Phantom

Bid-hub
Design Report
For Web Application

Version <1.1>

Revision History

Date	Version	Description	Author
01/Nov/22	1.0	First Version of Mini Ebay	Meng Wai Chan Sam Philips Milan Thapa Sean Iori Cometa Umali
24/Nov/22	1.1	Design Report	Meng Wai Chan Sam Philip Milan Thapa Sean Iori Cometa Umali

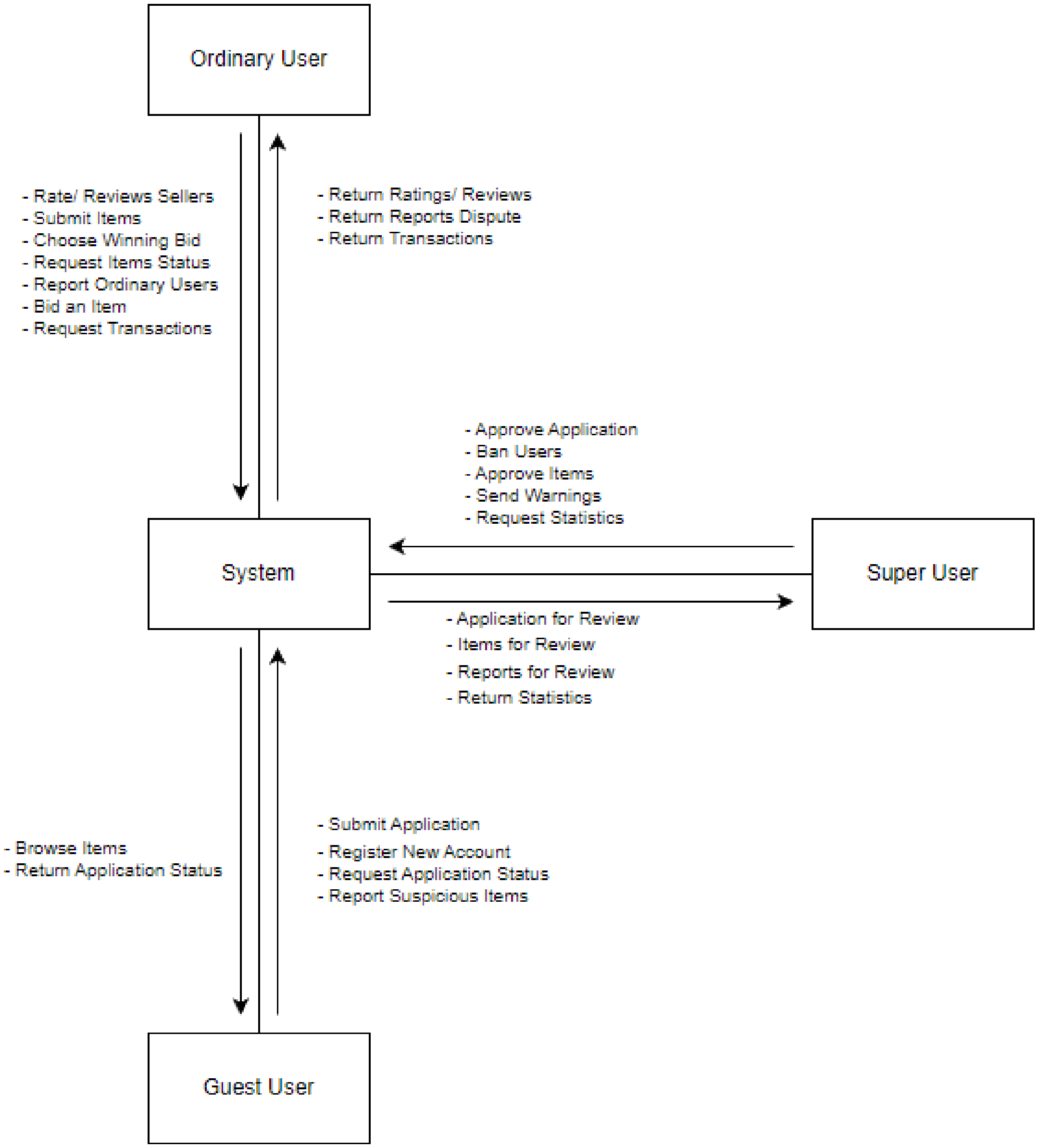
Table of Contents

Design Report

1. *Introduction*

This design report will provide an overview of this program, how it will function and how it looks. The purpose of this report is to provide readers a better understanding of this system.

1.1 *Collaboration Class Diagram*

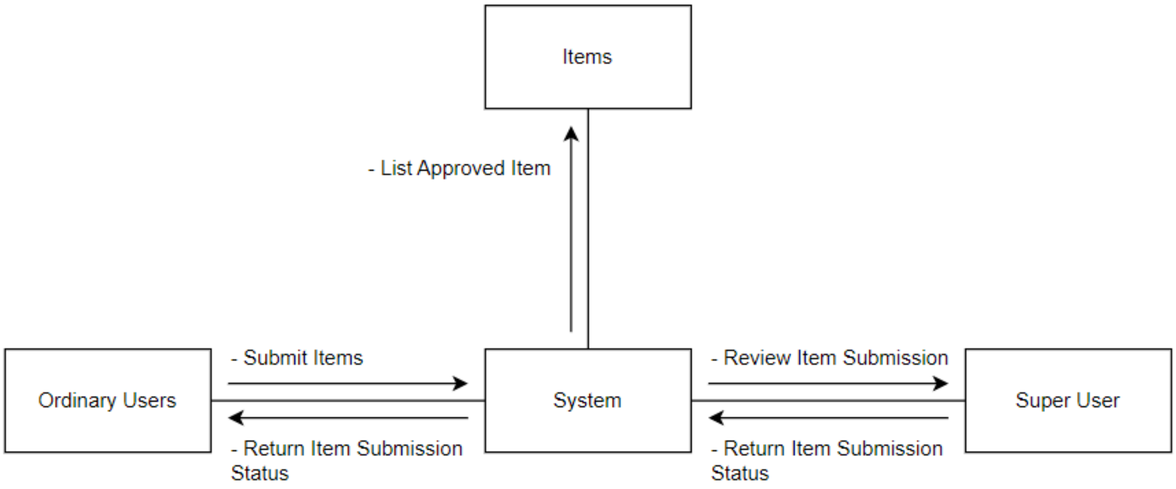


2. *Overall Description*

2.1 / 2.2 Scenarios For Each Case / Class Diagrams for Cases

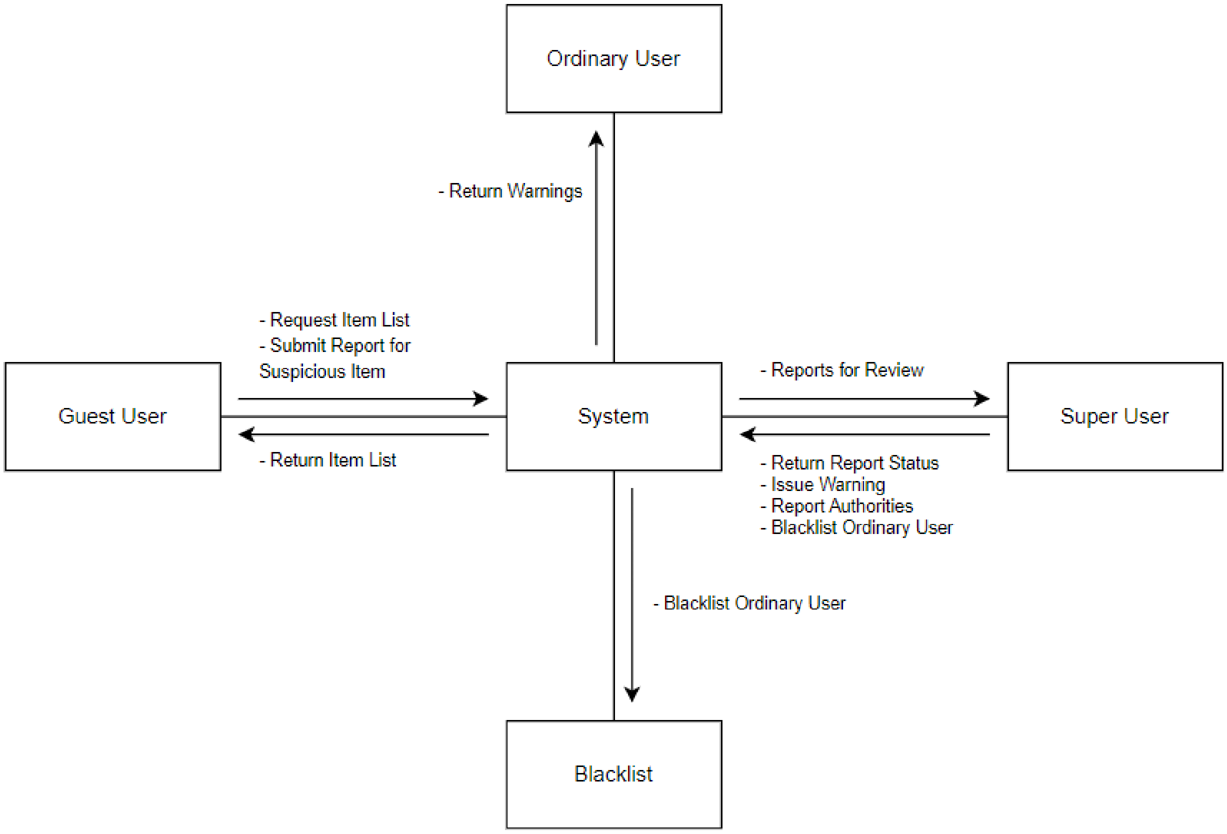
2.1.1 Item Submission system

The following diagram is to represent how ordinary users will submit items for Super User to review for bidding.



2.1.2 Guest User Browsing and Reporting Suspicious Items

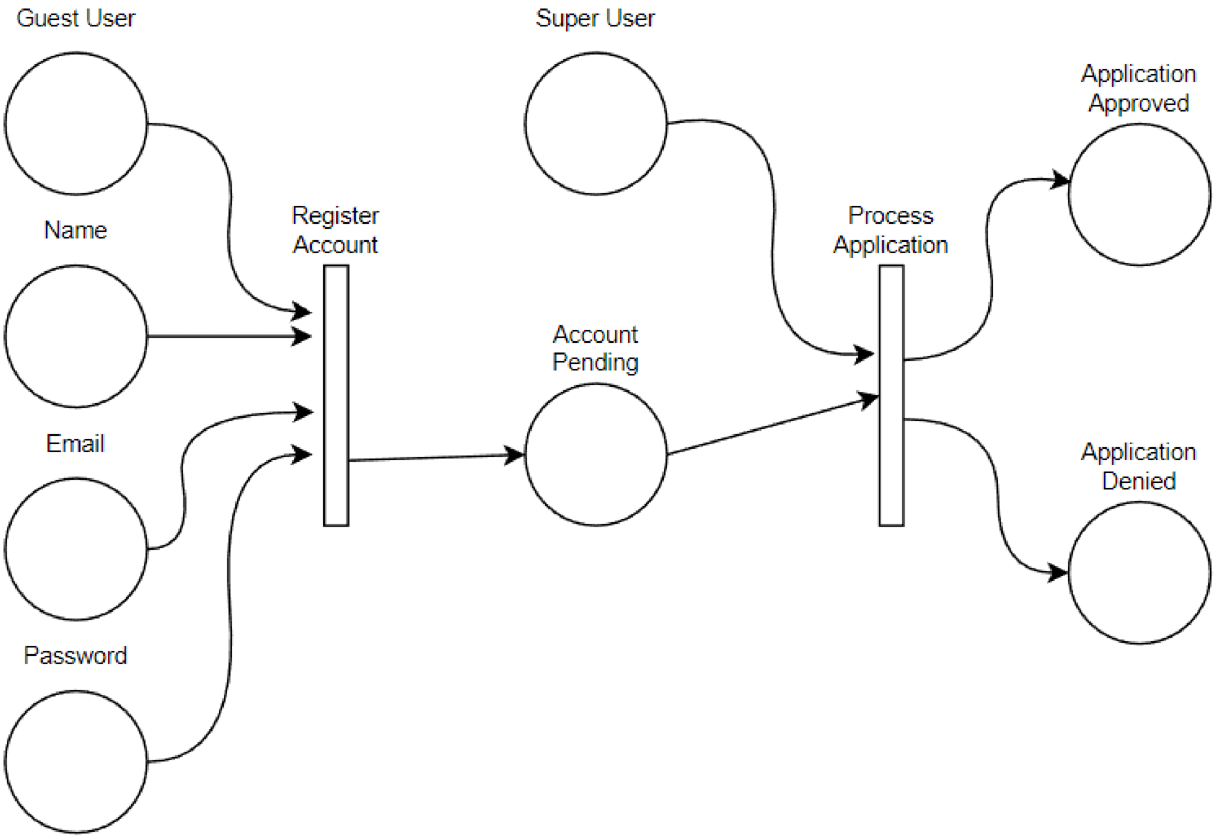
The following diagram demonstrates how Guest Users will be able to browse listed item on the website and report items for review for Super Users .If the item is deemed suspicious Super User will issue warnings to the suspicious Ordinary User, if too many warnings has been issued, the user will be blacklisted from the site.



2.3 Petri-net Diagram

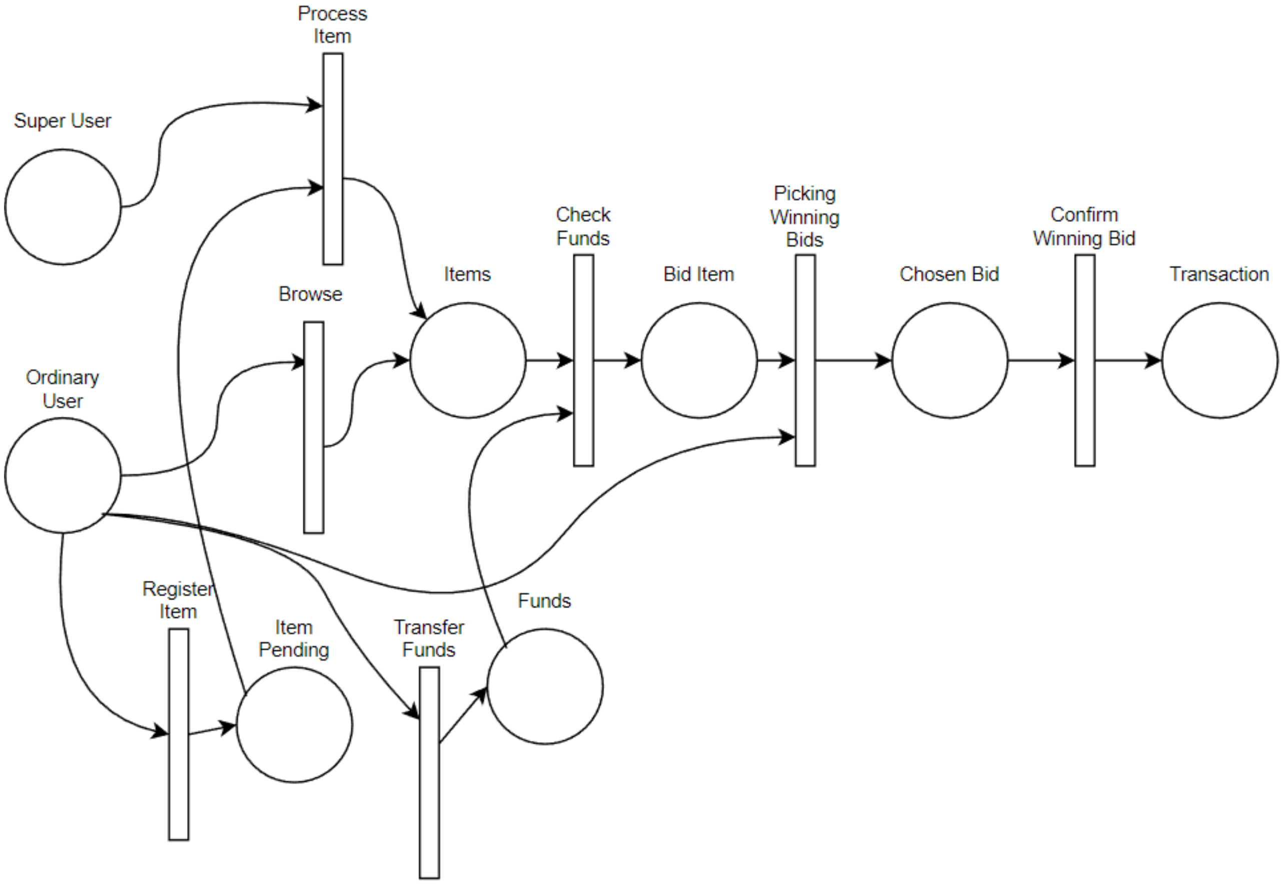
2.3.1 Registering Accounts

The following Petri-net diagram is to showcase how a Guest User will be able to register a new account in the system, A super User will oversee the Applications for approval.

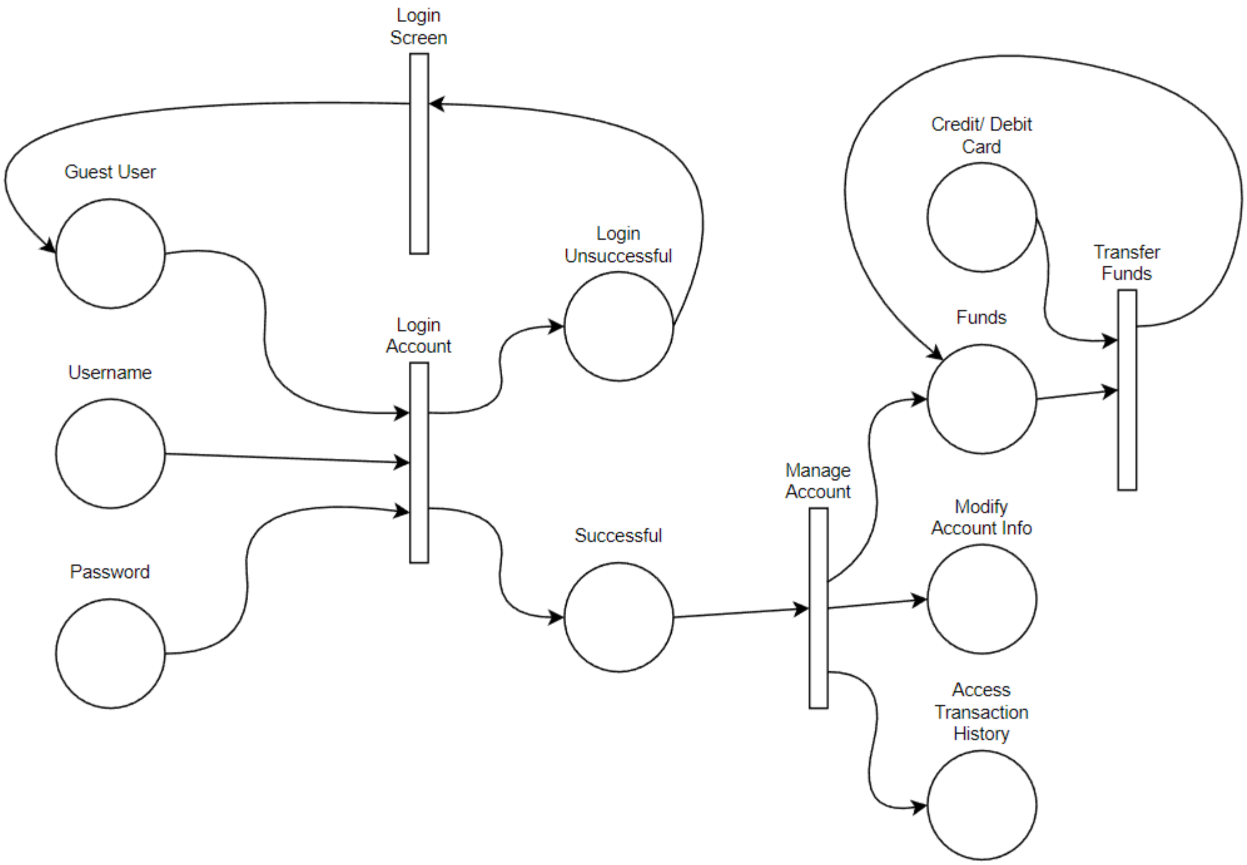


2.3.1 Bidding System

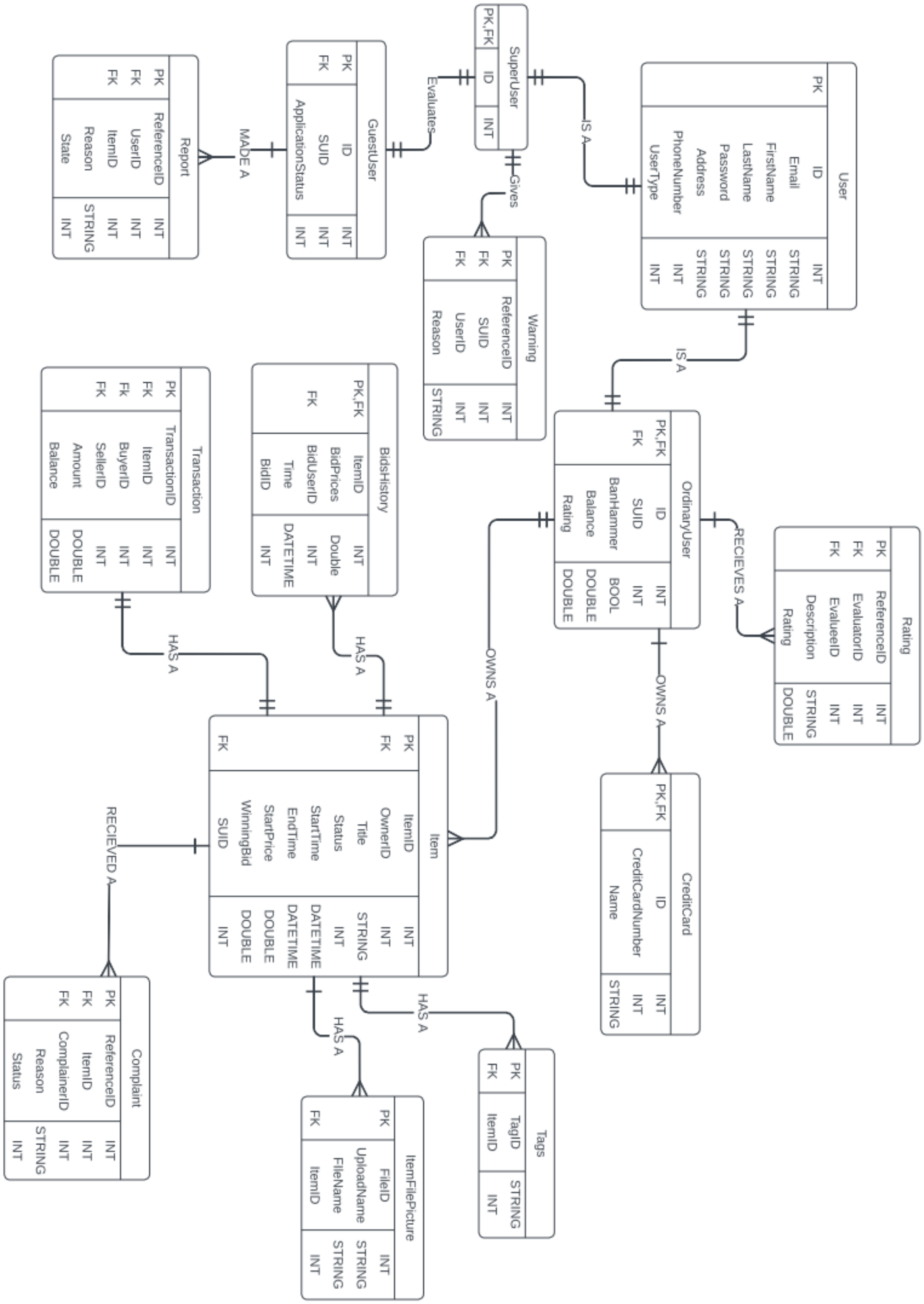
The diagram below is a representation of how users will bid items within the system, and how the process will proceed with Ordinary Users able to pick the winning bid if necessary.



2.3.1 Login System



3. *Entity-Relationship Diagram*



4. *Detailed Design*

4.1 AddNewItem

```
input
    title
    endTime
    startPrice
output
    status
def AddNewItem(title, endTime, startPrice):
    get current user from request headers
    add new item to db with given information
    if successful return 200 series code
    else return 400 or 500 depending on error type
```

4.2 *UploadPicture*

```
input
    fileData
    itemId
output
    status
def UploadPicture(fileData, itemId):
    get raw data stream of photo, upload to server and create DB entry or file
    Edit db entry to link to given item id
    if successful return 200 series code
    else return 400 or 500 depending on error type
```

4.3 *PlaceBid*

```
input
    itemId
    bidAmount
output
    status
def placeBid(itemId, bidAmount):
  get current user from request headers
  check if user has enough available funds for the bid
  if not, return 400 error
  if so create bid and return 200 status
```

4.4 *ChooseBid*

```
input
    itemId
    bidId
output
    status
def chooseBid(itemId, bidId):
    set winningBid to BidID
    if successful return 200 series code
    else return 400 or 500 depending on error type
```

4.5 *login*

```
input:
    username
```

password

output:

statusCode - Tells if the login was successful

sessionInformation - Returns information about the requested users session if successful

```
def login(username, password):
    check for user in db
        if not return 400 status
    hash PW and compare to PW in DB
        if incorrect return 400 status (don't send different error than user check, to make brute force attacks harder)
    return 200 and session information
```

4.6 *register (Apply for account)*

input:

username

email

firstname

lastname

password

address

phoneNumber

output:

statusCode - returns success if application was queued successfully, or an error if it was not submitted

```
def register(username, email, firstname, lastname, password, address, phoneNumber):
    if username is already in use return error
    if any input is invalid return error
else create user in db, set status to pending and return success
```

4.7 *processUserApplication*

```
input
        applicationId
        isApproved
output
        status - was a related transaction based on isApproved status successful?
def processUserApplication(applicationId, isApproved):
    get SUid from headers
    update application status to match isApproved
    if approved then enable user
    if rejected then update information with rejection information
```

4.8 *FileComplaint*

```
input:
        itemId
        message
output:
        status - was successfully entered into the report/complaint queue
def fileComplaint(itemId, message):
    get submitterID from headers
    create complaint in db with information
    return success code
```

4.9 *SearchItems*

input: - Not all inputs are required here

name

tagIds

page - allows to only request a specific offset of the items

limit - allows to only request a certain amount of items

output:

items - collection of items that match the input criteria. Given in an array format where each element is a serialized version of the item so the frontend can render them

```
def searchItems(name, tagIds, page, limit):

    query DB for items with a similar name, and/or matching tagIds. Limit result amount to limit requested, and offset by page * limit

    return db query results
```

4.10 *UpdateProfileInformation - only user id is required, any other inputs are used to update the existing users information*

input

userId - required

firstname

lastname

address

phoneNumber

email

password

output

status

```
def updateProfileInformation(userId, firstname, lastname, address, phoneNumber, email, password):

    find user with given id and update any given information

    For password only the hashed version is stored

    return 200 if successful, or an error code otherwise
```

4.11 *transferFunds*

```
input

        sourceId

        targetId

        amount

output

        status

def transferFunds(sourceid, targetid, amount):

        transfer funds from source to target. Ensure the source has at least the required amount to transfer.

        Update target and sources balances to new amount

        if any error then return error status code (400/500)
```

4.12 *submitReview*

```
input

        itemId

        rating

        description

output

        status

def submitReview(itemid, rating, description):
```

create a new rating for itemId, with given rating and description. Submitter information is retrieved from headers. person being evaluated is retrieved from item listing information

4.13 *banUser*

```
input
    userID
    reason
output
    status
def banUser(userId, reason):
    update user with userID to be disabled with reason for ban
    suid who made the ban is given from request headers
    return status code
```

4.14 *processItem*

```
input
    itemId
    newStatus
    reason
output
    status
def processItem(itemId, newstatus, reason):
    update item with given id to either be approved or denied (given from newstatus), and reason
    return status code
```

4.15 *processReport*

input

- reportId
- newState

output

- status

```
def processReport(reportId, newState):
        update given report to new state
        return status code
```

4.16 *issueWarning*

input

- referenceId
- reason
- userId

output

- status

```
def issueWarning(referenceID, UserID, reason):
    create warning in db with given reference id, reason and userId, and get SUid from request headers
```

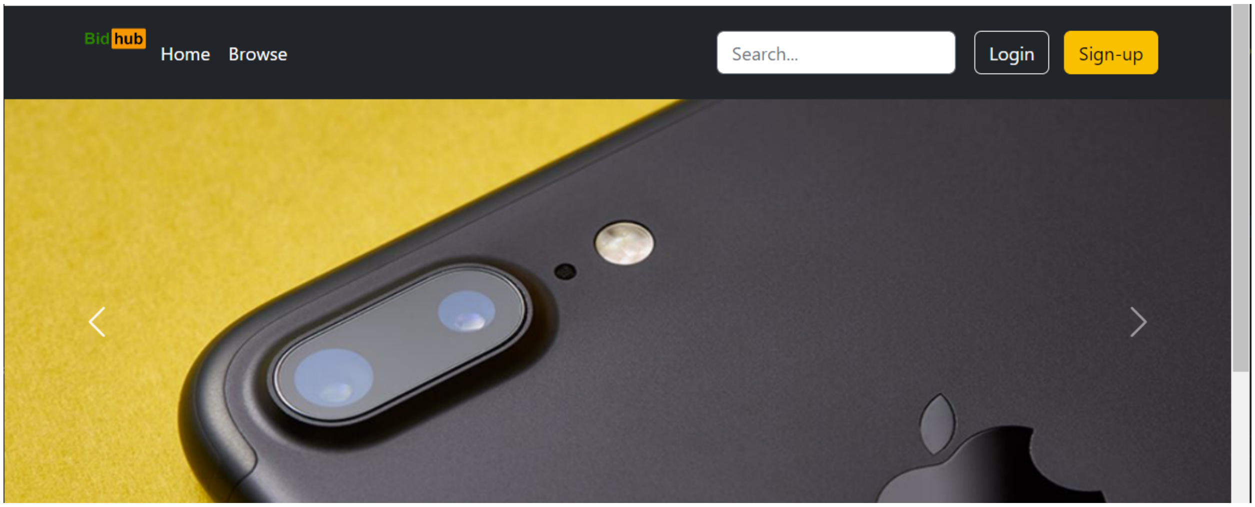
4.17 *viewStatistics*

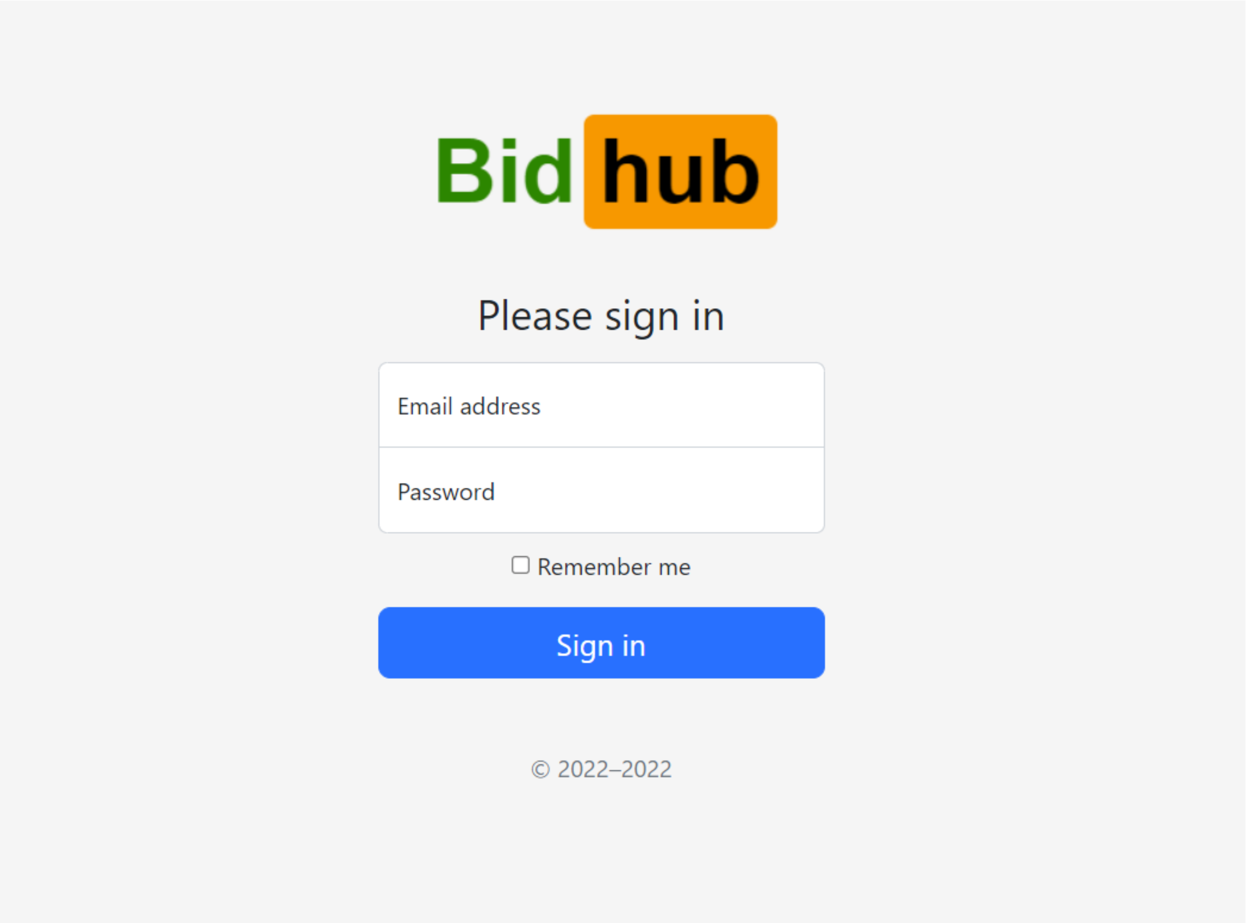
input - at least one must be provided for any given request

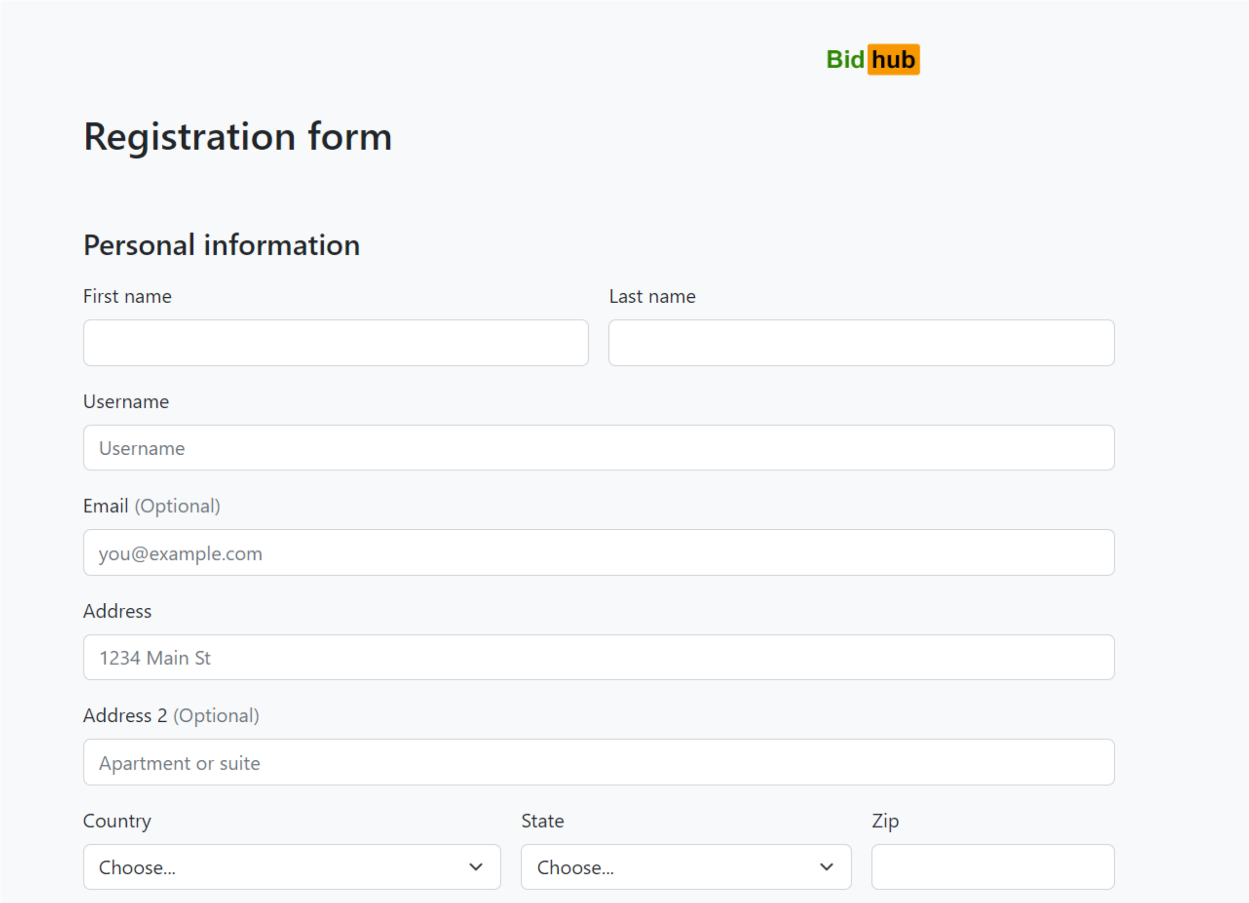
- startDateRange
- endDateRange
- name
- tagIds

```
        sellerId

        ratingRange

        buyerId

output

        collection of items

def viewStatistics(startDateRange, endDateRange, name, tagIds, sellerId, ratingRange,
buyerId):

        return collection of items that match the given criteria
```

5. *System Screens*




Bid hub
Please sign in
Email address
Password
□ Remember me
Sign in
© 2022–2022



Bid hub

Registration form

Personal information

First name

Last name

Username

Username

Email (Optional)

you@example.com

Address

1234 Main St

Address 2 (Optional)

Apartment or suite

Country

Choose...

State

Choose...

Zip

6. *Memos of Group Meetings*

Sep 13, 2022

- Group members Meet and Greet
- All member is present

Sep 28, 2022

- Choosing Programming Language

Oct 11, 2022

- Started working on Design Report Phase I
- Distributed work to all members

Oct 28, 2022

- Finishing Design Report Phase I

Nov 10, 2022

- Started working on Design Report Phase II
- Distributed work to all members

Nov 21, 2022

- Finalizing Design Report Phase II

7. *Github Repo*

https://github.com/armyf35/322_proj