Company: Triple-S

Software Requirement Specification
For
Electronic Business System

<Version 2.0>

Prepared By
Jin Chen
Yuemin Tang
Hongzhi Pan
Yuting Yang

Revision History

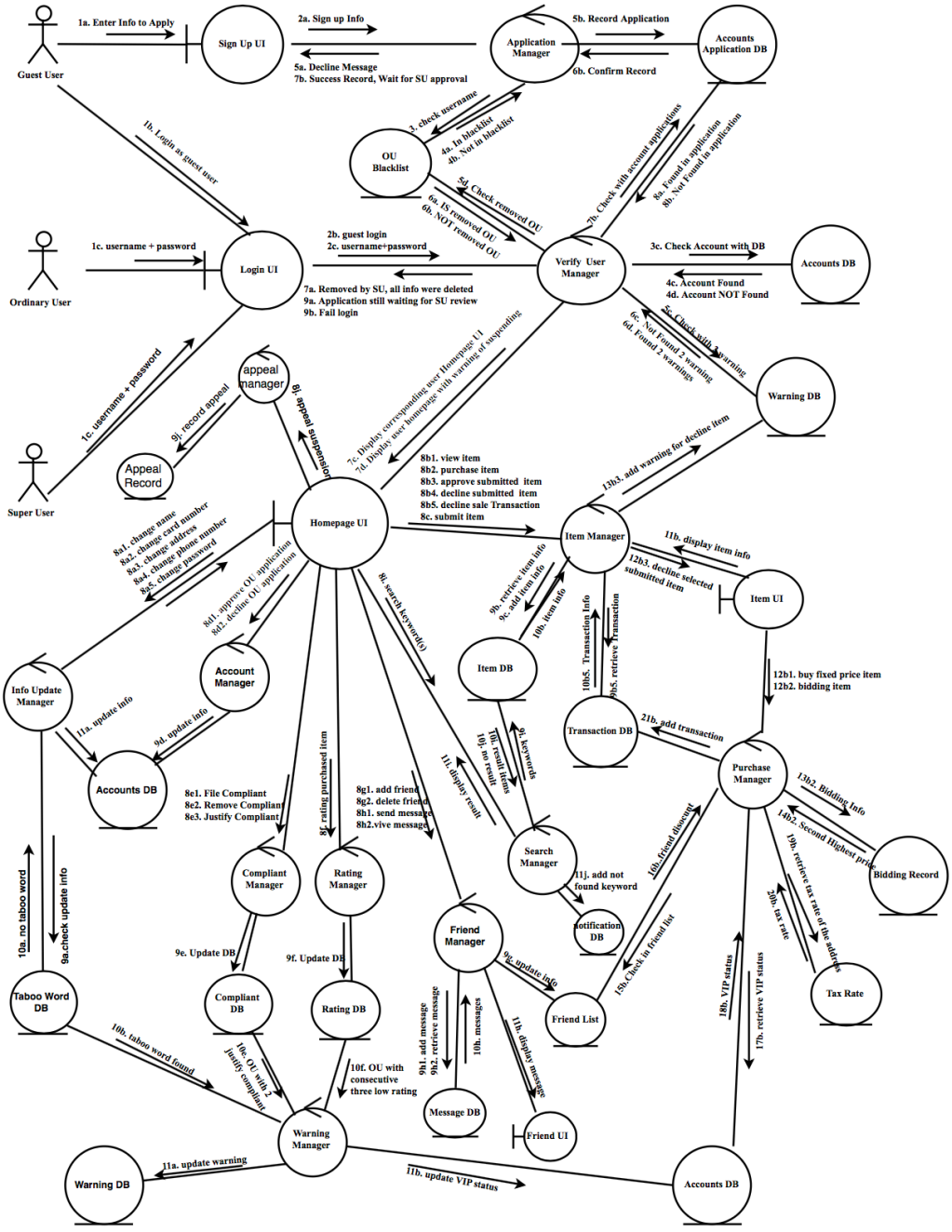
Date	Version	Description	Author
<23/03/2019>	<1.0>	Initial Release	Jin Chen, Yuemin Tang, Hongzhi Pan, Yuting Yang
<13/04/2019>	<2.0>	First Version of EbyMazon online selling system	Jin Chen, Yuemin Tang, Hongzhi Pan, Yuting Yang

Table of Contents

Phase II: Design Report

1. Introduction

This report gives the general overview of the design for our eByMazon system and it shows the details of majority functionalities that will carried out by the system. Below is the collaboration class diagram of an overview of our system.

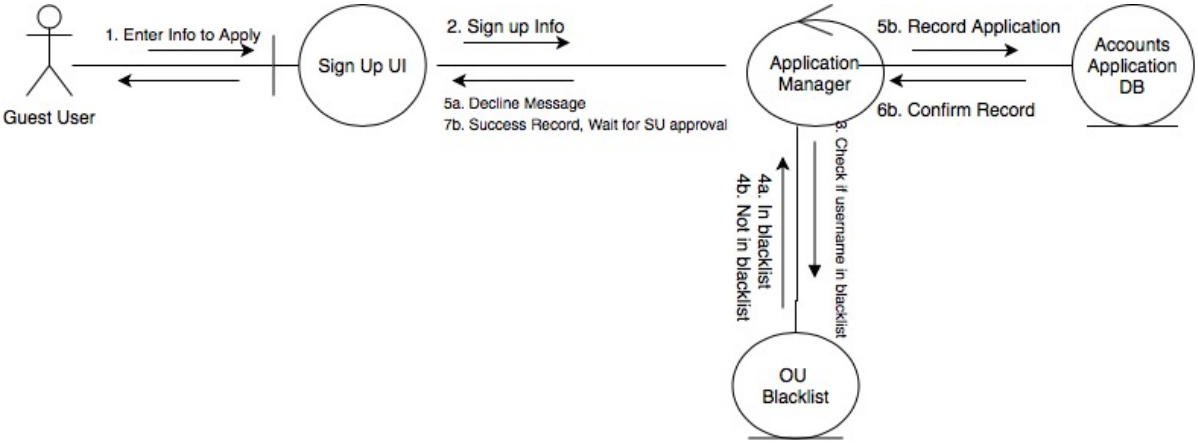


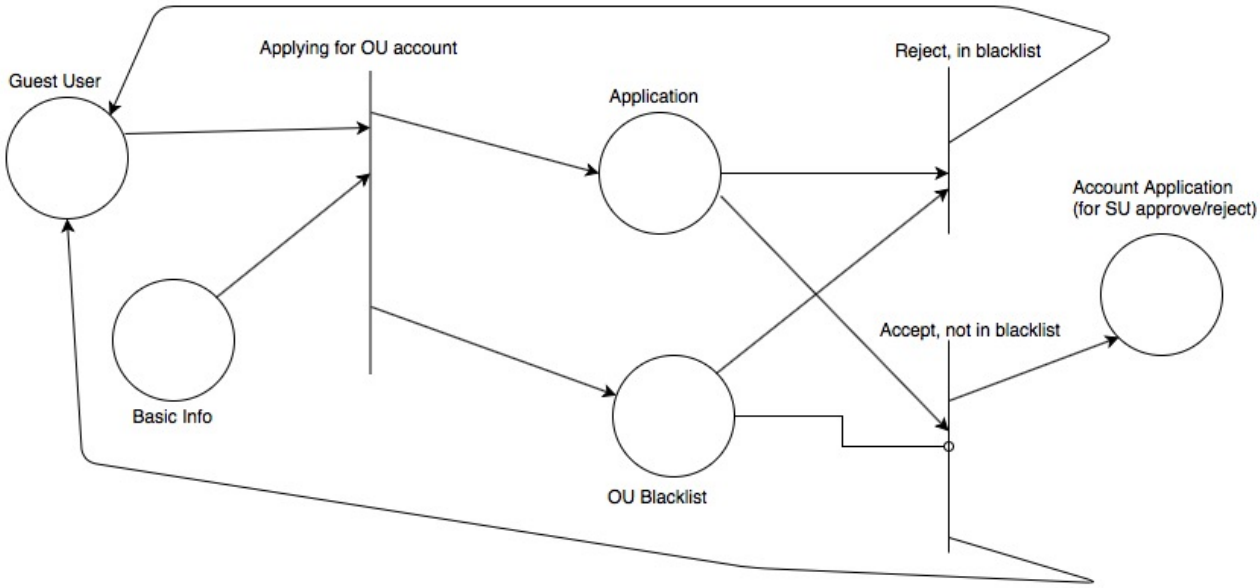
2. Use Case Analysis

The collaboration class diagram and petri-net will be shown for each use case.

2.1. Sign Up

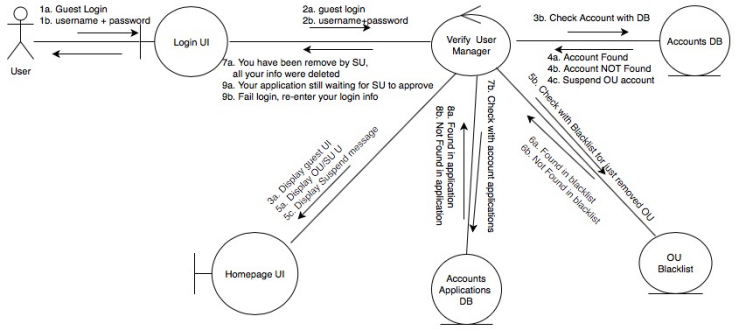
Only guest users are allowed for applying to be ordinary users. If the username that guest user used to sign up the new OU account is in blacklist or already used by other OU, then the application will be automatically decline, else wait for super user to approve.

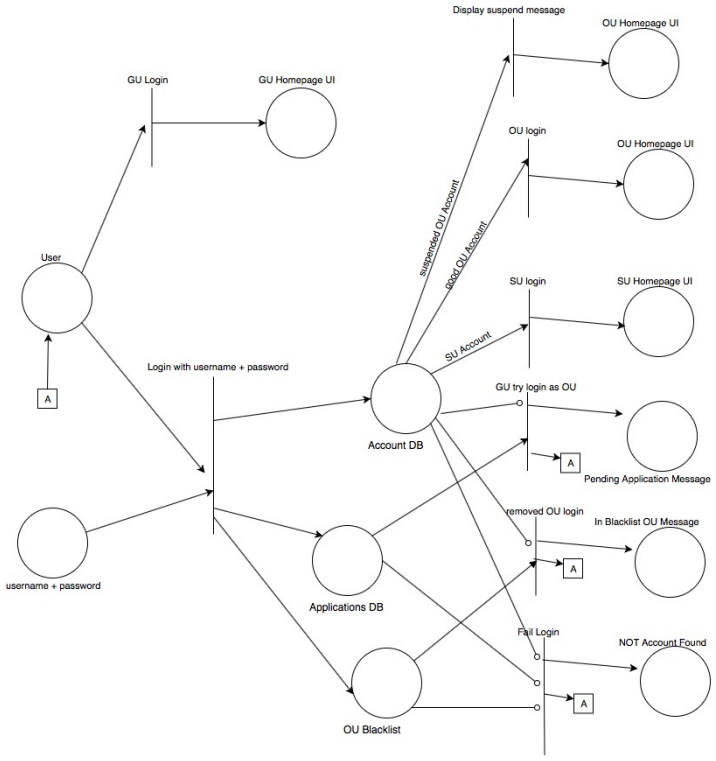




2.2. Login

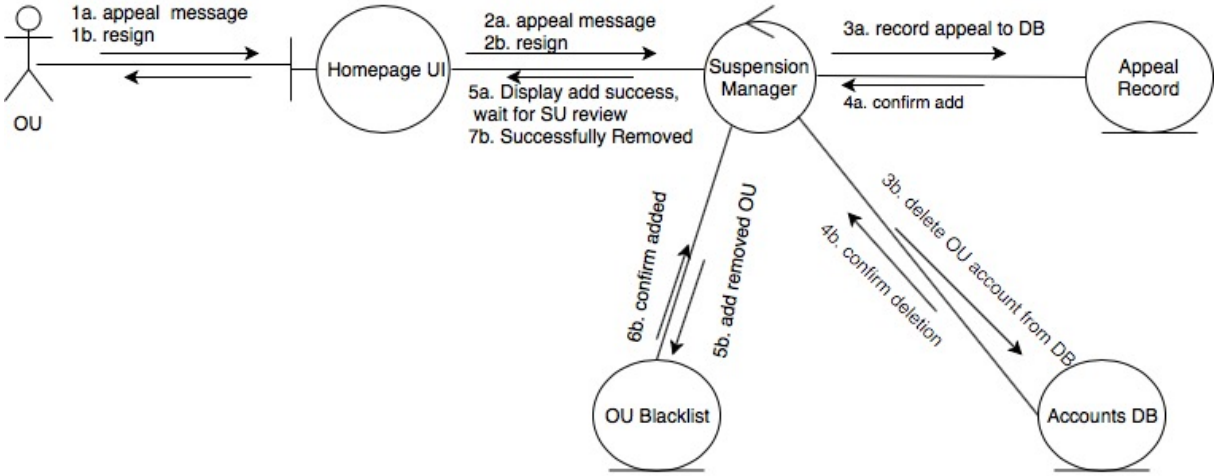
Only OU and SU that are already registered in the system are allowed to login. They have to enter their username and password, then verify with the information store in the database. If the information does not match, they cannot go to their home page and require to login again or stay in the GU homepage. OUs that are removed by SU can only login once to confirm their information are deleted. GU don't require login to browse and search the items, and if GU apply for OU account and try to login, they will receive the pending message if the application is still pending for SU to review. Each user will have their unique ID that is given by our system and unique username for login. Below is the user case and petri-net for login to user homepage.

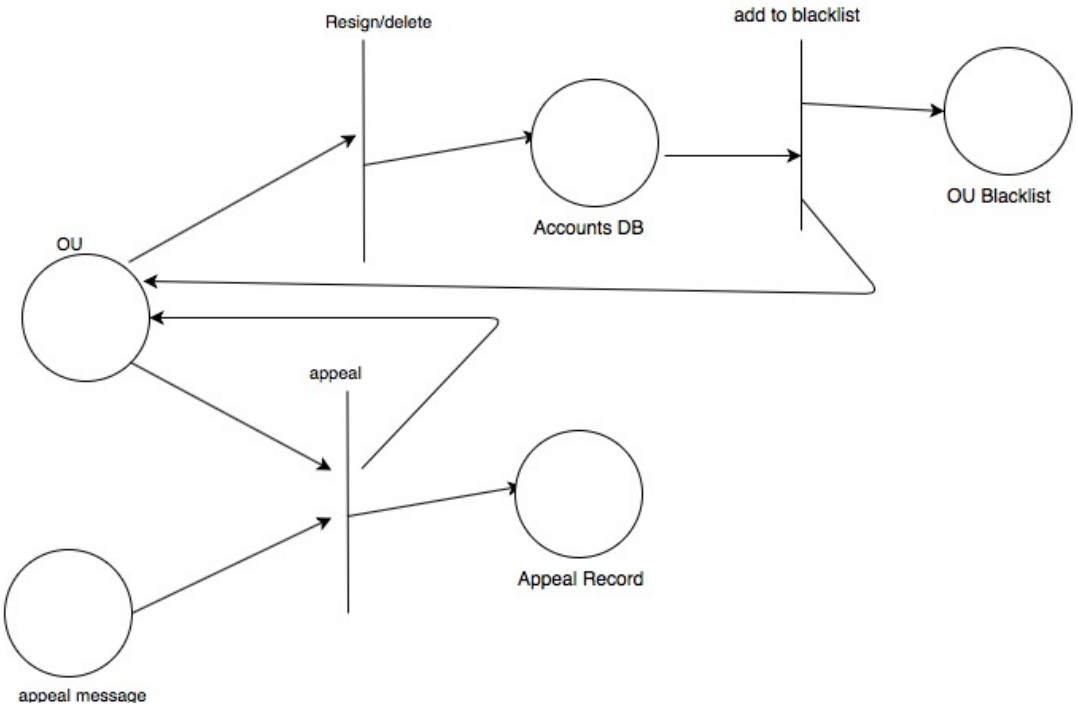




2.3. Appeal/Resign Suspension

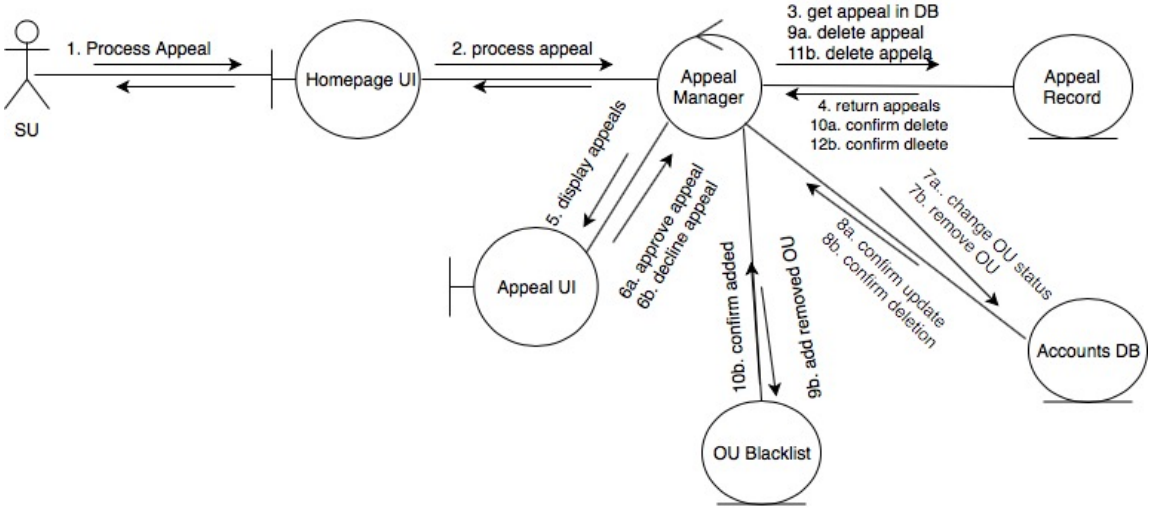
OU account will be suspended if they receive two or more warnings. They can choose to appeal or resign the suspensions. If they choose to appeal, then they need to write the appeal message and wait for SU to approve or deny.

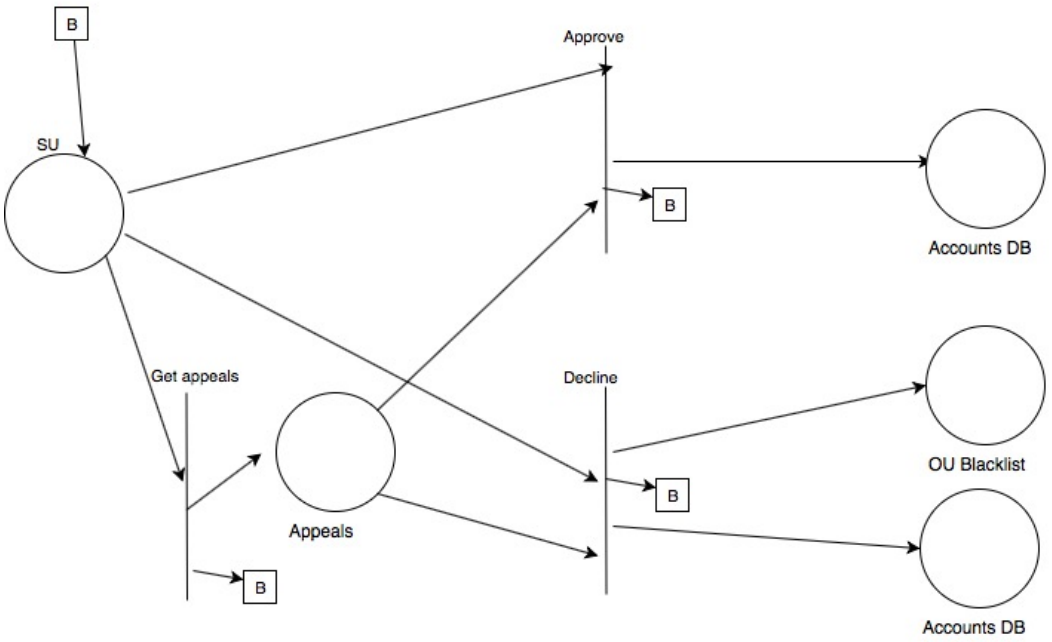




2.4. Process Appeal

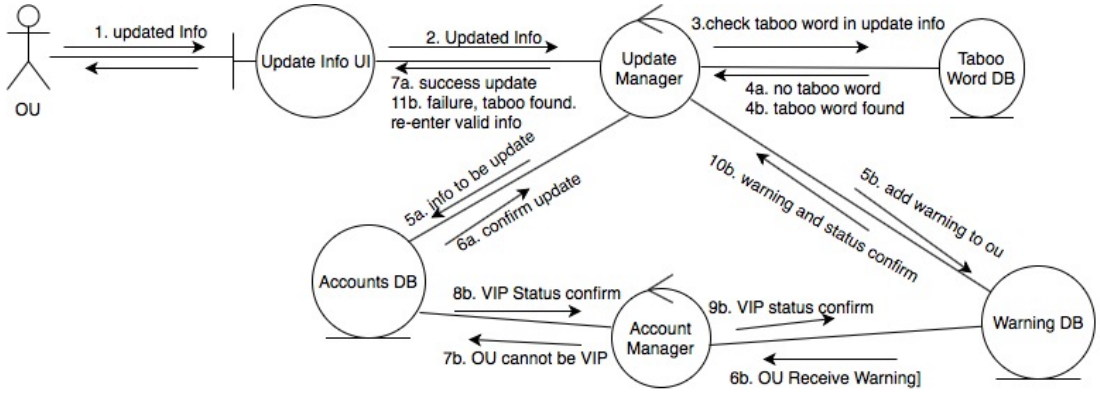
SU can either approve or deny the appeal submitted by OU. If approve, it will change the OU status. If deny, delete the OU from account database and add to blacklist.

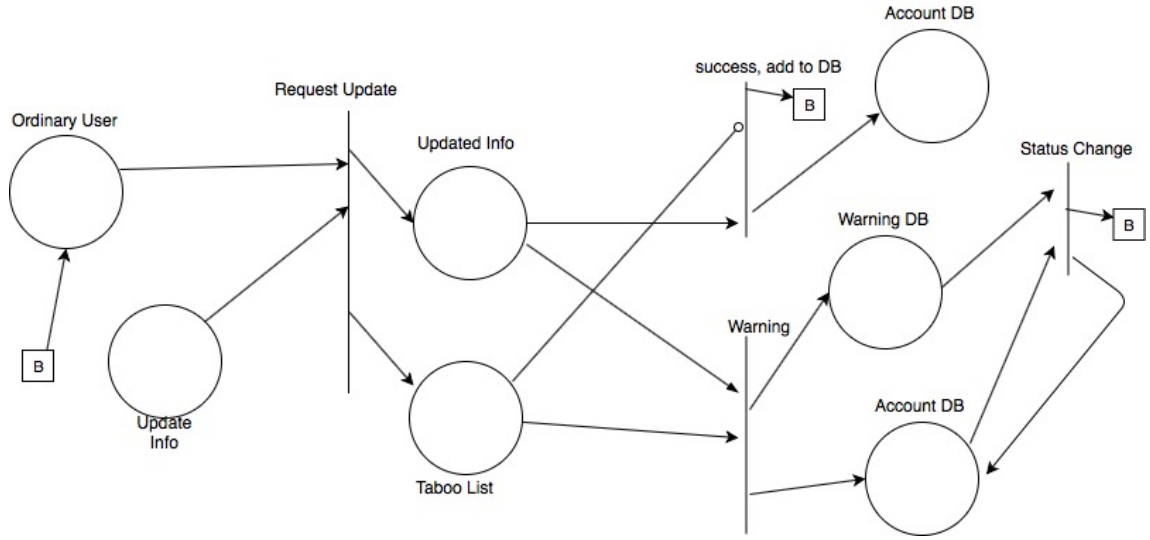




2.5. OU Update Info

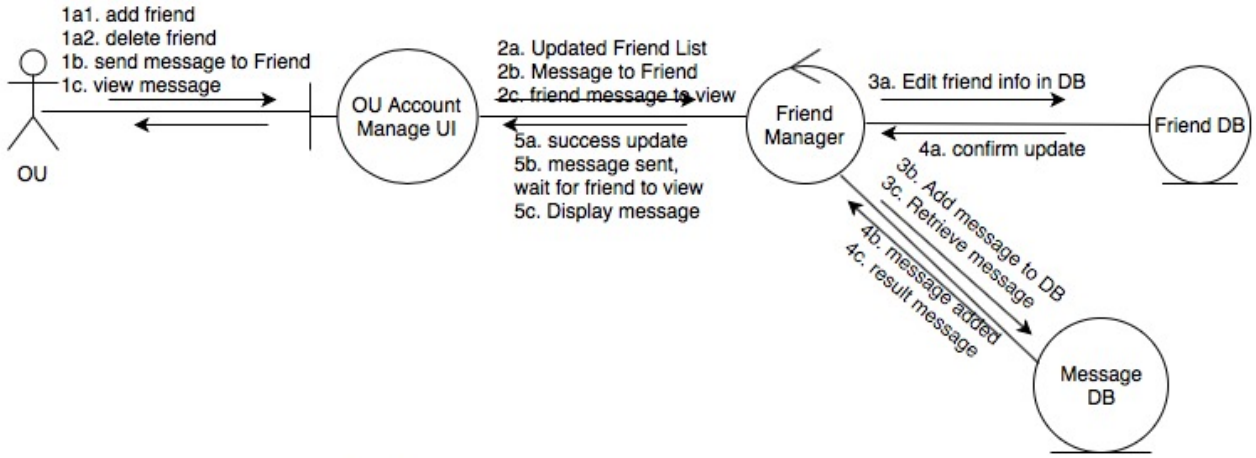
OU can change their password and update their profile information, including name, address, phone, and credit card number. As they submit the updates, system will check if any taboo word appeared in the information they enter. If no taboo word founded, then the update success. If founded, OU will receive a warning and require changing the update information and remove the taboo words. Also, a warning can lead to de-promote to OU if he/she previously is VIP user, this condition of warning apply to all other use cases below. Below is the user case and petri-net for OU update info.

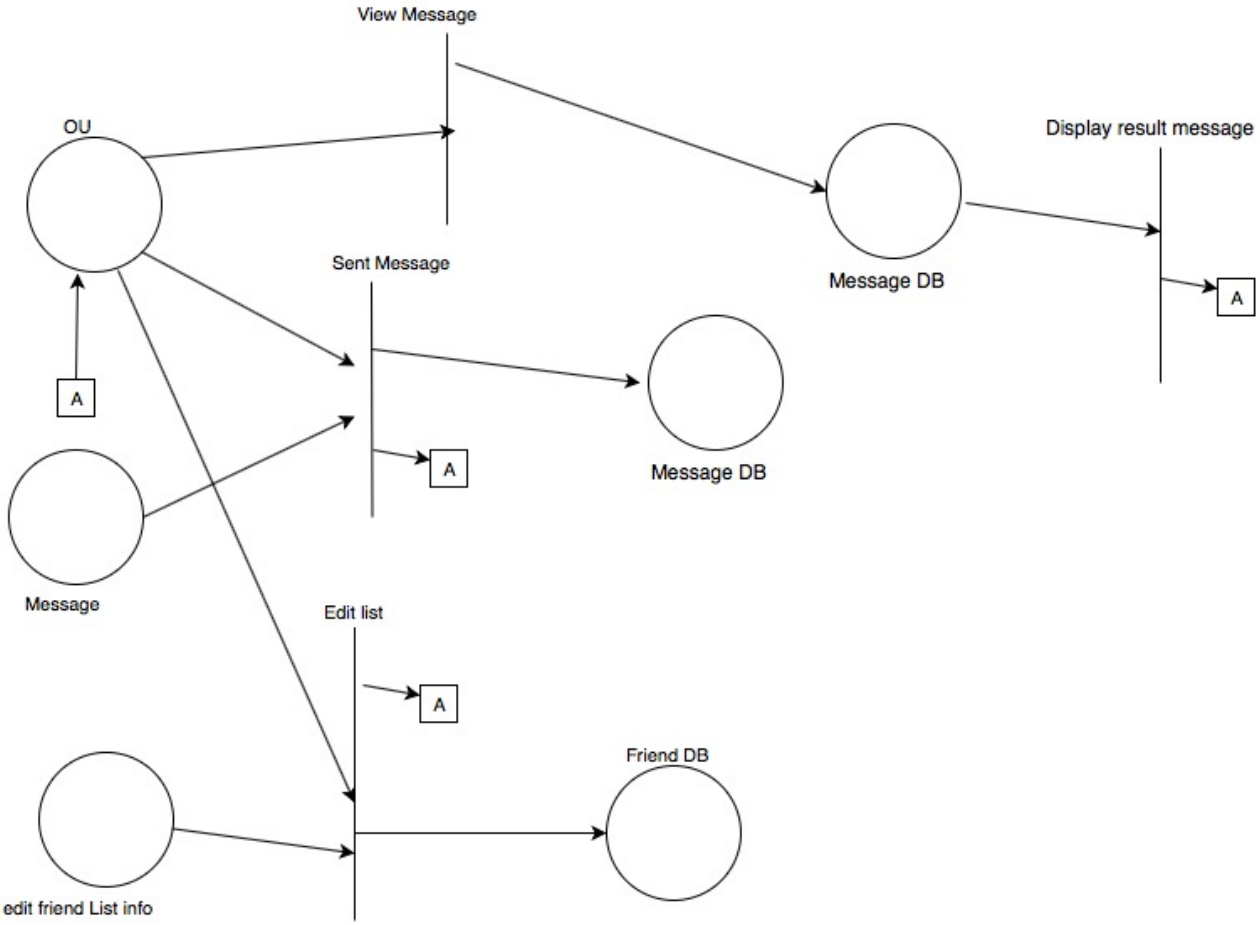




2.6. Friend List

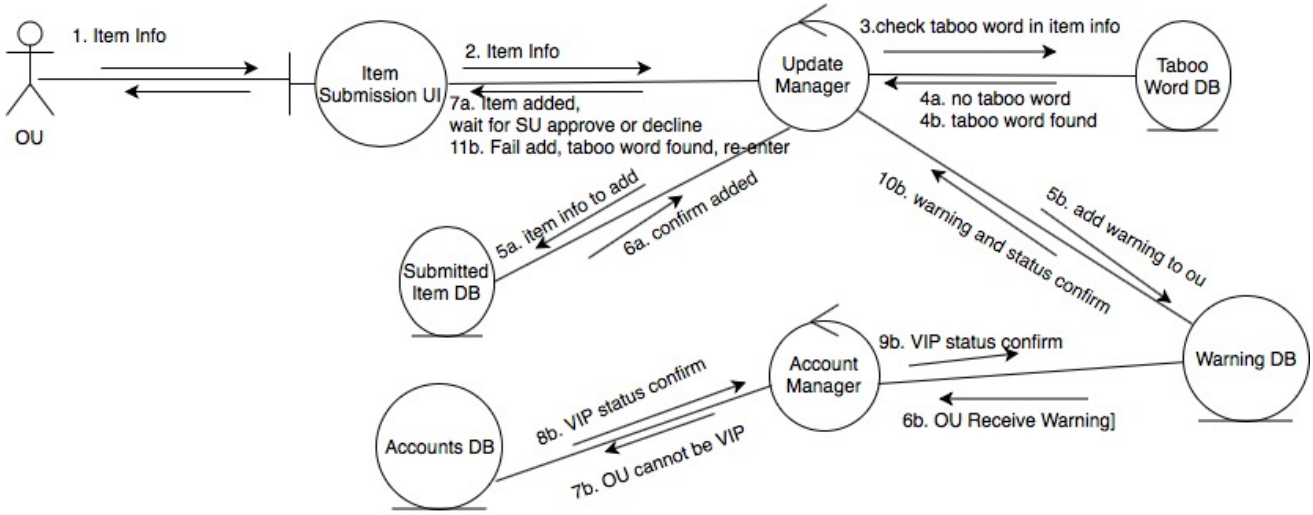
OU can add/delete other OUs to his/her friend list, as well as view and send message to the friend(s). OU receive discount at check out if he/she is the friend of owner of the item that he/she purchase. Below is the user case and petri-net for OU friend list.

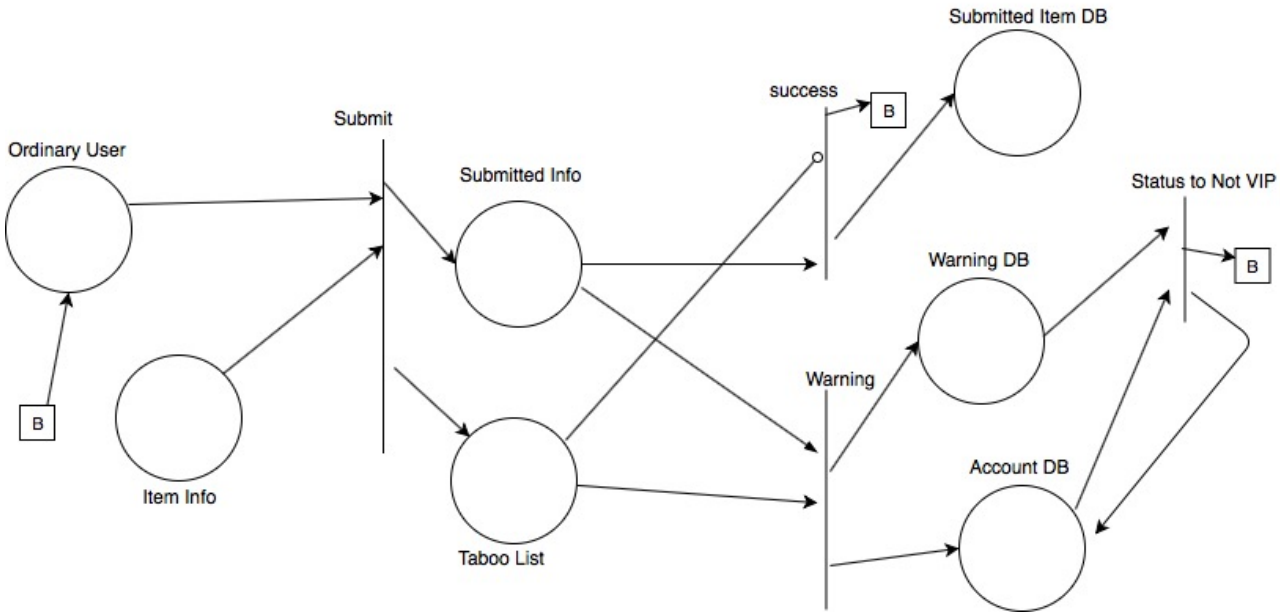




2.7. Submit Item by OU

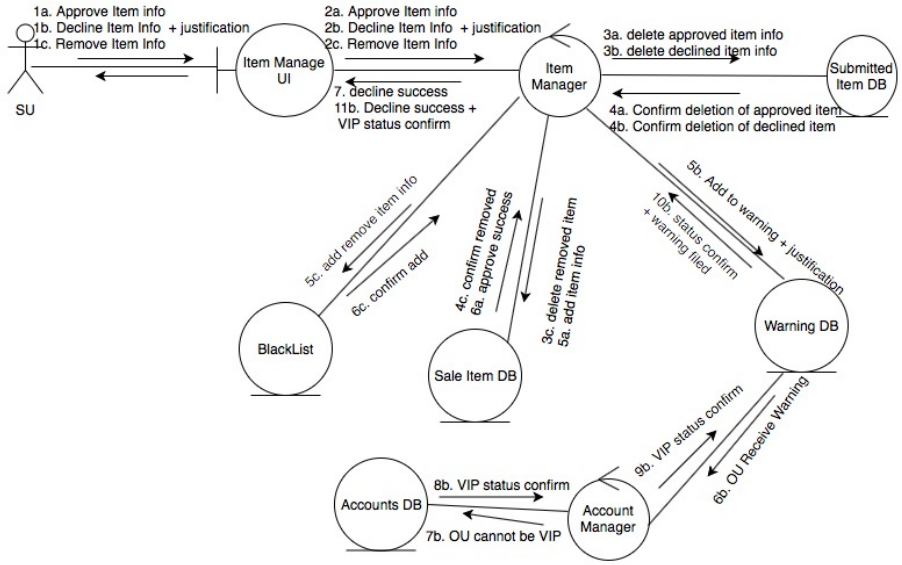
OU can submit the item he/she wants to sell with all item information to SU. If the item information contains taboo word, then OU will receive warning and the submission will not be accepted to database, else the item will be accepted and hold in the item database wait for SU to approve or decline. Below is the user case and petri-net for submit item

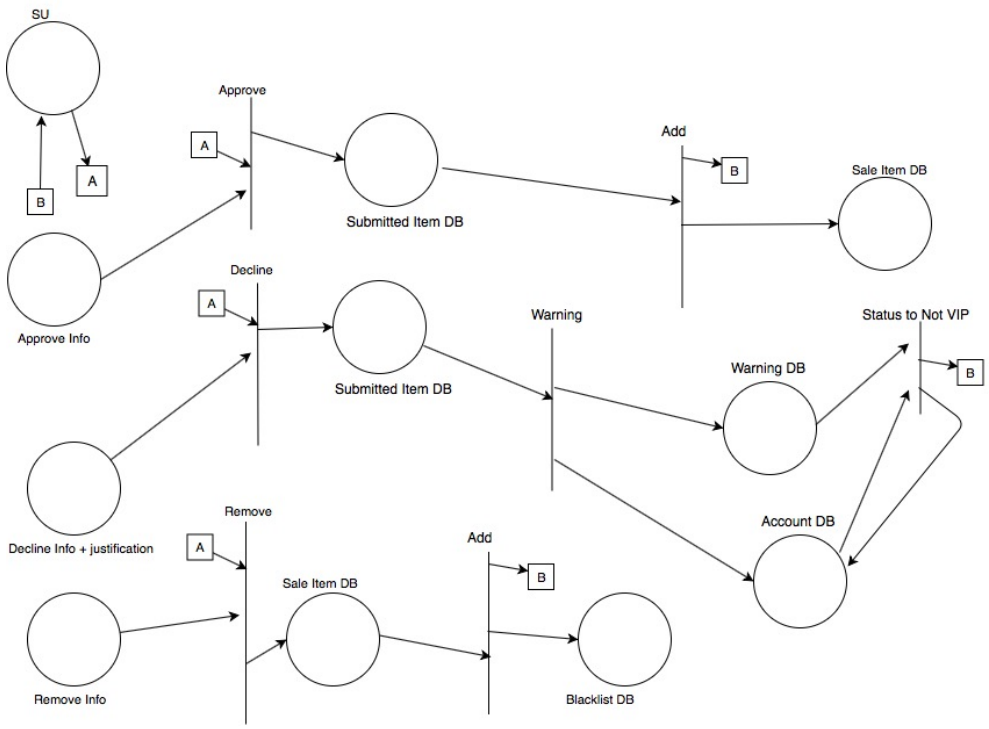




2.8. SU Manage Item

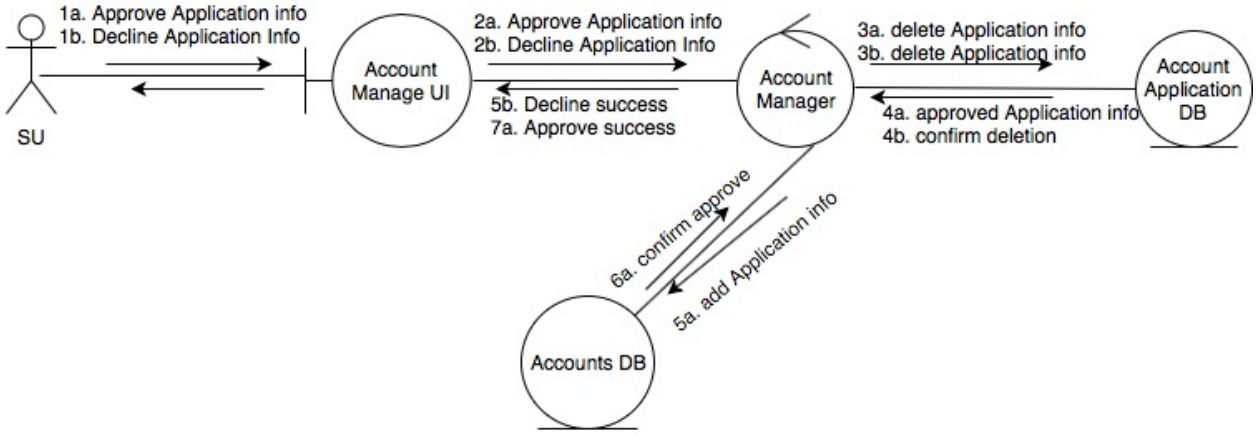
Only SU have the right to approve or decline the item that OU submitted with justification. If the item is declined, the OU who submitted this item will receive a warning. SU can also remove item from the system and removed item will be hold in the blacklist. Below is the user case and petri-net for SU manage item.

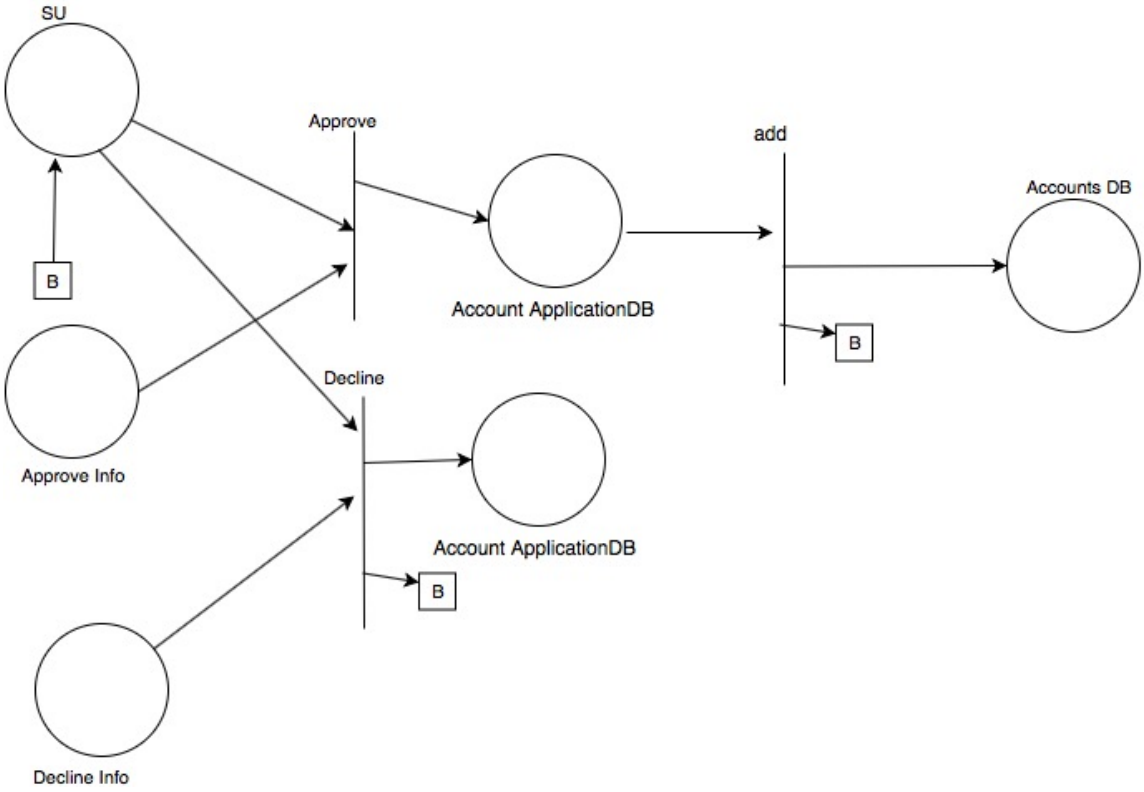




2.9.Approve/Decline OU application

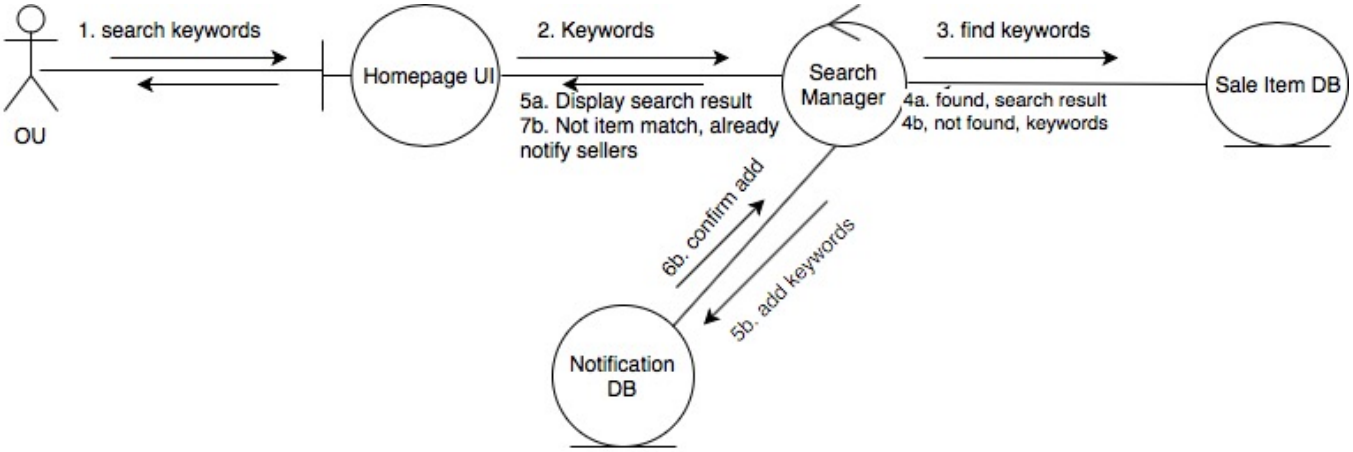
SU can approve or decline the GU application to become OU. If the application get approved, the login password for approved GU will be same as the username he/she registered for the application, and he/she can change the password at the first time they successfully login to the system. Below is the user case and petri-net for SU approve and decline application.

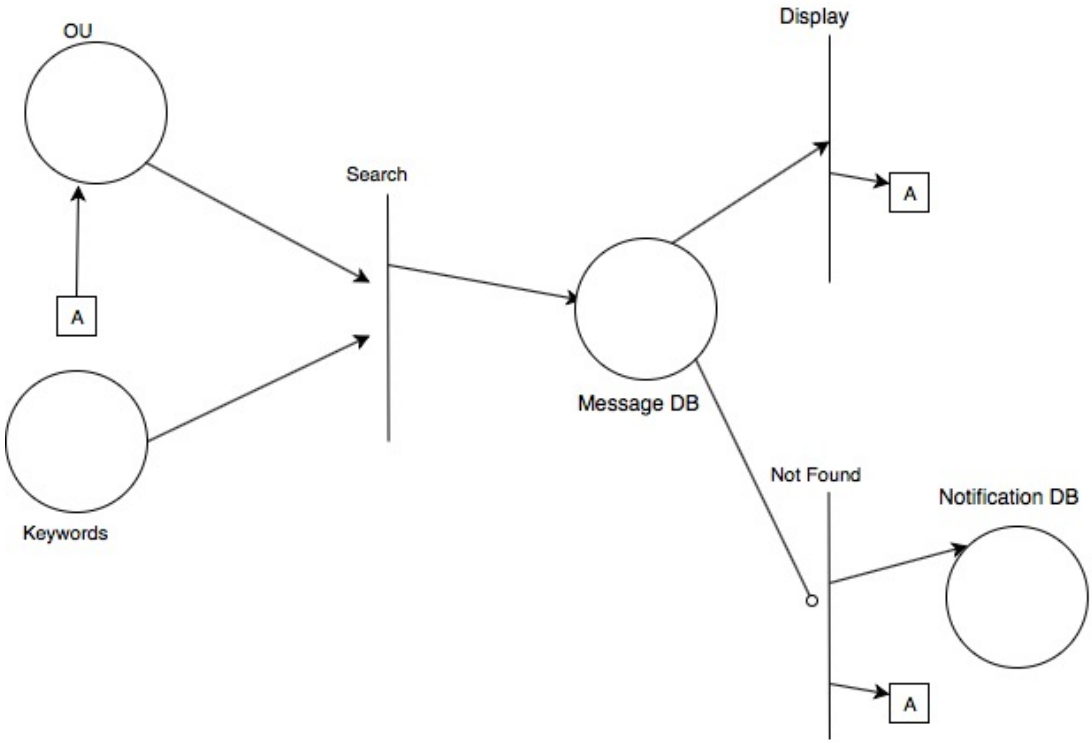




2.10. Search Item

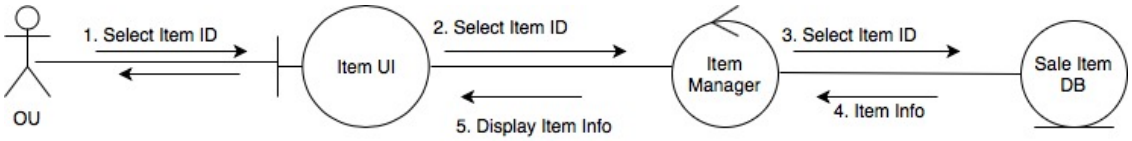
OU and GU can search for any items and the results will be displayed based on the keywords enter. Keywords will be match with the title of the items. If no item match, then the keyword will be in the notified list for OU. Below is the user case and petri-net for search item by OU and GU.

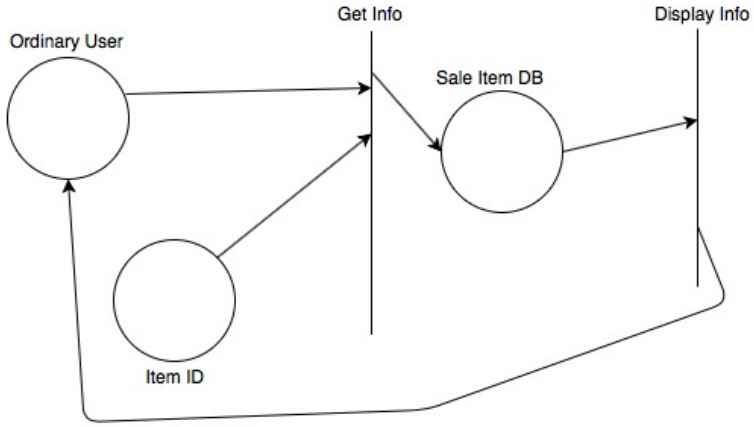




2.11. View Item

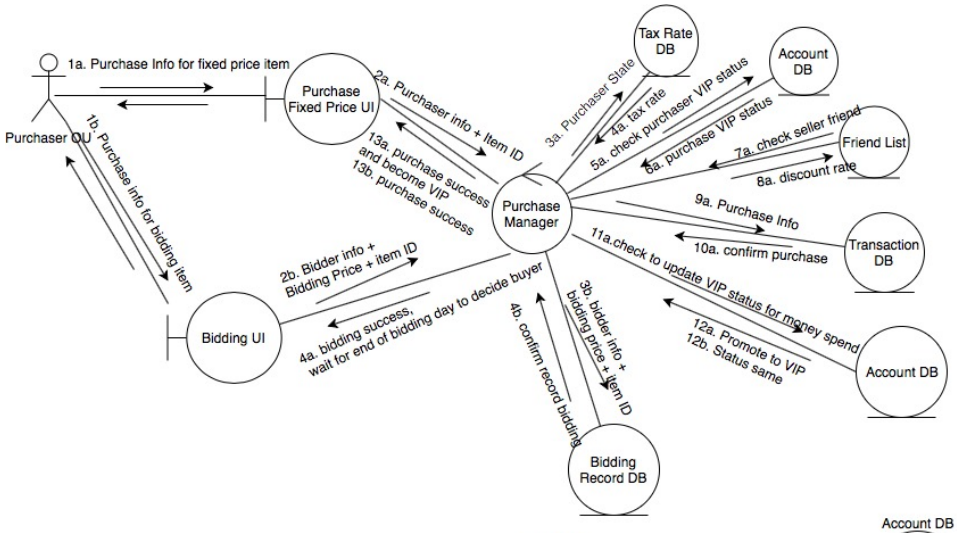
GU and OU can select the item they want to know more about. Below is the user case and petri-net for OU and GU view item.

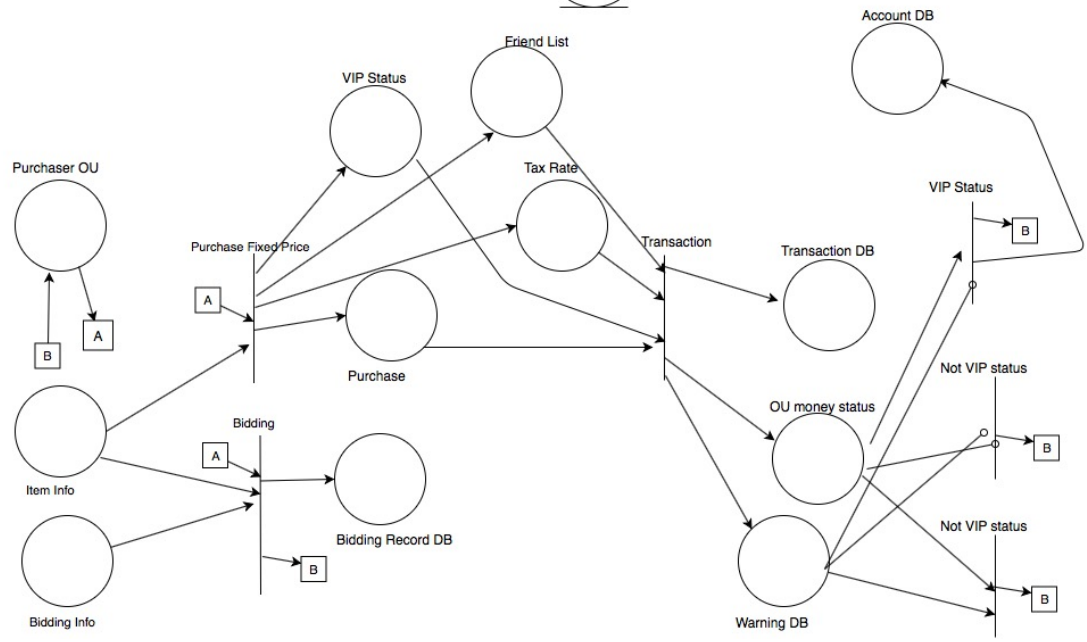




2.12. Purchase Item

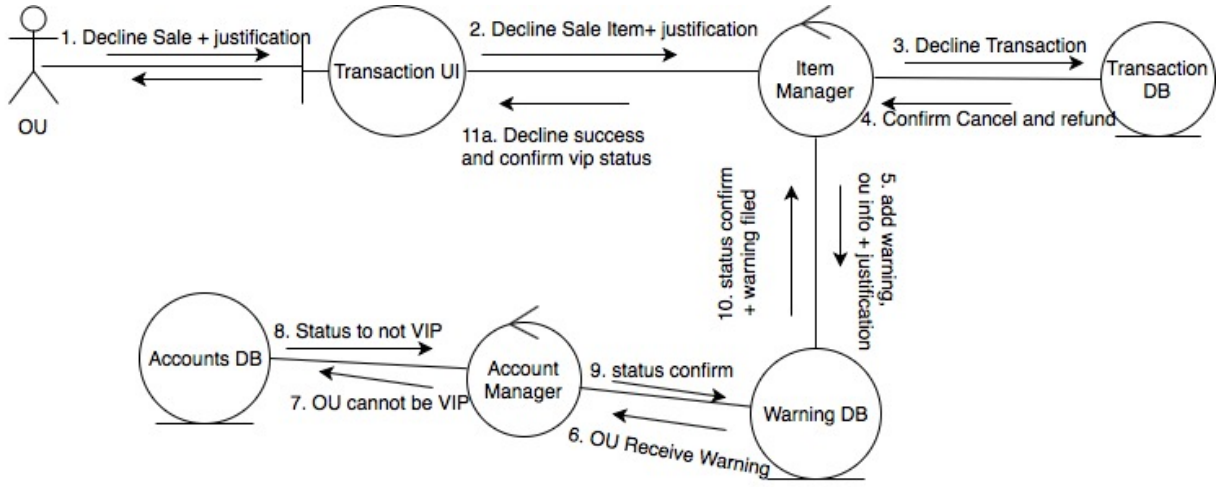
OU can purchase item that is available in the system. Item can either be in fixed price or bidding price. If the item is in fixed price, OU can only buy the item with that price. If the item is for bidding, then OU can enter the price they want to bid for this item. All the bidding has to set an end day, and by the end of that day, second highest bidder will be the buyer of that item. All the purchases will be charge for taxes depend on the address of the purchaser. Discount will be given if the purchaser is VIP or is friend of the seller. After purchasing transaction is done and if the purchaser OU spends $500 without warning, he/she will be promoted to VIP. Below is the user case and petri-net for OU purchase item.

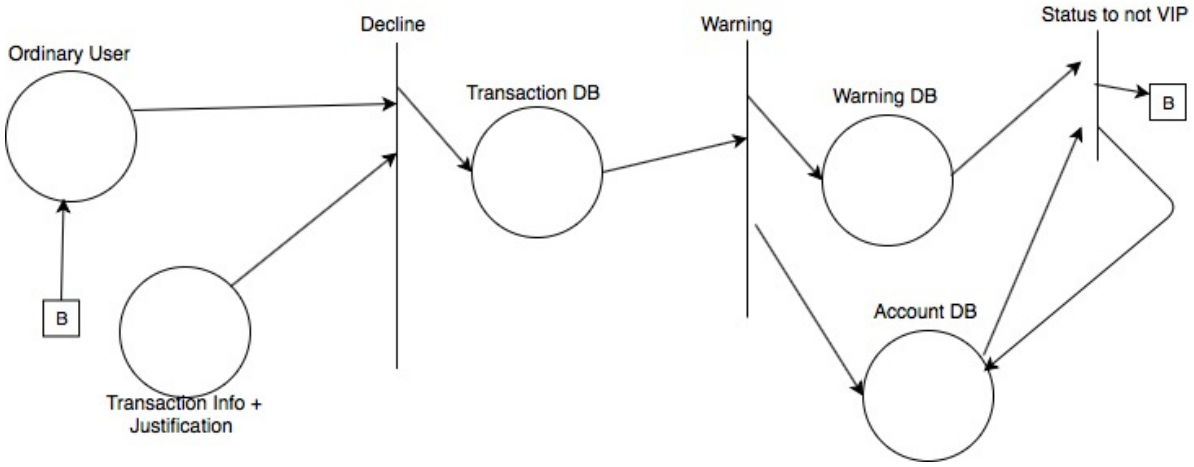




2.13. Decline Transaction

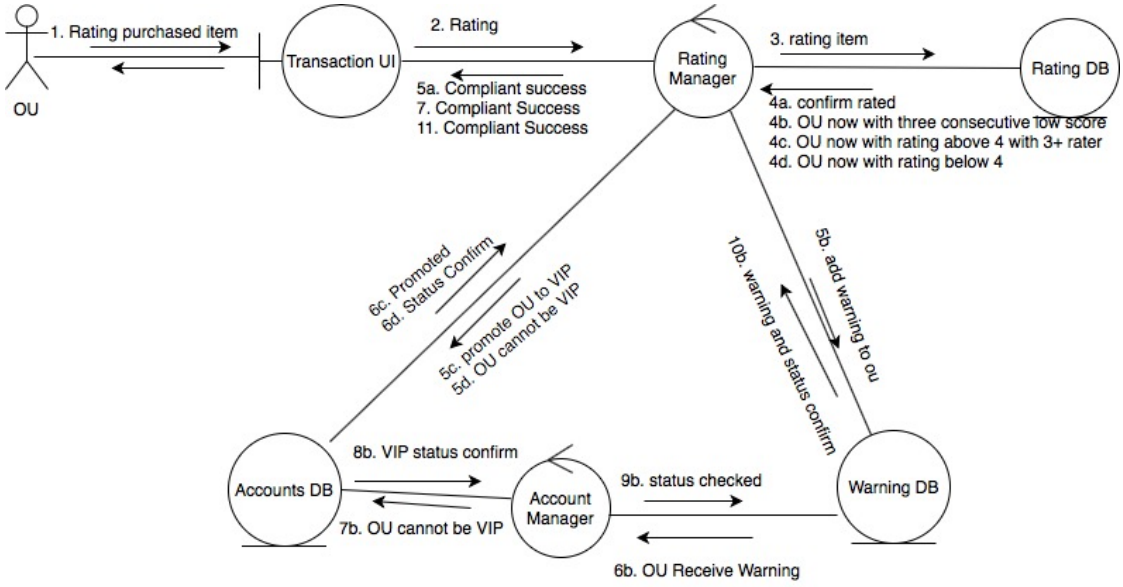
The seller OU can decline the sale with justification, but once declined, seller will receive a warning. Below is the user case and petri-net for OU decline transaction.

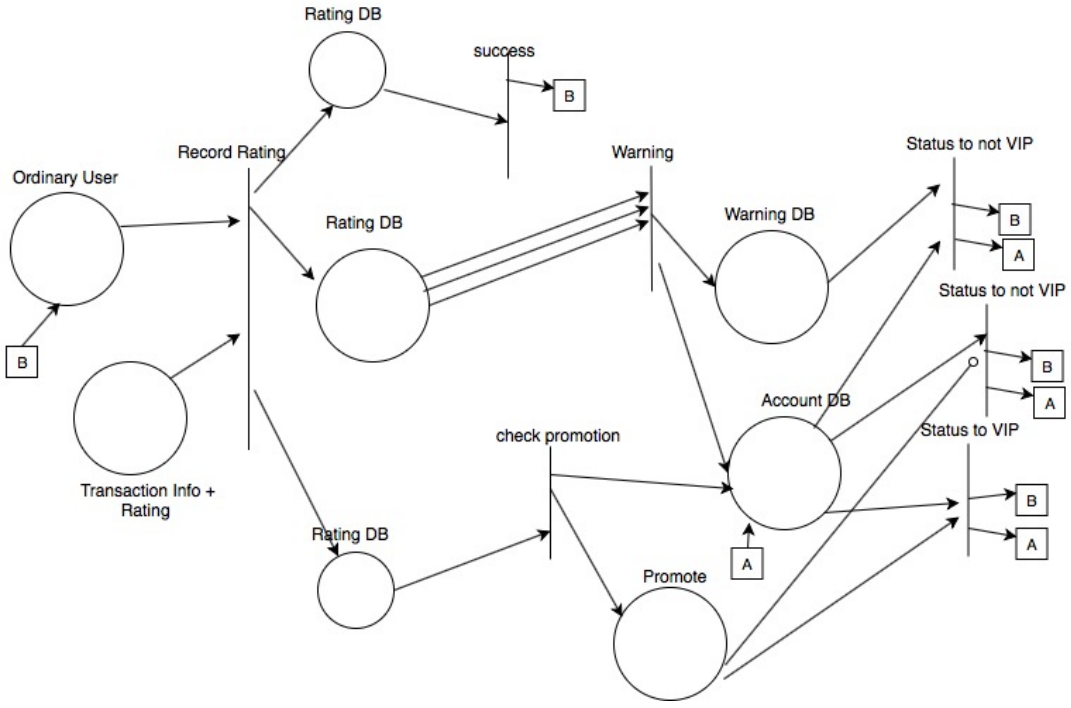




2.14. Rating

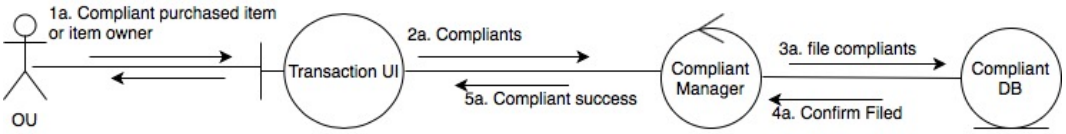
Only OU can rate the purchased item. OU with three 0 or 1 rating in a row will receive a warning. If OU's average rating is above 4 and it is rated by at least 3 different raters, then he/she can be promote to VIP. Or if the VIP OU's rating is below 4, then he/she will get de-promote to OU. Below is the user case and petri-net for OU rating.

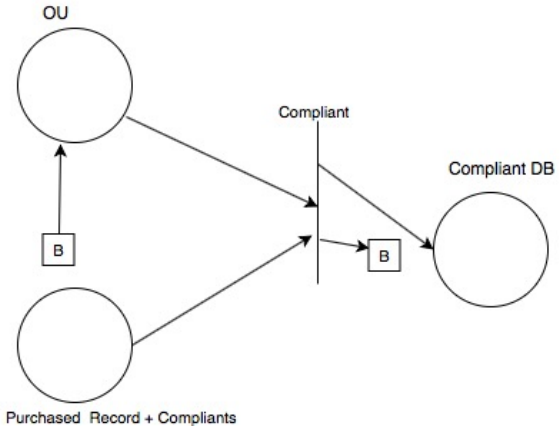




2.15. File Compliants

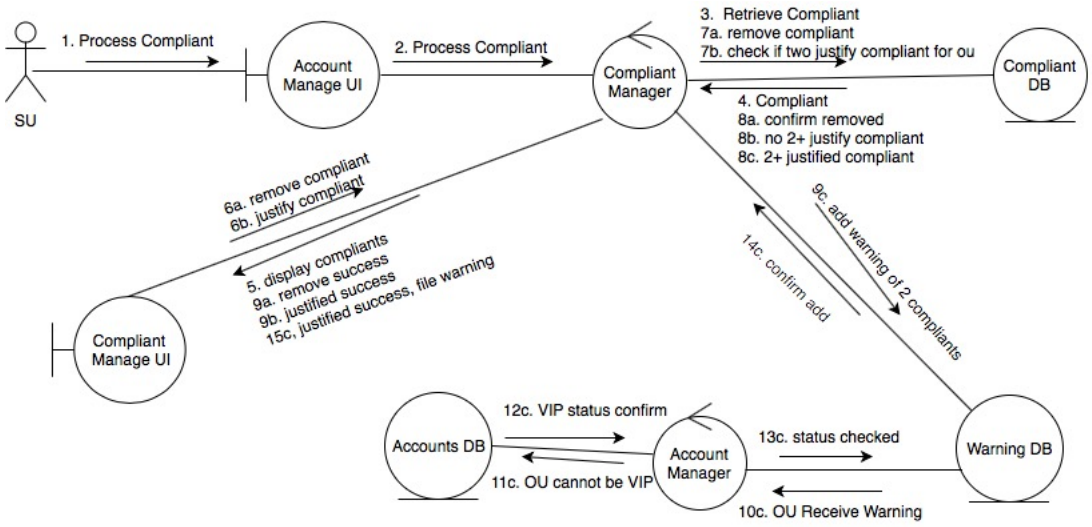
OU who purchased item can submit complaint about the item he/she purchased or complaint about the owner OU of the item.

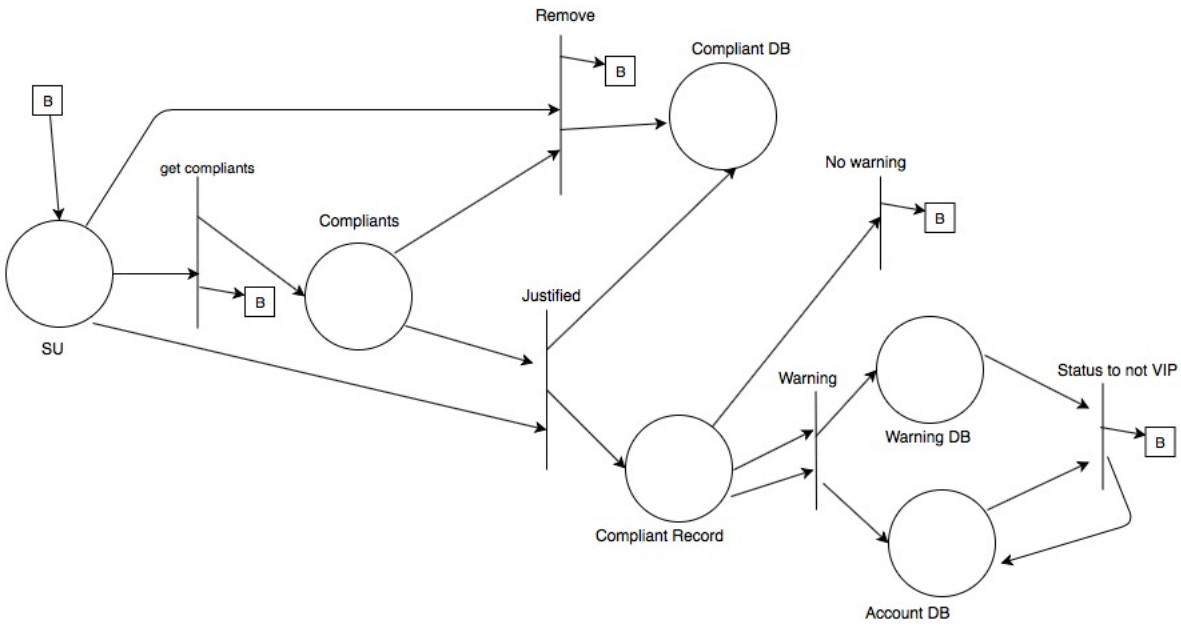




2.16. Process Compliants

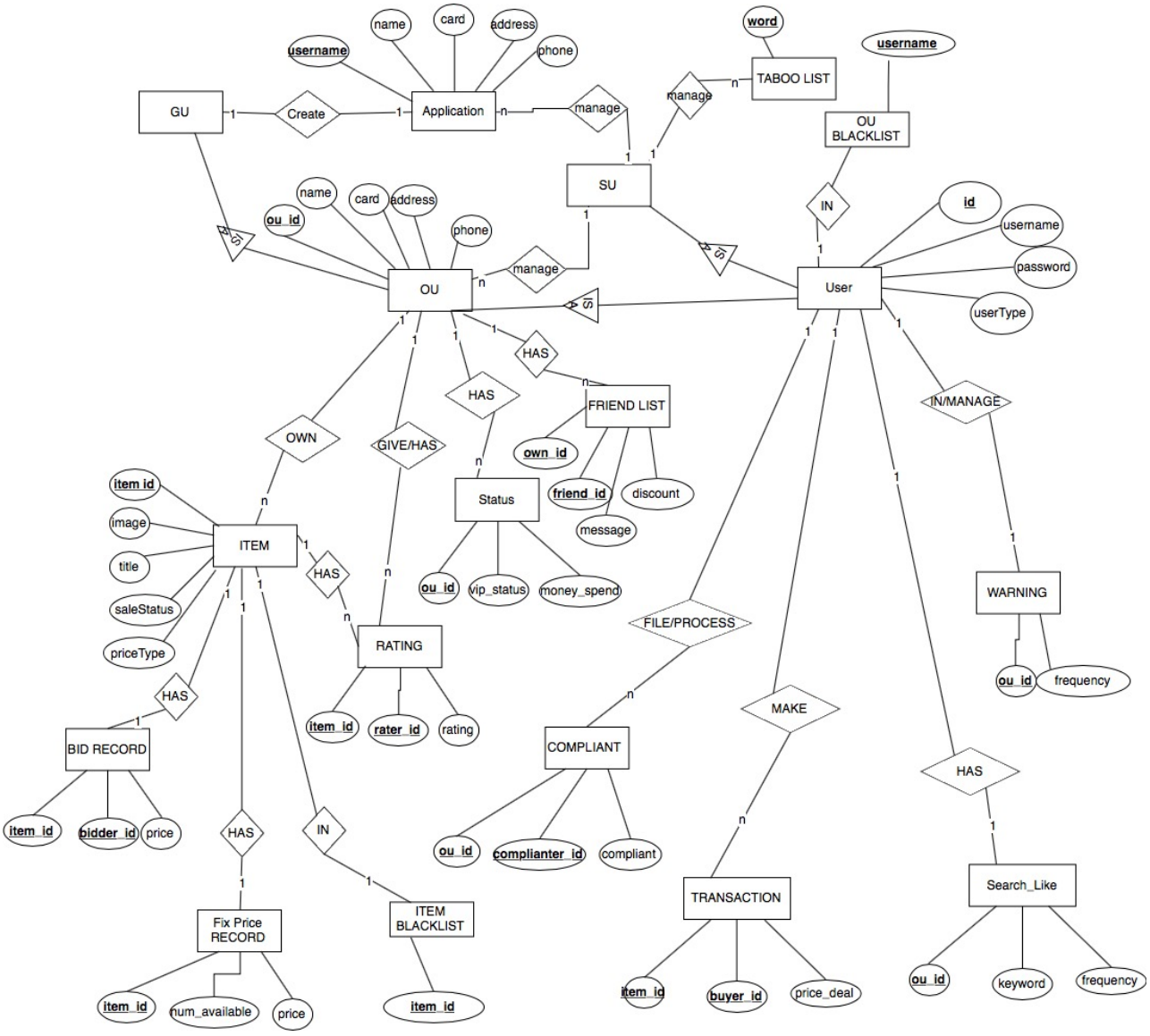
SU can either remove the compliant or approve the compliant. OU with two approved complaints will receive an warning. Below is the user case and petri-net for file and process compliant.





3. ER Diagram

Below is the detailed entity relationship diagram for our system. The bold and underlined words are the keys for respective class. Rectangular boxes represent the class, diamonds represent the type of the relation, ovals represent the attributes of the class and the triangles represent the is-a relationship between the classes.



4. Detail Design

4.1. General Functions

```python
class General():
    def __init__(self,cursor):
        self.cursor = cursor

    def login_check(self, username, password):
        # check login info with DB
        # still need to implement for check OU status if is OU and GU application
        self.cursor.execute(
            "SELECT ID, userType FROM User WHERE username=%s, password=%s",
            (username,password))

        ID, userType = -1, -1
        for user in self.cursor:
            ID = user[0]
            userType = user[1]

        if ID == -1:
            return False
        return {'ID':ID, 'userType':userType}

    def checkStaus(self, ouID):
        # return status of the select OU

    def changeStaus(self, ouID, newStatus):
        # update new status
        # get call when receive warning or low rating
        # return True: for success update
        #        Fasle: for encount error

    def manageWaring(self,ouID):
        # called when add a warning
        # check number warning receive, if greater or equal to 2, suspend OU
        # else make sure that ou is not VIP
        # return True: for success update
        #        Fasle: for encount error


    def ouBlacklist(self,ouID):
        # remove the ou from all DB and all active sale item post by that ou,
        # but not the sold item by he/she, the seller will be None
        # add the username to ouBlacklist
        # return True: for success update
        #        Fasle: for encount error

    def itemBlacklist(self,itemID):
        # remove all the occurance of this item in DB
        # add to itBlacklist
        # return True: for success update
        #        Fasle: for encount error
```

4.2. Item Functions

```python
class Item():
    def __init__(self,cursor, itemID):
        self.cursor = cursor
        self.itemID = itemID


    def itemProfile(self,itemID):
        # need for view item action
        # return image, title, priceType, saleStatus, postTime of the selected item

    def searchItem(self,keywords):
        # look through all items' title, compare with all capitalize letters
        # if found return list of itemProfile, call self.itemProfile(itemID)
        # if not found, add to notification DB and return False
```

4.3. Guest User

```python
class GU():
    def __init__(self,cursor):
        self.cursor = cursor

    def checkUsername(self,username):
        # check if username not exist in DB and not in blacklist return False else return True


    def apply(self, username, email, name, card, address, state, phone):
        # check if username is already exist, not allow for duplicate username
        # If unique username, save the application to DB
        # return True: for success add to DB
        #        Fasle: for encount error
        unique = self.checkUsername(username)
        if unique:
            return False

        qry = ("INSERT INTO GUapplications( username, email, name, cardNumber, address, state, phone) "
               "VALUE (%s,%s,%s,%s,%s);")
        self.cursor.execute(qry,(username, email, name, card, address, state, phone))
```

4.4. Super User

```python
class SU():
    def __init__(self,cursor,userID):
        self.cursor = cursor
        self.ID = userID

    def manageApplication(self, applicationID, action):
        # action == True: approve: register GU to OU account and delete application, set password same as username
        # action == False: decline: delete it from application accounts
        # return True: for success update
        #        Fasle: for encount error

        self.cursor.execute("SELECT * FROM GUapplications WHERE applicationID = %s;" % applicationID)
        for info in self.cursor:
            username, email, name,card,address,state,  = info[2]


    def manageItem(self, itemID, action):
        # action == True: approve: change approvalStatus in ItemDB to True
        # action == False: decline: remove the item in ItemDB, add warning to post OU in warningDB
        # return True: for success update
        #        Fasle: for encount error

    def viewCompliant(self):
        # Get all compliant from DB, return array of dict{itemID, complianerID, description, compliantTime}

    def manageCompliant(self, itemID, complianerID, action):
        # action == True: remove: delete compliant in DB
        # action == False: justified: change the justified to True in DB
        #                  check for two justified compliant that will cause warning
        # return True: for success update
        #        Fasle: for encount error
```

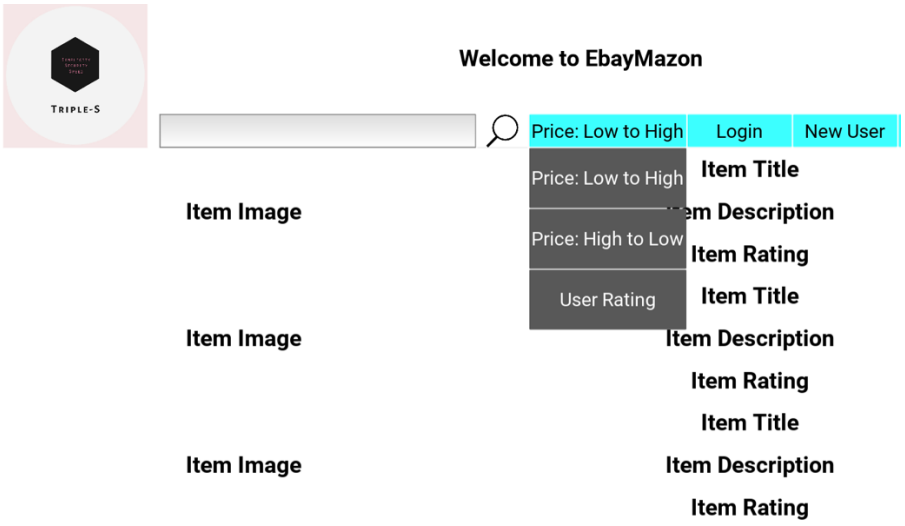
4.5. Ordinary User

```python
from mysql.connector import OperationalError
class OU():
    def __init__(self,cursor,ouID):
        self.cursor = cursor
        self.ID = ouID
        profile = self.getOUInfo()
        self.name, self.card, self.address, self.state, self.phone = self.getOUInfo(ouID)

    def getOUInfo(self):
        # Return ou information in strings: name, card number, address, state, phone
        self.cursor.execute("SELECT * FROM OU WHERE ouID = %s"% self.ID)
        for info in self.cursor:
            return [info[1],info[2],info[3],info[4],info[5]]


    def updateOUInfo(self, name, card, phone, address, state):
        # update OU info in DB
        qry = "UPDATE OU SET name = %s, cardNumber= %s, address =%s, state =%s, phone=%s WHERE ouID=%s;"

        try:
            self.cursor.execute(qry,(name, card,address,state,phone,self.ID))
            return True
        except OperationalError:
            print("Error in update OU Info")
            return False

    def changePassword(self,password):
        #update password in DB
        # return True: for success update
        #        Fasle: for encount error

    def editFriend(self,ownID,friendID,discount = 0.05):
        # add friend relation to DB,
        # each friend can have customer discount, if not provide, default is 5%
        # return True: for success update
        #        Fasle: for encount error

    def submitItem(self,image, title, priceType, usedStatus):
        # add submit item to itemDB
        # return True: for success update
        #        Fasle: for encount error

    def submitFixedPrice(self,itemID, price, numAvailable):
        # add price for fixed price item
        # return True: for success update
        #        Fasle: for encount error

    def submitBiddingInfo(self, itemID, startPrice, endDay):
        # add price for bidding item
        # return True: for success update
        #        Fasle: for encount error
```

```
    def bidding(self,itemID, bidderID, price):
        # record bidding in BidRecord
        # return True: for success update
        #        Fasle: for encount error

    def calculateTotal(self, price, buyerID,itemID):
        # get taxrate from taxDB by buyer address
        # get vip status from buyer
        # check if buyer if friend of owner in friendDB
        # can combine two discount
        # return total cost

    def purchaseFixedPrice(self, itemID, buyerID, numBuy):
        # get price from itemDB by itemID
        # get finalPrice by call self.calculateTotal
        # add to transactionDB and update buyer money spend
        # return True: for success update
        #        Fasle: for encount error

    def purchaseBidding(self, itemID):
        # get called when reach to the endDay of bidding
        # get second highest bidder from BidRecordDB
        # get finalPrice by call self.calculateTotal
        # add to transactionDB and update buyer money spend
        # return True: for success update
        #        Fasle: for encount error

    def declineTransaction(self, itemID, buyerID):
        # No available if the shipping status is true for item
        # Else: remove from transaction, deduct moneyspend from buyer status
        # Add warning to OU
        # return True: for success update
        #        Fasle: for encount error

    def viewTransactionHistory(self,ouID):
        # get transaction History from DB
        # Separate for sell and purchase, return dict{'sell','buy'}
        # each is list of transaction in form[itemID, buyerID, priceDeal, dealTime]

    def submitRating(self,itemID, raterID, rating):
        # add rating to DB
        # check for bad rating that will cause warning or depromotion
        # check for good rating that will cause promotion
        # return True: for success update
        #        Fasle: for encount error

    def submitCompliant(self,itemID, complianterID, description):
        # add compliant to DB
        # return True: for success update
        #        Fasle: for encount error
```

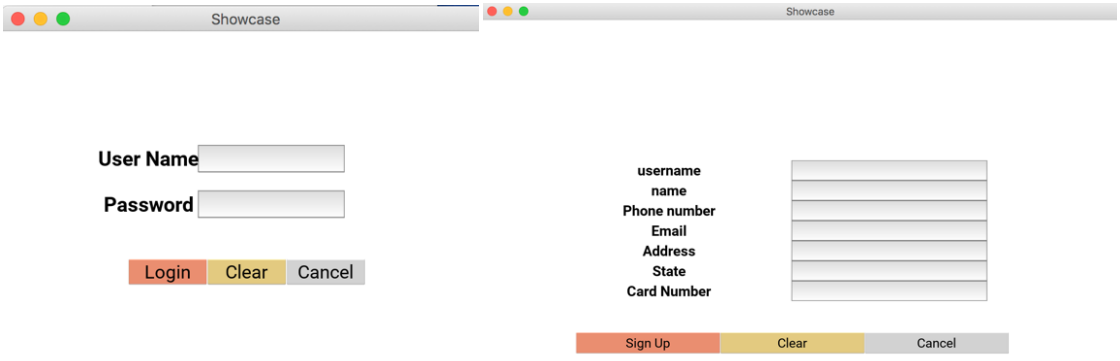
5. System Design

5.1. Home Page



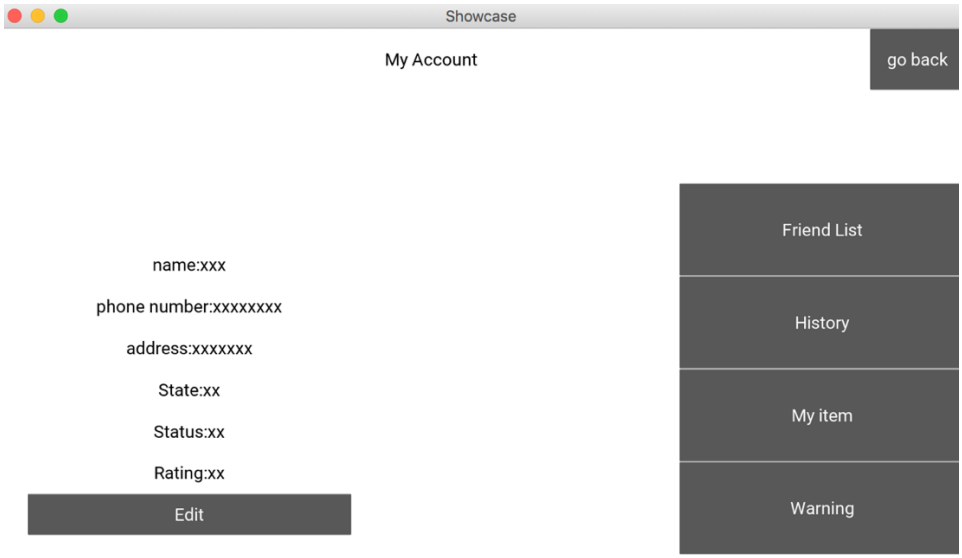
Our home page where display most popular items, currently still need to replace the *'item image'* with actual images from our database. At this page, the GUs are allowed to search for the items they want and can sort by selected order. However, only OUs are able to click into the item to see the details of the item information and perform further actions, such as bidding, purchasing and viewing comments. The upper left corner image is the logo for our website.

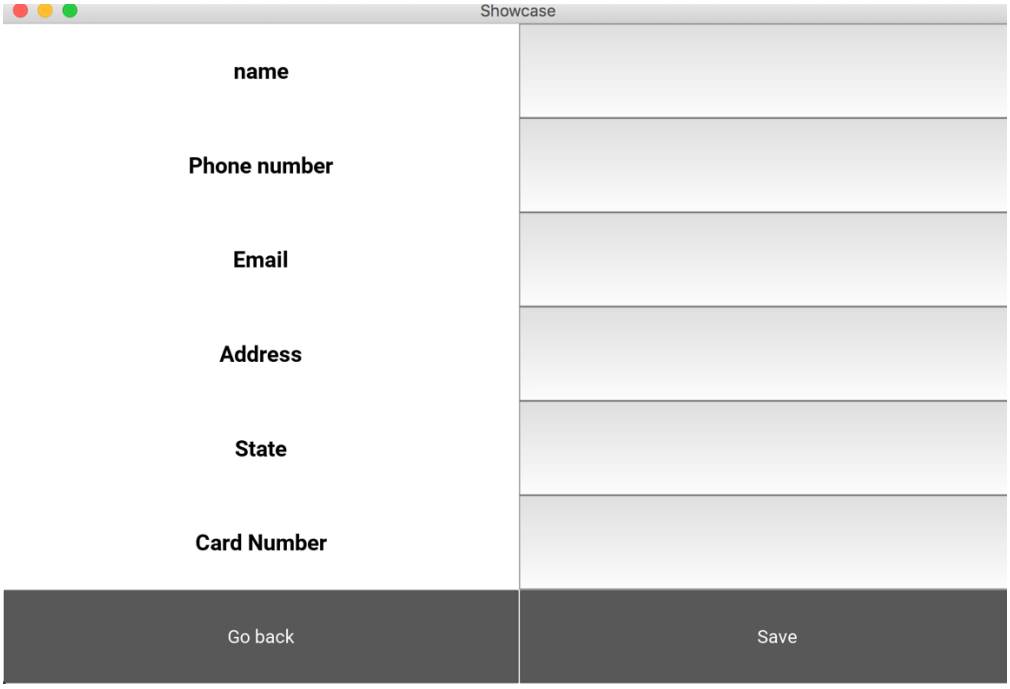
5.2. Login Page and Sign up Page



GU needs to register a valid account in order to become the OU of our website. It is required a unique username for registration since the username is one of the login credentials. Detail will show later.

5.3. OU Profile and edit Profile Info



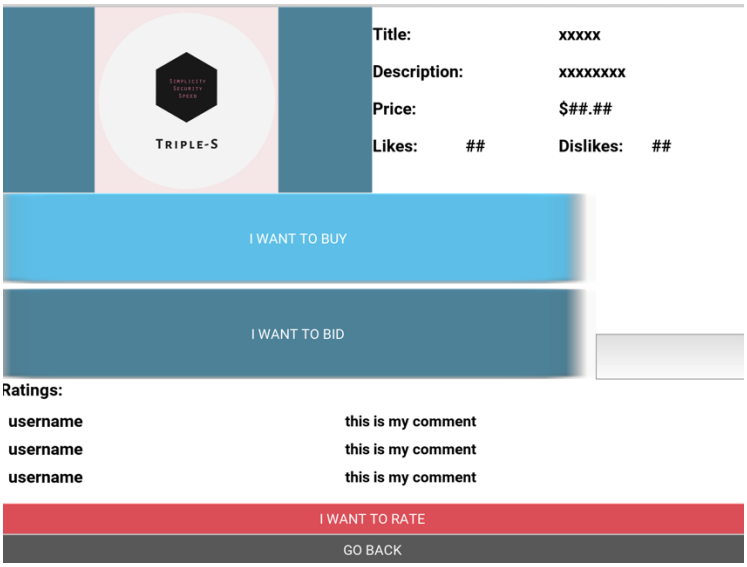


The first image in this page is the account page for OUs, where they can see their information and manage their accounts. The second image will appear when they click on the edit button of the first image to edit the information.

5.4. SU Home Page

The above image is the SU home page, in which each button will lead to different page that allow SU to perform their actions. This page will appear after SU successfully login.

5.5. Select Item Page



This is item page and it will have two versions, one for bidding items and another for fixed price items. The above is combine of both types, we will separate the page later. For bidding items page, it will not have comments from other OUs, but other information will be same. For fixed price items page, it will show how many items are still available and the input will be the number of items purchaser wants to buy.

5.6. Sign Up Functions Demo

5.6.1. User current store in our database

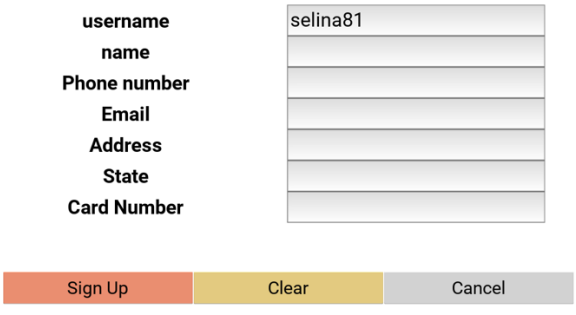
5.6.1.1. userType == 1 are for SU

5.6.1.2. userType == 0 are for OU

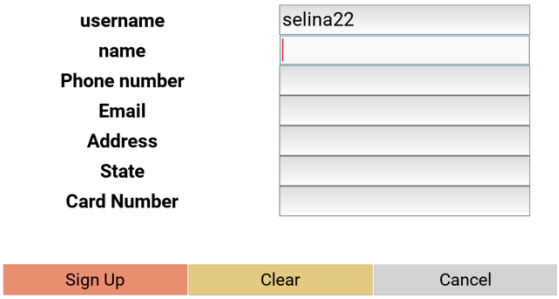
	ID	username	password	userType
1	1	admin	password	1
2	2	selina81	1031	0
3	3	hebe83	0330	0
4	4	ella84	0618	0
5	5	lu7	0420	0

5.6.2. Try to Sign up

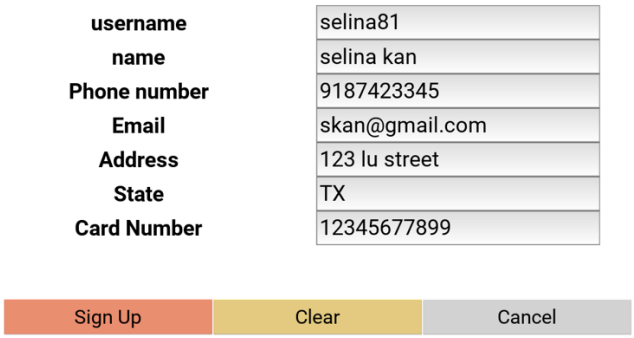
Username already Existed



Username can be used



Fail to Apply



Username will be check after user enter input for username, system will notify whether the username can be used or not, and usernames are not allowed duplicate. Since *'selina81'* already existed in system database, so the system will not accept this application to the database unless user change the username to a unique username. If application is accepted, it will go back to the homepage. We still need to check with Blacklist for the username to ensure that the username is not blocked.

5.6.3. Code

5.6.3.1. GUI Layout

```
Signup:
    id: signup
    name: "signupPage"
    title: "Sign Up"
    BoxLayout:
        pos_hint: {'center_y': 0.5, 'center_x': 0.6}
        orientation: "vertical"
        BoldLabel:
            id: userRepeat
            text: ""
            color:
InfoInput:
    size_hint_x: 0.7
    id: GUusername
    on_text_validate: root.checkUsername(GUusername.text)
LoginButton:
    text: "Sign Up"
    on_press:root.signUp(GUusername.text, GUname.text, GUphone.text,
                         GUemail.text, GUaddress.text,GUState.text,GUcard.text)
    pos_hint: {'center_y': .7, 'center_x': 0}
    background_color: [2.37, 1.13, .68, 0.7]
    size_hint_y: .15
```

5.6.3.2. Main File Call

```
def signUp(self,username, name, phone, email,address,state,card):
    applied = guest.apply(username, name, email,card,address,state,phone)
    if applied:
        self.ids['userRepeat'].text = ""
        self.ids['screenmanager'].current = "homepage"
    else:
        self.ids['userRepeat'].text = "Fail to Apply"


def checkUsername(self, username):
    nameCheck = guest.checkUsername(username)
    if not nameCheck:
        self.ids['userRepeat'].text = "Username already Existed"
    else:
        self.ids['userRepeat'].text = "Username can be used "
```

5.6.3.3. Database Function

```
def checkUsername(self,username):
    # check if username not exist in DB and not in blacklist return False else return True
    self.cursor.execute("SELECT * from User;")
    k1 = self.cursor.fetchone()
    qry = "SELECT EXISTS(SELECT * from User WHERE username='%s');" % username
    self.cursor.execute(qry)

    k = self.cursor.fetchone()[0]
    if k:
        return False
    return True
    # pass

def apply(self, username, name, email, card, address, state, phone):
    # check if username is already exist, not allow for duplicate username
    # If unique username, save the application to DB
    unique = self.checkUsername(username)
    if not unique:
        return False
    try:
        qry = ("INSERT INTO GUapplications( username, email, name, cardNumber, address, state, phone) "
               "VALUE (%s,%s,%s,%s,%s,%s,%s);")
        self.cursor.execute(qry,(username, email, name, card, address, state, phone))
        return True
    except mysql.connector.Error as ERR:
        print(ERR)
        return False
```

6. Meeting Record

Meeting Day	Meeting Time	Purpose
March 18	3:30pm – 5:00 pm	Discuss project detail and design database schemas. Spilt work for first report.
March 20	3:30pm – 5:00 pm	Update current process, draw the Use-Case Diagram
April 10	3:30pm – 5:00 pm	Discuss the layout for GUI pages, separate work. Create the git repository.
April 15	3:30pm – 5:00 pm	Update the work and further discuss the page in detail. Modify the pervious design.

7. GitHub Repository

eByMazon (https://github.com/jinchen1036/eByMazon)