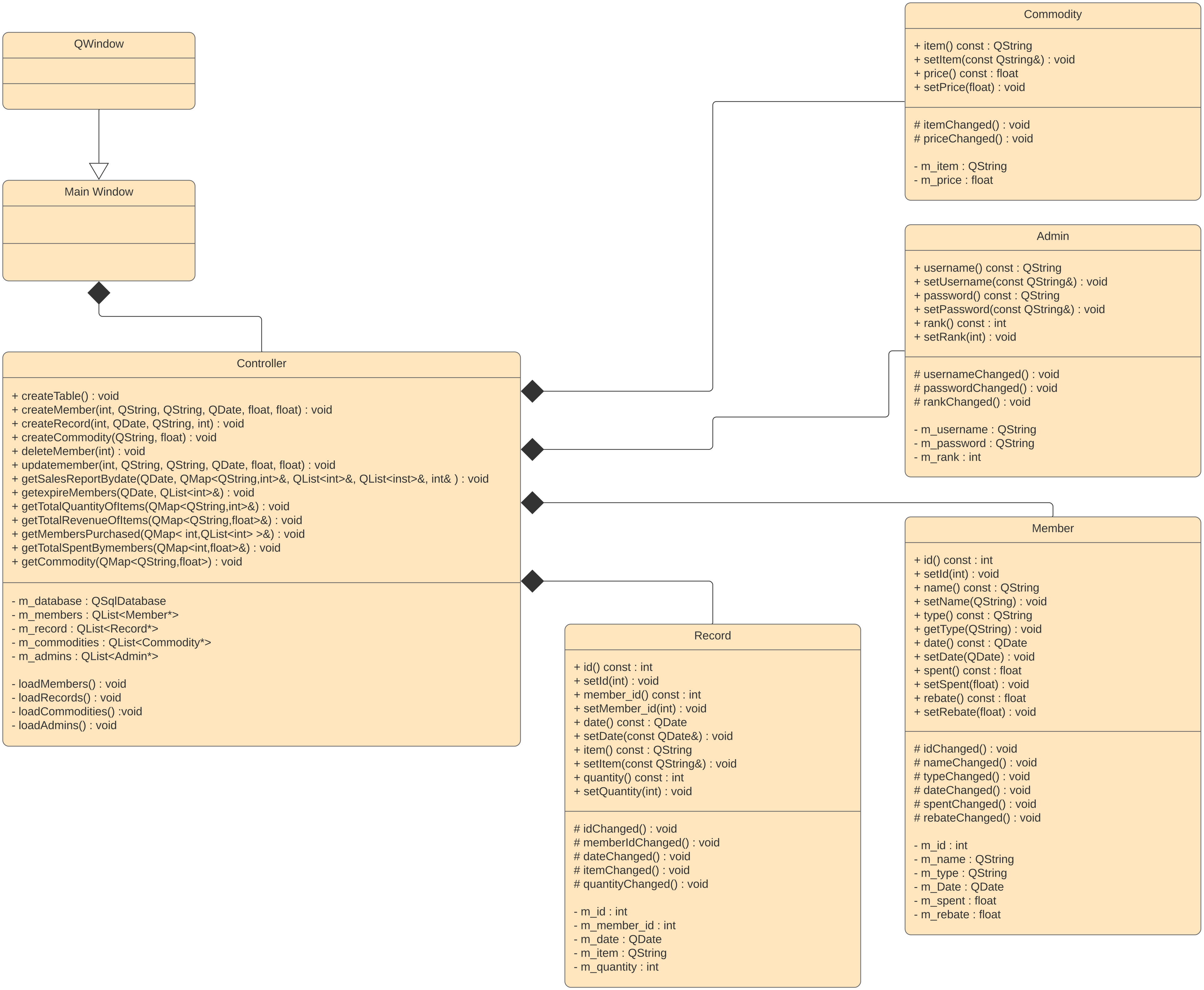
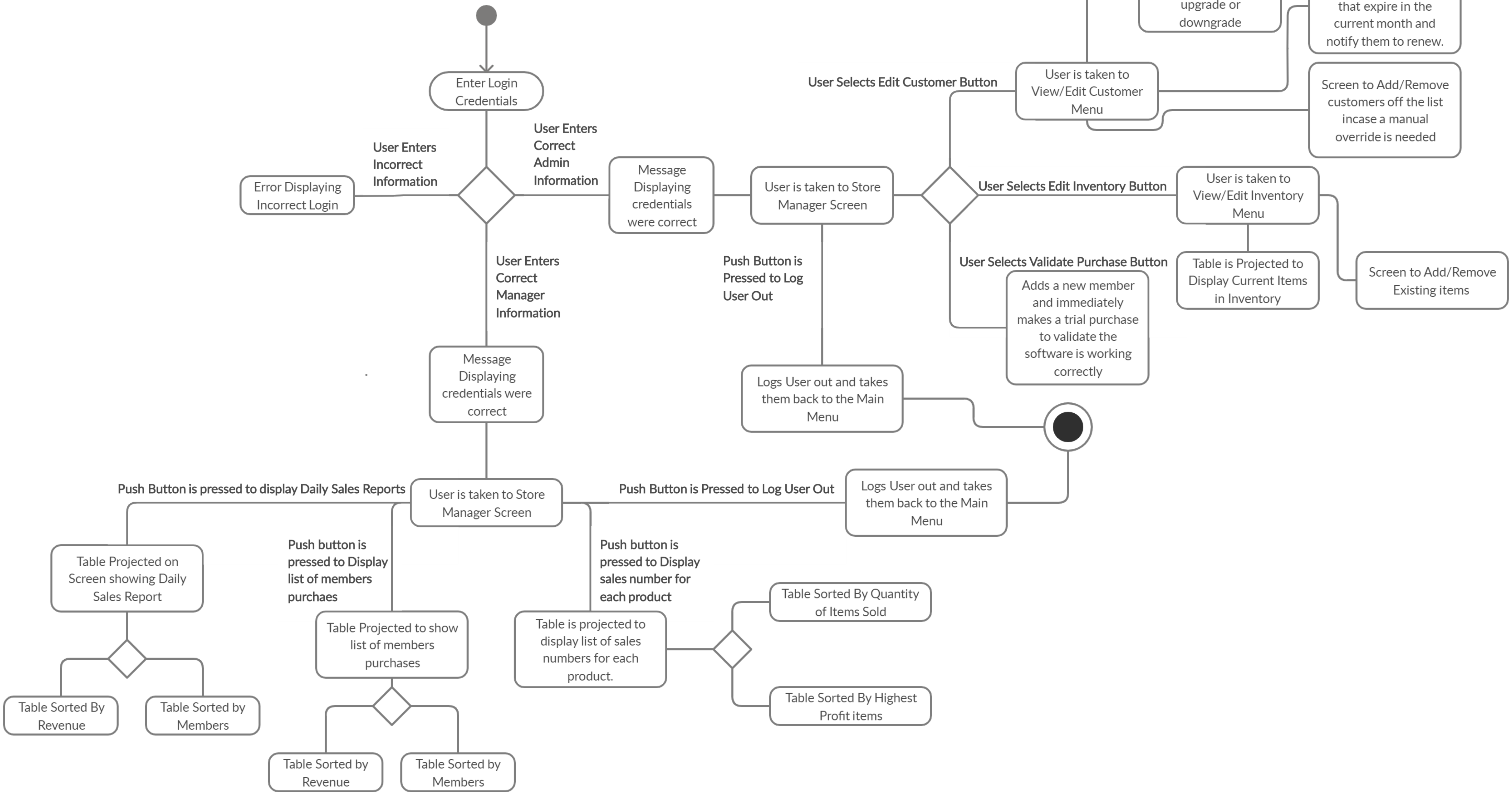
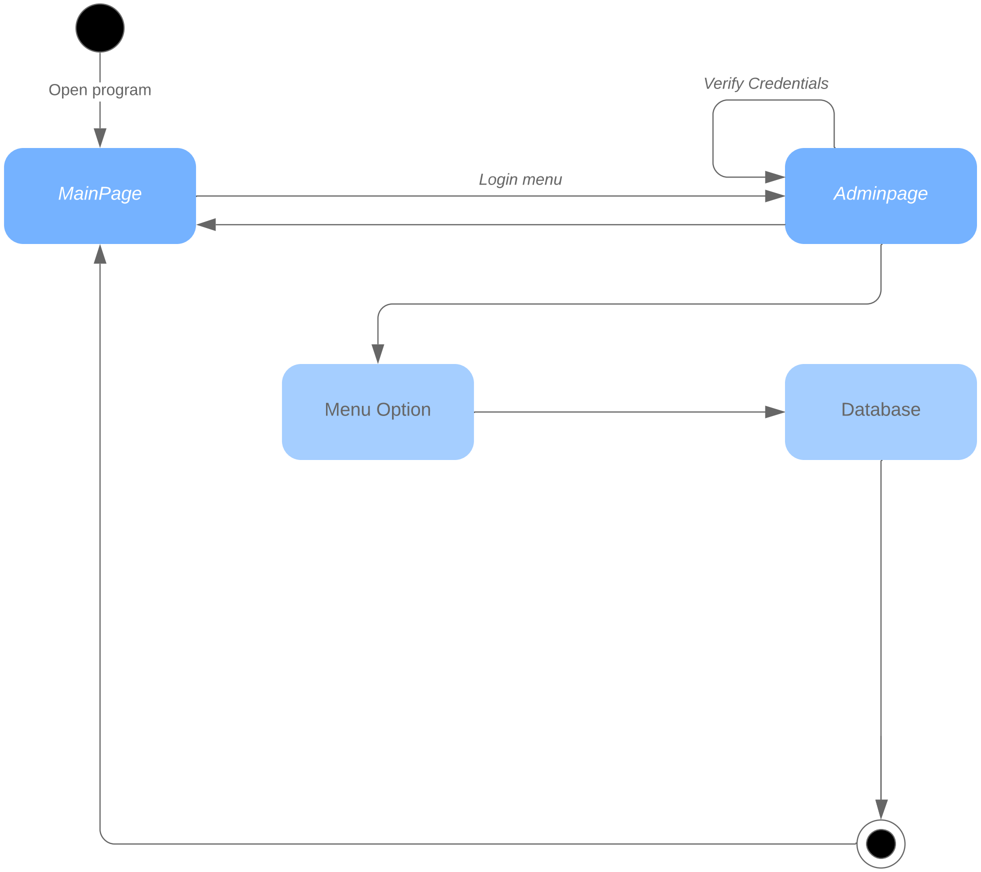

QWindow
Main Window
Controller
+ createTable() : void
+ createMember(int, QString, QString, QDate, float, float) : void
+ createRecord(int, QDate, QString, int) : void
+ createCommodity(QString, float) : void
+ deleteMember(int) : void
+ updatemember(int, QString, QString, QDate, float, float) : void
+ getSalesReportBydate(QDate, QMap<QString,int>&, QList<int>&, QList<inst>&, int&) : void
+ getexpireMembers(QDate, QList<int>&) : void
+ getTotalQuantityOfItems(QMap<QString,int>&) : void
+ getTotalRevenueOfItems(QMap<QString,float>&) : void
+ getMembersPurchased(QMap< int,QList<int> >&) : void
+ getTotalSpentBymembers(QMap<int,float>&) : void
+ getCommodity(QMap<QString,float>) : void
- m_database : QSqlDatabase
- m_members : QList<Member*>
- m_record : QList<Record*>
- m_commodities : QList<Commodity*>
- m_admins : QList<Admin*>
- loadMembers() : void
- loadRecords() : void
- loadCommodities() :void
- loadAdmins() : void
Commodity
+ item() const : QString
+ setItem(const Qstring&) : void
+ price() const : float
+ setPrice(float) : void
# itemChanged() : void
# priceChanged() : void
- m_item : QString
- m_price : float
Admin
+ username() const : QString
+ setUsername(const QString&) : void
+ password() const : QString
+ setPassword(const QString&) : void
+ rank() const : int
+ setRank(int) : void
# usernameChanged() : void
# passwordChanged() : void
# rankChanged() : void
- m_username : QString
- m_password : QString
- m_rank : int
Member
+ id() const : int
+ setId(int) : void
+ name() const : QString
+ setName(QString) : void
+ type() const : QString
+ getType(QString) : void
+ date() const : QDate
+ setDate(QDate) : void
+ spent() const : float
+ setSpent(float) : void
+ rebate() const : float
+ setRebate(float) : void
# idChanged() : void
# nameChanged() : void
# typeChanged() : void
# dateChanged() : void
# spentChanged() : void
# rebateChanged() : void
- m_id : int
- m_name : QString
- m_type : QString
- m_Date : QDate
- m_spent : float
- m_rebate : float
Record
+ id() const : int
+ setId(int) : void
+ member_id() const : int
+ setMember_id(int) : void
+ date() const : QDate
+ setDate(const QDate&) : void
+ item() const : QString
+ setItem(const QString&) : void
+ quantity() const : int
+ setQuantity(int) : void
# idChanged() : void
# memberIdChanged() : void
# dateChanged() : void
# itemChanged() : void
# quantityChanged() : void
- m_id : int
- m_member_id : int
- m_date : QDate
- m_item : QString
- m_quantity : int


Enter Login Credentials
User Enters Incorrect Information
Error Displaying Incorrect Login
User Enters Correct Admin Information
Message Displaying credentials were correct
User is taken to Store Manager Screen
User Selects Edit Customer Button
User is taken to View/Edit Customer Menu
Table displays showing the amount of rebate per customer, and displays reccomendations to upgrade or downgrade
Table displays to see all customer's membersips that expire in the current month and notify them to renew.
Screen to Add/Remove customers off the list incase a manual override is needed
User Selects Edit Inventory Button
User is taken to View/Edit Inventory Menu
Table is Projected to Display Current Items in Inventory
Screen to Add/Remove Existing items
User Selects Validate Purchase Button
Adds a new member and immediately makes a trial purchase to validate the software is working correctly
Push Button is Pressed to Log User Out
Logs User out and takes them back to the Main Menu
User Enters Correct Manager Information
Message Displaying credentials were correct
User is taken to Store Manager Screen
Push Button is pressed to display Daily Sales Reports
Push Button is Pressed to Log User Out
Logs User out and takes them back to the Main Menu
Table Projected on Screen showing Daily Sales Report
Table Sorted By Revenue
Table Sorted by Members
Push button is pressed to Display list of members purchaes
Table Projected to show list of members purchases
Table Sorted by Revenue
Table Sorted by Members
Push button is pressed to Display sales number for each product
Table is projected to display list of sales numbers for each product.
Table Sorted By Quantity of Items Sold
Table Sorted By Highest Profit items

UML STATE DIAGRAM PROJECT 2

Mohamed | October 28, 2019



Case number 1
Systems: Qt, SQL
Use Case: Add items

Description: This allows the user (an administrator) to add or delete items from the store inventory. The store inventory is a table within the main database that holds item names and price.

Actors:

- User (must be admin)

Triggers:

- The user clicks on the “Add item” button from the admin menu.

Preconditions:

- There is a table for the item inventory in the database
- The user is logged in as an administrator.
- The user has the item name and price to be added.

Postconditions:

- The user will receive a confirmation window that the item has been added.
- The table will updated with the item information.

Normal Flow:

1. The user clicks on the button to add an item.
2. The system will display a dialog box with a field for the user to enter the item name and price.
3. The system will display a confirmation window with item information to be added.
4. The user clicks “Accept”.
5. The system reads in the item name and price, create a record, and inserts it into the database table.
6. The user is returned to the admin menu.

Alternate Flows:

4A1. The user notices an error and clicks “Cancel”.
4A2. The system returns to the original dialog box for the user to enter a different item name and/or price.

5A1. The system fails to properly insert the new record. An error message is displayed.
5A2. The user is returned to the admin menu.

Case number 2
Systems: Qt, SQL
Use Case: Display Sales

Description: The store manager needs to be able to view the sales of all items for any given day. The list should be sortable by total sales, revenue, number of members who purchased a given items, and types of members.

Actors:

- Store Manager

Triggers:

- The user clicks on the “Display daily sales” button from the main menu.

Preconditions:

- There is a table for the item inventory in the database
- The user is logged in as a store manager.
- The table has functions to be sorted by the required criteria.

Postconditions:

- The table is displayed and sorted according to the proper criteria.

Normal Flow:

1. The user clicks on the button to display daily sales
2. The program goes to a new page (stack widget) with a table view and a dropdown menu for each sale day available.
3. The user selects which day to display from the dropdown menu.
4. The table view will be updated to display all item sales for that day.
5. The user selects one of the following buttons: “Sort by total sales”, “Sort by total revenue”, “Sort by total member purchases”, and “Sort by member type”.
6. The table is sorted to display the information in descending order.
7. The user can select other buttons to sort by other criteria.
8. When the user has finished viewing the tables, the user clicks “Close”.
9. The user is returned to the main menu (home page).

Alternate Flows:

3A. No sales are available for a given day. The user is unable to complete any operations.

Case number 3
System: Qt, SQL
Use Case: Promote Customer

Description: The program should determine if a Regular customer should upgrade to Executive Status, based on the potential for the rebate to offset the increased cost of membership.

Actors:

- Customer

Triggers:

- The customer checks out at a sales counter.

Preconditions:

- There is a table for member status and total purchases made by that member in the database

Postconditions:

- The customer is alerted to the beneficial savings of upgrading to an Executive member.

Normal Flow:

1. The customer checks out at a sales counter or register.
2. The total sale of all purchased items are added to the customer's annual total.
3. The program performs the following calculation: if 2% of the customer's sales are greater than or equal to the difference in cost from Executive to Regular membership (0.02 * (total sales) >= (120-65 = 55)), then a system message should display to inform the customer that a membership upgrade is warranted.
4. Note that sales tax is not taken into account when calculating total sales.

Alternate Flows:

3A. If the total hypothetical rebate is less than the membership cost difference ($55), then no upgrade is suggested and no message is displayed.

Case number 4
System: Qt, SQL
Use Case: User Accounts

Description: Only logged-in users should have access to the program. A login screen is displayed on program startup, and only a valid login will allow access to the program.

Actors:

- User (Admin or Store Manager)

Triggers:

- The program is started.

Preconditions:

- There is a table for user accounts that contain the list of acceptable usernames and passwords.

Postconditions:

- The user is logged in and directed to the main menu.

Normal Flow:

1. The user starts the program.
2. The program displays a window (stack widget) with text fields for username and password.
3. The user enters their information and clicks “login”.
4. The program checks the username and password against the table.
5. If the username and password is correct, then the user is logged in as that account and is shown the main menu.

Alternate Flows:

3A. The user clicks “cancel” instead and exits the program.

5A. The username and/or password is incorrect. An error message is displayed. the user is returned to the login screen to try another login.

Case number 5
System: Qt, SQL
Use Case: Add/Remove Customers
Summary: Administrator uses valid credentials to enter the database table and either remove or add customers to the list.
Actor: Administrator
Precondition: Admin open the program login window shows on the screen.
Description:

- Administrator uses login credentials to login to database.
- Once system recognize credentials administrator will have full access to customer's database.
- Database page will have a variety of buttons which represent different actions that can be done on the list.
- Administrator then clicks the add button to add a new customer to the database.
- The system will prompt the administrator to add customer information to store in the database.
- Once Administrator input the information, the information will be stored in the database.