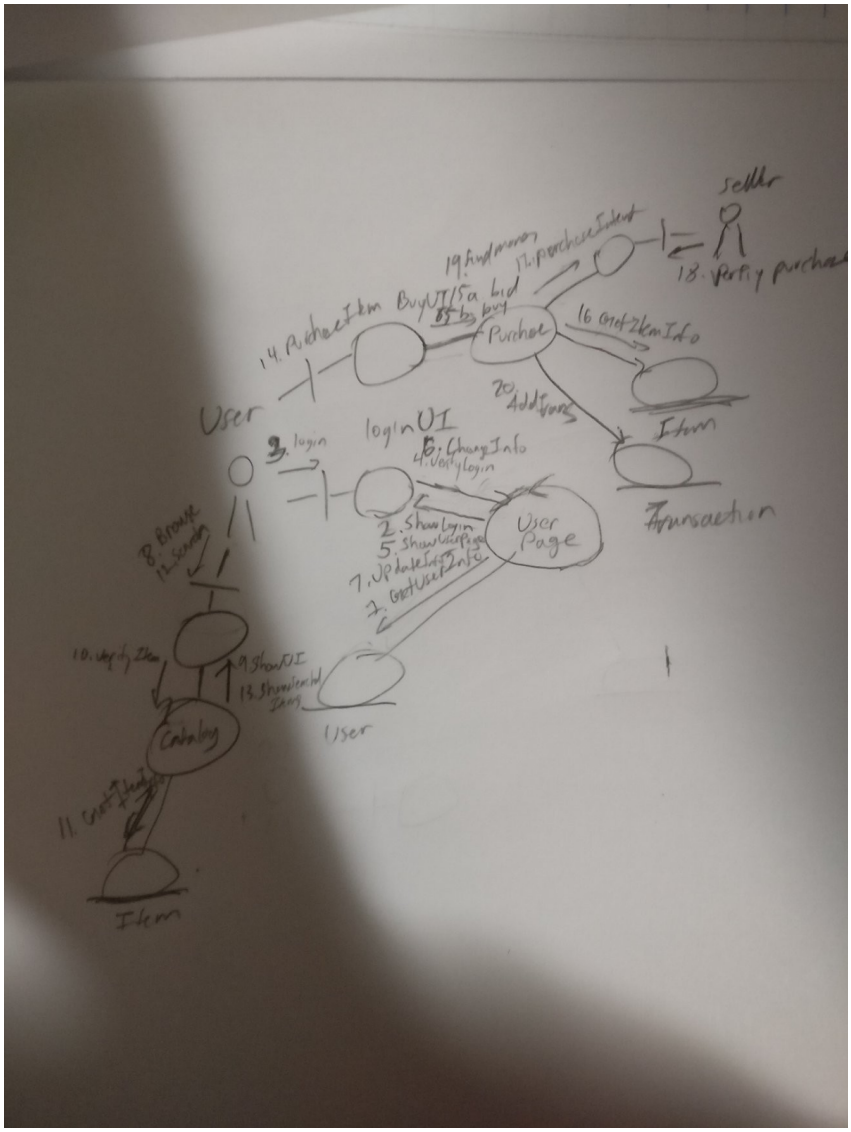
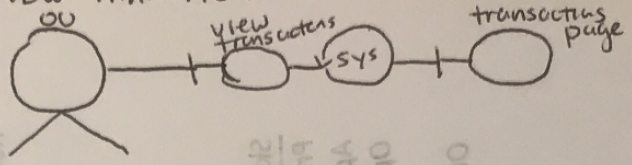
1. Introduction



2. All use cases

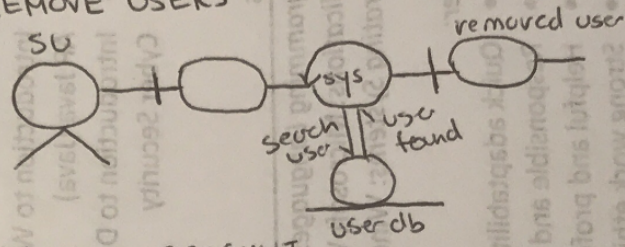
- **Collaboration or sequence class diagram for each use case**

VIEW TRANSACTIONS

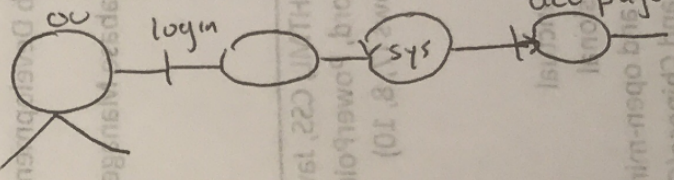


(same for SU)?

REMOVE USERS

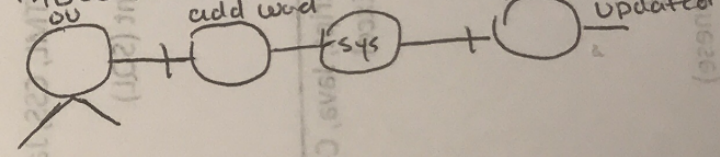


MANAGE ACCOUNT



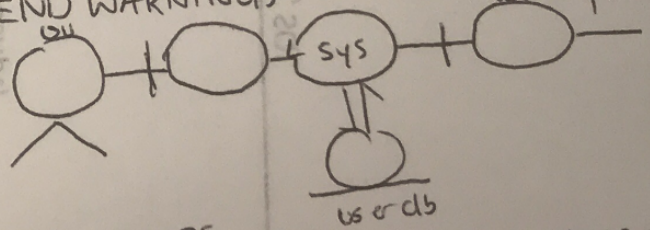
(same for SU)?

TABOO WORD LIST



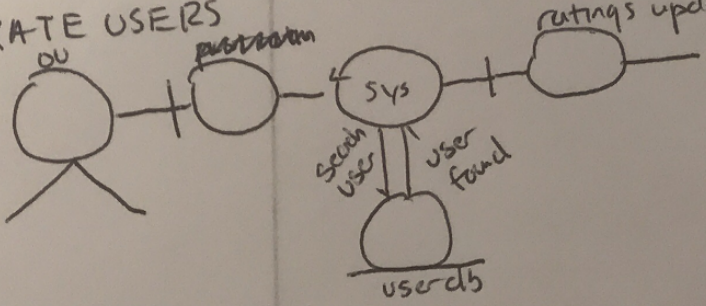
(same for SU)?

SEND WARNINGS



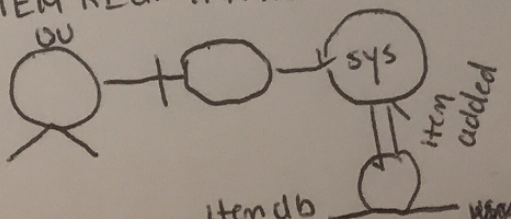
(same for SU)?

RATE USERS



(same for SU)?

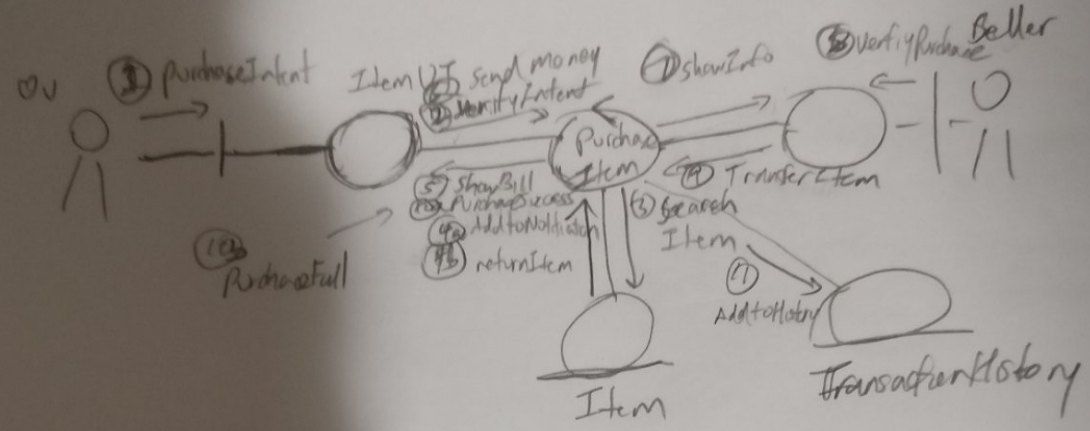
ITEM REGISTRATION



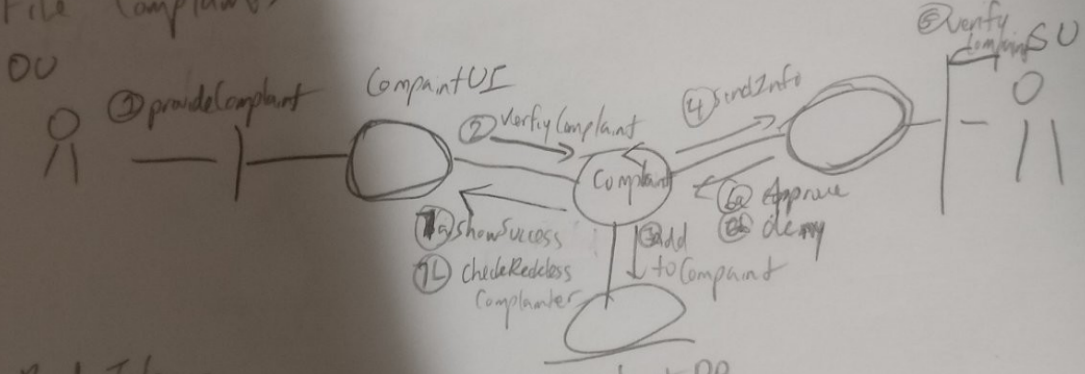
(same for SU)?

Seller

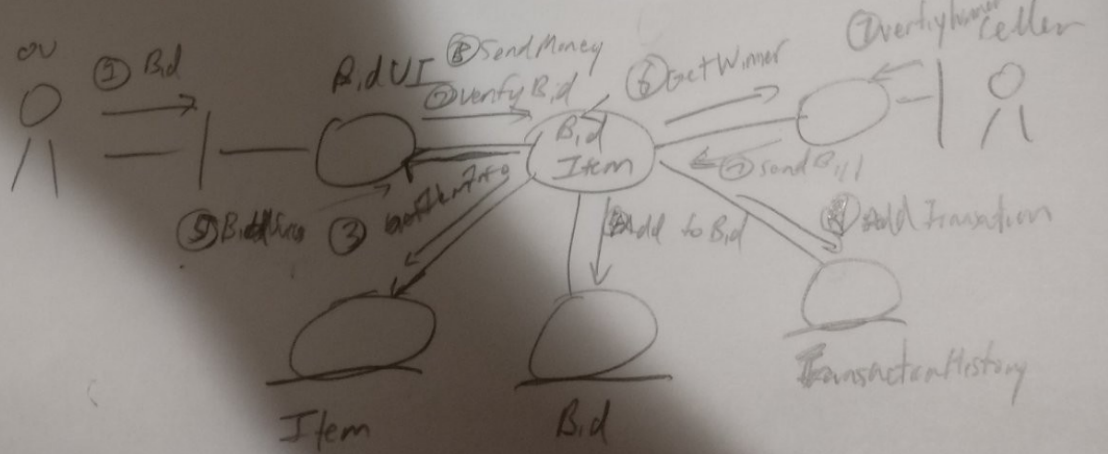
Buy Items



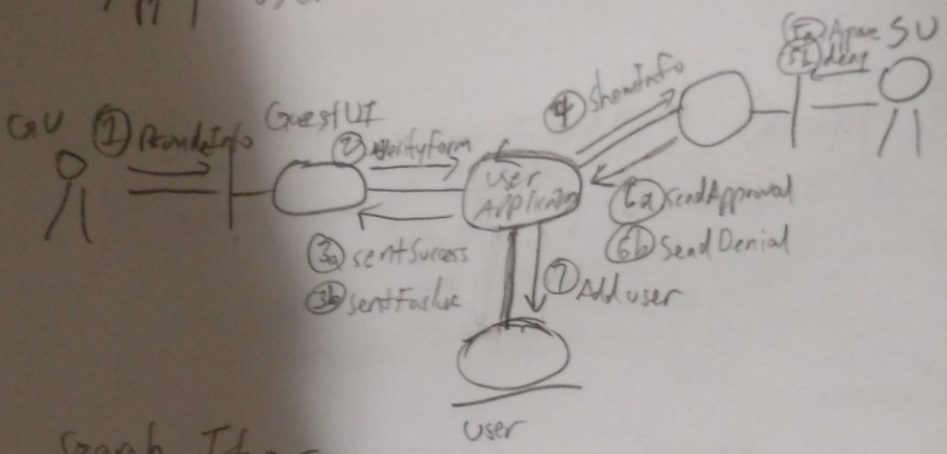
File Complaints



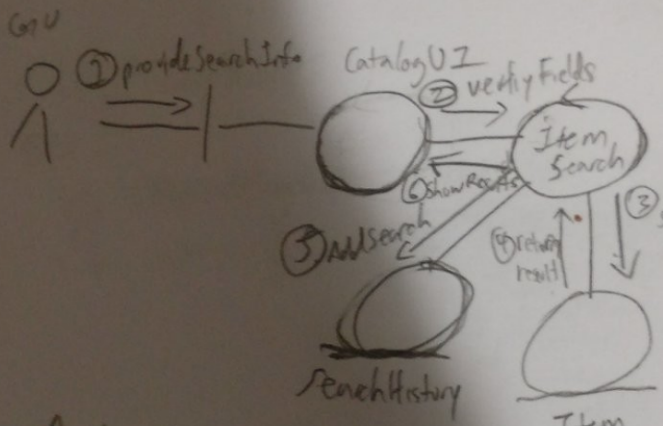
Bid Items



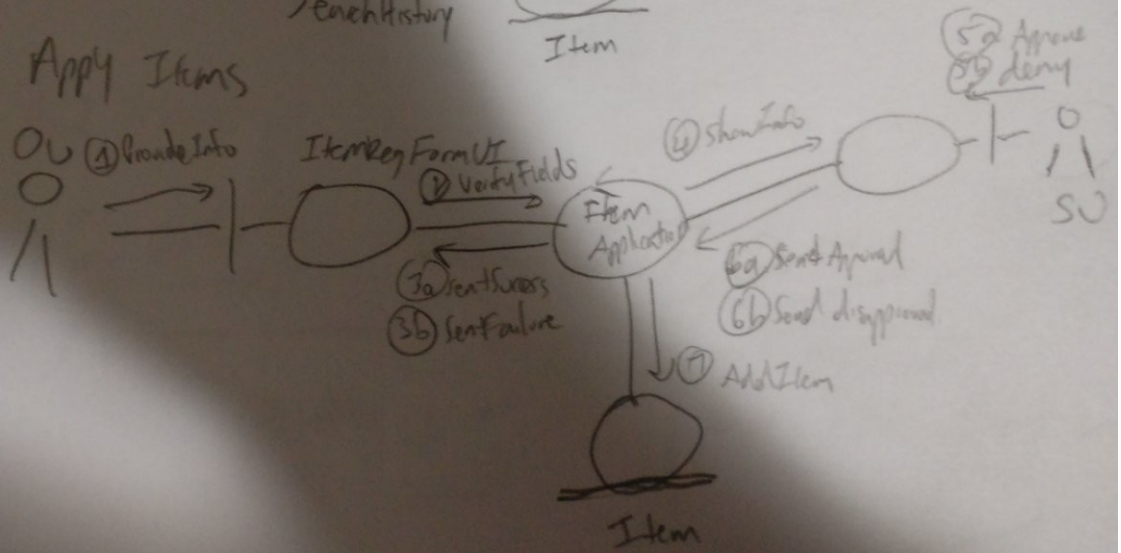
Apply User



Search Items

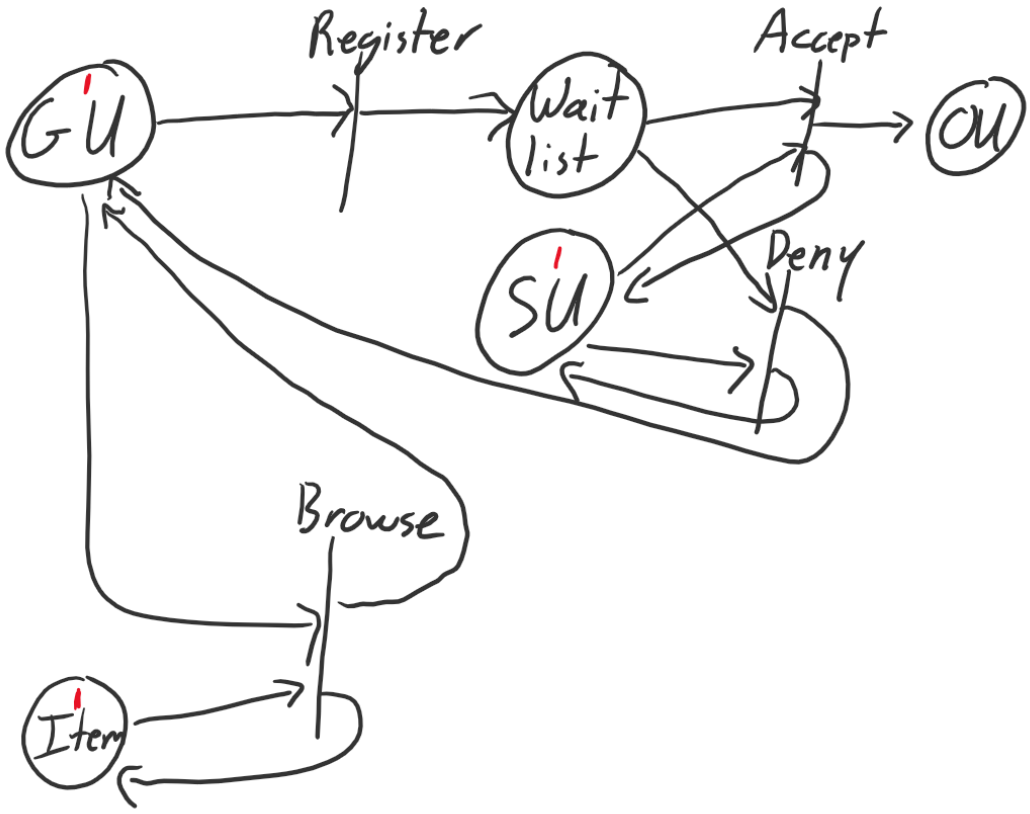


Apply Items

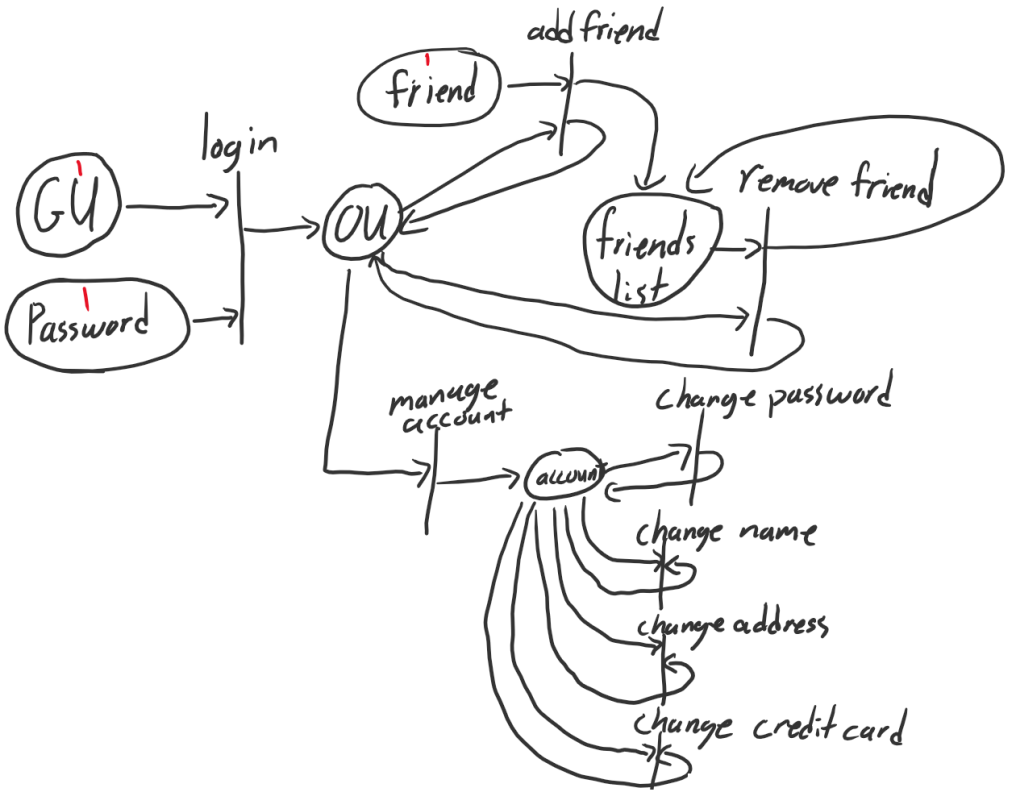


o Petri-net

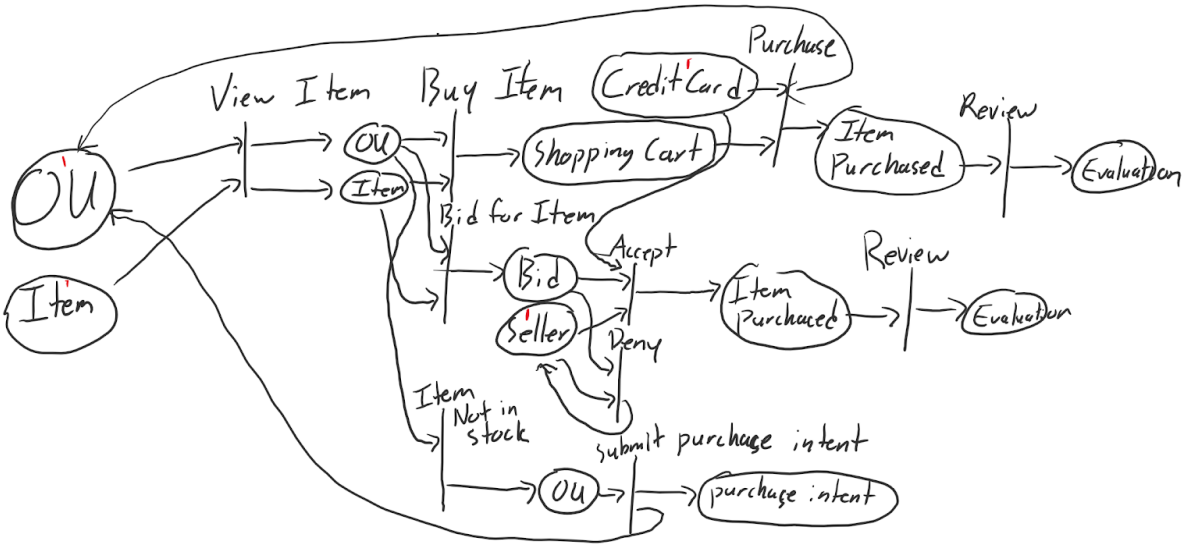
Register, User Application, Browse:



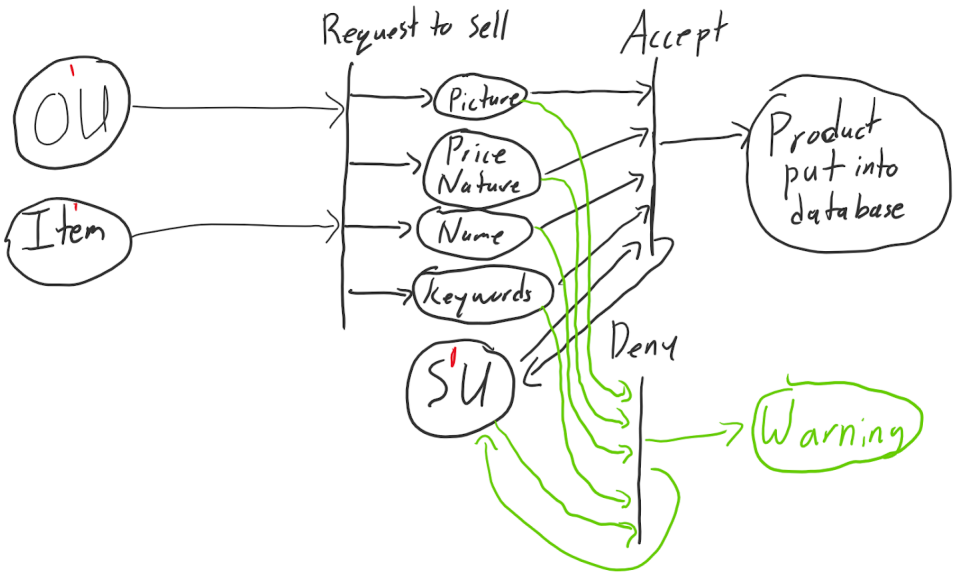
Login, Friend List, Update user information:



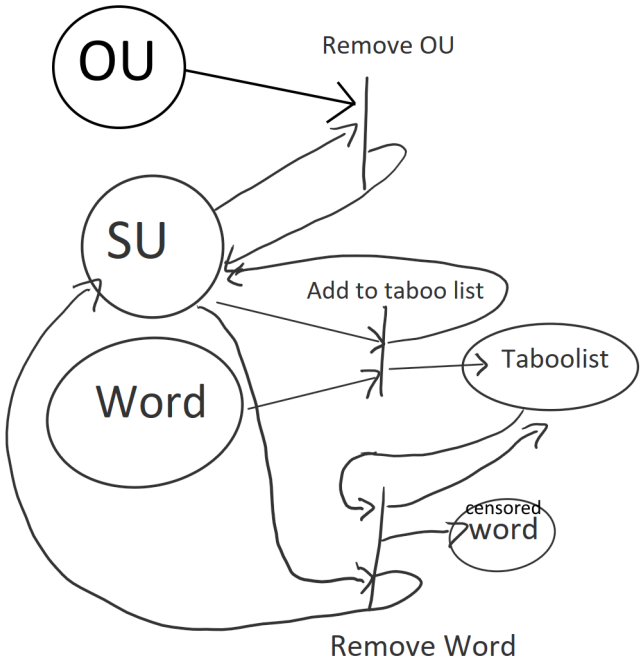
Buy Items, Bid Items, Reviews, Submit Purchase Intent:



Item Application:

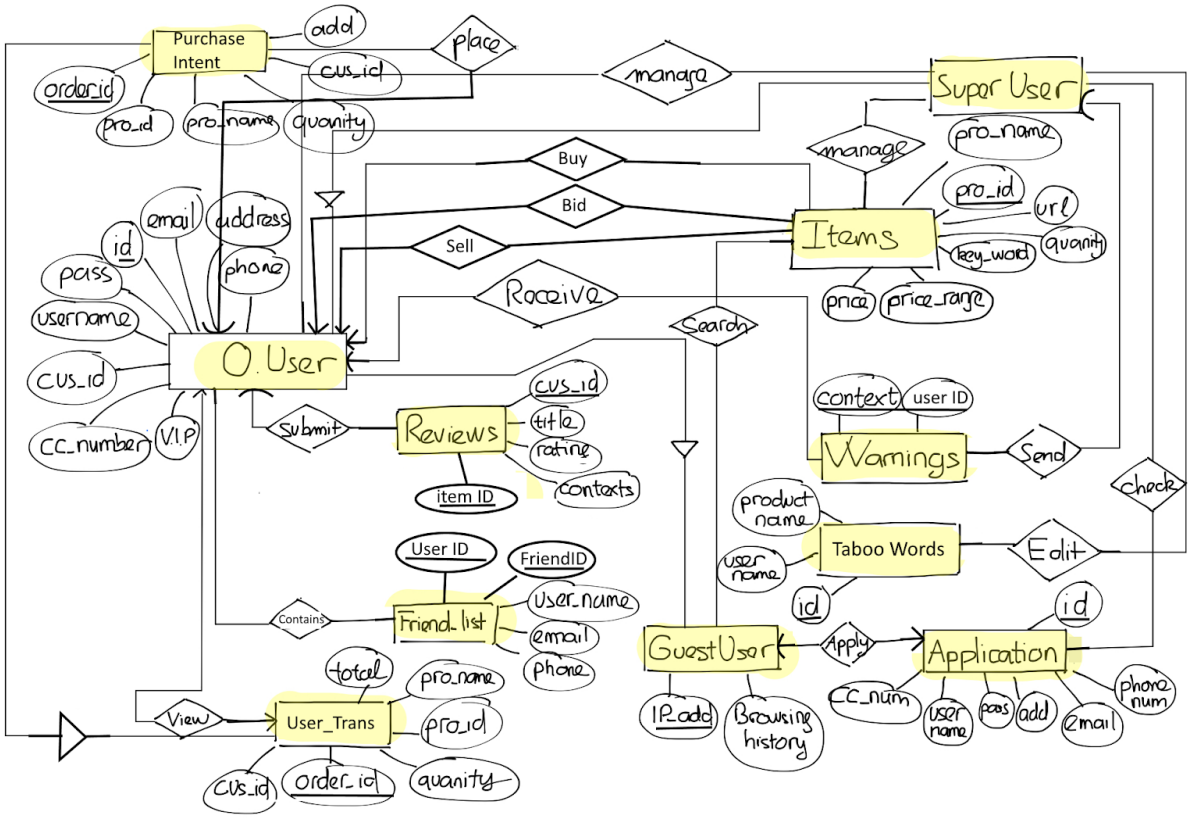


Taboo List:



---

3. E-R diagrams for the entire system


Purchase Intent
add
place
order_id
cus_id
pro_id
pro_name
quanity
manage
Super User
manage
pro_name
Buy
Bid
Sell
pro_id
url
key_word
quanity
Items
price
price_range
email
address
id
phone
pass
username
cus_id
cc_number
V.I.P
Receive
Search
O. User
cus_id
title
rating
contents
Reviews
submit
item ID
context
user ID
Warnings
Send
check
product name
user name
Taboo Words
Edit
id
User ID
FriendID
user_name
email
phone
Friend_list
Contains
GuestUser
Apply
Application
id
IP_add
Browsing history
cc_num
user name
pass
add
email
phone num
total
pro_name
pro_id
quanity
order_id
cus_id
User_Trans
View

4. Detailed design:

Browse/search available items based on title, keywords, price (fixed or a range for bidding) and ratings;

- Input search title, keywords, price nature, and rating range and outputs all items within the search parameters. If a parameter not provided, a default value queried to omit from search.

```
SELECT * FROM Items
      WHERE title LIKE searchTitle
      AND keywords LIKE searchKeywords
      AND priceNature IS searchIsBidBoolean
      AND rating BETWEEN searchBottomRange AND searchTopRange
```

Submit to SUs the application to be an OU, which should include a unique id, the name, a valid credit card number, address, phone number.

- Input unique id, name, credit card information, address, and phone number into a form and submits into a table where the superusers approve or deny. Approved users are then added to user table and initial password set to unique_id from application.

```
INSERT INTO UserApplication (unique_id, name, cc_info, addresss, phone_num)
VALUES (appUnique_id, appName, appCC_info, appAddress, appPhone_num)
```

Process applications and decide who can be OU and who should be denied.

```
if approve:
      INSERT INTO Users (username, password, name, cc_info, address, phon_num,
... other user columns)
      SELECT unique_id, unique_id, name, cc_info, address, phon_num

      FROM UserApplication
```

The initial password is the same as the id which the new OU is required to change the first time s/he logs in.

```
if loginSuccess
      if unique_id equals password:
      redirect to password change screen
      UPDATE Users
      SET password = NewPassword
      WHERE user_id = currentUser_id
      else
      continue to home page
else
      return to login screen
```

Submit the bid to buy an available item; for price range, the one with the second highest bids should be chosen, otherwise the owner should provide a note of justification.

- Input desired bid price and it gets added into a bid table with (item_id, bidder_id, price) as a primary key. When inputting bid, checks if the input bid is higher than the current highest bid and adds in if it is. When the bid ends, returns the person who is the 2nd highest bidder to find the winner.

```
INSERT INTO Bid (item_id, bidder_id, price, startDate, endDate)
      SELECT bidItem_id, bidDidder_id, bidPrice, bidStartDate, bidEndDate
      WHERE NOT EXISTS (SELECT price
                        FROM Bid
                        WHERE item_id = bidItem_id AND price >= bidDidder_id)

SELECT bidder_id
      FROM (SELECT * FROM Bid
            WHERE item_id = bidItem_id
            AND bidder_id = bidDidder_id
            AND  price = bidPrice
            ORDER BY price DESC)
      WHERE ROWNUM = 2
```

Submit to SUs the information (title, key words, price nature, at least one picture) of the item(s) s/he wants to sell.

- Inputs title, key words, price nature, picture into a form and submits into a table where superusers approve or deny. Approved items are then added to Item table. Item table contains a Boolean for if it is an item for bid and a price value. The price value serves as the fixed price for a regular selling item or a starting bid price if it is a bid item.

```
CREATE TABLE Items {
      item_id INT PRIMARY KEY,
      owner_id INT,
      title varchar(255) ,
      keywords varchar(255),
      isBiddable boolean,
      price INT,
      picture URL,
      ... other item parameters
      FOREIGN KEY (owner_id) REFERENCES Users(user_id)
}

INSERT INTO ItemApplication (owner_id, title, keywords, isBiddable, picture)
      VALUES (appUser_id, appTitle, appKeywords, appIsBiddable, appPicture)

INSERT INTO Items (owner_id, title, keywords, isBiddable, picture)
      SELECT (owner_id, title, keywords, isBiddable, picture)
      FROM ItemApplication
```

Sell the item s/he posted; for fixed price, the first one who posted the purchase intention should be chosen otherwise the owner should provide a note justifying why the first one is not chosen

- Input an item and buyer id into a purchase intention table. If there is no note, the first intent is posted is selected, otherwise, another intent is selected.

```
if note:
      desiredSeller = position of desired seller to sell to
else:
      desiredSeller = 1
SELECT buyer_id
      FROM (SELECT * FROM PurchaseIntent
            WHERE item_id = sellingItem_id
            AND buyer_id = sellingBuyer_id
            ORDER BY date DESC)
      WHERE ROWNUM = desiredSeller
```

Submit purchasing intentions if no such item exists in the system, once someone post item(s) matching the keyword(s), the OU will receive notifications.

```
IF NOT EXISTS (SELECT * FROM Item WHERE item_id = purchaseIntentItem_id)
BEGIN
      INSERT INTO PurchaseIntent(item_id, buyer_id, date)
      VALUES (item_id, user_id, currentDate)
END
ELSE
BEGIN
      INSERT INTO AvailableItemNotification(keywords, user_id)
      VALUES (purchaseIntentItemKeywords, user_id)
END
```

File complaints to SUs against certain OUs whose item or payment has some problem or explain to the SU about any complaint received from others.

- Inputs a form with the target user and complaint details into a complaint table where superusers can see.

```
INSERT INTO Complaints (user_id, target_id, complains)
      VALUES (user_id, target_id, complains)
```

Grade an OU after buying an item from him/her, 0 being the worst and 5 being the best. An OU who has submitted too many low ratings, three 0 or 1 ratings in a row, or high ratings, three 5 ratings in a row, will be warned by SU as possible reckless graders.

```
INSERT INTO Ratings (rater_id, target_id, rating, comments)
      VALUES (rater_id, target_id, rating, comments)

def checkRecklessGrader
      past3ratings = [-1,-1,-1] # -1 = no ratings
      past3ratings.append(rating)
      past3ratings.pop(0)
      for i in range(1, len(past3ratings)):
      if past3ratings[i] is 0 or 1 and past3ratings[i-1] is 0 or 1
            continue check

      else if past3ratings[i] is 5 and is equal past3ratings[i-1]
            continue check
      else
            past3ratings[i-1] not equal past3ratings[i] and past3ratings[i]
not equal 0, 1, or 5:
            return # exit check


      send warning:
      INSERT INTO PossibleWarning (rater_id, details)
      VALUES (rater_id, details)
```

See his/her own transaction history.

```
INSERT INTO Transactions (user_id, buyer_id, date, ... other transaction
details)
      SELECT (item_id, buyer_id, date)
      FROM PurchaseIntent

SELECT * FROM Transactions
```

Change his/her password, name, address, phone and credit card number, but not the id.

```
UPDATE Users
SET password = desiredPassword, name = desiredName, address = desiredAddress,
phone = desiredPhone and cc_num = desiredCC_num
WHERE user_id = currentUser_id
```

An OU will become a VIP OU if her/his rating is >= 4 (by at least 3 different OUs) or spent more than $500 and without warnings, any VIP will receive 5% discount when checking out. A VIP is moved to ordinary OU if his/her rating is below 4 automatically or received one warning.

```
def checkVIPStatus:
      query = SELECT * FROM ratings WHERE target_id = currentUser_id
      if query rows >= 3  # 3 unique ratings, rater_id + target_id is primary
key of ratings, no rater_id can be the same for the same target
      if rating >= 4
            user.isVIP = true
      else if spent > 500 and no results from warning query with the user_id
as target
      user.isVIP = true
      else if rating < 4 or there is results from warning query with the
user_id as target
      user.isVIP = false

if isVIP = true
      price *= 0.95
```

An OU can maintain a friend list who will receive discount and friend-only messages, the OU is free to add/delete any OU from the list.

```
INSERT INTO FriendRequest (user_id, target_id)
      VALUES (user_id, target_id)

if target accept:
INSERT INTO Friendlist (user1_id, user2_id)
      SELECT (user_id, target_id)
      FROM FriendRequest
```

Process item information the OUs submitted intending to sell, some can be publicly available on the platform and some may be denied with justifications. An OU whose item is denied will be warned once.

```
def ItemApplicationApproveOrDeny (boolean approve)
      if approve:
      INSERT INTO Item (... parameters)
      SELECT (... parameters)
      FROM ItemApplication
      else
      INSERT INTO Warnings (user_id, warning_message ...)
      VALUES (itemApplicant_id, warning_message ... )

      DELETE FROM ItemApplication WHERE itemApplicantion_id =

thisApplication.id
```

Send warnings to OUs who received two justified complaints or the average grade are lower than 2 with at least 3 different evaluators, should an OU receive two warnings this OU should be suspended from the system.

```
def sendWarnings:
      complaints_count = SELECT COUNT(target_id) FROM Complaints
                         WHERE target_id = currentUser_id
                         AND  isJustified IS TRUE
      average_grade = SELECT AVG(rating) FROM (SELECT * FROM Ratings WHERE
target_id = currentUser_id)
                                      WHERE ROWNUM > 3

      if complaints_count >= 2 or average_grade < 2:
      INSERT INTO Warnings (target_id, details, ...)
      VALUES (currentUser_id, details, ...)

def checkSuspended:
      warnings_count = SELECT COUNT(target_id) FROM Warnings
                         WHERE target_id = currentUser_id AND addressed IS
FALSE

      if warnings_count >= 2:
      currentUser.isSuspended = true
      return true
      else
      currentUser.isSuspended = false
      return false
```

A suspended OU will be notified when s/he logs in and can choose to appeal or resign. The appeal can be either approved or denied (then the user is removed).

```
if login = success:
      if checkSuspended is true:
      redirect to appeal form page
      choose appeal or resign

      if appeal
            reason = user input from form

            INSERT INTO AccountAppeals (user_id, reason)
            VALUES (currentUser_id, reason)
      else if resign
            DELETE FROM Users WHERE user_id = currentUser_id


def AppealApproveOrDeny (boolean approve):
      if approve
      currentUser.isSuspended = false
      UPDATE Warnings
      SET addressed = TRUE
      WHERE user_id = currentUser_id
      else
      removeUser(currentUser)
```

Have the right to remove any OU based on his/her judgment. A removed OU can log in to the system for the last time to clear matters, then the account is deleted afterwards.

```
def removeUser (selectedUser):
      selectedUser.isRemovedOnLogout = true

if loginSuccess:
      if currentUser.isRemovedOnLogout:
      show notification of account removal

if logoutSuccess:
      if currentUser.isRemovedOnLogout:
      DELETE FROM Users WHERE user_id = currentUser_id
```

Collect transaction statistics for a certain period (a day or a week) or a certain OU.

```
transactions = SELECT * FROM Transactions
      WHERE user_id = selectedUser_id
      AND TransDate BETWEEN startDate AND endDate

calculateSomeStatistic(transactions)
```

Maintain a taboo list with all inappropriate words, any OU's keywords/name or items containing taboo words will be replaced by *** and the OU received one warning and required to change the word next time s/he logs in.

```
tabooList = ['words', ...]
def filter (String str):
      for words in str:
      if words in tabooList:
            words = '***'
            INSERT INTO Warnings (user_id, warning, ...)
            VALUES (currentUser_id, warning, ...)

if loginSuccess:
      IF EXISTS (SELECT * FROM Warnings
      WHERE user_id = currentUser_id
      AND addressed IS FALSE)
      AND Warnings = 'Taboo List Warning'
      redirect to form to change keywords/name
```

When logging in, the system should have personalized recommendations to the OU based on his/her past search/purchase records, if no such information is available, the top 3 most popular items and OUs are displayed in the GUI.

```
PurchasedItems = SELECT item_id FROM Transactions
            WHERE user_id = currentUser_id)

def displayReccommended (String basedOn):
      if basedOn = 'purchase':
      parameters = PurchasedItems parameters
      else if basedOn = 'search'
      parameters = SELECT searchTitle, searchKeywords, ... FROM SearchHistory
WHERE user_id = currentUser_id

      SELECT * FROM Items
      WHERE title LIKE searchTitle
      AND keywords LIKE searchKeywords
      AND priceNature IS searchIsBidBoolean
      AND rating BETWEEN searchBottomRange AND searchTopRange
      where search parameters are parameters
```

OUs removed from the system by SUs will be blocked based on the name for future re-application.

```
blockedUserList ['name', ...]
if application.name in blockedList:
      deny user application
```

Items removed from the system by SUs will be blacklisted from system.

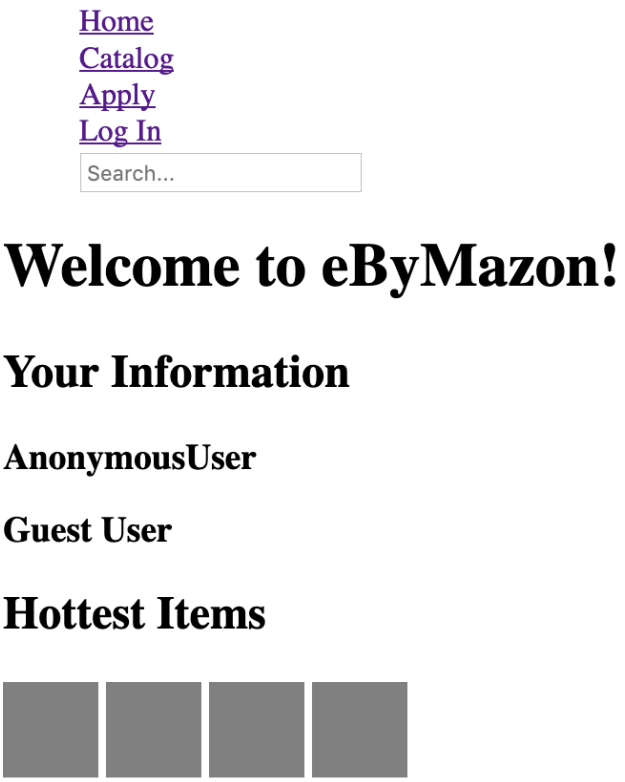
```
blockedItemList ['name', ...]
if item.name in blockedList:
      deny item application
```

The tax of every sold item is automatically evaluated based on the address of the buying OU

```
def taxedPrice:
      tax = price * taxDict.get(user.address.state)
      return price + tax
```

5. System screens:

Main Page (GU)



Main Page (OU)

Home
Catalog
Dashboard
Log Out
Search...

Welcome to eByMazon!

Your Information

kwan2

Ordinary User

Recommended Items

Based on Recently Purchased Items

Hottest Items

Main Page (SU)

Home
Catalog
Dashboard
Log Out
User Applications
Search...

Welcome to eByMazon!

Your Information

kwan

Super User

Recommended Items

Based on Recently Purchased Items

Hottest Items

Catalog

Home
Catalog
Apply
Log In
Search...

Catalog

Product Name:	Apple MacBook Pro (13" Retina, Touch Bar, 2.3GHz Quad-Core Intel Core i5, 8GB RAM, 512GB SSD) - Space Gray (Latest Model)
Seller:	kwan
Price Nature	For Sale

Apply Membership Page

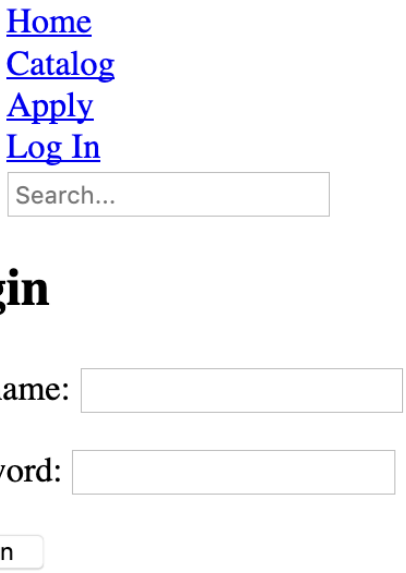
Home
Catalog
Apply
Log In
Search...

Apply for Membership

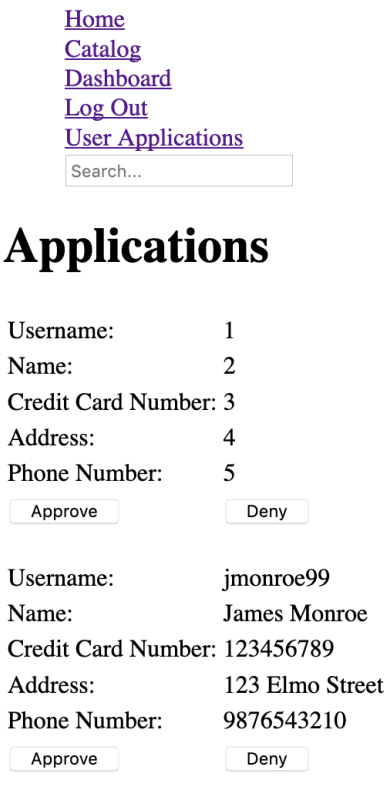
Your signup details will be sent to a Super User for approval.

Username:
Name:
Credit Card Number:
Address:
Phone Number:
Sign up

Log in page



Applications Approval Page



6. Minutes of group meetings

__ minutes

7. Address of the git repo

Github

https://github.com/kfung9564/eByMazon