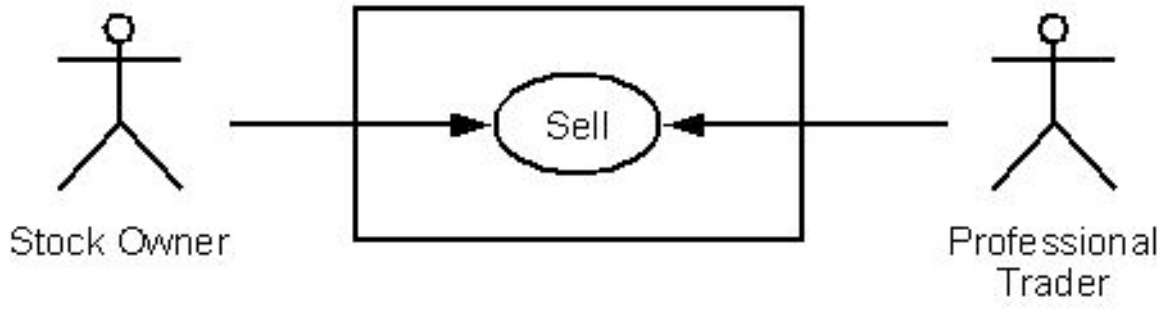


FIGURE S.1
Sell Use-Case Diagram

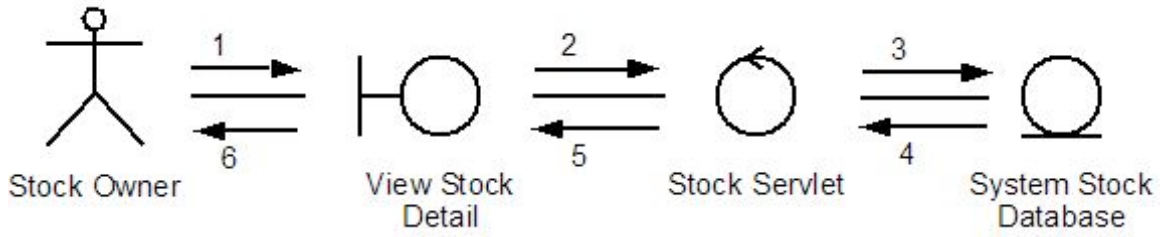


FIGURE S.2
Stock Owner Sell Collaboration Diagram

1. Stock Owner clicks on “Sell”.
2. The stock being sold is passed to the Stock Servlet.
3. The stock is processed as being on sale and is passed to the System Stock Database.
4. The stock is returned to Stock Servlet.
5. The data is passed back to the View Stock interface.
6. The Stock Owner is now can now see his stock on sale.

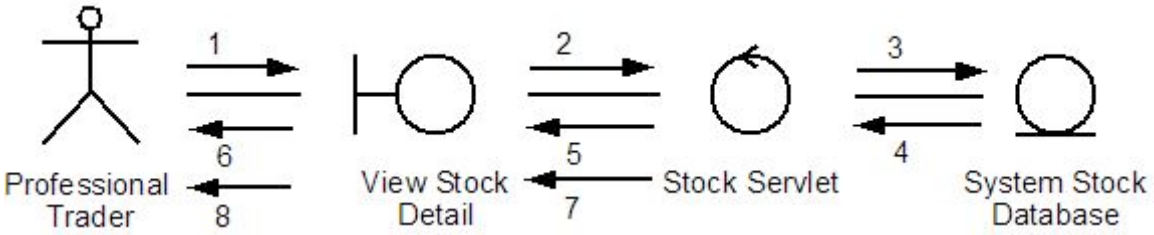


FIGURE S.3
Professional Trader Sell Collaboration Diagram

1. Professional Trader clicks on “Sell”.
2. The stock being sold is passed to the Stock Servlet.
3. The stock is processed as being on sale and is passed to the System Stock Database.
4. The stock is returned to Stock Servlet.
5. The data is passed back to the View Stock interface.
6. The Professional Trader is now can now see his stock on sale.

Exceptional Case:

1. Professional Trader clicks on “Sell”.
2. The stock being sold is passed to the Stock Servlet.
7. The Stock Servlet returns the stock back to the View Stock interface because the Professional Trader does not have enough money.
8. The Professional Trader is told that he does not have enough money to sell his stock.

T. View All Users

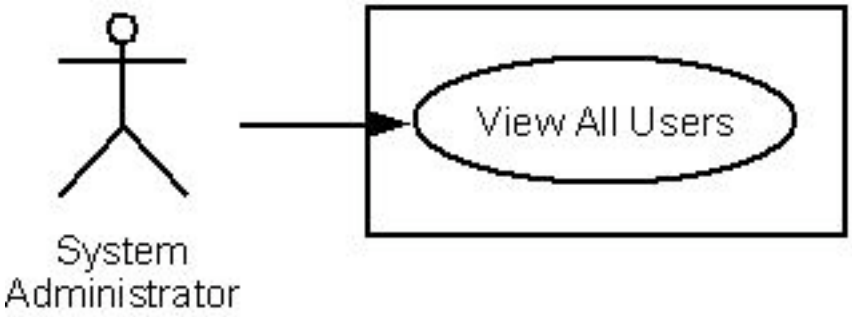


FIGURE T.1
View All Users Use-Case Diagram

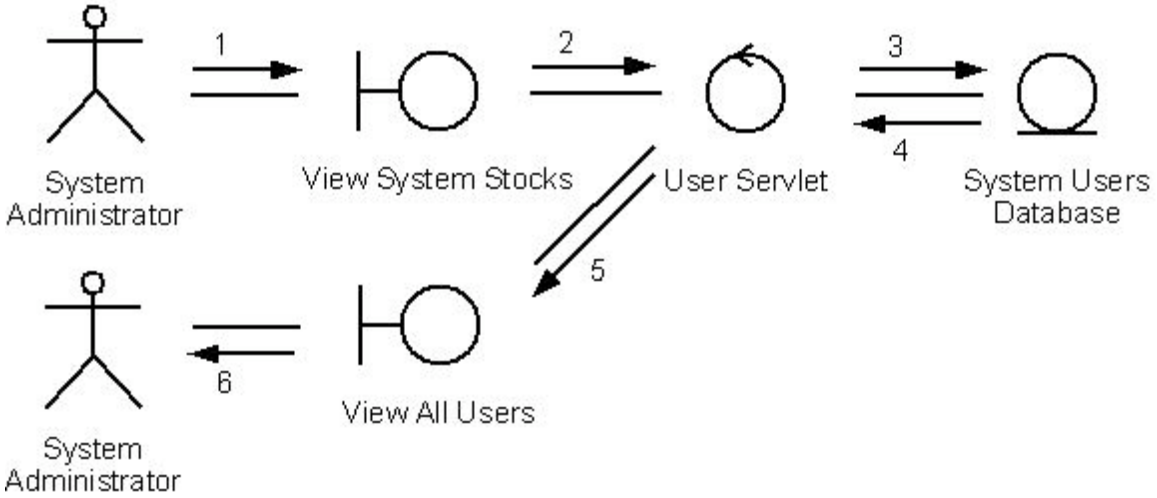


FIGURE T.2
View All Users Collaboration Diagram

1. System Administrator clicks on “View All Users”.
2. The information is passed to the User Servlet.
3. All the users are processed and gathered.
4. The database returns all the users in the system.
5. User Servlet processes the information gathered and sends it to the View All Users interface.
6. The System Administrator is shown a list of all the users in the system and a link to their profiles.

U. Quit System

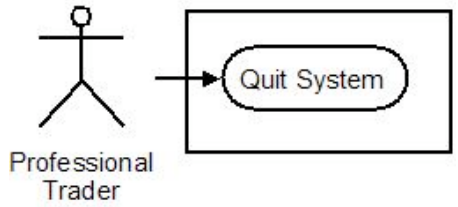


FIGURE U.1
Quit System Use-Case Diagram

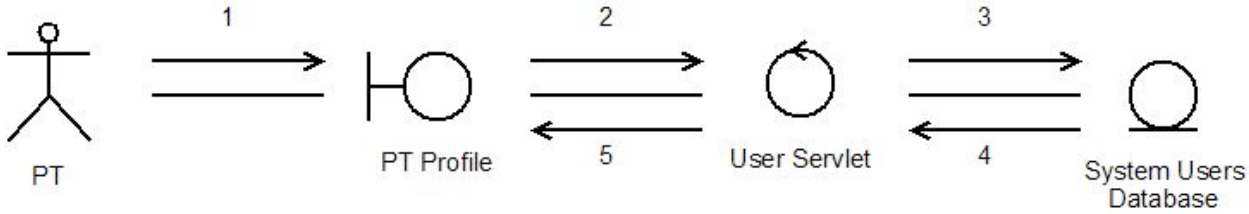
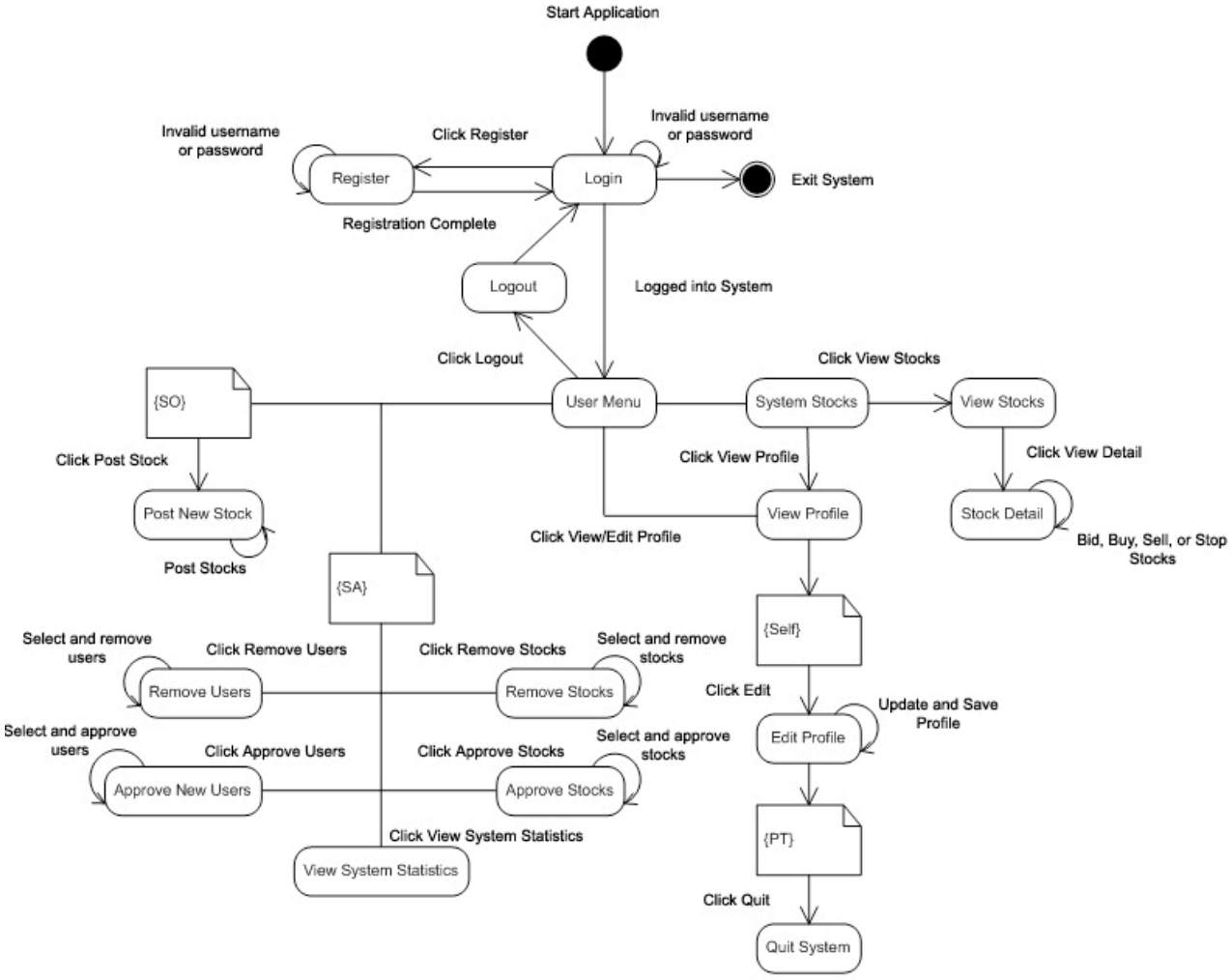


FIGURE U.2
Quit System Collaboration Diagram

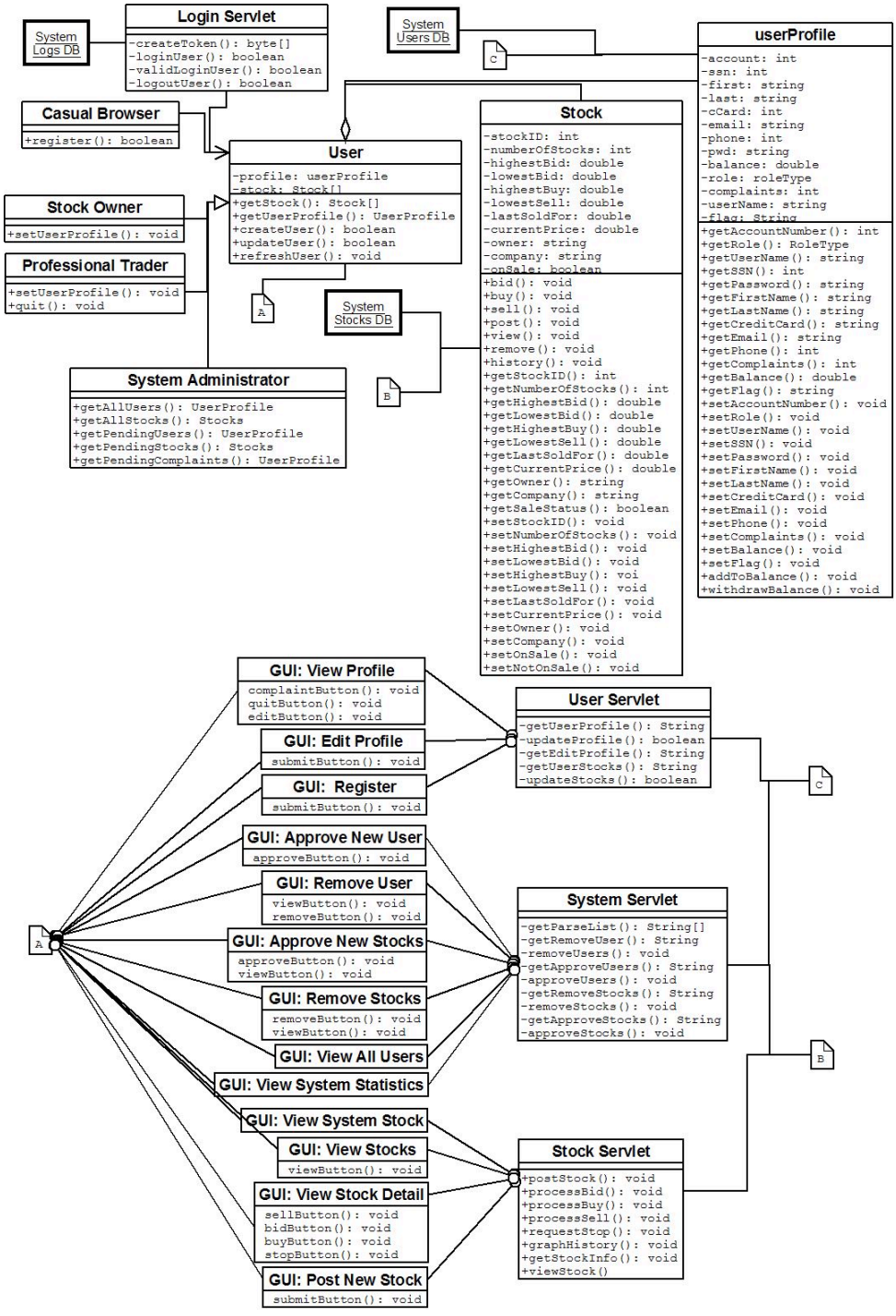
1. PT clicks the “Quit from System” button.
2. Pass information to the User Servlet.
3. Pass information to the System Users Database, flagging the user.
4. Pass information back to the User Servlet.
5. Display message on PT Profile for successful quit request.

2.3 ESC System State Diagram

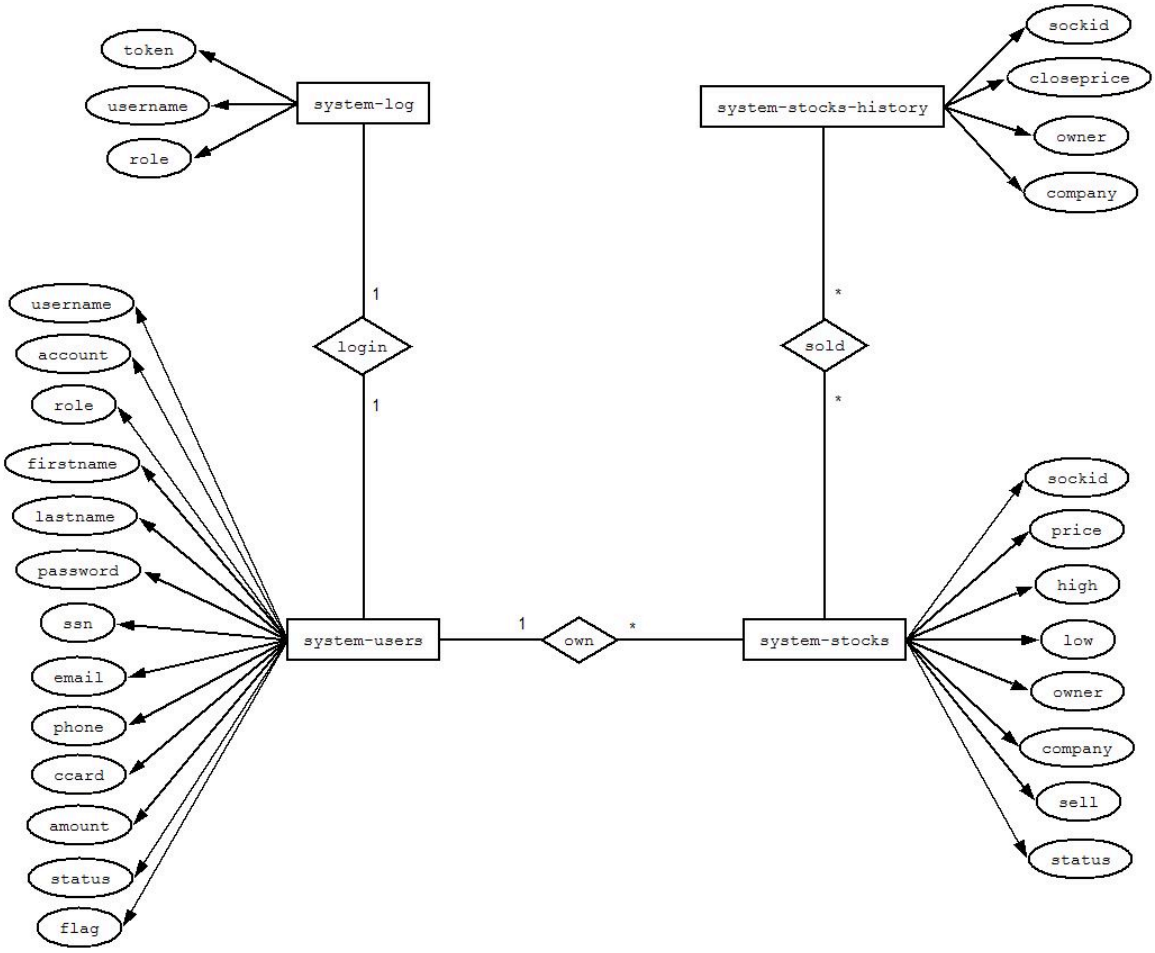


3. ER Diagrams

3.3 ER Class Diagrams



3.2 ER Database Diagram



4. Detailed Pseudo-Code

4.1 Classes

a.
Stock Class

Attributes:

- private int stockID = 0;
- private int numberOfStocks = 0;
- private double highestBid = 0.0;
- private double lowestBid = 0.0;
- private double highestBuy = 0.0;
- private double lowestSell = 0.0;
- private double lastSoldFor = 0.0;
- private double currentPrice = 0.0;
- private String owner = “”;
- private String company = “”;
- private boolean onSale = false;

Functions:

- public Stock()
  //This is the constructor for the Stock class

- public int getStockID()
  //This returns the stockID of a stock

- public int getNumberofStocks()
  //This returns numberOfStocks or shares that aStock has

- public double getHighestBid()
  //This returns highestBid, the current highest bid of a stock

- public double getLowestBid()
  //This returns lowestBid, the lowest bid that was placed on a stock

- public double getHighestBuy()
  //This returns highestBuy, the highest amount that a share in the stock was bought for

- public double getLowestSell()
  //This returns lowestSell, the lowest amount that a share was sold for

- public double getLastSoldFor()
  //This returns lastSoldFor, the amount of money that a stock was last sold for

- public double getCurrentPrice()
  //This returns currentPrice or the current price of a stock

- public String getOwner()
  //This returns the name of the owner of the stock

- public String getCompany()
  //This returns the name of the company that owns the stock

- public boolean getSaleStatus()
  //This returns true if the stock is on sale or false if the stock is not on sale

- public void setStockID(int newval)
  //This function sets the ID of a stock to newval

- public void setNumberOfStocks(double newval)
  //This function sets the number of stocks to newval

- public void setHighestBid(double newval)
  //This function sets the highest bid that was placed on a stock, highestBid, to newval

- public void setLowestBid(double newval)
  //This function sets the lowest bid that was placed on a stock, lowestBid, to newval

- public void setHighestBuy(double newval)
  //This function sets the highest bought price of a stock, highestBuy, to newval

- public void setLowestSell(double newval)
  //This function sets the lowest price that a stock was sold for, lowestSell, to newval

- public void setLastSoldFor(double newval)
  //This function sets the price that the stock was last sold for, lastSoldFor, to newval

- public void setCurrentPrice(double newval)
  //This function sets the current price of the stock to newval

- public void setOwner(String newOwner)
  //This sets the owner of a stock to newOwner

- public void setCompany(String newval)
  //This sets the company of a stock to newval

- public void setOnSale()
  //This sets the boolean onSale to true, making the stock available to buy

- public void setNotOnSale()
  //This sets the boolean onSale to false, no longer making the stock available to sell

- public void bid(double placedBid)
  //If placedBid is higher than the highest current bid on a stock, then highest bid becomes placedBid. If placedBid is lower than the current highest bid, then a error is displayed to the user making the bid

- public void buy(double soldPrice, String newOwner)
  //What this does is place the stock under a new owner by setting owner to newOwner and we set the last sold price, lasSoldFor, to soldPrice

- public void sell()
  //This function checks to see if the stock is on sale. If it is then the stock can be sold, bid on, and bought.

- Public void post(int initNumOfStocks, double initialPrice)
  //This functions posts a new stock. First, check to see if the initial number of stocks, initNumOfStocks is not equal to zero. If it is not, check to see if initialPrice is not equal to zero. If it is not, then we make a new stock object and we set the number of stocks to initNumOfStocks. We also set the current price of the initialPrice. If the initialPrice is equal to zero, then we tell the user to set initial price of stocks. If the initial number of stocks is equal to zero then we tell the user to declare the number of shares/stocks in the stock.

- Public void view()
  //This retrieves the name of the company, the number of stocks, the highest bid, and the lowest bid and displays it.

- public void remove()
  //This stops a stop and removes it. It sets all the attributes of stock back to null if it is of String type, 0 if it is of int type, and 0.0 if it is of double type. It also takes the stock off sale.

- public void history()
  //This retrieves all information about a stock and displays it

b.

UserProfile Class

Attributes:

- public enum RoleType { SA, SO, PT };
- private int account = 0;
- private String username = “”;
- private String pwd = “”;
- private String first = “”;
- private String last = “”;
- private String email = “”;
- private String address = “”;
- private int ssn = 0;
- private int cCard = 0;
- private int phone = 0;
- private int complaints = 0;
- private double balance = 0.0;
- private RoleType role = null;
- private String flag = “”;

Functions:

- public UserProfile()
  //This is the constructor for the UserProfile class

- public int getAccountNumber()
  //This returns the user’s account number (user’s ID)

- public RoleType getRole()
  //This returns the user’s role (SO/PT/SA)

- public String getUserName()
  //This returns the user’s username within the system

- public int getSSN()
  //This returns the user's social security number

- public String getPassword()
  //This returns the user's password

- public String getFirstName()
  //This returns the user's first name

- public String getLastName()
  //This returns the user's last name

- public int getCreditCard()
  //This returns the user's credit card number

- public String getEmail()
  //This returns the user's email address

- public String getAddress()
  //This returns the user's address

- public int getPhone()
  //This returns the user's phone number

- public int getComplaints()
  //This returns the number of complaints filed against the user

- public double getBalance()
  //This returns the user's balance – how much money s/he has within the system

- public String getFlag()
  //This returns the user's flag type

- public void setAccountNumber(int newval)
  //This sets the user's account number to newval

- public void setRole(RoleType newval)
  //This sets the user's role to newval

- public void setUserName(String newval)
  //This sets the user's username to newval

- public void setSSN(int newval)
  //This sets the user's social security number to newval

- public void setPassword(String newval)
  //This sets the user's password to newval

- public void setFirstName(String newval)
  //This sets the user's first name to newval

- public void setLastName(String newval)
  //This sets the user's last name to newval

- public void setCreditCard(int newval)
  //This sets the user's credit card number to newval

- public void setEmail(String newval)
  //This sets the user's email address to newval

- public void setAddress(String newval)
  //This sets the user's address to newval

- public void setPhone(int newval)
  //This sets the user's phone number to newval

- public void setComplaints(int newval)
  //This sets the user's complaint counter to newval

- public void setBalance(double newval)
  //This sets the user's account balance to newval

- public void addToBalance(double newval)
  //This increases the user's account balance by newval

- public void withdrawBalance(double newval)
  //This decreases the user's account balance by newval. However, if balance<newval, an error message is displayed because the user cannot withdraw more money than what he has in his account.

- public void setFlag(String newval)
  //This sets the user's flag attribute to newval. "C" stands for complaints – which appears once the user's complaint counter reaches 3+. "Q" stands for quit – which appears once a Professional Trader has sent a request to quit the system.

c.

User Class

Attributes:

- private UserProfile profile = null;
- private Stock ownedStocks[] = null;

Functions:

- public User ()
  //Makes a new UserProfile object and initialize it to null. Make a new Stock object and initialize to null.

- public boolean createUser(String username, String userpassword)
  // Select * from system-users.xml where user=username;
  If selected password = userpassword then
  Set all attributes with selected values;
  Return true;
  Else
  Return false;

- public boolean updateUser()
  // **boolean** result = false;
  Try
  {
  Update system-users.xml with all current user attributes;
  result = true;
  }

  Return result;

- public void refreshUser()
  // Select * from system-users.xml where user=username;
  Set all attributes with selected values;

- public UserProfile getProfile()
  //This returns the profile of a user.

- public Stock[] getStock()
  //This returns the ownedStocks, the array of stocks that a user owns.

d.

CasualBrowser Class

Attributes:

- private UserProfile CBuser;

Functions:

- public CasualBrowser()
  //This is the constructor for the CasualBrowser class

- public void register(int CBssn, String CBusername, int CBcredit, int CBstatus, String CBpassword)
  //This allows a casual browser to register. He will enter his social security number, intended username, credit card number, intended status, and a temporary password. The information will be sent to the System Administrator who will approve/reject the registration request.

e.

StockOwner Class

Attributes:

- private UserProfile SOprofile;

Functions:

- public StockOwner()
  //This is the constructor for the StockOwner class

- public void editProfile(String newPassword, String confirmNewPwd, String newEmail, String newAddress, int newPhone, double addAmount, double withdrawAmount)
  //This sets the corresponding attributes of the user to the new values specified in the input

f.
ProfTrader Class

(extends the User class)

Attributes:

- private UserProfile PTprofile;

Functions:

- public ProfTrader()
  //This is the constructor for the ProfTrader class

- public void editProfile(String newPassword, String confirmNewPwd, String newEmail, String newAddress, int newPhone, double addAmount, double withdrawAmount)
  //This sets the corresponding attributes of the user to the new values specified in the input

- public void quitSystem()
  //This sets the user's flag attribute to "Q" – indicating that the user (a PT) has requested to quit the system

g.
SystemAdmin Class

(extends the User class)

Attributes:

- private UserProfile SAprofile;

Functions:

- public SystemAdmin()
  //This is the constructor for the SystemAdmin class

- public UserProfile getAllUsers()
  //This returns the information of all of the users that are currently in the system

- public Stocks getAllStocks()
  //This returns the information of all of the stocks that are currently in the system

- public UserProfile getPendingUsers()
  //This returns a list of casual browsers who have sent in register applications/requests

- public Stocks getPendingStocks()
  //This returns a list of new stocks posted by stock owners and that await approval

- public UserProfile getPendingComplaints()
  //This returns a list of users who have been reported (one user who was reported many times will appear multiple times)

4.2 Servlets

a.
LoginServlet

Functions:

- **private boolean** loginUser(String userName, String userPassword, HttpServletRequest req, HttpServletResponse res)
  //Call UserServlet;
  If request was successful then
  createToken(username, userrole);
  insert token into system-log.xml;
  return true;
  else
  return false;

- **private** String createToken(String userName, String userRole)
  //Return MD5-hash(username, role, unique key);

- **private boolean** validLoginUser(String userToken, HttpServletRequest req, HttpServletResponse res)
  // Select * from system-log.xml where token=usertoken;
  If hasRows then
  Return true;
  Else
  Return false;

- **private boolean** logoutUser(String userName, String userPassword, HttpServletRequest req, HttpServletResponse res)
  // Call UserServlet (Diallocate user);

  If request was successful then
  delete token from system-log.xml;
  return true;
  else
  return false;

b.

UserServlet

Functions:

- private String getUserProfile()
  // Build HTML containing user information;
  Return HTML;

- private String getUserProfile(String account)
  //Select * from system-users.xml where useraccount=account;
  Build HTML containing selected user information;
  Return HTML;

- private String getEditProfile()
  //Build HTML containing user editable information;
  Return HTML;

- private boolean updateProfile(HttpServletRequest req)
  //Get request parameters;
  Set user profile attributes to parameter values;
  user.updateUser();

- private String getUserStocks()
  //For i = 0; i < stocks[].length; i++; loop
  Build HTML containing the user stocks[i] description;
  Next

  Return HTML;

- private boolean updateStock(HttpServletRequest req)
  //Get request parameters;
  Set user stock[n] attributes to parameter values;
  user.updateUser();

c.

SystemServlet

Functions:

- **private** String[] getParseList(String itemList, **int** itemSize)
  //j = 0;
  For i = 0; i < itemList.length; i+=itemSize; loop
  List[j++] = substring(itemList, i, itemSize);
  Next;
  Return List[];

- **private** String getRemoveUsers()
  //Select * from system-users.xml where status={FL, QT, IF};
  Build HTML containing selected user description;
  Return HTML;

- **private void** removeUsers(String[] user)
  // For i = 0; i < user.length; i++; loop
  Delete user from system-users.xml where user=user[i];
  Next;

- **private** String getApproveUsers()
  //Select * from system-users.xml where status={PD};
  Build HTML containing selected user description;
  Return HTML;

- **private void** approveUsers(String[] user)
  //For i = 0; i < user.length; i++; loop
  Update user from system-users.xml to status={OK}
  where user=user[i];
  Next;

- **private** String getRemoveStocks()
  //Select * from system-stocks.xml where status={ST};
  Build HTML containing selected stock description;
  Return HTML;

- **private void** removeStocks(String[] stock)
  //For i = 0; i < stock.length; i++; loop
  Delete stock from system-stocks.xml where stock = stock[i];
  Next;

- **private** String getApproveStocks()
  //Select * from system-stocks.xml where status={PD};
  Build HTML containing selected stock description;
  Return HTML;

- **private void** approveStocks(String[] stock)
  //For i = 0; i < stock.length; i++; loop
  Update stock from system-stocks.xml to status={OK}
  where stock = stock[i];
  Next;

d.

StockServlet

Functions:

- private String getStockInfo()
  //Build HTML containing stock information and return it.

- private String postStock()
  //This builds HTML containing editable fields for a user to input his stock. Returns HTML.

- private String processBid()
  //This builds HTML containing editable fields for placing a bid. The bid is then updated for the stock in the system-stocks database. Returns HTML.

- private String processBuy()
  //This processes the bought stock and returns HTML on information regarding the transaction.

- private boolean processSell()
  //This updates the system-stocks database and puts the stock on sale.

- private String requestStop()
  //This builds HTML to stop a stock and returns HTML regarding the pending removal of the stock.

- private String graphHistory()
  //This updates the system-stocks database and returns HTML with the new graph of the stock.

4.3 GUI

a.
View Profile GUI

Functions:

- complaintButton()
  //When a user visits another user's profile, they will press this button to mark a complaint against that user.

- quitButton()
  //When PT is visiting his own profile, he can click on this button to quit from the system, after SA's approval.

- editButton()
  //When a user visits his own page, this button will appear. Once the user clicks on this button they will be able to edit their profile.

b.
Edit Profile GUI

Functions:

- submitButton()
  //When a user is done editing his profile, they click on this button to save their changes.

c.
Register GUI

Functions:

- submitButton()
  //When a user is done filling in the fields to register, they click on this button to submit their application to the SA.

d.
Approve New User GUI

Functions:

- approveButton()
  //The SA clicks on one or multiple new users to approve and once he clicks this button, they will be approved into the system.

e.
Remove User GUI

Functions:

- viewButton()
  //A list of users pending removal shows up and each one has this button next to them. When the SA clicks on this button, they will be taken to the profile of the user they clicked on.

- removeButton()
  //The SA clicks on one or multiple users pending removal and one he clicks on this button they will be removed from the system.

f.
Approve New Stocks GUI

Functions:

- viewButton()
  //A list of stocks requesting to be added into the system is shown and each one has this button next to them. The SA can click on this button and they will be taken to the stock's information.

- approveButton()
  //A list of new stocks shows up and the SA can select one or multiple stocks and once he clicks on this button they will be added into the system.

g.
Remove Stocks GUI

Functions

- removeButton()
  //Clicking this button, which is positioned next to each stock, opens up the corresponding stock's "view stock detail" page

- viewButton()
  //Clicking this button removes the selected stock(s) and all of the stock(s)'s corresponding records/attributes from the system

h.
View Stocks GUI

Functions:

- viewButton()
  //Clicking this button opens up the stock's "view stock detail" page

i.
View Stocks Detail GUI

Functions:

- sellButton()
  //Clicking this button will pass information to the Stocks Servlet, which will process the transaction and changes the owner of the stock

- bidButton()
  //Clicking this button will pass information to the Stocks Servlet, which will process the information and generates a bid record for the stock

- buyButton()
  //Clicking this button will pass information to the Stocks Servlet, which will process the transaction

- stopButton()
  //Clicking this button will pass information to the Stocks Servlet, which will flag the stock

j.
Post New Stock GUI

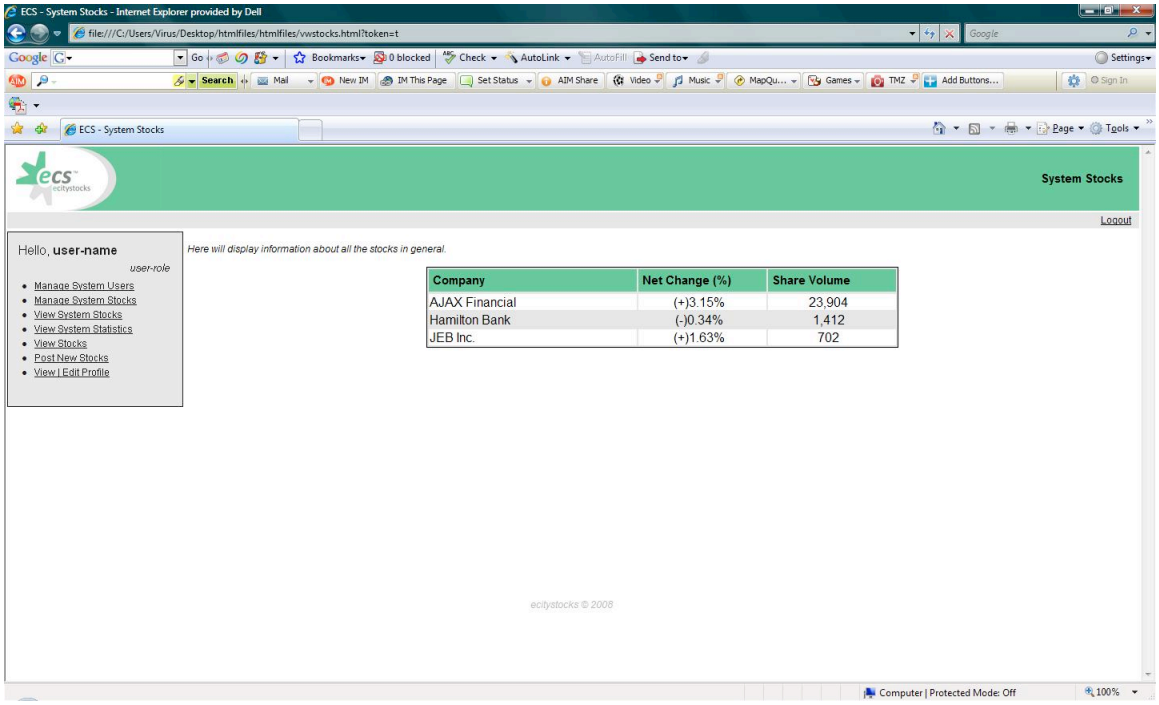
Functions:

- submitButton()
  //Clicking this button will send the information of the new stock to the System Administrator for approval

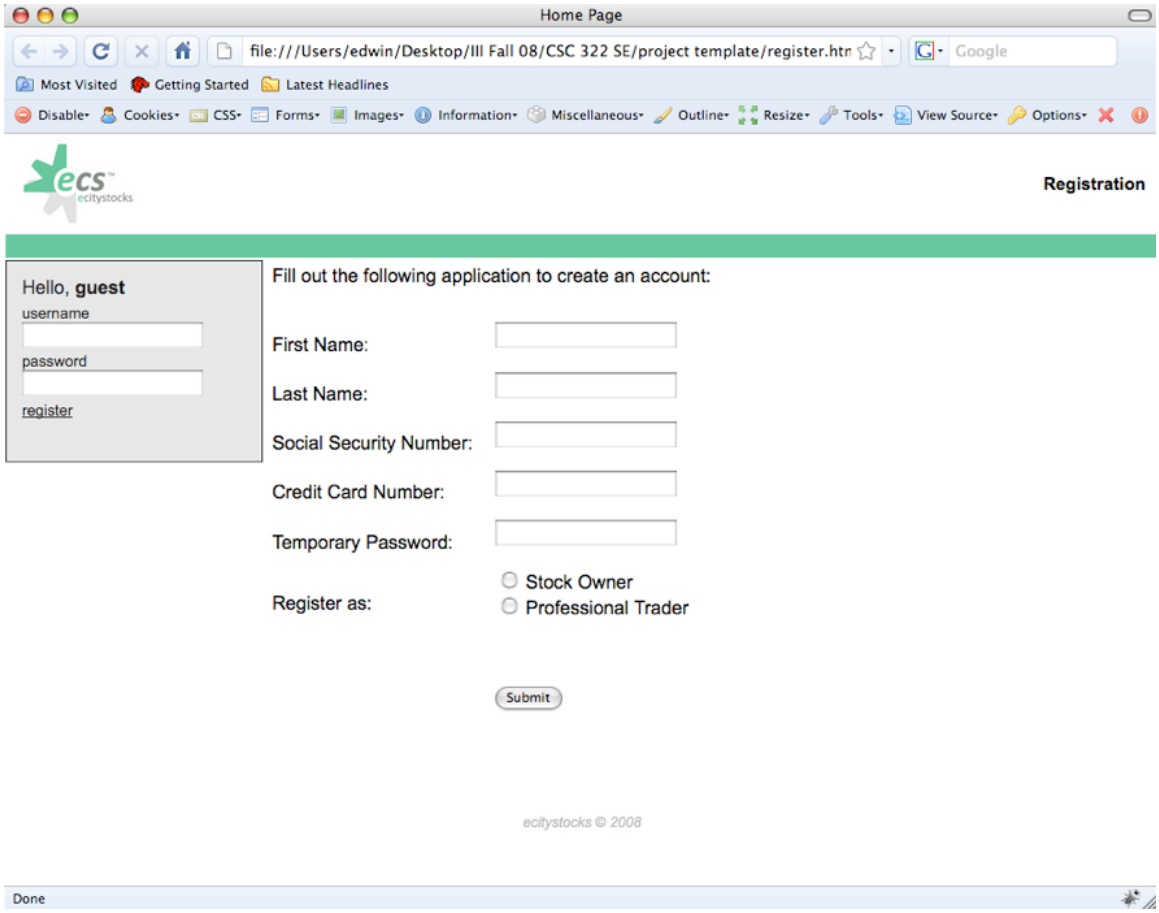
5. System Screens

5.1 User-Interface Prototypes:

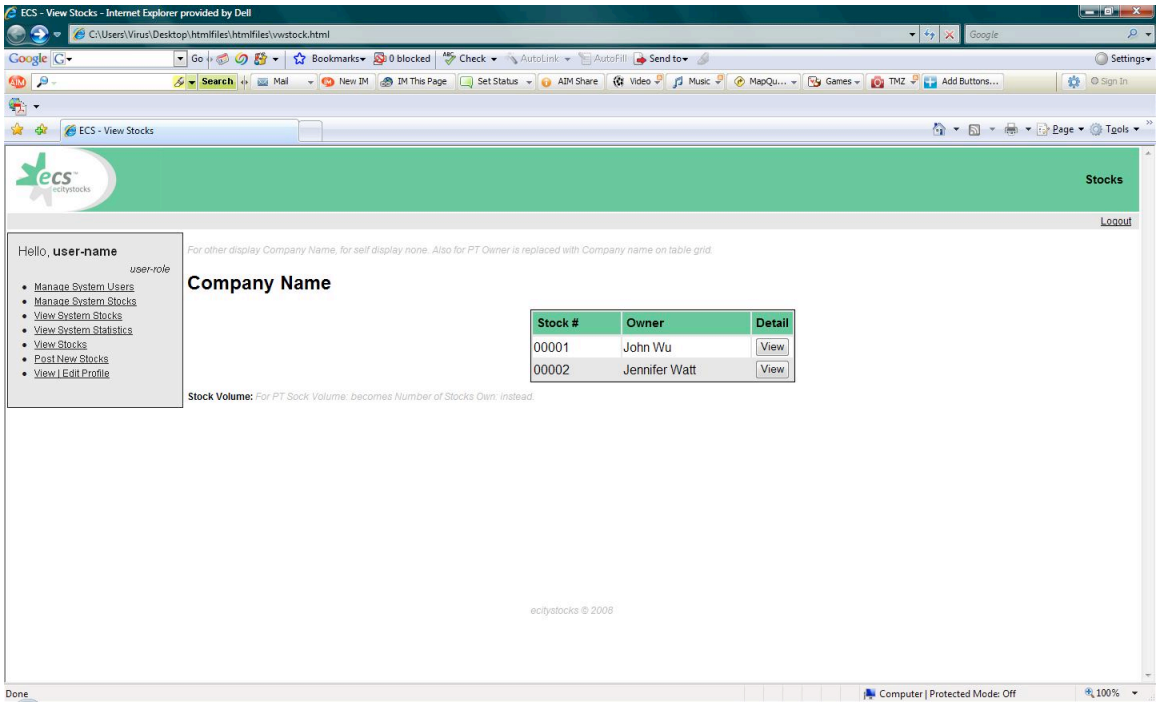
a. View System Stock – This is the storefront that all users see as their homepage.



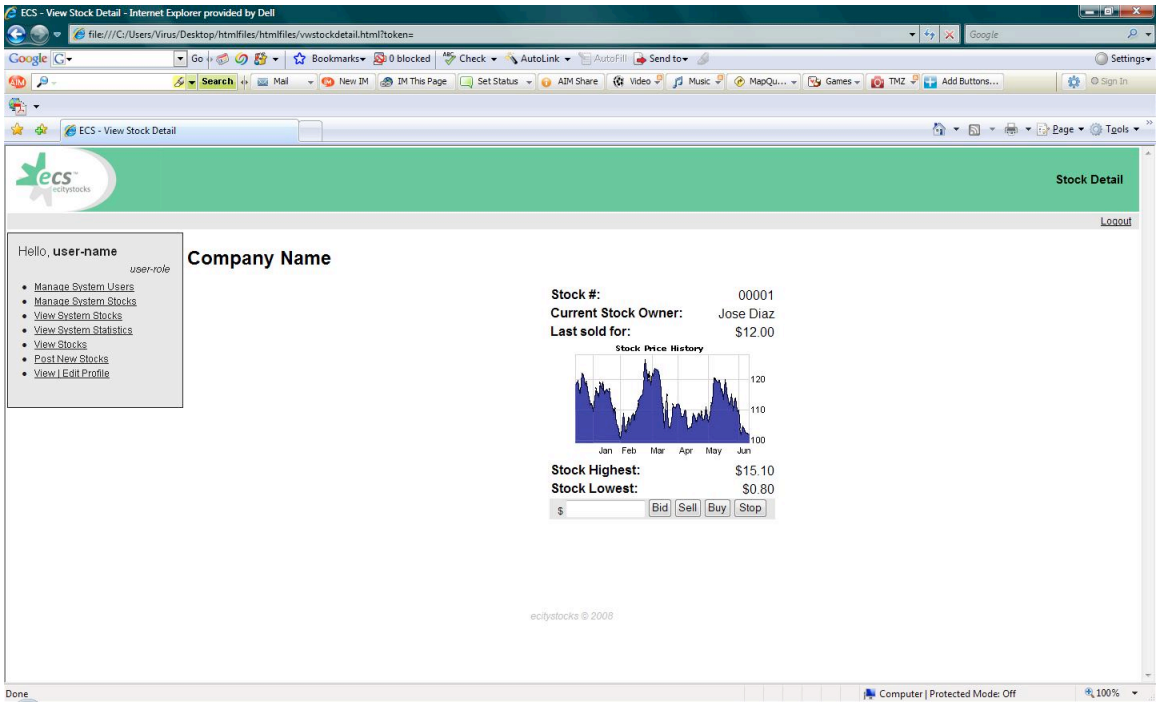
b. Register - This contains the Registration interface as well as the Login interface on the left side.



c. View Stocks – This contains the list of stocks that a company owns.

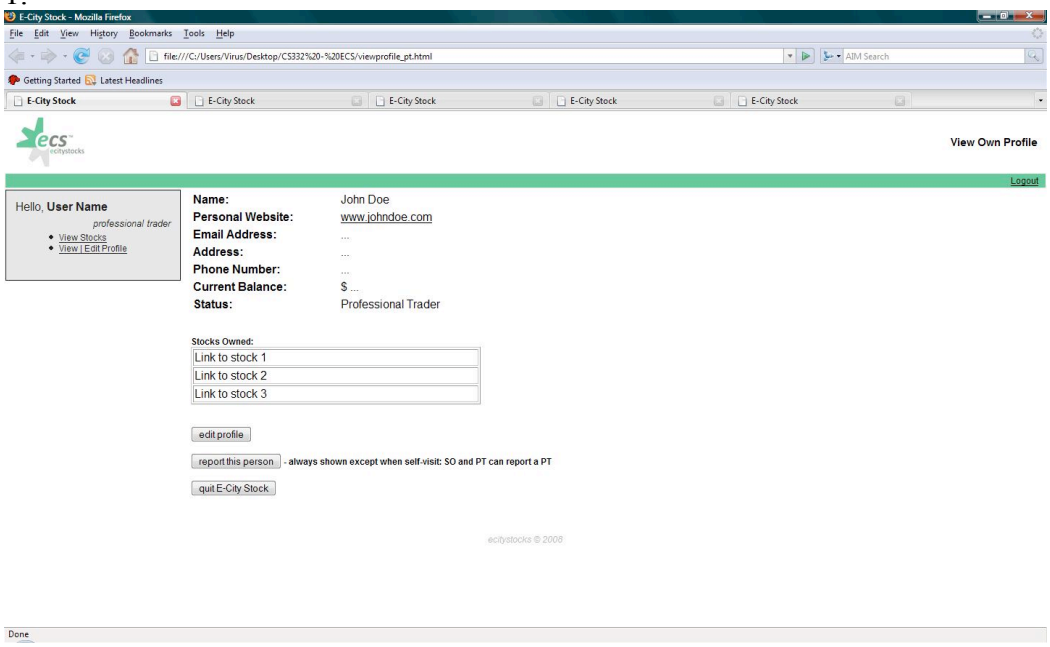


d. View Stock Detail – This is where a SO or PT can bid, buy, or sell stock and view stock history.

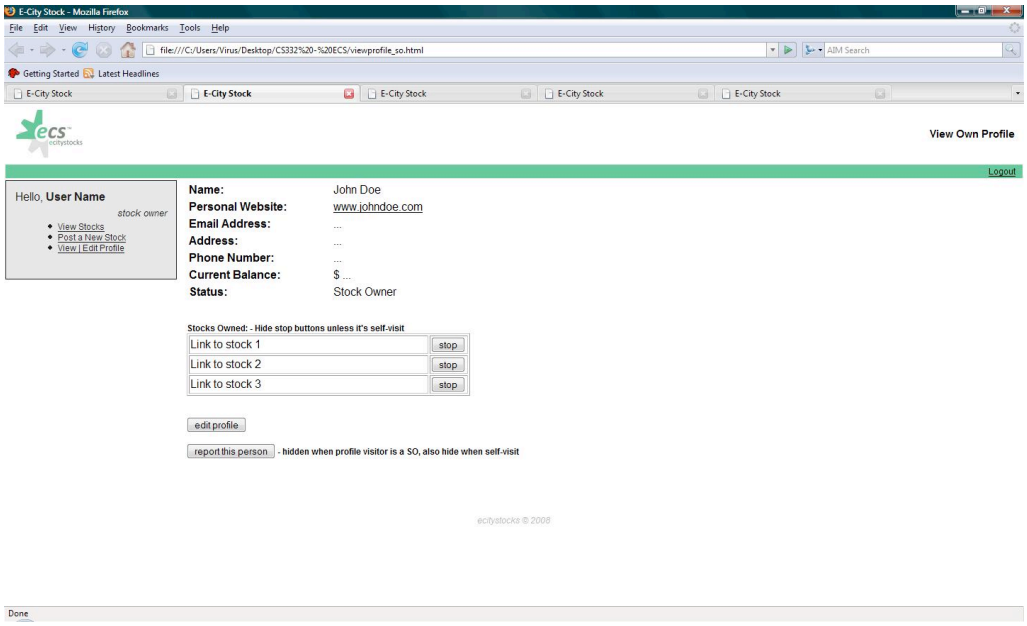


e. View Profile – The first interface is for PT and the second interface is for SO. This is where SO and PT can edit their profile and file a complaint. SO can also stop a stock and PT can quit from the system.

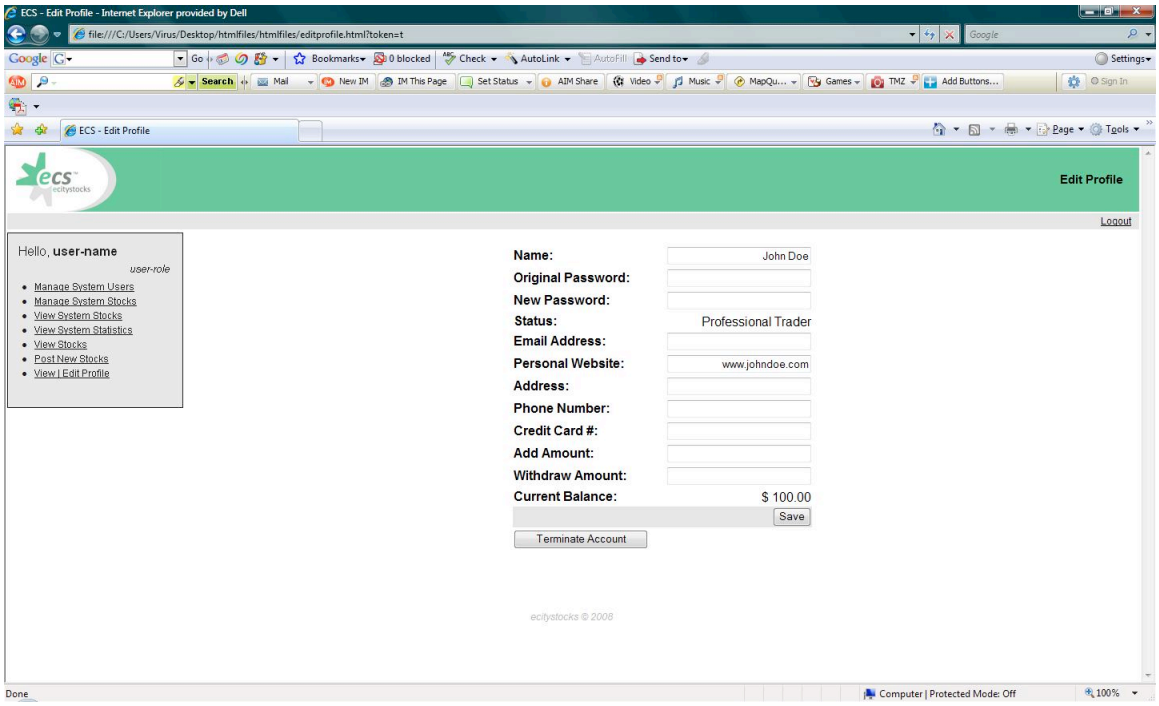
1.



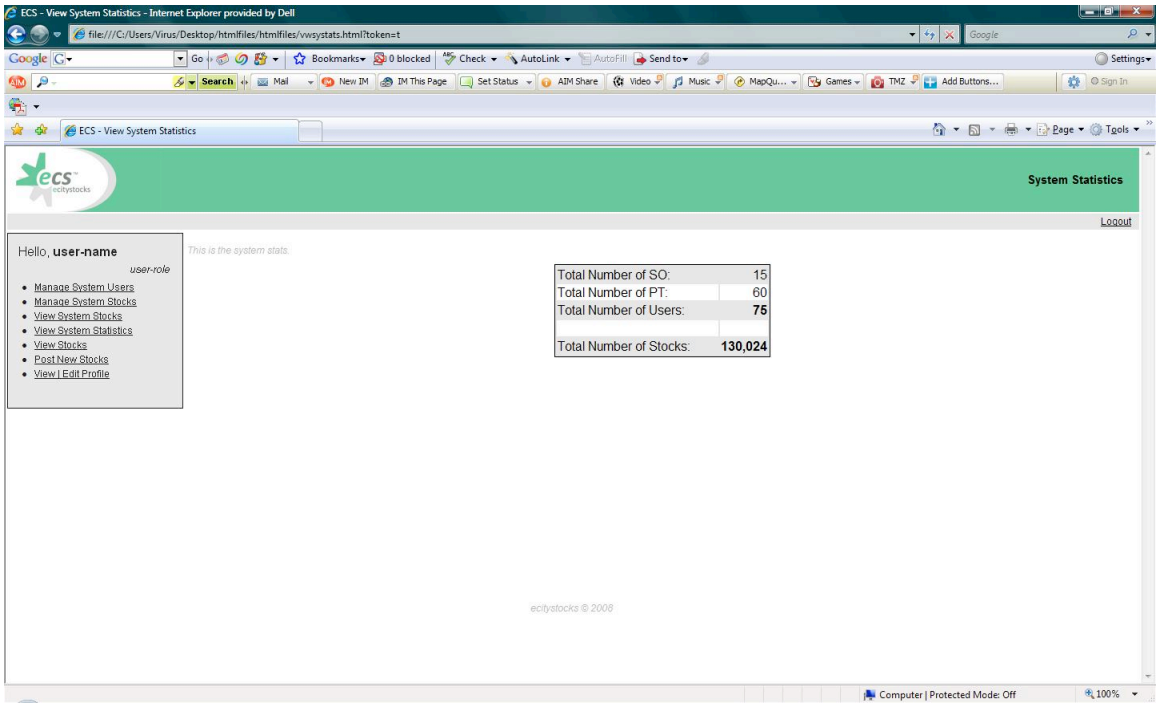
2.



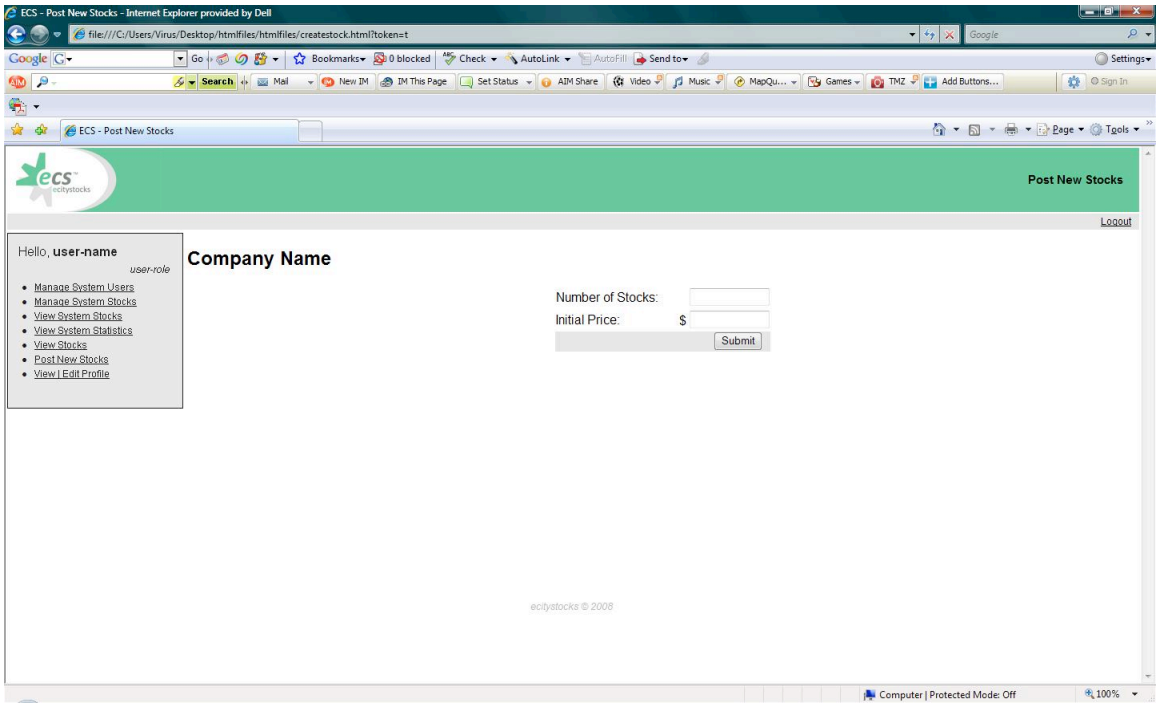
f. Edit Profile – This is where SO and PT can edit their profile and add or withdraw money.



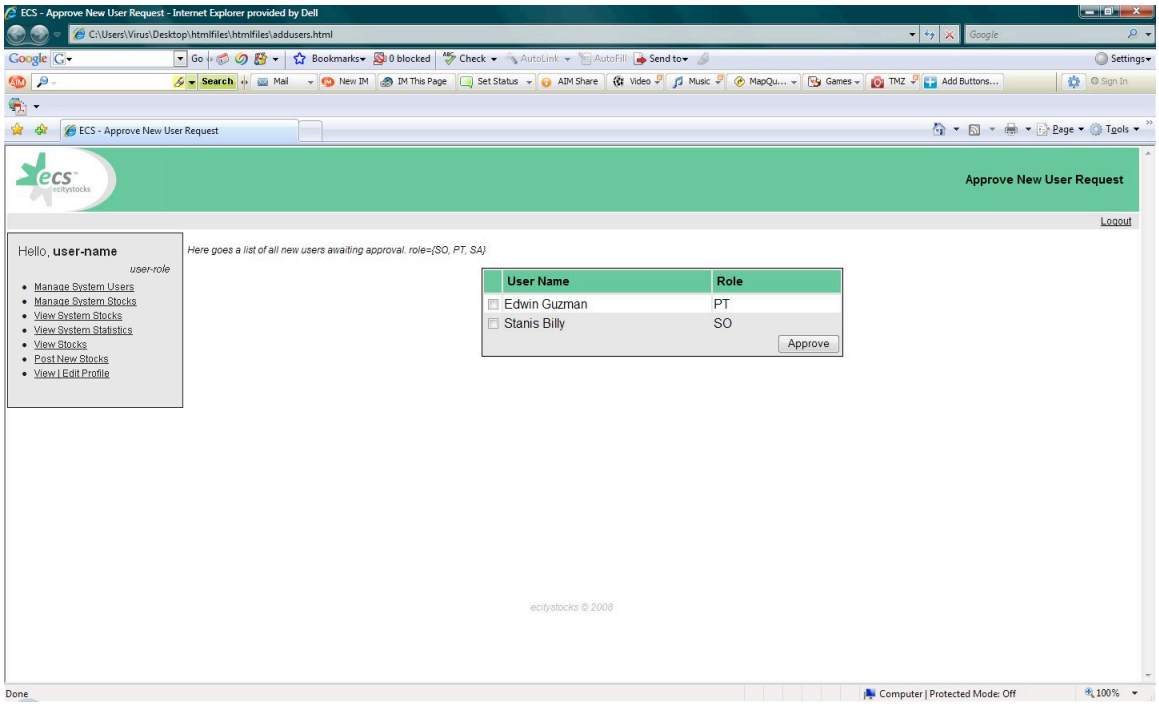
g. View System Statistics – This is where the SA can view system-wide statistics.



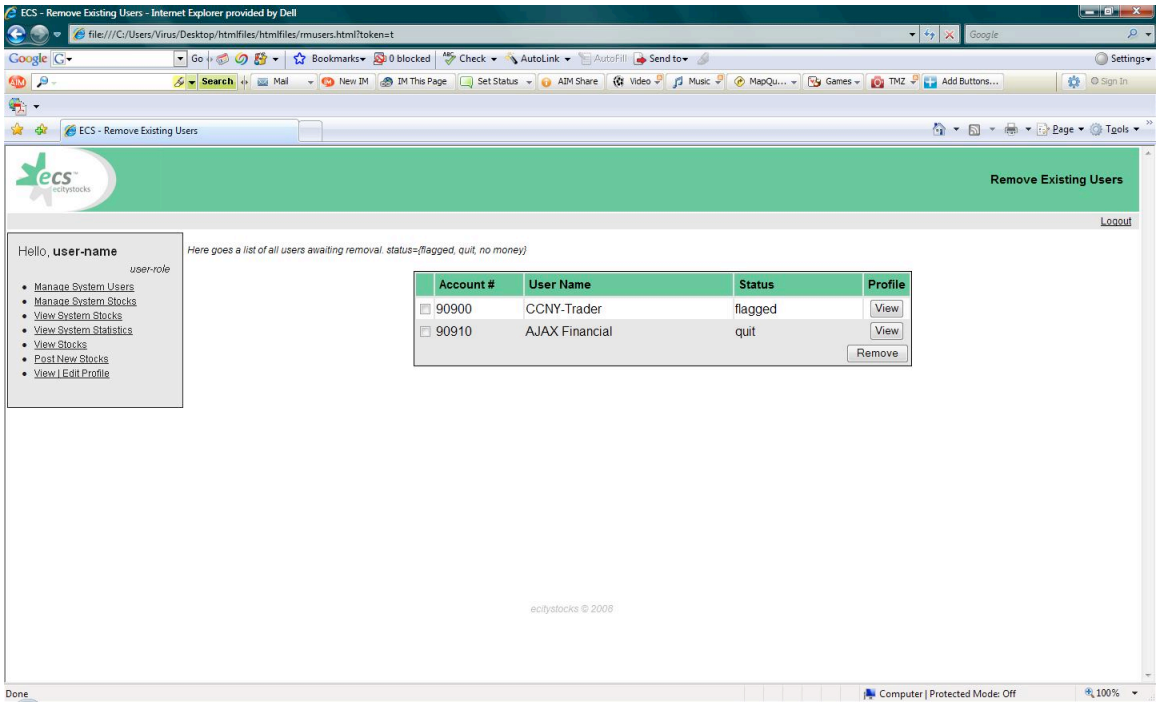
h. Post New Stock – This is where the SO can post a new stock.



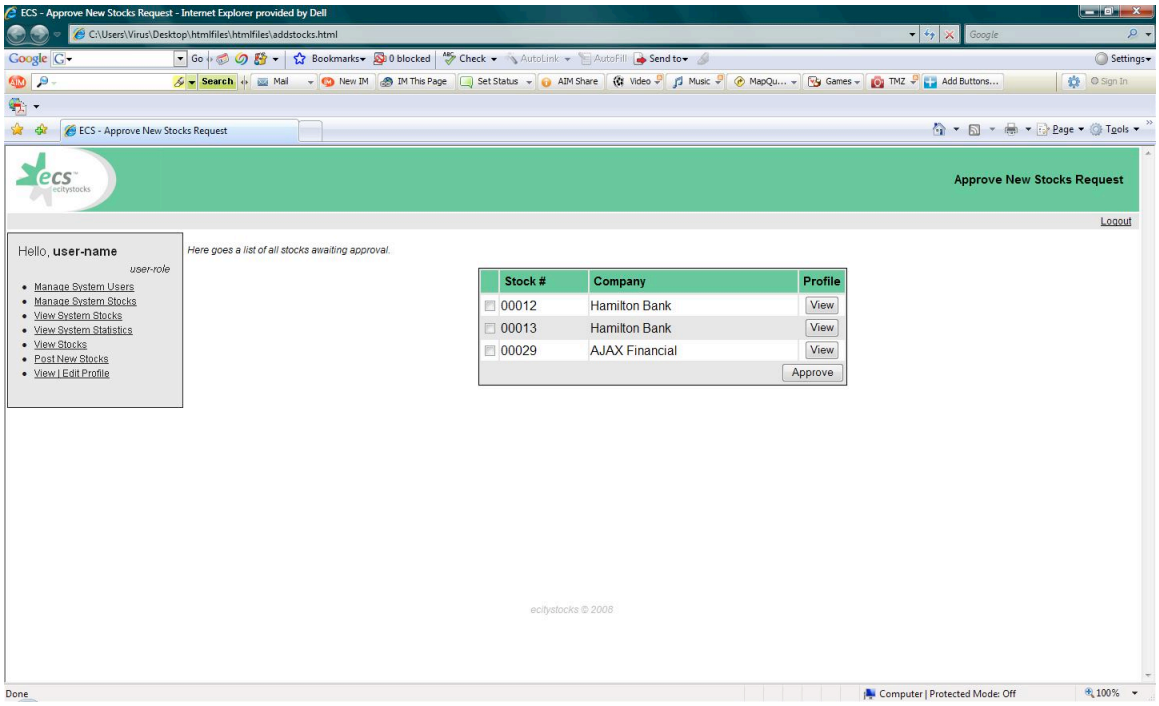
i. Approve New User – This is where the SA approves users who have recently registered to ECS.



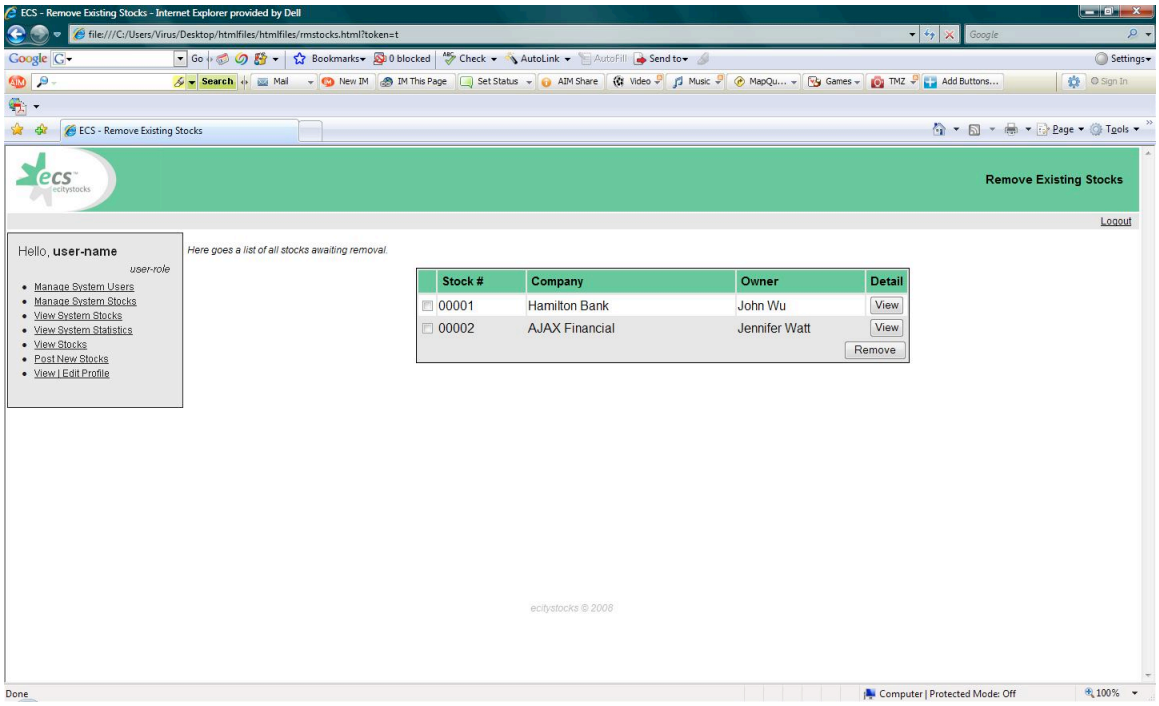
j. Remove User – This interface shows the SA users that have been flagged or who want to quit from ECS.



k. Approve New Stock – This interface gives a list of stocks that wait approval from the SA.



l. Remove Stocks – This interface gives a list of stocks that have been stopped by SO and await removal from the SA.



m. View All Users – This shows the SA all the users in the system along with ID and user type.