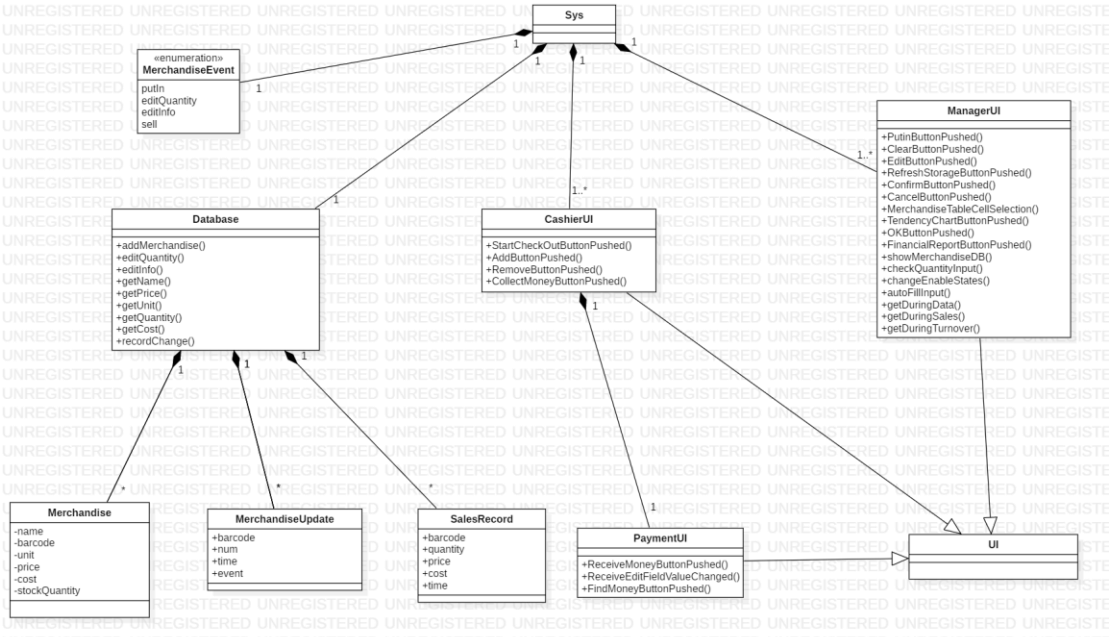
Specification for Supermarket Cashier

v1.0.0

System Architecture



Software Specifications

S1: Database

S1.1: Merchandise info

S1.1.1:

- Define a class: *Merchandise* and it has properties: *product name, cost, price, stock quantity, barcode, unit.*

S1.2: Financial data

S1.2.1: Merchandise change info

- Define a class: *MerchandiseUpdate* and it has properties: *barcode, num, time,*

event. It stores change of the merchandise with barcode *barcode*, data after being updated *num* and event type *event*, at time *time*.

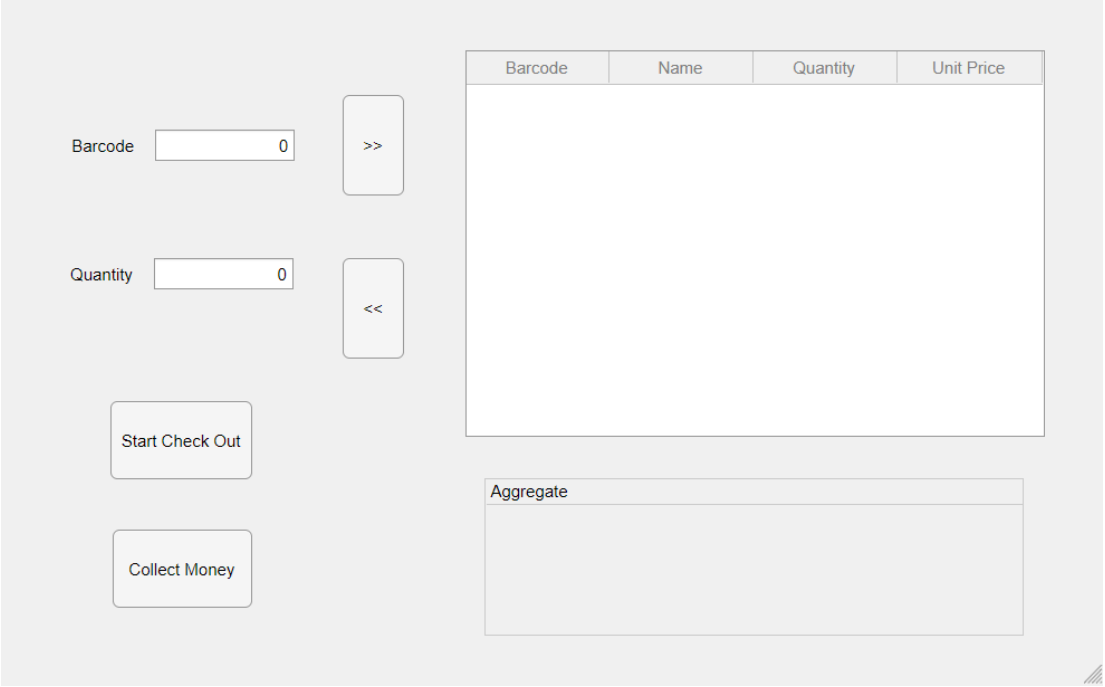
S1.2.2: Merchandise change event type

- Define an enumeration class: *MerchandiseEvent* and it has enumeration: *putIn, editQuantity, editInfo, sell.*

S1.2.3: Sales record

- Define a class: *SalesRecord* and it has properties: *barcode, quantity, price, cost, time. quantity* is the sale quantity for each merchandise at each sale time.

S2: Interface for cashier



S2.1: Check out items

S2.1.1: Check out

1. *StartCheckOutButton* is for starting check out.
2. Use edit fields *BarcodeEditField, QuantityEditField* and buttons *AddButton*

and *RemoveButton* to enter barcode and quantity. Use two buttons to add or cancel. The merchandises added are displayed in the table *CheckTable*.

a) If item is not in *database*, it cannot be added.

b) If item is not already in the table, add the item to the table.

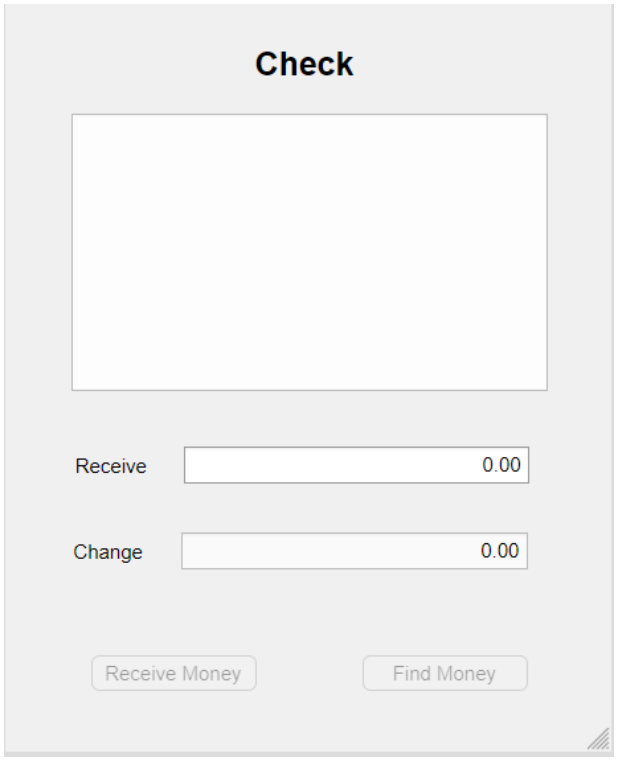
c) If item is already in the table, increase the quantity.

d) If quantity in the table is equal to or less than zero, remove the item from the table.

e) If item is not in the table, it cannot be removed.

3. *CollectMoneyButton* is for finishing check out and going to payment.

S2.1.2: Payment

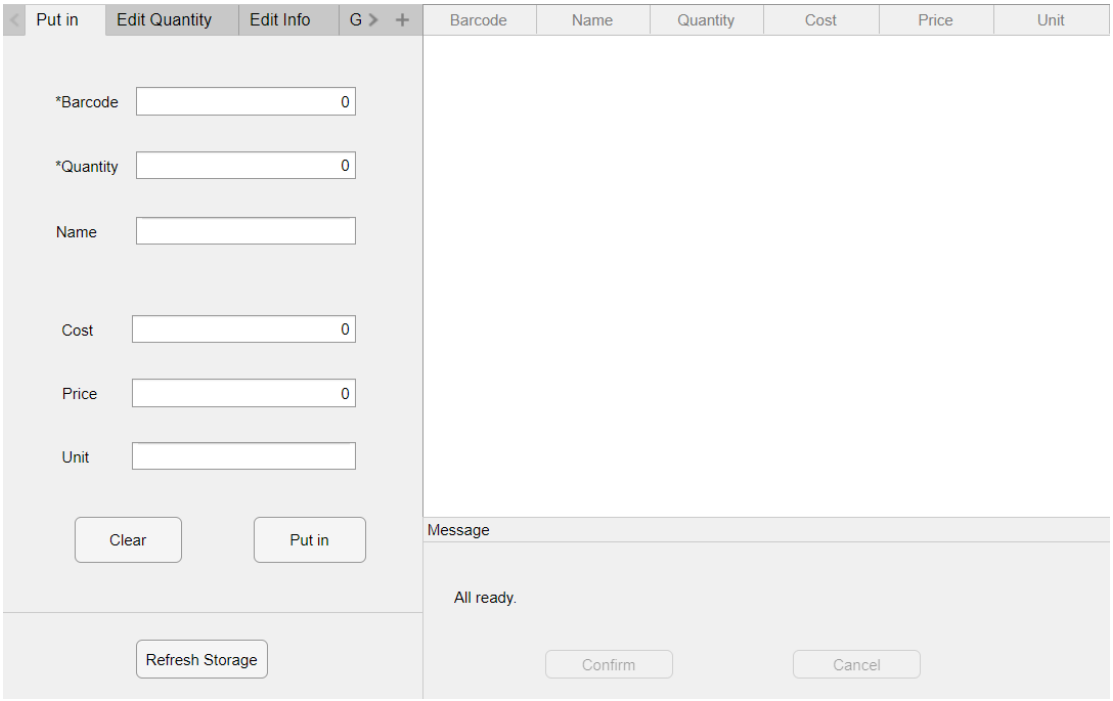


1. Define a *PaymentUI*, and use edit field *ReceiveEditField* to enter money collected.

2. Click *ReceiveMoneyButton* to calculate change money.

3. Change money is calculated by the UI and displayed in edit field *ChangeEditField*.
4. Click *FindMoneyButton* to finish the payment and go back to *CashierUI.* For all merchandises sold, new *MerchandiseUpdate* objects will be generated and stored in *database,* with event type *sell*. And new SalesRecord objects will be generated and stored in *database*.

S3: Interface for manager



S3.1: Manage inventory

S3.1.1: Put in storage

1. Use edit fields and buttons to enter merchandise (to be put in) information.
    a) If the merchandise is already in *database*, increase the quantity and update the info.
        i. If some info is not entered, it means no change on them.

b) If the merchandise is not in database, generate a new *Merchandise* object and store it in *database*.

2. Error messages are displayed in message panel. The number in *QuantityEditField* cannot be negative. If the quantity is zero, it will cause an error.
3. When putting in a merchandise, confirming messages are also displayed in message panel, and two buttons are for confirming or cancelling. When waiting for confirming, any other operations are disabled. The change message will show in message panel.
4. Any change of *MerchandiseList* will refresh the table which displays the inventory information.
5. A new *MerchandiseUpdate* object will be generated and stored in *database,* with event type *putIn*.

S3.1.2: Edit quantity

1. Use edit fields and buttons to enter the barcode and quantity which are needed for editing.
   a) Negative quantity means decrease.
2. Error messages are displayed in message panel.
   a) If the barcode is not found in *database,* or the quantity is zero, or the quantity will make the stock quantity negative, it will cause an error.
3. Any change of *MerchandiseList* will refresh the table which displays the inventory information and the change message will show in message panel.

4. A new *MerchandiseUpdate* object will be generated and stored in *database,* with event type *editQuantity*.

S3.1.3: Edit info

1. Use edit fields and buttons to enter merchandise information which are needed for editing. Empty or default values (like 0's) mean no change.
2. Error messages are displayed in message panel. If the barcode is not found in *database,* it will cause an error.
3. When editting the merchandise, confirming messages are also displayed in message panel, and two buttons are for confirming or cancelling. When waiting for confirming, any other operations are disabled. The change message will show in message panel.
4. Any change of *MerchandiseList* will refresh the table which displays the inventory information.
5. A new *MerchandiseUpdate* object will be generated and stored in *database,* with event type *editInfo*.

S3.2: Generate reports

S3.2.1: Get merchandise change info

1. Use date pickers to select the statistics start time and end time. Use edit field to enter the barcode of the merchandise need to be analysed.
2. Click *TendencyChartButton,* corresponding cost, price, profit and stock quantity updating records and sale quantity records in *database* will be found or calculated.

3. The change tendency charts of all data above during selected time will be displayed on the whole window.
    a) If all records are empty, nothing will be displayed.
4. *OKButton* is used to return to the manager UI.
5. The confirm message will be on the message panel.

S3.2.2: Generate financial report

1. Use date pickers to select the statistics start time and end time.
2. Click *FinancialReportButton,* total cost, total turnover and total profit during selected time will be displayed on the right of the window.
3. The confirm message will be on the message panel. Click *ConfirmButton* to return.

S3.3: Get inventory info

S3.3.1: Refresh storage

- Click *RefreshStorageButton* to show the latest inventory info.

S3.3.2: Quick input

- Select any information of any merchandise in the table, all barcode edit fields will be filled with the barcode of the corresponding merchandise.

S3.4: Clear all current inputs

S3.4.1:

- All clear buttons are for making all texts and values of edit fields in current tab return to default values.