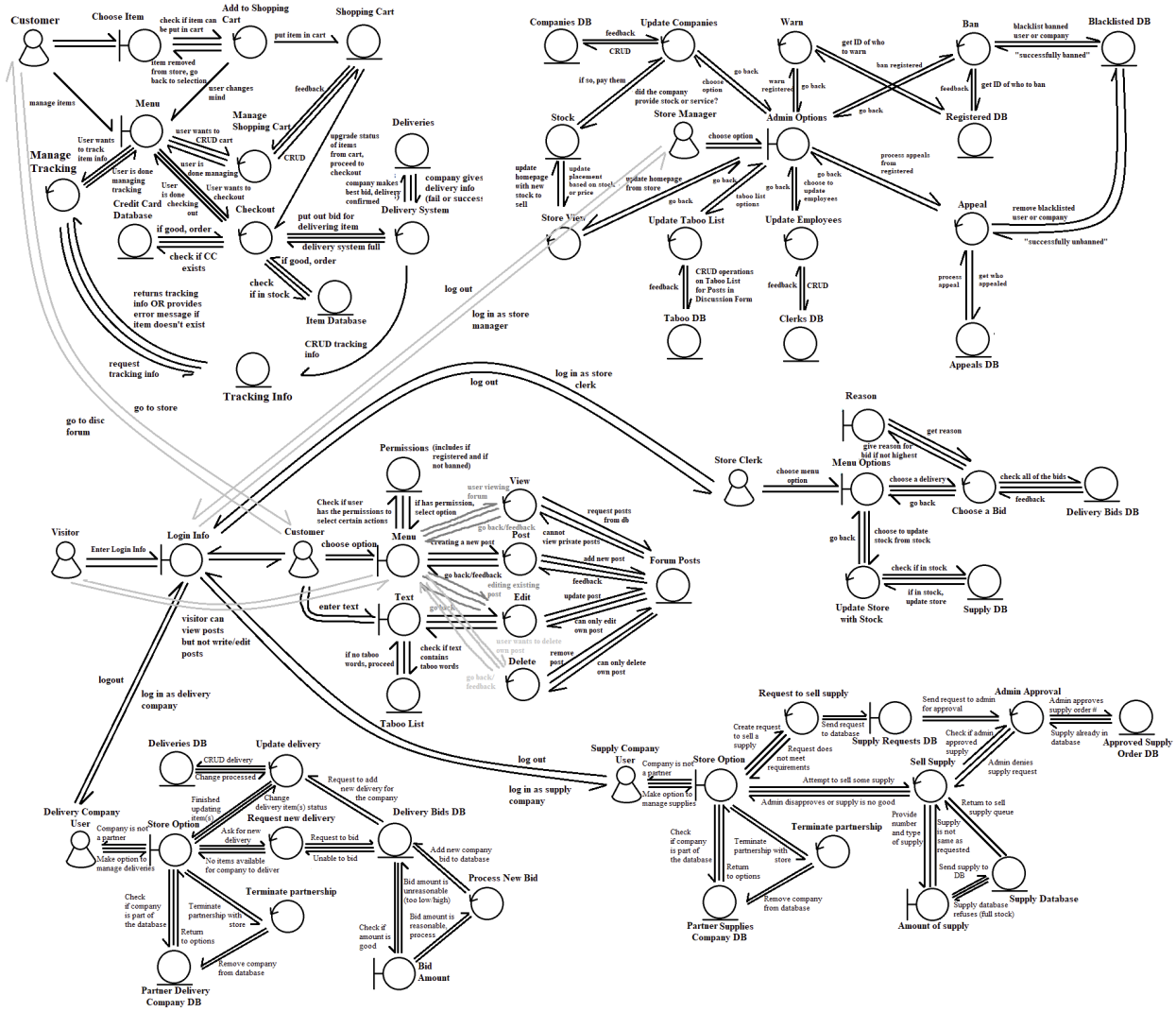
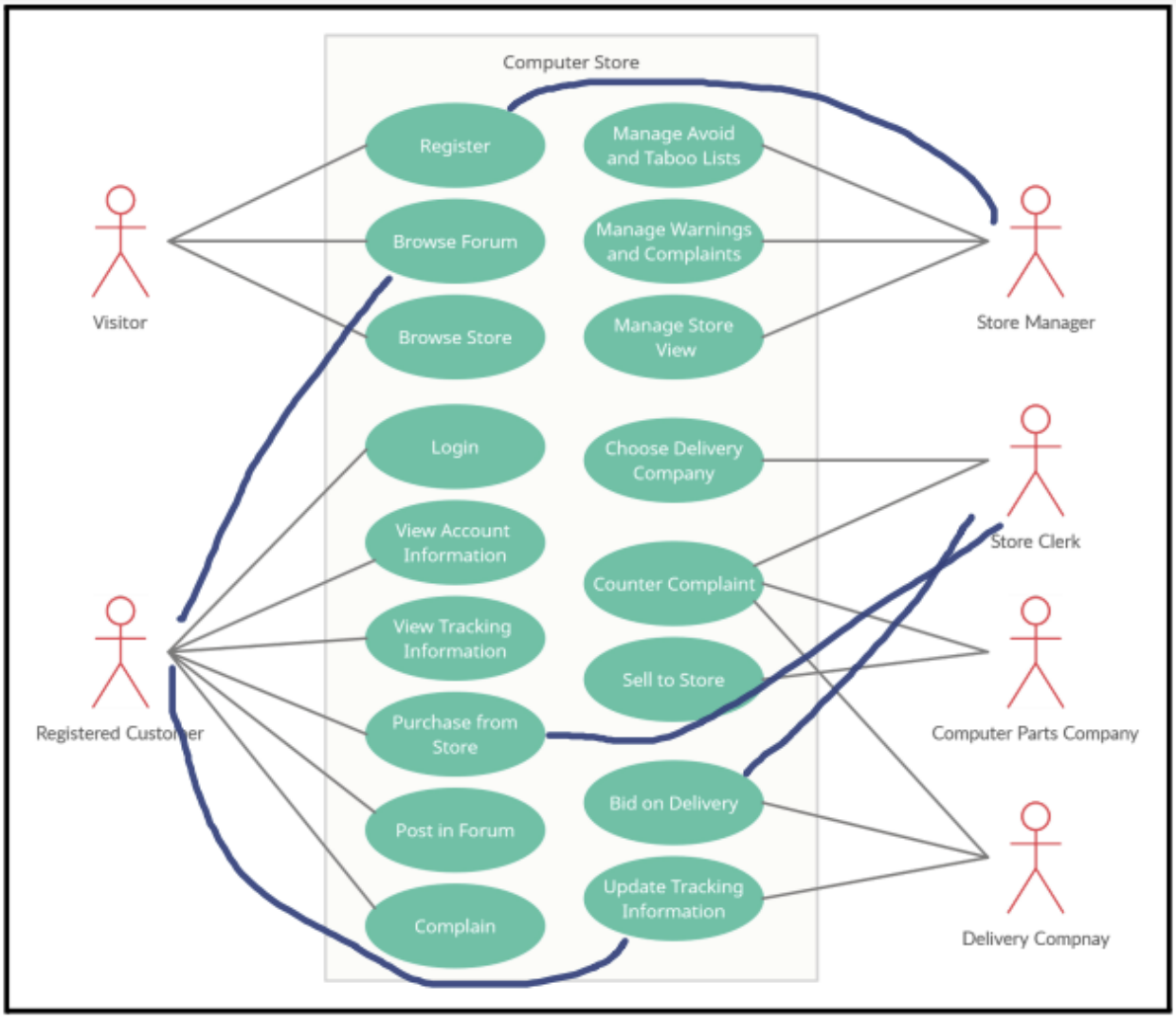
# City Electronics LLC

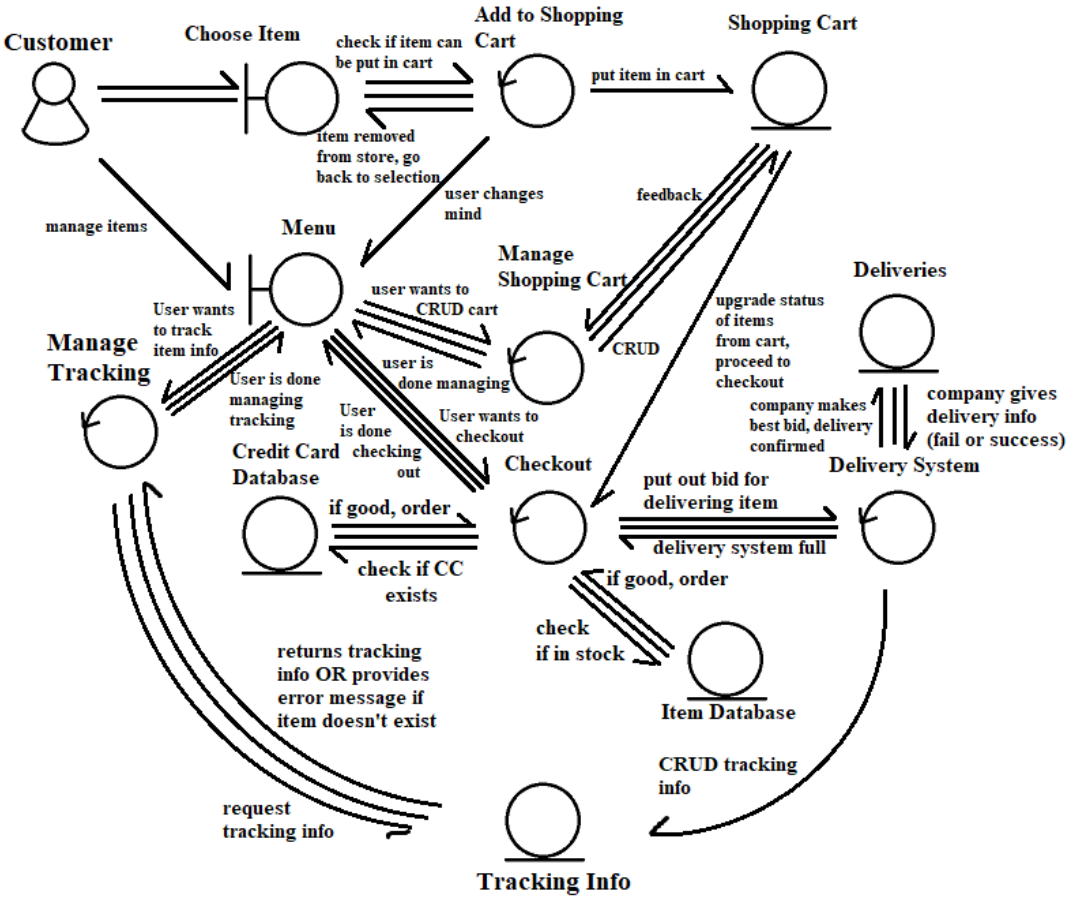
# Parts Authority
# Design Report
# For Computer Parts Store

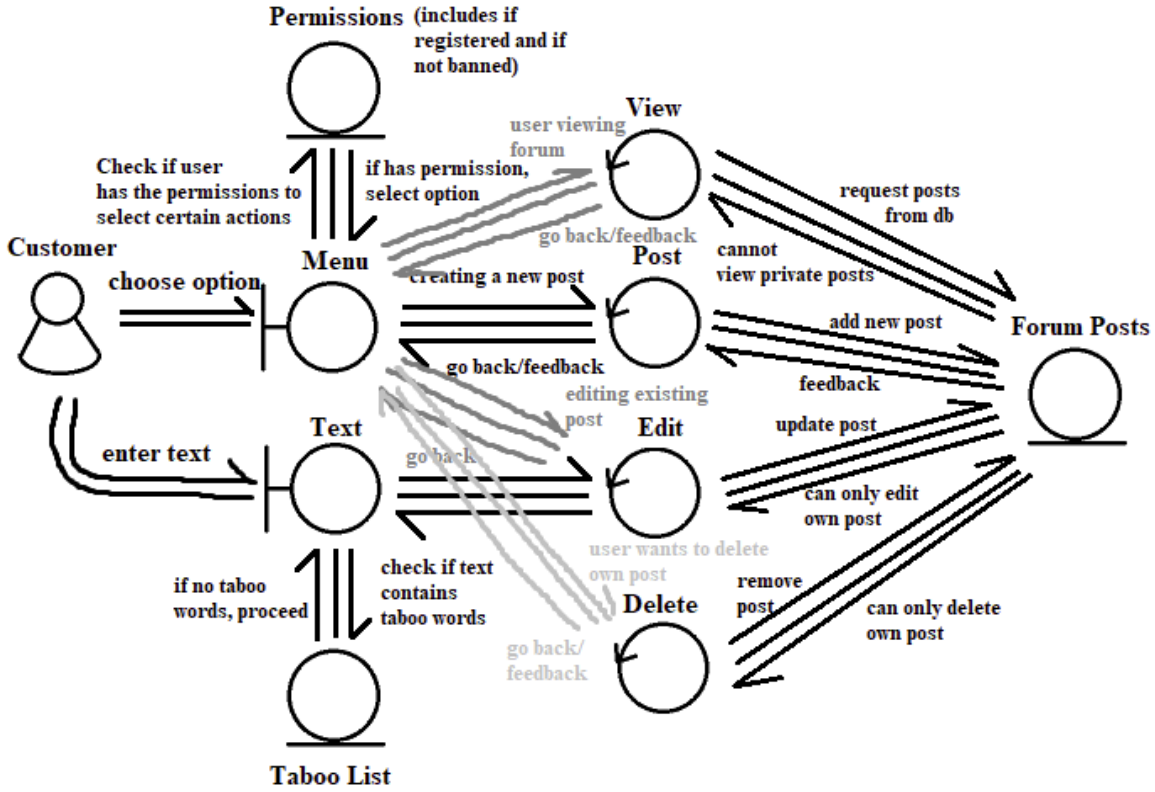
# 1. Introduction

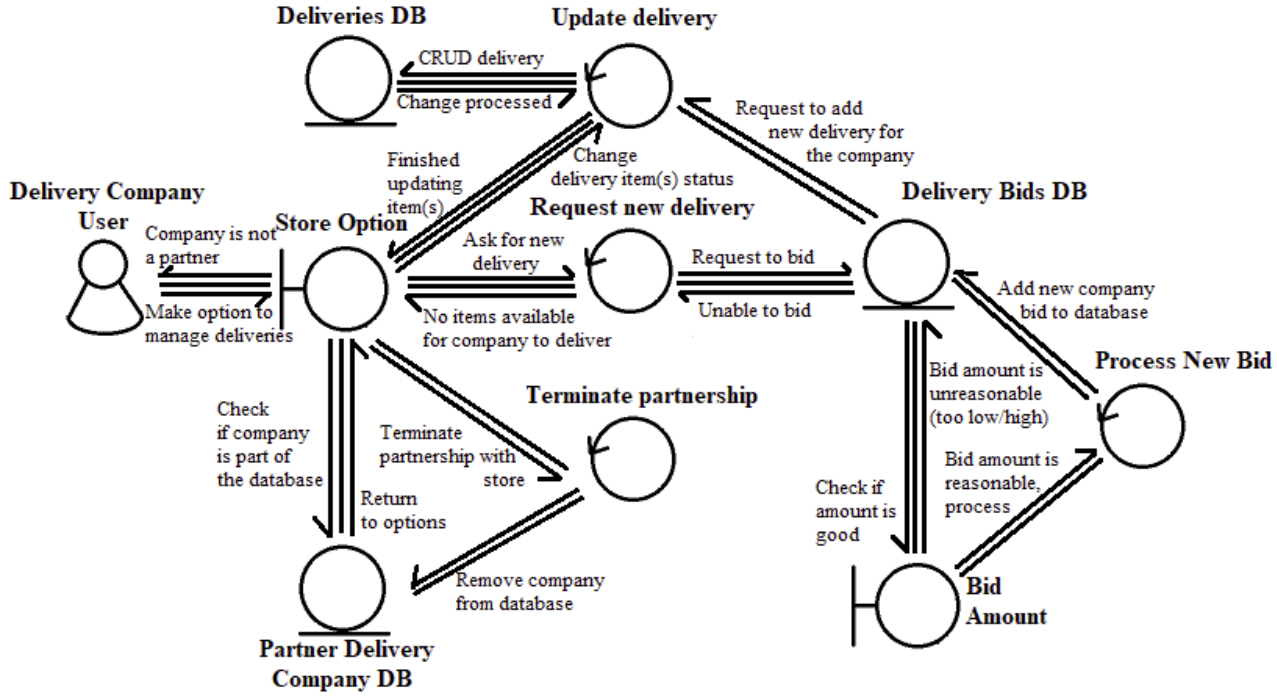


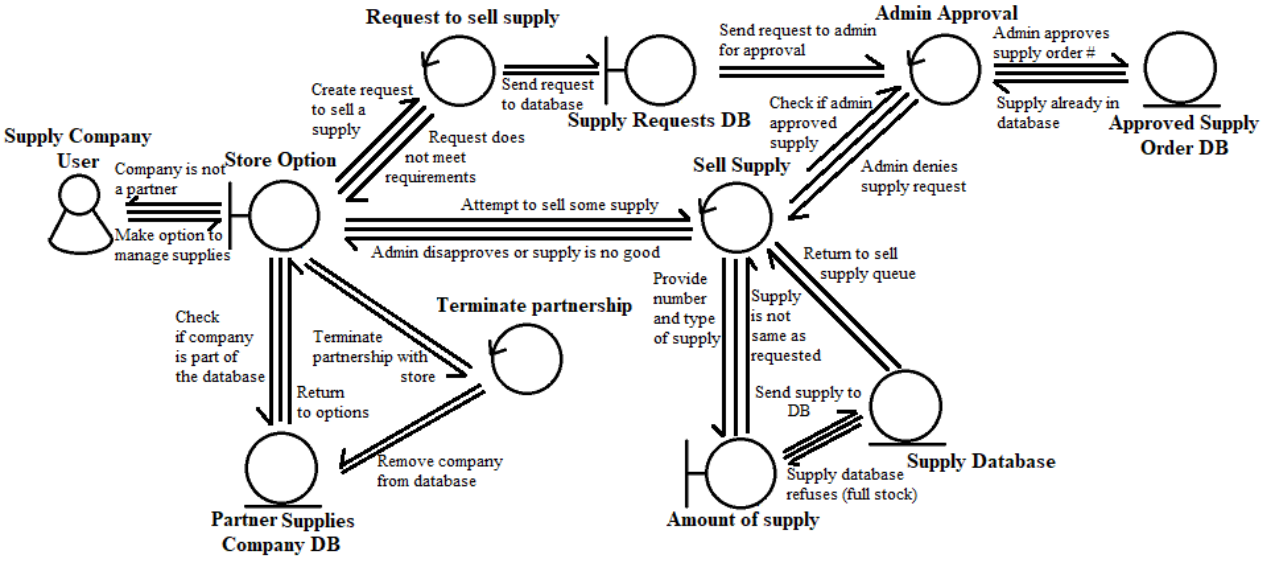
## 2. All Use Cases

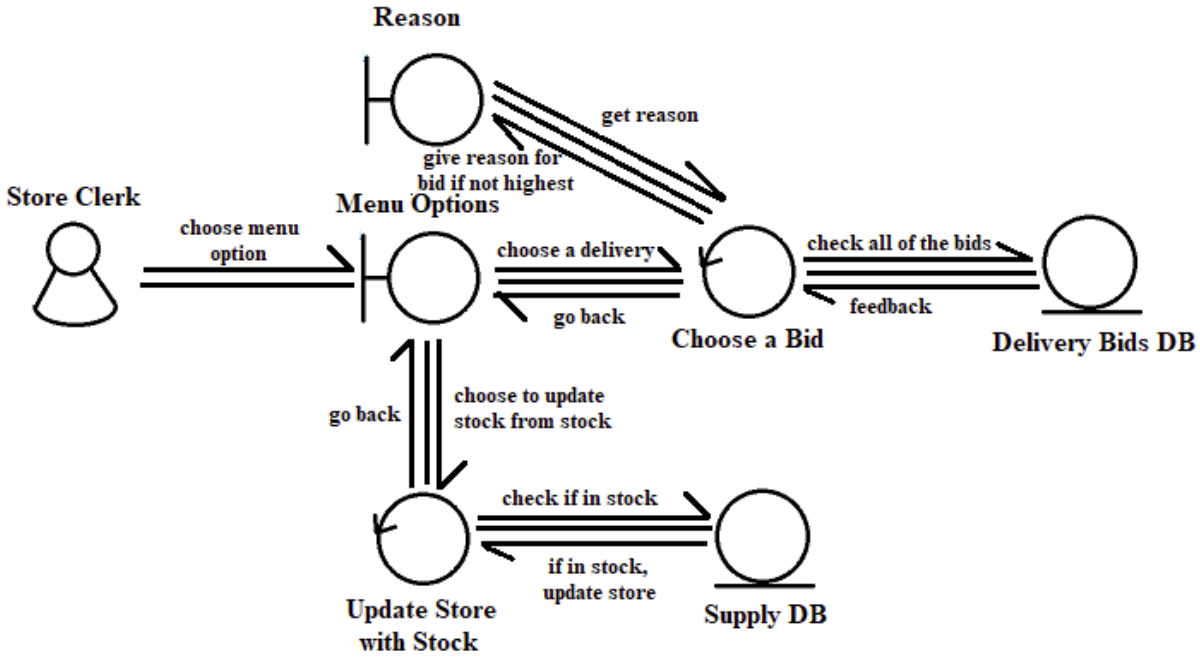


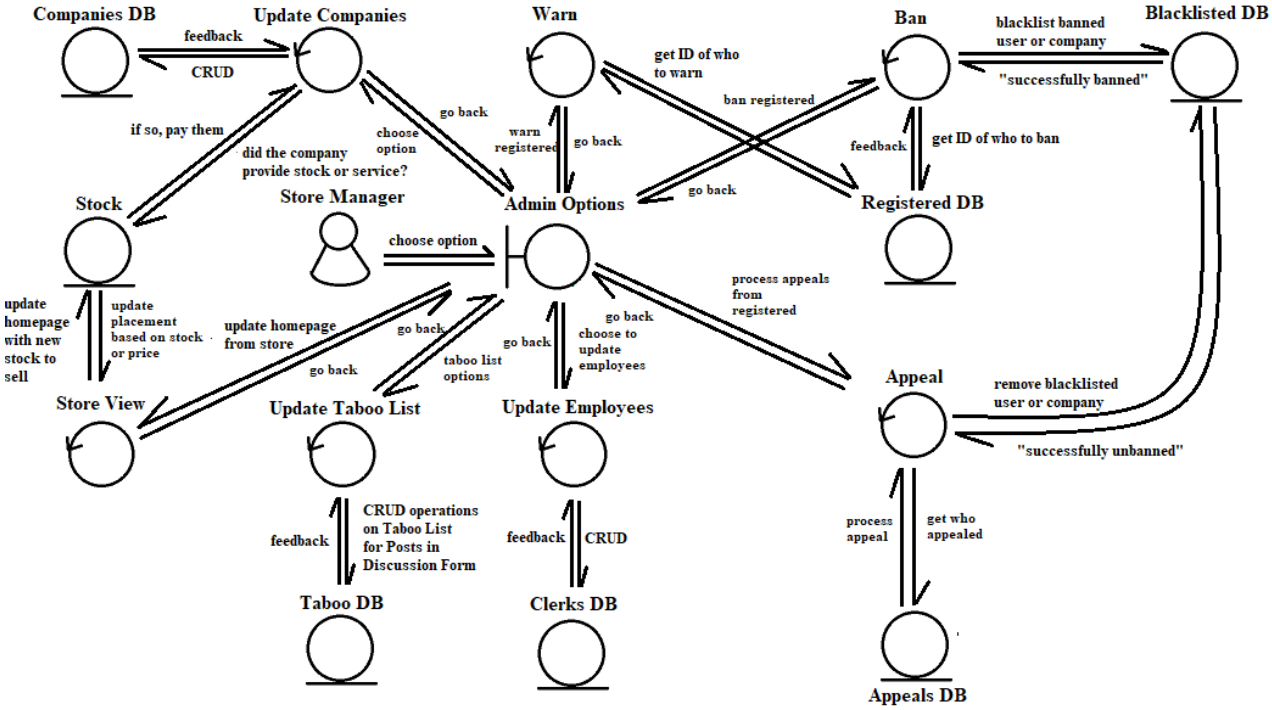

Customer
Choose Item
check if item can be put in cart
Add to Shopping Cart
Shopping Cart
put item in cart
item removed from store, go back to selection
manage items
user changes mind
feedback
Menu
Manage Shopping Cart
Deliveries
User wants to track item info
user wants to CRUD cart
upgrade status of items from cart, proceed to checkout
Manage Tracking
user is done managing
CRUD
User is done managing tracking
company makes best bid, delivery confirmed
company gives delivery info (fail or success)
User is done checking out
User wants to checkout
Credit Card Database
Checkout
Delivery System
put out bid for delivering item
if good, order
delivery system full
check if CC exists
if good, order
check if in stock
returns tracking info OR provides error message if item doesn't exist
Item Database
CRUD tracking info
request tracking info
Tracking Info


Permissions
(includes if registered and if not banned)
View
user viewing forum
Check if user has the permissions to select certain actions
if has permission, select option
request posts from db
go back/feedback
cannot view private posts
Customer
Menu
Post
choose option
creating a new post
add new post
Forum Posts
go back/feedback
feedback
editing existing post
Text
Edit
update post
enter text
go back
can only edit own post
user wants to delete own post
if no taboo words, proceed
check if text contains taboo words
remove post
Delete
can only delete own post
go back/ feedback
Taboo List

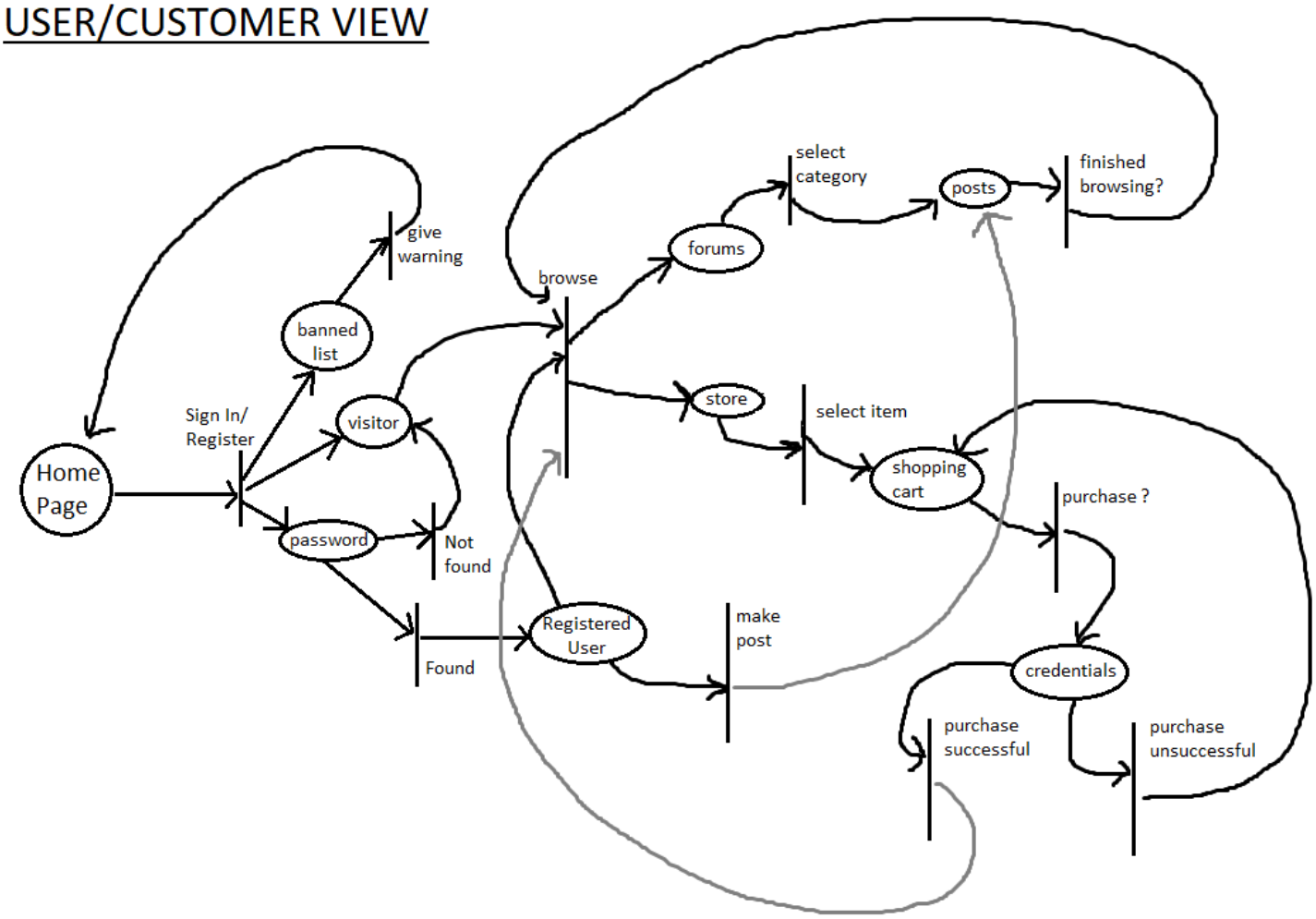

Deliveries DB
Update delivery
CRUD delivery
Change processed
Request to add new delivery for the company
Finished updating item(s)
Change delivery item(s) status
Delivery Company User
Store Option
Request new delivery
Delivery Bids DB
Company is not a partner
Make option to manage deliveries
Ask for new delivery
No items available for company to deliver
Request to bid
Unable to bid
Add new company bid to database
Process New Bid
Bid amount is unreasonable (too low/high)
Bid amount is reasonable, process
Check if amount is good
Terminate partnership
Check if company is part of the database
Terminate partnership with store
Return to options
Remove company from database
Partner Delivery Company DB
Bid Amount


Request to sell supply
Send request to database
Supply Requests DB
Admin Approval
Send request to admin for approval
Admin approves supply order #
Supply already in database
Approved Supply Order DB
Supply Company User
Store Option
Company is not a partner
Make option to manage supplies
Create request to sell a supply
Request does not meet requirements
Check if admin approved supply
Admin denies supply request
Sell Supply
Attempt to sell some supply
Admin disapproves or supply is no good
Return to sell supply queue
Provide number and type of supply
Supply is not same as requested
Send supply to DB
Supply Database
Supply database refuses (full stock)
Amount of supply
Terminate partnership
Check if company is part of the database
Terminate partnership with store
Return to options
Remove company from database
Partner Supplies Company DB

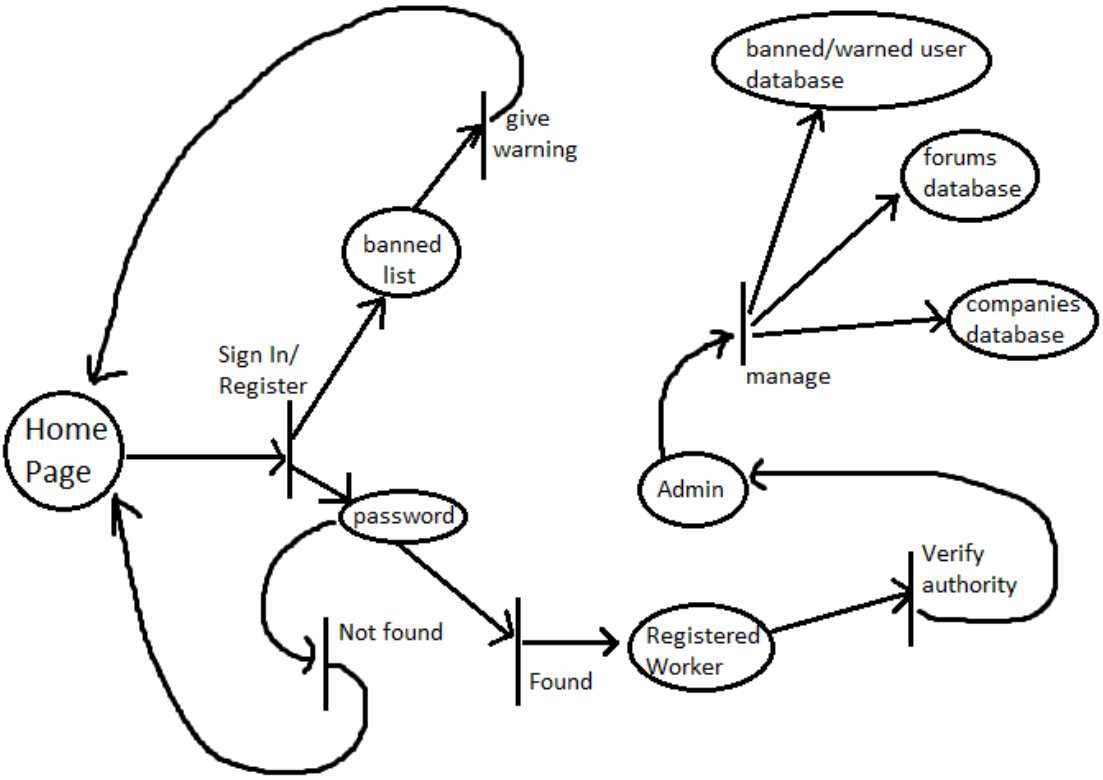

Reason
get reason
give reason for bid if not highest
Store Clerk
choose menu option
Menu Options
choose a delivery
go back
Choose a Bid
check all of the bids
feedback
Delivery Bids DB
go back
choose to update stock from stock
check if in stock
if in stock, update store
Update Store with Stock
Supply DB


Companies DB
feedback
CRUD
Update Companies
Warn
get ID of who to warn
Ban
blacklist banned user or company
Blacklisted DB
"successfully banned"
ban registered
if so, pay them
did the company provide stock or service?
choose option
go back
warn registered
go back
feedback
get ID of who to ban
Stock
Store Manager
Admin Options
go back
Registered DB
choose option
process appeals from registered
update homepage with new stock to sell
update placement based on stock or price
update homepage from store
go back
go back
taboo list options
go back
choose to update employees
Store View
Update Taboo List
Update Employees
Appeal
remove blacklisted user or company
"successfully unbanned"
feedback
CRUD operations on Taboo List for Posts in Discussion Form
feedback
CRUD
process appeal
get who appealed
Taboo DB
Clerks DB
Appeals DB

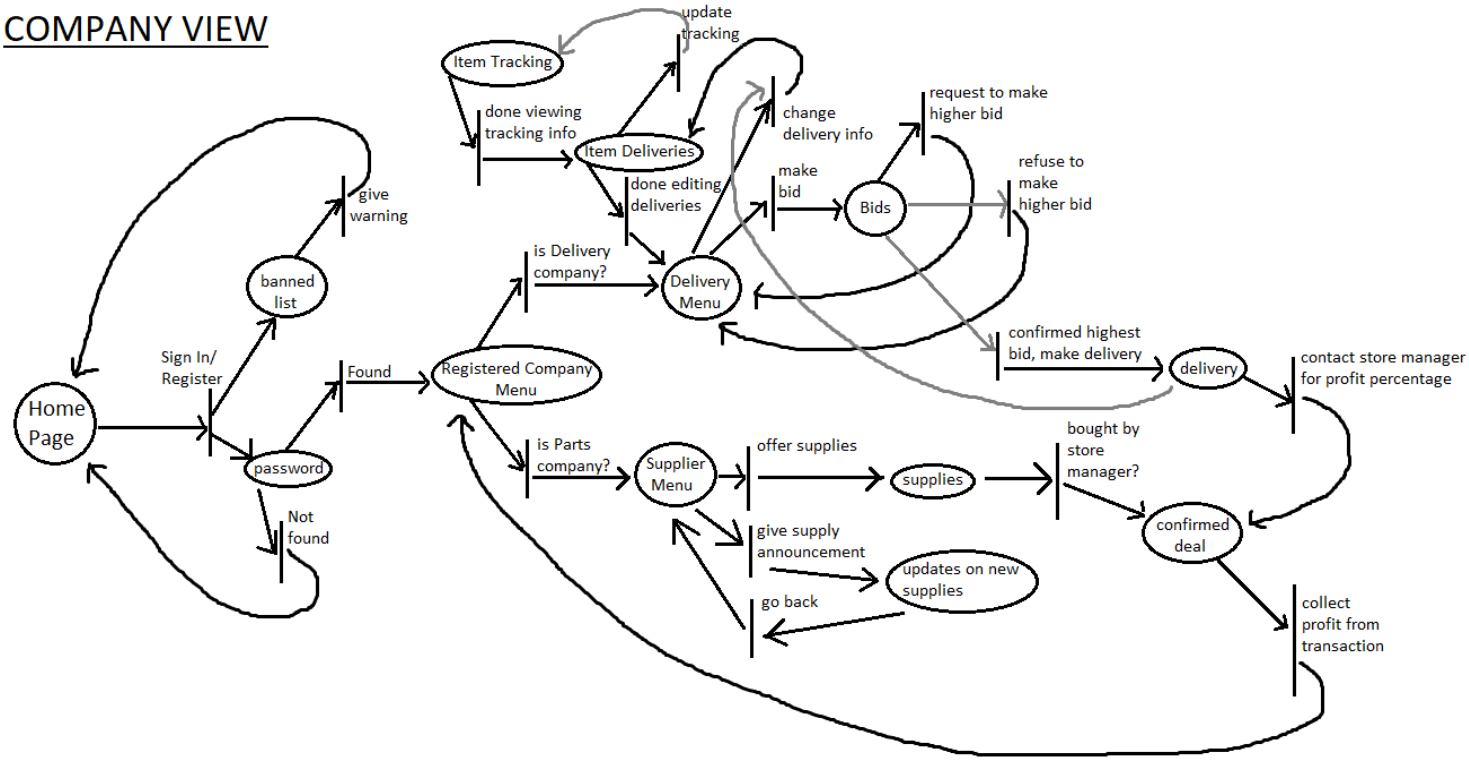
# USER/CUSTOMER VIEW



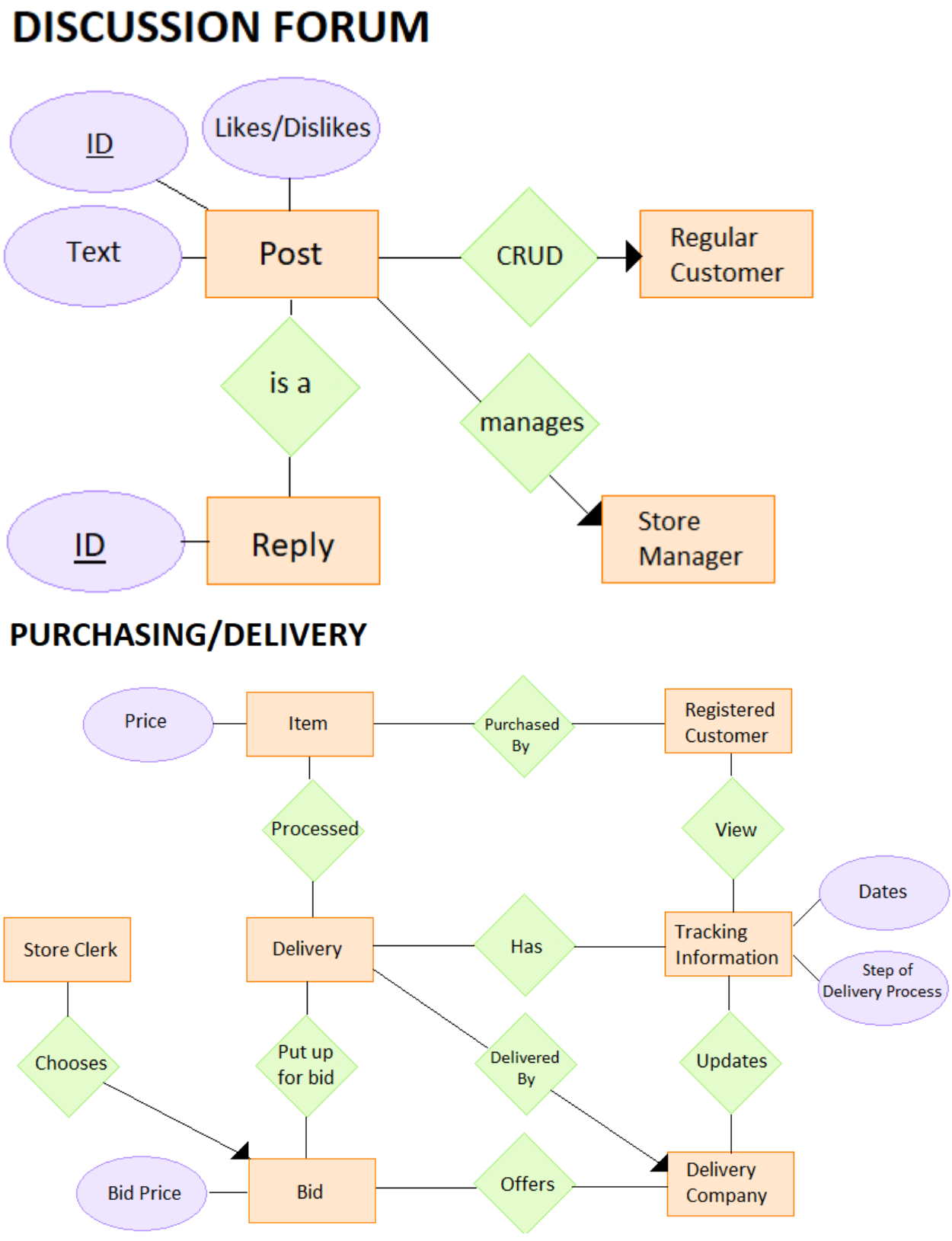
# STORE CLERK/MANAGER VIEW



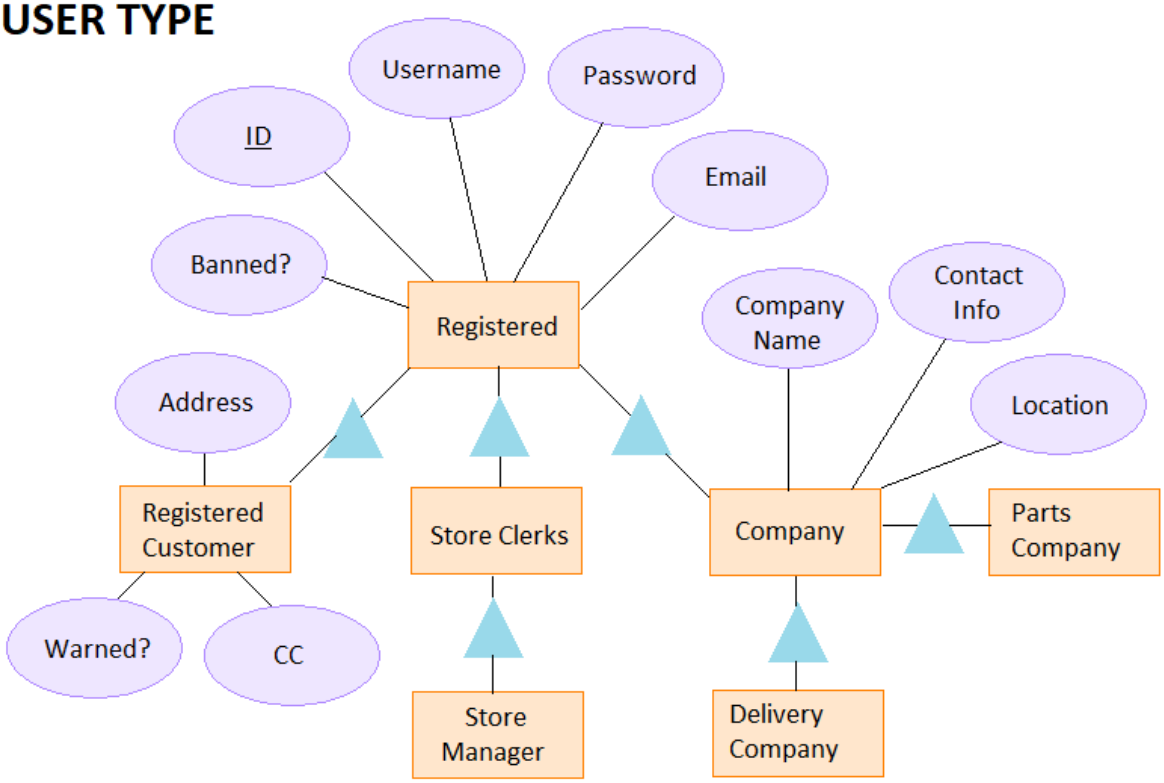

## COMPANY VIEW



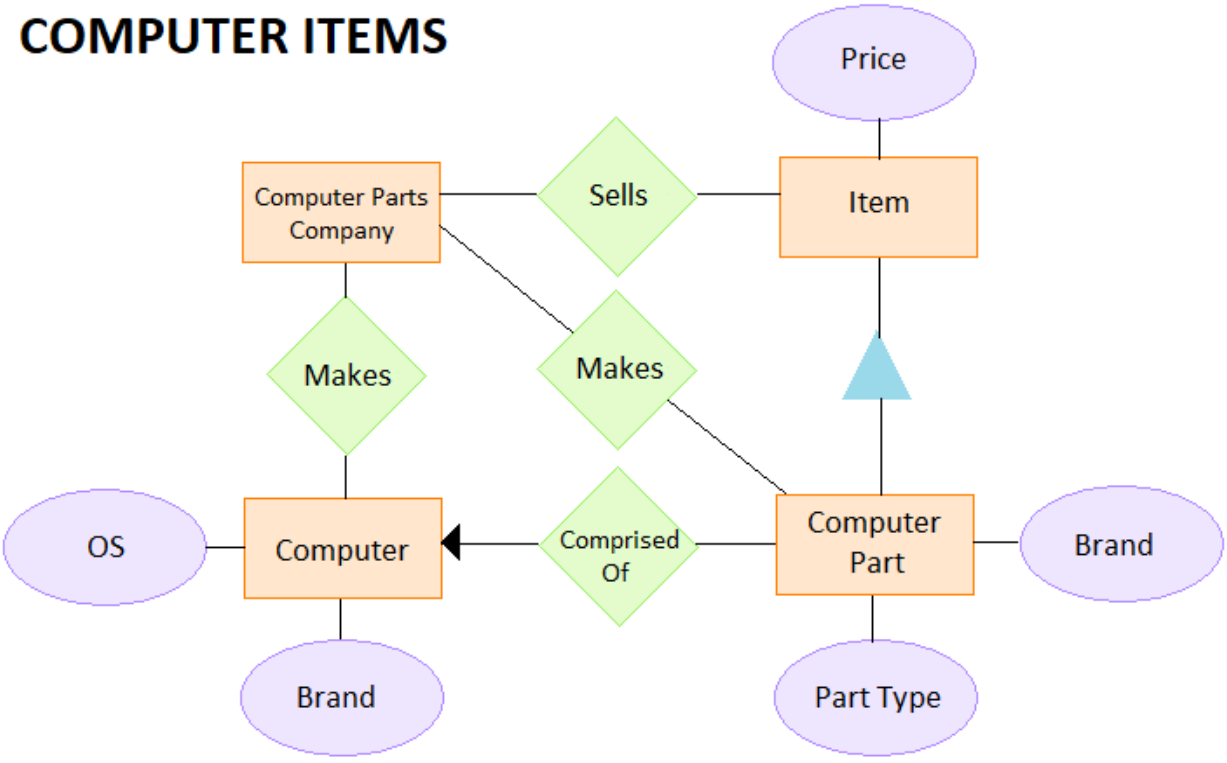
### 3. E-R Diagram for Entire System



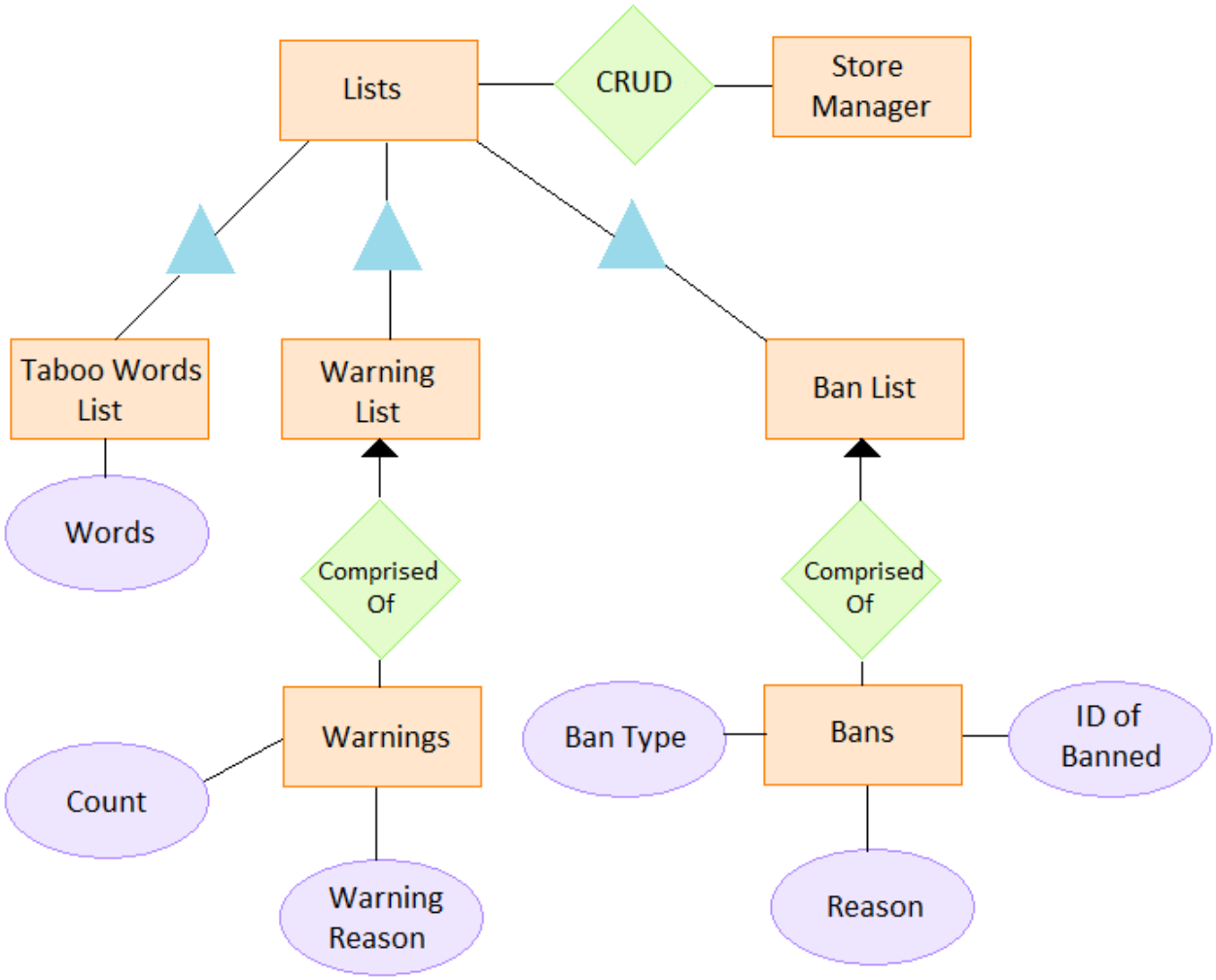
## USER TYPE




## COMPUTER ITEMS



# BAN/TABOO LIST



**4. Detailed Design (Pseudo Code)**

## Customer Store

```
cart_database = SQL_user_cart
def addtoCart(item):
    cart_database.update(item)

def removefromCart(item):
    for i in cart_database:
        if item == i:
            cart_database.remove(item)

credit_database = SQL_credit_cards

def checkCredentials(credit_card):
    if credit_database.contains(credit_card):
        true
    else:
        false:

item_database = SQL_items
def itemQuantityExists(item):
    if item_database.contains(item):
        true
    else:
        false

purchased_database = SQL_user_purchased
def checkOut(cart, credit_card)
    for item in cart:
        if itemQuantityExists(item):
            if creditCredentials(credit_card):
                    charge(credit_card, item.cost())
                    purchased_database.update(item)
            else:
                print("This credit card is not valid"
        else:
            print("This item is out of stock")
```

# Discussion Forum

```
post_database = SQL_Forum_Posts
taboo_list_database = SQL_Taboo_List

# CRUD operations on forum posts database
def addPost(post_ID, text):
    post_database.add(post_ID, text)

def editPost(post_ID, new_text):
    post_database.update(post_ID, new_text)

def getPostData(post_ID):
    return post_database.get_from_post_ID(post_ID)

def deletePost(post_ID):
    post_database.delete(post_ID)

# Getting user text from GUI
def getUserText():
    input = QInputText()
    text = input.getText()

    for words in text:
        if not words in taboo_list_database:
            return text

    input.show("Please do not include taboo words!")

# GUI for user adding a post
def userAddPost():
    screen = QScreen()
    screen.display("Please enter the text of your post")
    text = getUserText()
    confirm_buttom -> (on_click) = # Look below #
    if post is reply:
        addPost(current_user, text, post_ID_ref)
    else:
        addPost(current_user, text)

# Creating a viewable and interactable post GUI
def createGUIPost(post):
    postView = XML()
```

```
    image = post.getImage()
    username = post.getUsername()
    text = post.getText()
    price = post.getPrice()
    rating = post.getRating()
    postView.add(image, username, text, price, rating)

    reply_button -> (on_click) = userAddPost()
    if (current_user = post.getAuthor() or current_user.isAdmin()):
        delete_button -> (on_click) = deletePost()
        edit_button -> (on_click) = editPost()
        postView.add(delete_button)
        postView.add(edit_button)

# Creating the list view of posts
def createGUIPostListView():
    view = QListView()
    for post in post_database:
        view.add(createGUIPost(getPostData(post)))
        view -> (on_click) = createGUIPost()

# Creating a forum
def createNewForum(forum_name, text = "New Forum")
    view = QListView()
    view.enlarge(addPost(ROOT_USER, forum_name, text))
    new_post = QButton()
    new_post.setText("Click me to add a new post!")
    view.add(new_post)

# Profile Button (posts by user) which can be edited
# + Button (can add a post to a forum)
# Home Button
    # Forum Tabs (GPUs, CPUs, etc)
    # Complaints Tab
def createDiscussionForumNavigationBar():
    profile_button -> (on_click) = getProfile(user_ID)
    add_button -> (on_click) = addPost()
    home -> (on_click) = navigateToHome()
    home.add(gpu_button -> (on_click) -> createNewForum("GPU", "Graphics
for you Computer!"))
    home.add(cpu_button -> (on_click) -> createNewForum("CPU", "Processing
Power for your Computer!"))
    home.add(gpu_button -> (on_click) -> createNewForum("Cases", "Cases for
your Computer!"))
```

```
    home.add(complaints_button -> (on_click) ->
createNewForum("Complaints", "Unjustified?"))
```

## Delivery

```
delivery_company_db = SQL_Delivery_Companies
dc_id = signed in company's id

# Checks eligibility of Delivery Company
def checkEligible(company_id):
    return if company_id exists in delivery_company_db

# Delivery Company no longer wants to work with store
def terminatePartnership(company_id):
    statement = "DELETE {company_id} FROM DeliveryPartner"
    cursor.execute(statement)

# Delivery Company searches for new delivery
def requestNewDelivery(company_id):
    openBids = fetchOpenBids()
    if notEmpty(openBids):
        proceed to bidding screen
    else:
        display("No bids available for your company. Come back later!")

# Delivery Company makes a bid (generally, lower the better)
def processBid(company_id, amount, bid_id):
    if checkInRange(amount, bid_id) && (amount < curr_highest_bid(bid_id):
        if checkIfCompanyBidOnThisBefore(company_id, bid_id):
            statement = "UPDATE Bids SET amount = {amount} \
                         WHERE company = {company_id}, bid = {bid_id};"
            cursor.execute(statement)
        else:
            statement = "INSERT INTO Bids(company, amount, bid) \
                         VALUES({company_id},{amount},{bid_id});"
            cursor.execute(statement)
    else:
        display("Your price is unreasonable.")

# Delivery Company confirmed to deliver this item
```

```
def addBiddedDelivery(company_id, bid_id):
    statement = newDeliveryStatement(company_id,
                                     itemForOrder(bid_id),
                                     customerForOrder(bid_id),
                                     autoGenerateShippingID,
                                     "Shipment info sent.")
    cursor.execute(statement)

# Delivery Company changes status of delivery
def updateDeliveryStatus(company_id):
    shipping_id = promptUserToChangeWhichItem()
    new_status = promptUserToChangeItemShipStatus()
    statement = "UPDATE Deliveries SET status = {new_status} \
                 WHERE id = {shipping_id};"
    cursor.execute(statement)
```

## Supplies

```
supply_company_db = SQL_Supply_Companies
sc_id = signed in company's id

# Checks eligibility of Supply Company
def checkEligible(company_id):
    return if company_id exists in supply_company_db

# Supply Company no longer wants to work with store
def terminatePartnership(company_id):
    statement = "DELETE {company_id} FROM SupplyPartner"
    cursor.execute(statement)

# Supply Company requests to offer some supply
def requestOfferSupply(company_id, supply_offer_plea):
    company_name = getCompanyName(company_id)
    supply_number = generateRandomToken()
    # Admin determines if supply number token will be labeled as
    # approved in the database
    sendEmailToAdmin(subject="{company_name} would like to offer
supplies!",
                     body=supply_offer_plea,
                     application_number=supply_number)
```

```
# If supply request is approved, company can sell
def sellSupply(company_id, supply_number)
    if checkSupplyApproved(supply_number):
        sellSupply()
        display("Successfully sold supply for some money!")
    else:
        display("This request had not yet been approved.")

# Supply Company gives announcement
def giveAnnouncement(company_id, announcement):
    requestAdminToPostAnnouncementOnHomeScreen(company_id, announcement)
```

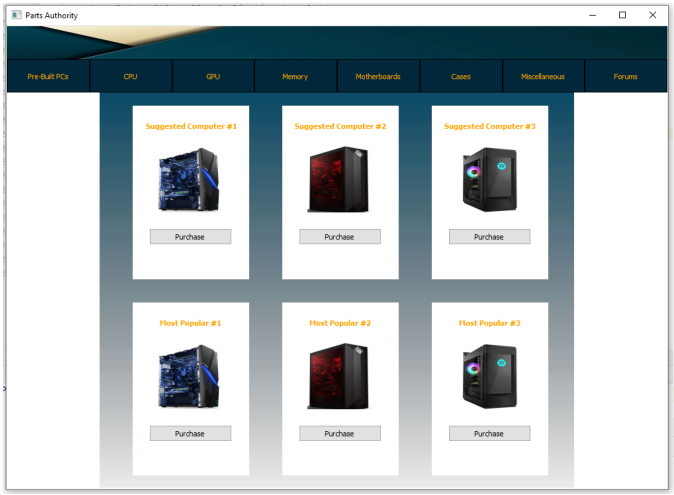
## Store Manager

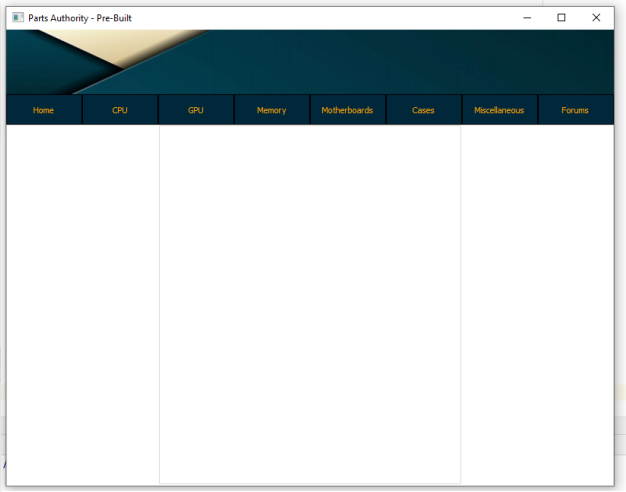
```
warning_list_database = SQL_Warning_List
ban_list_database = SQL_Ban_List
complaints_database = SQL_Complaints
items_database = SQL_Items

# CRUD operations on several databases
def banCustomer(customer_ID, ban_type, reason):
    ban_list_database.add([customer_ID, ban_type, reason])

def tabooList_add(word):
    taboo_list_database.add(word)

def warningList_add(warning_reason, count):
    warning_list_database.add([warning_reason, count])

# . pretend the rest of the RUD operations are there . #

# Create a nice view for the manager to see
def viewSQLTable(sql_table):
    view = QWindow()
    for entry in sql_table():
        view.add(entry.getInfo())
        if (current_user.isAdmin())
            delete_button -> (on_click) = delete()
            edit_button -> (on_click) = edit()
```

```
# Get the HomePage GUI that the Manager can edit/delete using buttons
def getHomePage():
    view = QWindow()
    list_of_views = []
    for item in items_database:
        if item is valid:
            list_of_views.add(getGUIView(item))
            if current_user.isAdmin():
                view.add(delete_button)
                view.add(edit_button)
        else:
            view.alert(item + "is not currently in stock")
    view.add(sorted(list_of_views, reverse = True))

# Ban a user/company/clerk
def ban(ID, reason):
    banCustomer.add(ID, reason)
    switch(ID):
    case(user):
        ID.setPermission(NO_POST_FORUM)
        ID.setPermission(NO_BUY_STORE)
    case(clerk):
        ID.setPermission(NO_CHOOSE_BID)
    case(deliver_company)
        ID.setPermission(NO_BID_STORE)
        break:
    case(computer_parts_company)
        ID.setPermission(NO_SELL_STORE)
        break:
```

## 5. System Screens

Demonstrate major GUI screens of the system and a prototype of one functionality of your own choice.




On the left is the home screen with the suggested computers and most popular computers, this screen is resizeable.
On the right is a raw layout design for the product listings, the products have not been listed yet so you don't see anything. Below is an image with the current functionality of the register screen, the gui of the register screen will be polished up with better designs.


## 6. Address of Git Repo

GitHub Link for Project: https://github.com/AshKhanNY/SoftwareEngineering

## 7. Meeting Minutes

| Date | TASK PERFORMED BY MEMBER ON DATE | |
|---|---|---|
| 4/11/2021 (2 hours) | **Ashraq Khan** | **Dewan Tahmid** |
| | Studied methods on the best way to store data for store, concluded on using SQL with the backend to CRUD data. Gave input on how to structure user interface and separate each user. | Researched PyQT5 for the Desktop GUI. Worked with Ash on the backend to create diagrams for all of the ER Diagrams for this report. |
| | **Azwad Shameem** | **David Diop** |
| | Researched PyQt5 for the Desktop GUI. Started implementing the user Interface and the overall design. | Researched PyQt5 for the Desktop GUI. |

| | | |
|---|---|---|
| 4/18/2021 (2 hours) | **Ashraq Khan** | **Dewan Tahmid** |
| | Developed schemas for the store database and what attributes to consider for each table (ex. store items, company delivery/supply requests, taboo list for forums, etc.). Wrote pseudo code on how companies can deliver or supply products. | Created prototypes and rough drafts for discussion forums and online computer store. Wrote pseudocode for the discussion forum. Used by Azwad to implement features of the GUI. |
| | **Azwad Shameem** | **David Diop** |
| | Improved the user Interface and the user roles in the implementation | Began prototyping code for the GUI (developed early version of login and register screen). |

| | | |
|---|---|---|
| 4/25/2021 (3.5 hours) | **Ashraq Khan** | **Dewan Tahmid** |
| | Worked with Dewan to create numerous diagrams for the second report. Such diagrams include petri-nets for user/customer/company view of application, use case and class collaboration diagrams for store partner companies, and polishing ER charts. | Worked with Ash to create numerous diagrams for the second report. Such diagrams include the pre-polished version of the ER diagram and class collaboration diagrams for customer forum and store clerk/manager and whole diagram. |
| | **Azwad Shameem** | **David Diop** |
| | Added more features to the user interface and started work on the required functionality. | Worked with Azwad to continue development of the GUI |