

Secured Online Auction

Project Report (Phase II)

*submitted in partial fulfillment of the requirements
for the award of the degree of*

Bachelor of Technology

in

COMPUTER ENGINEERING

by

Harshal Sanjay Somani (10303320171124513004)
Sudhir Thimmappa Moolya(10303320171124510012)
Vishal Manohar Kshatriya (10303320171124510007)

Under the guidance of

Asst. Professor Heena Gangrekar



DEPARTMENT OF COMPUTER ENGINEERING
DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY
Lonere-402103, Tal. Mangoan, Dist.Raigad(MS)INDIA

2020-21

Certificate

The report entitled **Online Secured Auction** submitted by **Harshal Sanjay Somani (10303320171124513004)**, **Sudhir Thimmappa Moolya(10303320171124510012)** and **Vishal Manohar Kshatriya (10303320171124510007)** is approved to the partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Engineering.

Asst. Professor Heena Gangrekar

Guide

Department of Computer Engineering

Dr.A.W.Kiwalekar

Head

Department of Computer Engineering

Examiners:

1.Prof. Heena Gangrekar

2. Prof. Rahul Rathod

Place: Dr Babasaheb Ambedkar Technological University, Lonere.

Date: 01-07-2021

Acknowledgements

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed guide, **Prof. Heena Gangrekar**, for providing us the right guidance and advice at the crucial junctures and for showing us the right way.

We extend our sincere thanks to our respected **Head of the department Dr. Arvind Kiwlekar**, for allowing us to use the facilities available. We would like to thank the other faculty members also, at this occasion. Last but not the least, we would like to thank our friends and family for the support and encouragement they have given us during the course of our work.

Abstract

Online auctions are now an immensely popular component of the electronic marketplace. However, there are many fraudulent buying/selling behaviours that can occur during an auction (e.g., shill bidding, bid shielding, etc.). While researchers are proposing methods for combating such fraud, it is extremely difficult to test how effective these countermeasures are. This is primarily due to it being unethical to engage in fraudulent behaviour just for the purpose of testing countermeasures.

Furthermore, there is limited commercial auction data available due to the sensitivities of an online auctioneer being willing to admit that fraud has, or is occurring. In order to test fraud countermeasures in a controlled environment, we have created our own online auction server for conducting auction-related research. This paper presents our experiences with designing and implementing our own online auction system which we call u-Auction. At present, there is limited useful literature on auction system design. We present an analysis and design of the auction system by employing Unified Modeling Language (UML) to show the architectural model, subsystems, use cases, activity workflows, class diagram, user interfaces, and system sequence diagrams. Our auction model is grounded in object-oriented techniques and is open source so that other researchers can expand upon our approach.

Contents

Certificate	ii
Acknowledgementsiii
Abstract	iv
Contents	v
List of Figuresvii
1. INTRODUCTION	1
2. PROBLEM MOTIVATION	2
3. PARTICIPANTS AND THE ONLINE AUCTION FORMAT	3
3.1 Online Auction Participants/Stakeholders	3
3.1.1 Seller	
3.1.2 Bidder	
3.1.3 Auctioneer	
3.2 Online Auction Format	5
3.2.1 Bid history of an online auction	5
4. SYSTEMS ANALYSIS	7
4.1 Auction subsystems	7
4.2 User account management	
4.3 Auction managements	7
4.4 Auction Searching	
4.5 Bidding	
4.6 Payment	
4.7 Fraud detection	
5. SYSTEM DESIGN	11
5.1 Software model	11
5.1.1 User	11
5.1.2 Auctioneer	11
5.2 Interface Design	11
5.2.1 The User Interface	11
5.2.2 The Auctioneer Interface	11
6. IMPLEMENTATION	22

7.	ADVANTAGES	48
8.	CONCLUSION	49
	Future scope	50
	References	52

LIST OF FIGURES

1. User Account Management	5
2. Auction Management	6
3. Searching	6
4. Bidding	7
5. Payment	7
6. Fraud Detection	8
7. Activity Diagram of Bid Submission	9
8. Sequence Diagram of Bid Submission	10
9. E-Auction Software Model	10
10. Navigation Process of Bidder	11
11. Home Page	26
12. Register Page	27
13. Login Page	28
14. Add Item	29
15. Watchlist	30
16. Auction Result	31
17. Categories	32
18. Admin Page	33
19. Django Admin Item Category	34
20. Django Item Comment	35
21. Django Change Item	36
22. Django User To Change	37

CHAPTER 1

INTRODUCTION

Online auction sites, such as eBay and Yahoo! Auctions, are experiencing a dramatic increase in their popularity. The number of auction items hosted by eBay has increased from 110 million to approximately 266 million between July 2010 and September 2014 . A seller lists an item online for a set amount of time and buyers must place a bid higher than the last bid in order to purchase. Online auctions have removed the physical and logistical limitations of geographic proximity, time to organise, physical space, and small target audience.

However, the online environment creates many unique opportunities for people to cheat. Auction fraud can occur prior to an auction (e.g., misrepresentation of items, selling of black market goods, and triangulation), during an auction (e.g., shill bidding), or after the auction terminates (e.g., buyer does not pay for the item). Much research has been conducted around pre and post auction fraud . However, in-auction fraud is typically the hardest to develop effective countermeasures for as it deals with human behaviours and strategies that are somewhat unclear.

Shill bidding is the practice whereby a seller bids on his/her own auction in order to artificially increase the price that the winning bidder must pay. While it is understood that this is a problem, there are multiple strategies a shill bidder can engage in. As such there is much confusion over what actually constitutes shill bidding and how to effectively detect and prevent shill bidding. An even more significant problem is how to test the effectiveness of in-auction fraud counter measure proposals.

A major factor in the difficulty of testing in-auction fraud counter measures is the lack of available commercial online auction data. Online auctioneers do not share their auction data, commonly citing privacy reasons. However, it is more likely due to fear of damage to their public image should it be discovered that fraud is rampant in their auctions. Another significant issue with testing fraud counter measures is due to ethics/legality. For example, it is actually illegal for a researcher to engage in shill bidding in commercial online auctions primarily for the purpose of testing fraud counter measures. Due to these two major impediments, an alternative proposal for in-auction fraud testing must be examined. We were driven to create our own online auction system due to there being limited useful literature available on auction software design. Moreover, the existing auction software literature are typically not based on Unified Modeling Language (UML) [6], [7], [14], [16]. While there are vendors who sell auction software [2], such software is expensive and cannot be customised for our research requirements. This paper presents an analysis and design of our auction system which we call uAuction.

CHAPTER 2

PROBLEM MOTIVATION:

Wurmann et al. provide a software design for on-line English auctions that supports both software and human agents. Their proposed auction server named the Michigan Internet AuctionBot provides for flexible specification of auctions considering different parameters so that agent researchers can explore the design space of auction mechanisms. However, the authors do not show how they developed their auction system. Furthermore, the AuctionBot has been decommissioned since the early 2000s.

Kumar and Feldman present a software architecture and describe the various processes that comprise the auction application. They present an auction system which implements a collection of auction types. The key features of the underlying objects, processes, and interaction models are described along with how an auction relates to some commonly used trading models such as brokerages, two party negotiations, and competitive bid-based procurement. Moreover, they describe issues such as delay, security, and collaboration. However the paper does not use UML for analysing and designing the proposed auction system. Furthermore, the paper is not research-oriented.

Considering the limitations of the existing literature, we have designed uAuction. Our main target research is to detect shill bidding in real-time. To do this, we need to develop our own auction system as it is illegal/unethical to engage in shill bidding using commercial auction sites (e.g., eBay, TradeMe, etc.) for testing purposes. This paper's aim is to describe our knowledge and experiences with developing a web-based auction model.

- The reasons for developing this auction system are as follows:
- To fully understand online auction system requirements;
- To gain experience with administering an auction server and participating in online auctions;
- To enable testing of fraud detection/prevention techniques; and
- To educate auction users about fraud/auctioning behaviours.

CHAPTER 3

PARTICIPANTS AND THE ONLINE AUCTION FORMAT

3.1 Online Auction Participants/Stakeholders: There are three main stakeholders in an online auction:

- **3.1.1 Seller** – A seller lists an item (or collection of items) for sale. The seller is typically after the highest price possible for the item(s).
- **3.1.2 Bidder** – A bidder submits a bid for an item listed by the seller. The amount the bidder bids is an indication of what the bidder is willing to pay for the item being auctioned. The bidder is typically after the lowest price possible in order to win.
- **3.1.3 Auctioneer** – The auctioneer is responsible for hosting the auction, providing the resources required for the auction, and conducting the auction proceedings according to the auction rules. The auctioneer is usually paid a listing fee by the seller. In some cases, the auctioneer may receive a commission based on the winning price. In this case, the auctioneer will typically want the item to sell for the highest price possible.

3.2 Online Auction Format:

3.2.1 Bid history of an online auction

There are different types of auctions such as English, Vickrey, Dutch, Continuous Double auction, etc. The English auction is an open bid, ascending-price auction in which bidders place competing bids against other bidders in order to purchase the auctioned item. When a given time expires, the highest bidder wins the auction and must pay an amount equal to the winning bid. This type of auction is commonly used in real estate. Many online auctions are modelled on the English auction, except that an auction finishes at a predetermined closing time

Bid ID	Bidder	Bid Amount	Bid Time
8	David	\$65	Jan-31-16 09:18:16
7	Jessica	\$59	Jan-31-16 09:17:40
6	William	\$55	Jan-30-16 09:12:58
5	Shannon	\$51	Jan-28-16 21:54:09
4	Jessica	\$50	Jan-28-16 20:35:05
3	Robert	\$45	Jan-28-16 21:05:14
2	Jessica	\$42	Jan-28-16 21:04:44
1	Shannon	\$40	Jan-28-16 21:04:08

In general, the format of a bid in an online auction is: <bidder, bid amount, bid time>. A bidder can request for a price quote of an auctioned item from the Auctioneer at any time. The price quote comprises the bid information of the current highest bid. The auction bid history is a list of all bids that have been placed by the participating bidders in the auction (see Table I). Here the winning bid is \$65.

An online auction can have three additional parameters:

- Starting price – the minimum price at which the bidding must start;
- Reserve price – the minimum price the seller will accept. If the final price is below the reserve price, the result of the auction is void; and
- Minimum bid increment – the minimal amount required to outbid the current highest

bid. There are many variations available in online auctions. For example, bidders can alter/cancel bids,

confirm their placed bids, auction multiple goods, Buy It Now auctions, etc. However, the auction model in this paper only focuses on English auctions with a predetermined closing time.

CHAPTER 4

SYSTEM ANALYSIS

4.1 Auction Subsystems:

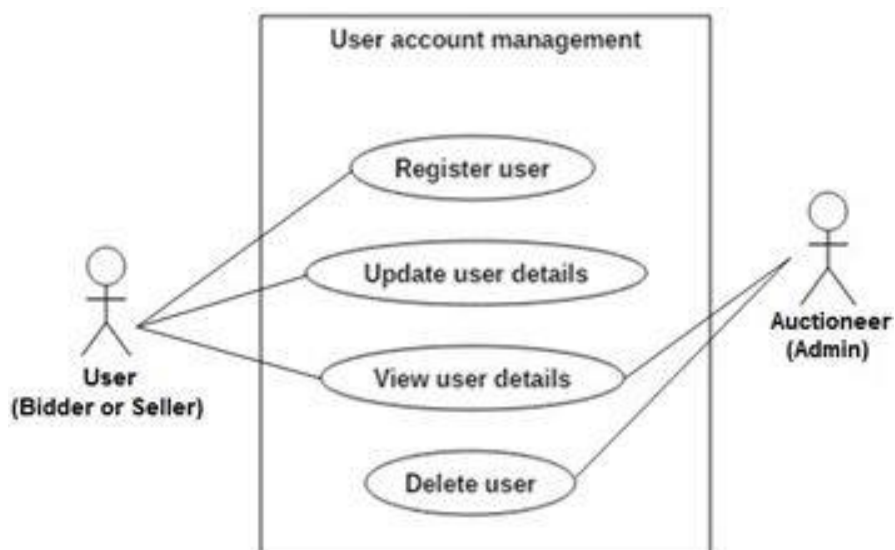
Software subsystems provide an intuitive way of visualising, understanding and analyzing the major functional requirements. Table II shows the uAuction's subsystems. Inconsistent: containing discrepancies in codes or names.

Subsystem	Users/Actors	Description
User-account management	Seller, Bidder, Auctioneer	A seller/bidder can register himself and can also update or view his details. The auctioneer can view/edit a user's details and delete users.
Auction management	Seller, Auctioneer	A seller lists products for auction. The auctioneer approves or disapproves an auction listing, and acknowledges the corresponding seller. He is also responsible for adding product categories.
Auction searching	Bidder	A bidder searches products on auction.
Bidding	Seller, Bidder, Auctioneer	Bidders submit bids in an auction and the auctioneer notifies the bidder who wins the auction.
Payment	Seller, Bidder, Auctioneer	A seller pays the auction listing fee to the auctioneer. The winning bidder has to complete transactions for paying the seller. The seller sends the product to the winner upon receiving the payment.
Fraud detection	Seller, Bidder, Auctioneer	The auctioneer can detect auction fraud. The bidder and seller are provided with information to determine whether fraud has occurred.

4.2 User account management:

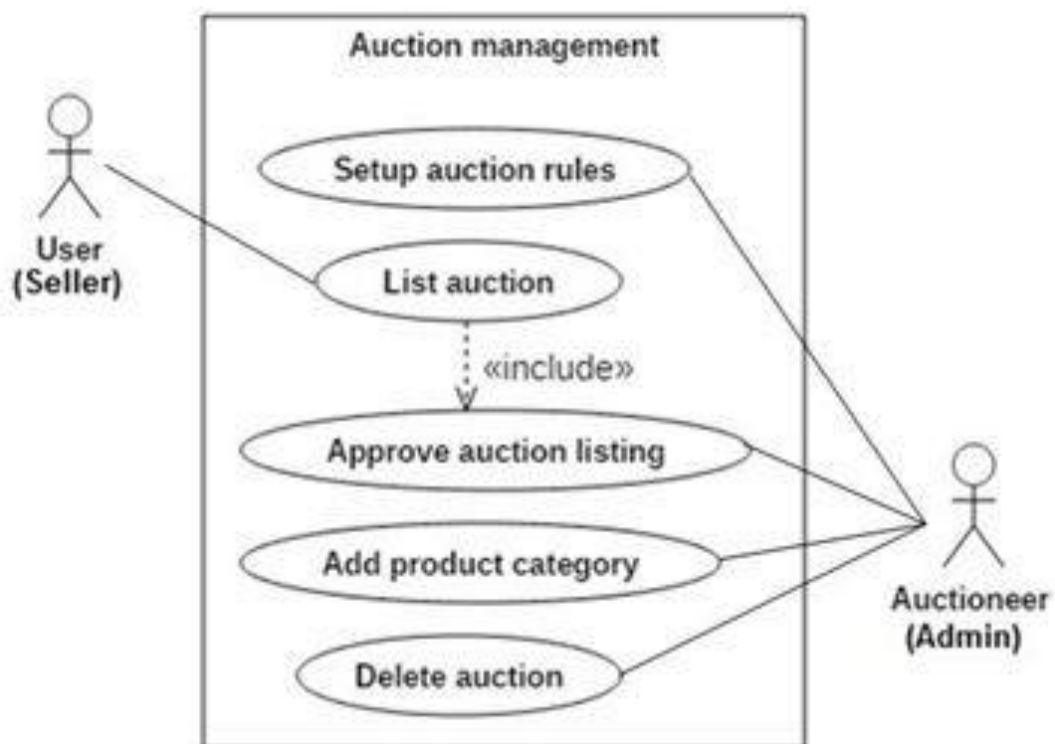
Fig. 1 presents the use case diagram that deals with the issues relating to the authentication of trading companies or users (sellers/bidders), creation of a profile for each user that reflects his interest in

different kinds of products, location details, and membership information. A seller/bidder can also update or view his details. The auctioneer can view the details of any user and delete any user account.



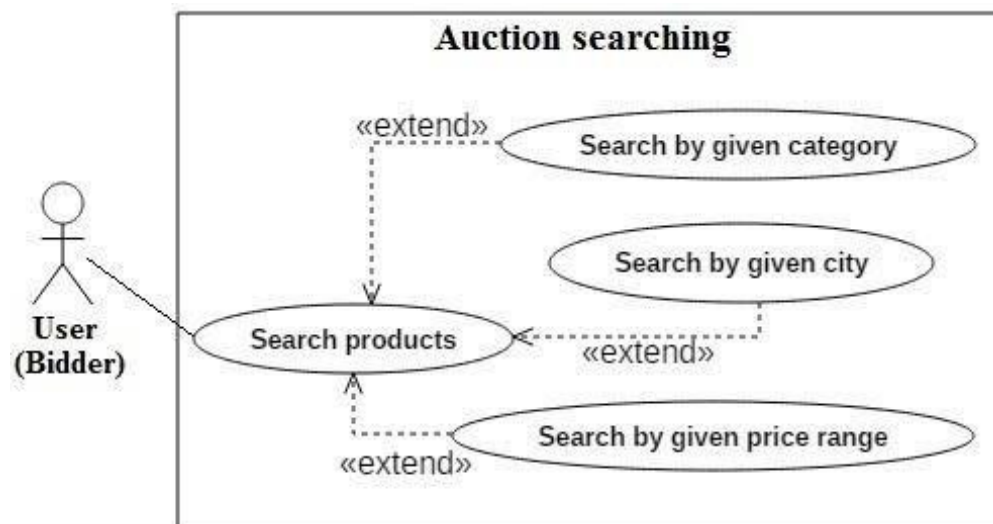
4.3: AUCTION MANAGEMENT:

A seller sets up the auction listing of products. This step deals with describing parameters such as price, reserve value, delivery dates, shipping details, payment type, starting and ending date and time of the auction, auction status, etc. The auctioneer approves the auction listing. After getting approval, the corresponding seller will be able to process his auction. Moreover, the auctioneer sets up the auction rules and guidelines. (see Fig. 2)



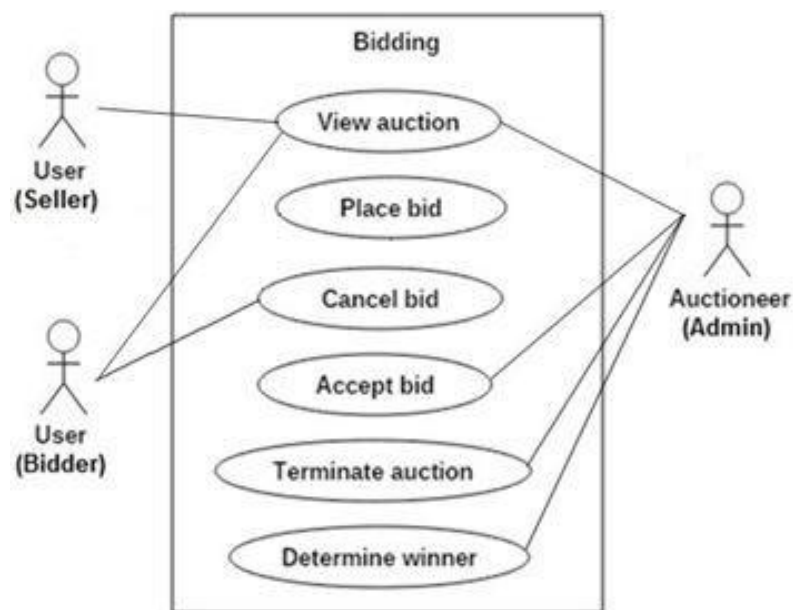
4.4: AUCTION SEARCHING:

Registered bidders are able to search for specific products being auctioned (see Fig. 3).



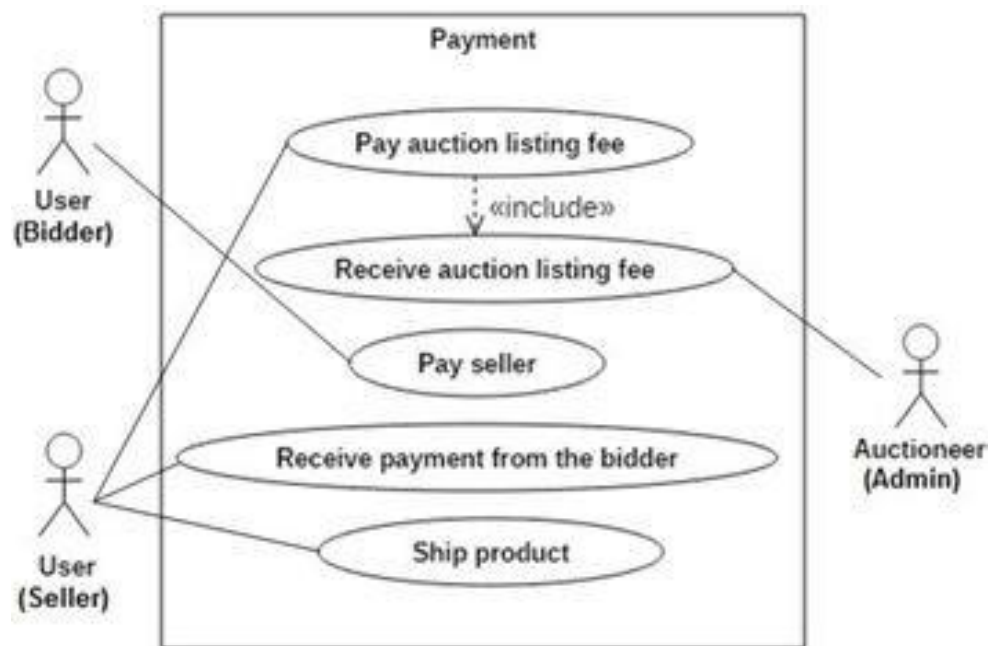
4.5: BIDDING:

Bidding: Fig. 4 shows the use case diagram of bidding subsystem that views the list of auctions and handles the collection of placing bids from different registered bidders and implements the bid control rules of the auction (e.g., minimum bid, bid increment, deposits required with bids). The auctioneer terminates an auction and then determines the winner according to the auction rules.



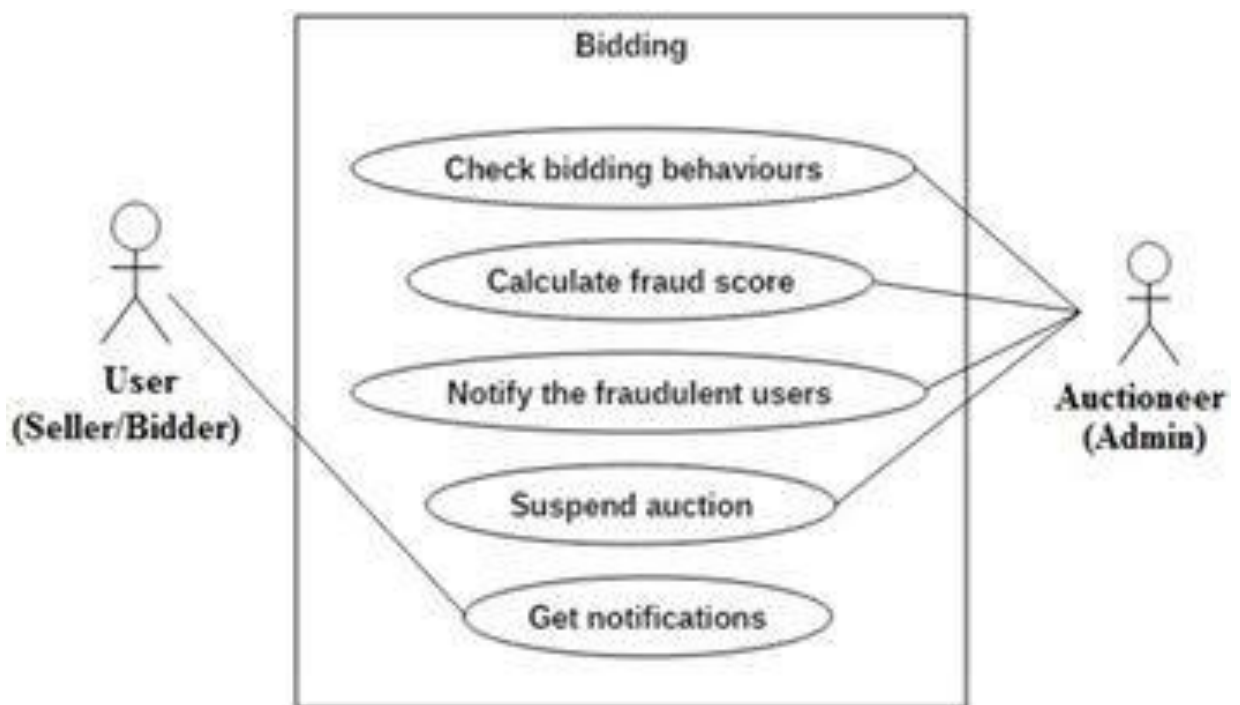
4.6: PAYMENT:

This subsystem handles the payment of auction listing fees to the auctioneer. A bidder who wins an auction sends payment to the corresponding seller. At the same time, the seller transfers the auctioned products to the bidder (see Fig. 5).



4.7: FRAUD DETECTION:

Fig. 6 illustrates the fraud detection subsystem. This subsystem monitors an auction for signs of fraudulent bidding behaviour. Bidders or sellers can provide information or are notified if fraud is suspected.



CHAPTER 5

System Design:

5.1 Software Model

Fig. 10 presents a high level software model for performing online auctions in uAuction. There are two main parties: a user (bidder or seller), and an auctioneer. A communication link is used to join the two parties. The participants of an online auction system are described as follows:

User: A user can be either a bidder or a seller. A bidder interacts with an auctioneer using an HTML browser. A two-way communication link is used for communicating with the auctioneer to place a bid or get information such as auction status from the auctioneer. A seller interacts with an auctioneer using an HTML browser. Using a two-way communication, the seller can post one or more items with the details of the items to the auctioneer for auctioning, as well as, can get notification from the auctioneer about the winner of the auction item after the end of the auction.

Auctioneer: The auctioneer runs a web server (e.g., MySQL server) and a scripting language like PHP. The auctioneer is responsible for taking information from the bidders and the sellers. The auctioneer provides registration service, log service, access control service, data persistence service, etc.

The entire auction is database driven. All state information (e.g., bids, timing, bidding amount, cancellation of bids, etc). about the auction is stored into the database. The transaction of the database starts when a bidder submits a bid or requests a price quote. Using the scripting language, the database generates dynamic web pages according to the bidder activity.

Interface Design:

In this section we first discuss the usability issues in auction software from the auctioneers perspective. As the software becomes more flexible, allowing a wide variety of auction styles to be used, the auctioneers task of specifying the complete set of rules for an auction becomes more arduous.

The User Interface: There are two types of user interface available in uAuction. Both users have to register into uAuction for performing selling and bidding activities. Our auction site, uAuction puts a user(seller/bidder) on the Welcome page from where a registered user can authenticate him to the site and initiate a secure session (login). An unregistered user will get the opportunity to fill in a registration form that may be processed online or off-line. Other additional features of these two user interfaces are discussed in this section.

i)

ii) **The Seller Interface:** A seller lists auction products by providing details of the product (e.g., the name of the product, starting price, regular price, product category, product description, image of the product, etc).

After listing an auction, a seller waits for approval from the auctioneer. The listing products will be ready for auctioning when the seller gets the notification of approval. Fig. 11 shows a seller interface of uAuction.

2. The Bidder Interface: Fig. 12 shows how the bidders navigate the auction web site in the auction implementation. Each bubble shows a web page and arcs from one page to another indicate that a hot link is available from the first page to the second.

The two-way communication link indicates that it is possible to browse the first page to the second and also possible to go back to the first page from the second. After registration, a bidder can browse through or search the products in the auction site, which will possibly result in a product being selected, and its description presented to the bidder (see Fig. 13). If the product is on auction, the rules of auction and bid history can be viewed, and bids can be submitted for that product. From the home page the bidder can also see a list of all auctions at uAuction or a subset of these that are in his personal auction watchlist. An auction gets added to a bidder's auction watchlist when the bidder explicitly takes an action to do so from certain auction pages, or implicitly when the bidder places his first bid. From lists, all auctions or auction watchlist, the bidder can select an auction and access the description of the product being auctioned, see the rules of the auction, or bid on the product.

The Auctioneer Interface: Fig. 14 shows an auctioneer interface of uAuction. The auctioneer interface consists of five components.

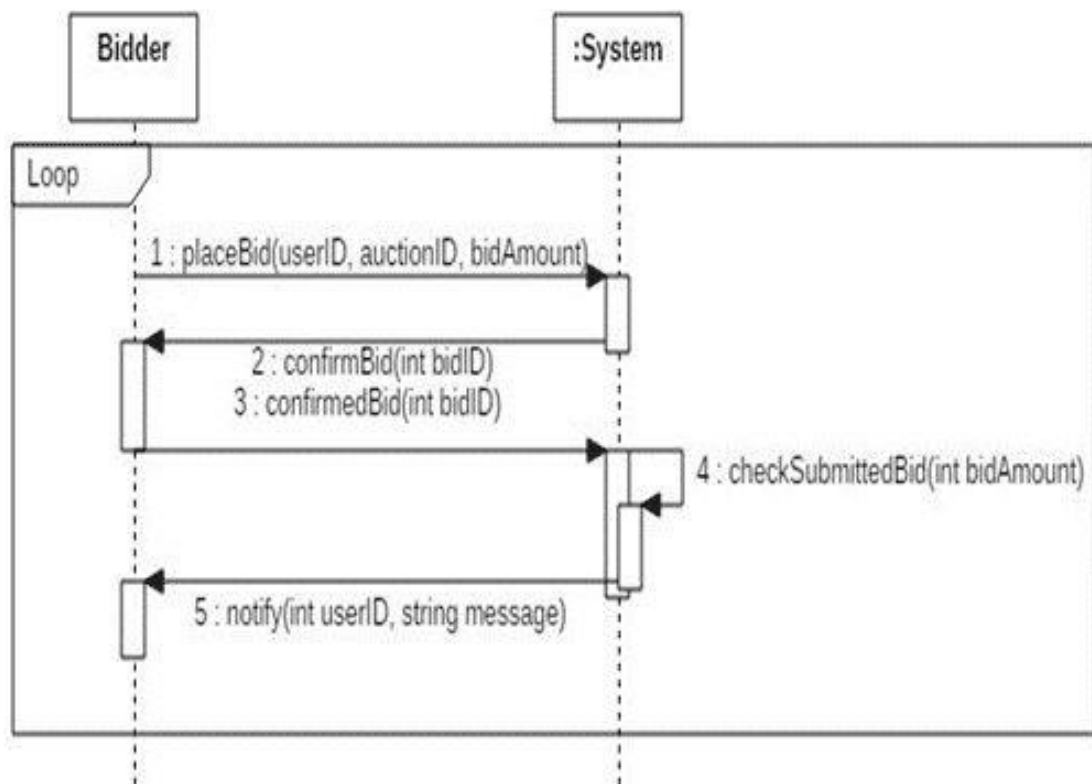
Provide approval to auction listings requested by a particular seller after which the product will be ready for auctioning process;

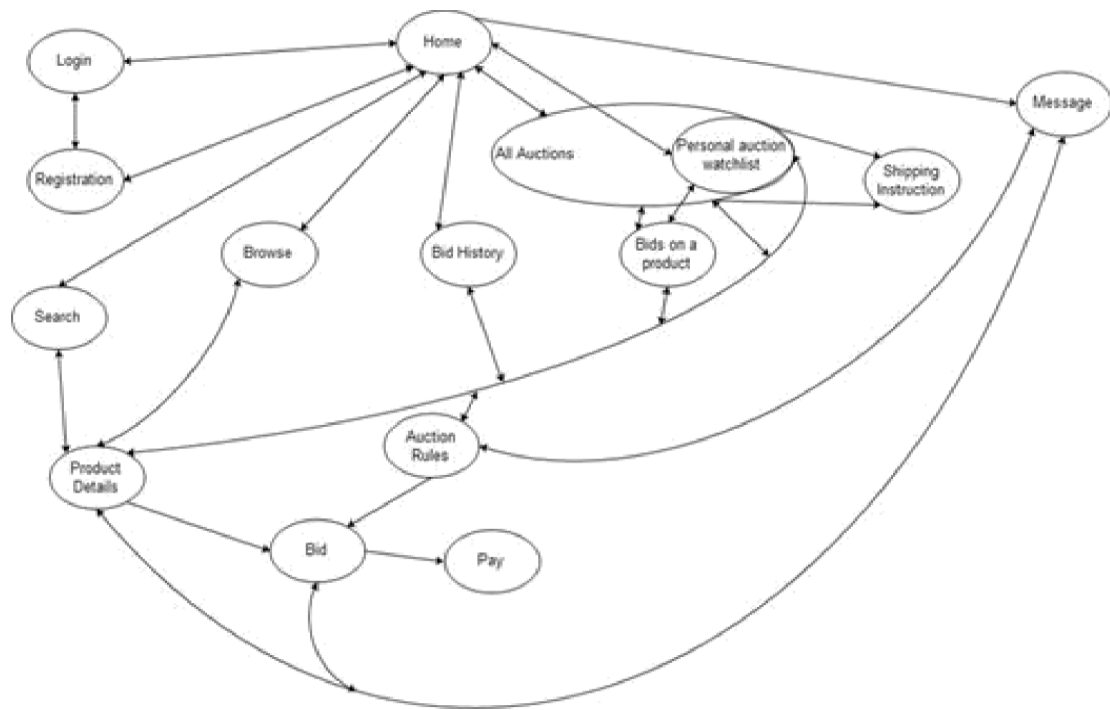
Allow the auctioneer to view the details of the products and monitor the progress of the various running auction;

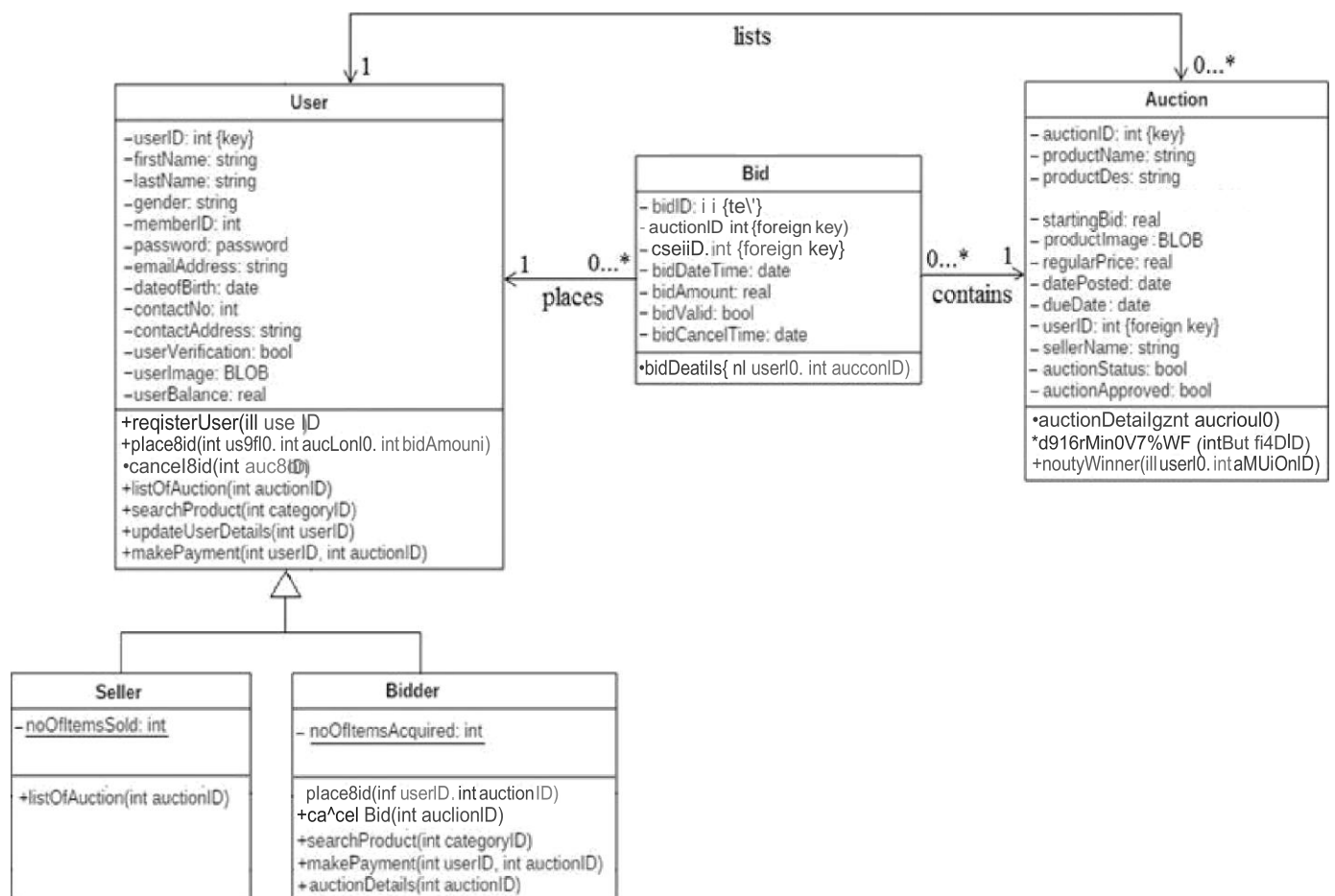
Add new categories of product;

Notify the seller if his product is not satisfied the auction rules or the bidder if he wins an auction;

Cancel any auction if any violation or fraud activity takes place.



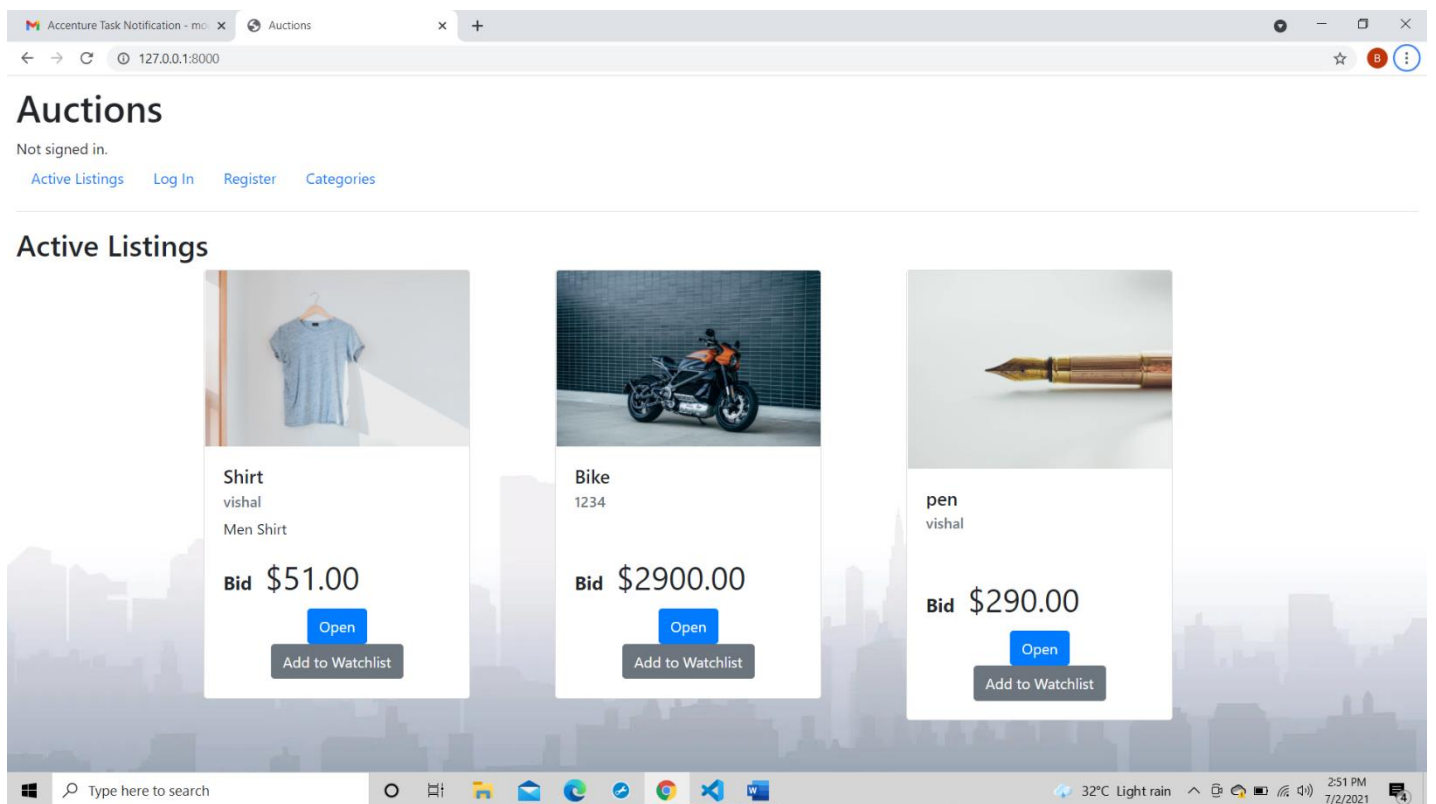




CHAPTER 7

IMPLEMENTATION

HOME PAGE



REGISTER PAGE

Accenture Task Notification - mo x Auctions x +

← → ↻ 127.0.0.1:8000/register ☆ B ⋮

Auctions


Not signed in.

[Active Listings](#) [Log In](#) [Register](#) [Categories](#)

Register

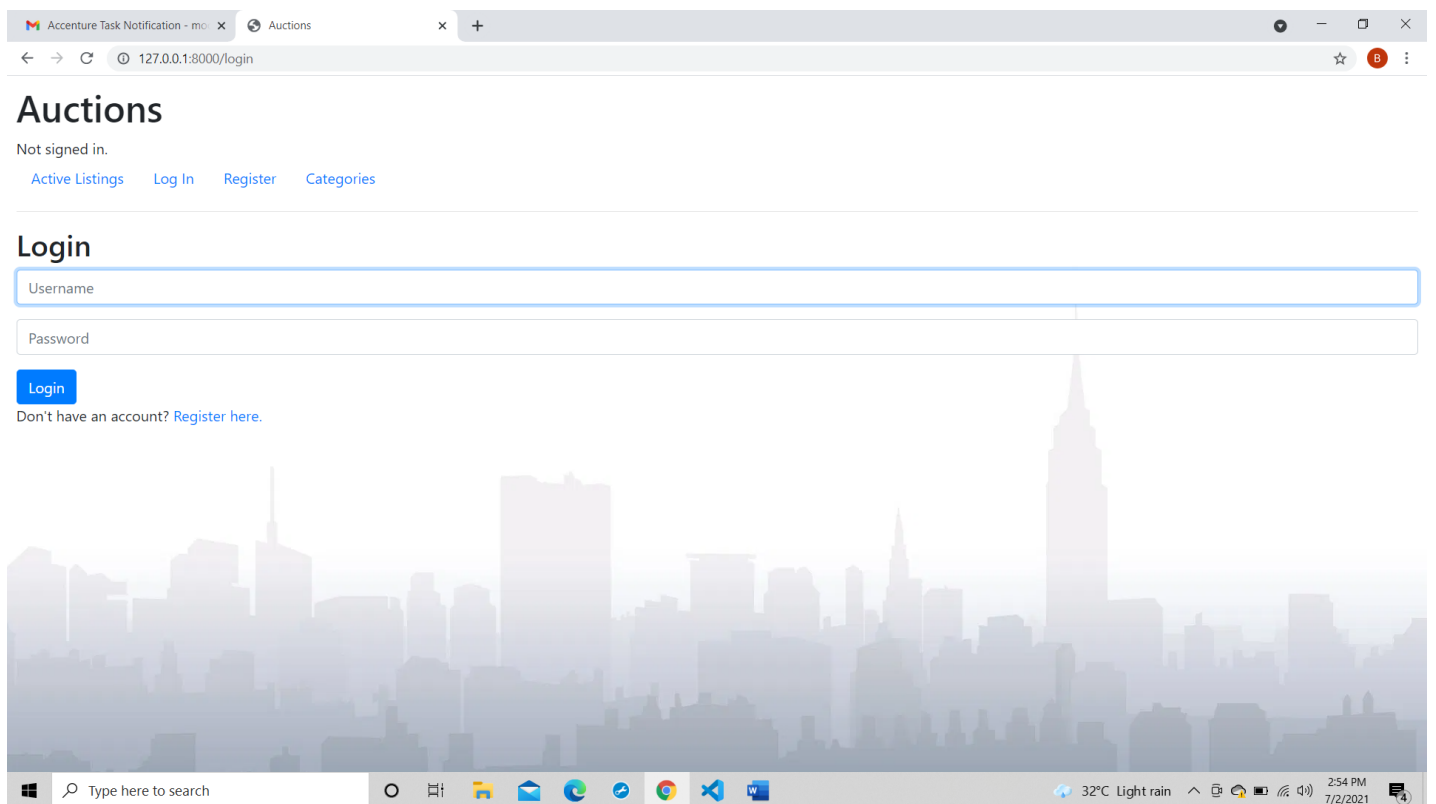
[Register](#)

Already have an account? [Log In here.](#)



Windows Taskbar: Type here to search | [Icons: File Explorer, Mail, Edge, Chrome, Word] | 32°C Light rain | 2:54 PM 7/2/2021

LOGIN PAGE



Accenture Task Notification - mo x Auctions x +

← → ↻ 127.0.0.1:8000/login ☆ B ⋮

Auctions

Not signed in.

[Active Listings](#) [Log In](#) [Register](#) [Categories](#)

Login

Don't have an account? [Register here.](#)

Windows taskbar: Type here to search, Task View, File Explorer, Edge, Chrome, Word, 32°C Light rain, 2:54 PM 7/2/2021

ADD ITEM

Accenture Task Notification - mo x Auctions x +

← → ↻ 127.0.0.1:8000/add ☆ B ⋮

Auctions

Signed in as **vishal**.

[Active Listings](#) [Log Out](#) [Add Item](#) [Watchlist](#) [Auctions History](#) [Categories](#)


Title:

Description:

Img url:

Starting bid:

Category: Clothing Submit



Windows

Type here to search

32°C Light rain 2:55 PM 7/2/2021

WATCHLIST

Accenture Task Notification - mo

Auctions

127.0.0.1:8000/watchlist


☆ B

Auctions

Signed in as **vishal**.

[Active Listings](#) [Log Out](#) [Add Item](#) [Watchlist](#) [Auctions History](#) [Categories](#)

Item Name	Starting Bid	Description	Action
Shirt	50.00	Men Shirt	Delete
Bike	2900.00		Delete



Windows

Type here to search

32°C Light rain

2:55 PM 7/2/2021

AUCTION RESULT

Accenture Task Notification - m...

Auctions


127.0.0.1:8000/auctions_history

Auctions


Signed in as **vishal**.

[Active Listings](#) [Log Out](#) [Add Item](#) [Watchlist](#) [Auctions History](#) [Categories](#)

Auctions you have won



Cycle
vishal
Cycle
Bid \$4000.00



Mobile
vishal
Mobile
Bid \$441.00

Type here to search

33°C Light rain

2:55 PM 7/2/2021

CATEGORIES

Accenture Task Notification - mo

Auctions

← → ↻

127.0.0.1:8000/category_list

☆

B

 ⋮


Auctions

Signed in as **vishal**.

[Active Listings](#) [Log Out](#) [Add Item](#) [Watchlist](#) [Auctions History](#) [Categories](#)

Categories

[Clothing](#)
[Toys](#)
[Electronics](#)
[Art](#)

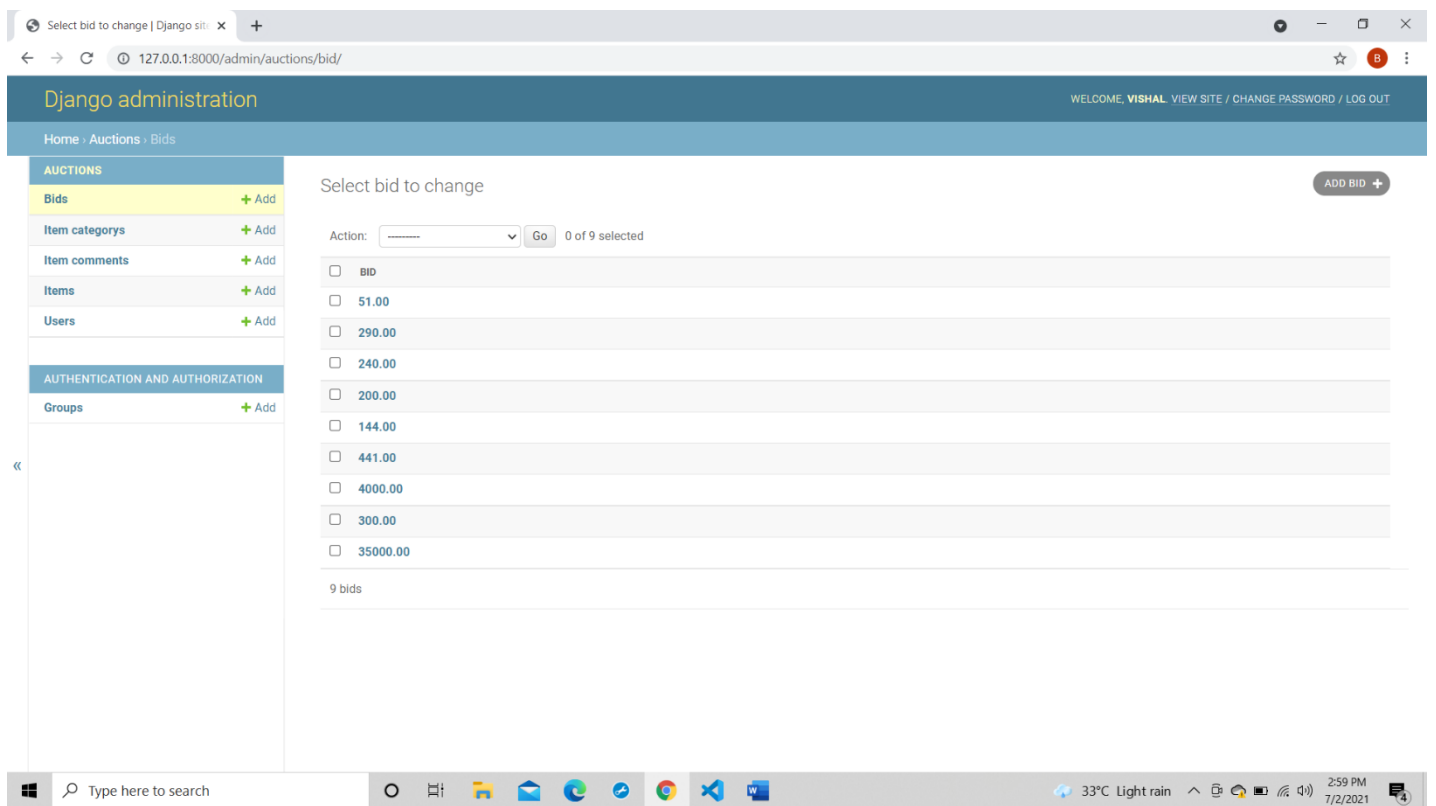


Type here to search

33°C Light rain

2:55 PM
7/2/2021

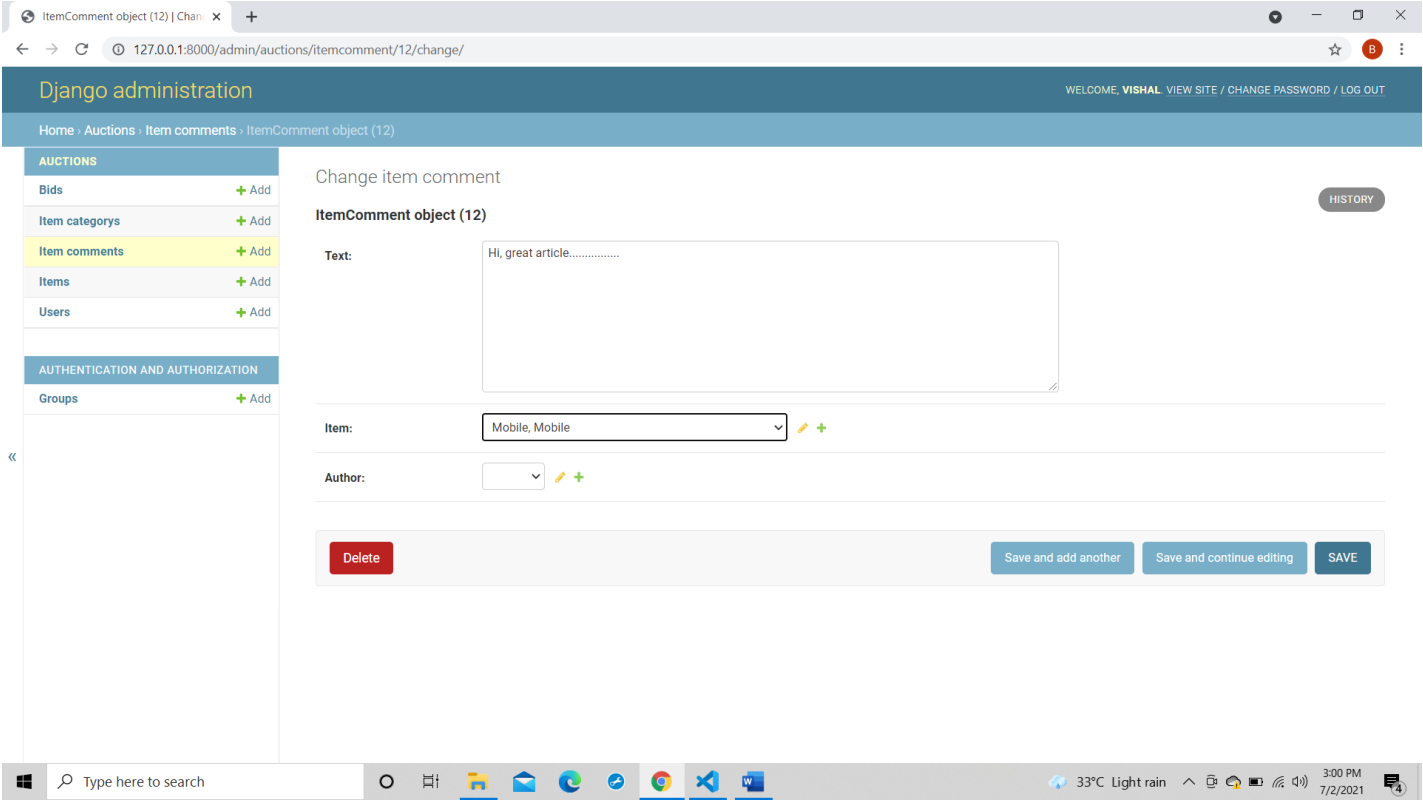
ADMIN PAGE



DJANGO ADMIN ITEM CATEGORY

The image shows a web browser window displaying a Django administration page titled "Django administration". The page is for editing an item category, specifically "clothing". The left sidebar contains a navigation menu with sections "AUCTIONS" and "AUTHENTICATION AND AUTHORIZATION". Under "AUCTIONS", there are links for "Bids", "Item categories" (which is highlighted), "Item comments", "Items", and "Users", each with a "+ Add" link. The main content area is titled "Change item category" and shows the current category "clothing". Below this, there is a form with a "Name:" field containing "clothing" and an "Item:" field containing "pen,". At the bottom of the form, there are three buttons: "Delete", "Save and add another", and "Save and continue editing", followed by a "SAVE" button. The top of the page shows the Django logo and the text "Django administration". The bottom of the page shows a Windows taskbar with various application icons and the system clock indicating 2:59 PM on 7/2/2021.

DJANGO ITEM COMMENT



DJANGO CHANGE ITEM

pen, | Change item | Django site

127.0.0.1:8000/admin/auctions/item/31/change/

☆ B

Django administration

WELCOME, VISHAL VIEW SITE / CHANGE PASSWORD / LOG OUT

Home · Auctions · Items · pen,

AUCTIONS

Bids + Add

Item categories + Add

Item comments + Add

Items + Add

Users + Add



AUTHENTICATION AND AUTHORIZATION

Groups + Add

Change item

pen,

HISTORY

User:  

Title:

Description:

Img url:

Starting bid:

Category:

☒ Status


Delete

Save and add another

Save and continue editing

SAVE

Type here to search



33°C Light rain

3:00 PM 7/2/2021

DJANGO USER TO CHANGE

Select user to change | Django si x +

127.0.0.1:8000/admin/auctions/user/

WELCOME, VISHAL. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home · Auctions · Users

AUCTIONS

Bids + Add

Item categorys + Add

Item comments + Add

Items + Add

Users + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Select user to change

Q

Search

Action: ----- Go 0 of 10 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	1234	rakesh@gmail.com			✖
<input type="checkbox"/>	Sudhir	moolyasudhir801@gmail.com			✖
<input type="checkbox"/>	admin	hello@gmail.com			✖
<input type="checkbox"/>	ale				✖
<input type="checkbox"/>	alejandro	alejandro@gmail.com			✖
<input type="checkbox"/>	alejo				✖
<input type="checkbox"/>	raj	raj@gmail.com			✖
<input type="checkbox"/>	santiago	santiagobn1@gmail.com			✔
<input type="checkbox"/>	suresh	suresh@gmail.com			✖
<input type="checkbox"/>	vishal	kshatriya789789@gmail.com			✔

10 users

FILTER

By staff status

All

Yes

No

By superuser status

All

Yes

No

By active

All

Yes

No

ADD USER +

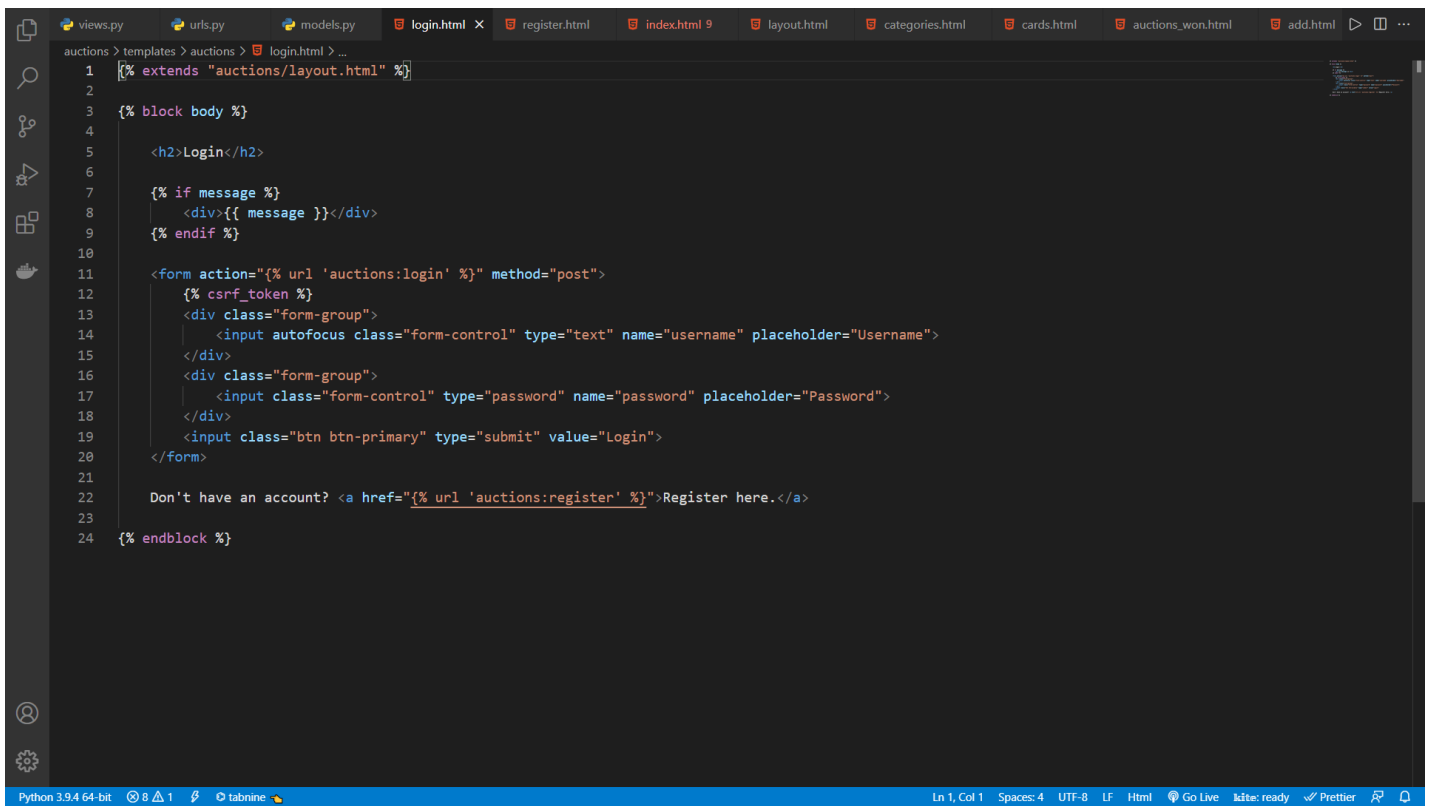
Type here to search

33°C Light rain

3:00 PM 7/2/2021

LOGIN.HTML

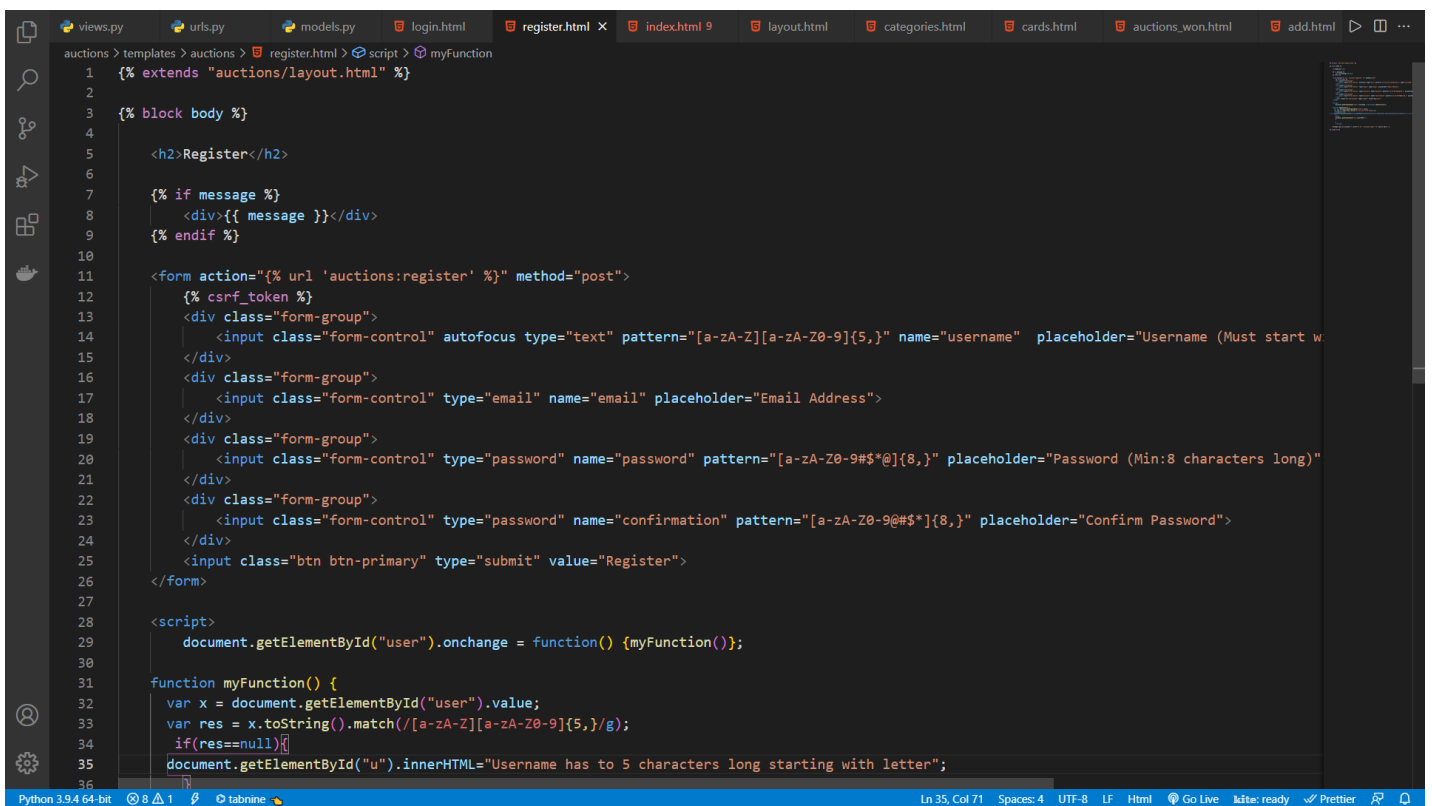
The Code of Login Page contains HTML and Django in it.
It is responsible for the user login once they have registered on the website.
This file is completely dynamic.



```
1  {% extends "auctions/layout.html" %}
2
3  {% block body %}
4
5      <h2>Login</h2>
6
7      {% if message %}
8          <div>{{ message }}</div>
9      {% endif %}
10
11      <form action="{% url 'auctions:login' %}" method="post">
12          {% csrf_token %}
13          <div class="form-group">
14              <input autofocus class="form-control" type="text" name="username" placeholder="Username">
15          </div>
16          <div class="form-group">
17              <input class="form-control" type="password" name="password" placeholder="Password">
18          </div>
19          <input class="btn btn-primary" type="submit" value="Login">
20      </form>
21
22      Don't have an account? <a href="{% url 'auctions:register' %}">Register here.</a>
23
24  {% endblock %}
```

REGISTER.HTML

The Register Page is responsible for the new user to create account on the website. It contains Username, Email and Password with the Username authentication. The Username must always start with an alphabet else the user won't be able to register. This page has completely dynamic content.



```
1 {% extends "auctions/layout.html" %}
2
3 {% block body %}
4
5 <h2>Register</h2>
6
7 {% if message %}
8 <div>{{ message }}</div>
9 {% endif %}
10
11 <form action="{% url 'auctions:register' %}" method="post">
12     {% csrf_token %}
13     <div class="form-group">
14         <input class="form-control" autofocus type="text" pattern="[a-zA-Z][a-zA-Z0-9]{5,}" name="username" placeholder="Username (Must start w
15     </div>
16     <div class="form-group">
17         <input class="form-control" type="email" name="email" placeholder="Email Address">
18     </div>
19     <div class="form-group">
20         <input class="form-control" type="password" name="password" pattern="[a-zA-Z0-9#%*]{8,}" placeholder="Password (Min:8 characters long)"
21     </div>
22     <div class="form-group">
23         <input class="form-control" type="password" name="confirmation" pattern="[a-zA-Z0-9#%*]{8,}" placeholder="Confirm Password">
24     </div>
25     <input class="btn btn-primary" type="submit" value="Register">
26 </form>
27
28 <script>
29     document.getElementById("user").onchange = function() {myFunction()};
30
31 function myFunction() {
32     var x = document.getElementById("user").value;
33     var res = x.toString().match(/[a-zA-Z][a-zA-Z0-9]{5,}/g);
34     if(res==null){
35         document.getElementById("u").innerHTML="Username has to 5 characters long starting with letter";
36     }
```

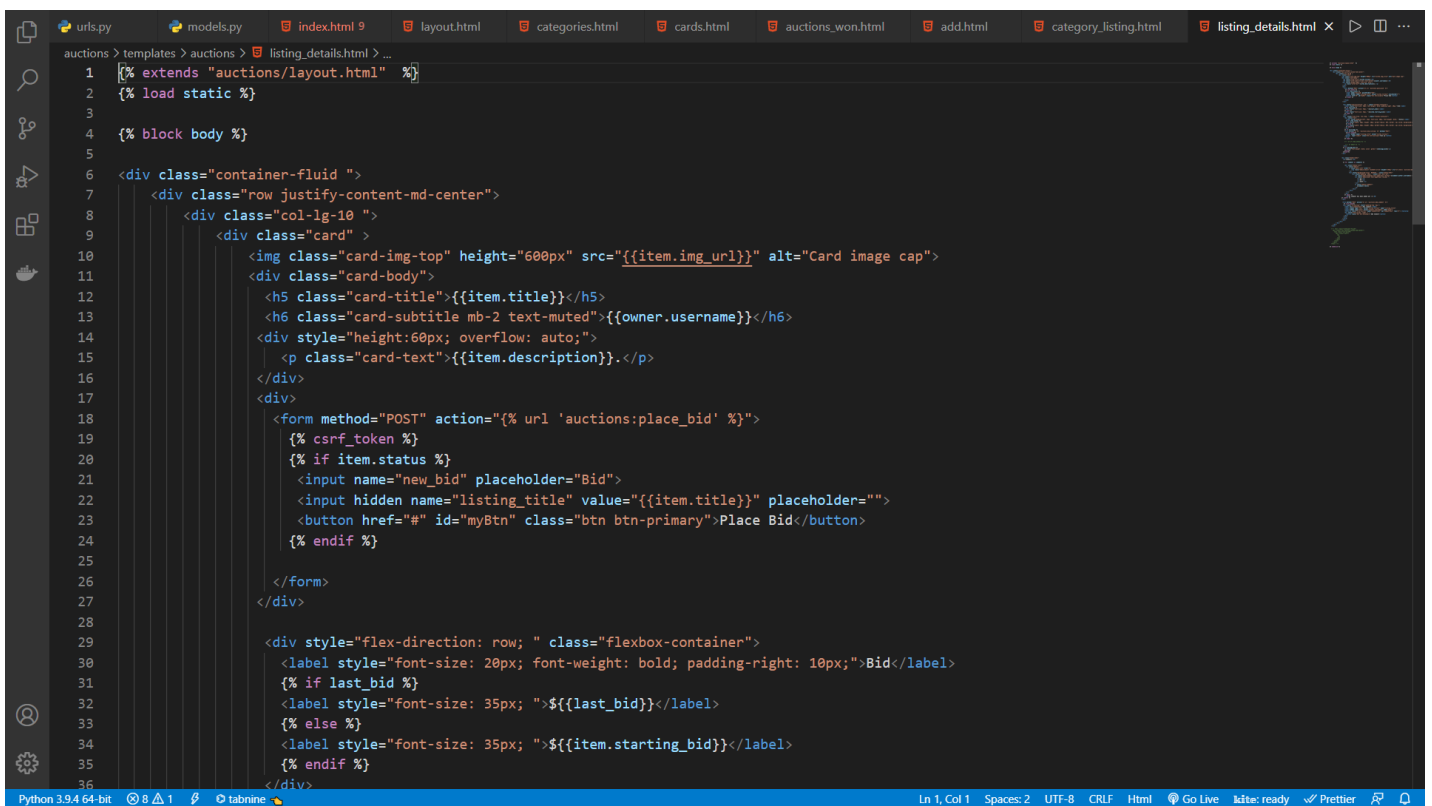
INDEX.HTML

The Index Page is the Home Page of the website.

It contains properties to Add Item for the auction, the Auction History and the Categories of the product available for the auction.

It also provides the link to navigate to the active listings.

Whenever the user adds a new product, the item is displayed on the Home Page.



```
1 {% extends "auctions/layout.html" %}
2 {% load static %}
3
4 {% block body %}
5
6 <div class="container-fluid">
7   <div class="row justify-content-md-center">
8     <div class="col-lg-10">
9       <div class="card">
10        
11        <div class="card-body">
12          <h5 class="card-title">{{item.title}}</h5>
13          <h6 class="card-subtitle mb-2 text-muted">{{owner.username}}</h6>
14          <div style="height:60px; overflow: auto;">
15            <p class="card-text">{{item.description}}</p>
16          </div>
17          <div>
18            <form method="POST" action="{% url 'auctions:place_bid' %}">
19              {% csrf_token %}
20              {% if item.status %}
21                <input name="new_bid" placeholder="Bid">
22                <input hidden name="listing_title" value="{{item.title}}" placeholder="">
23                <button href="#" id="myBtn" class="btn btn-primary">Place Bid</button>
24              {% endif %}
25            </form>
26          </div>
27
28          <div style="flex-direction: row; " class="flexbox-container">
29            <label style="font-size: 20px; font-weight: bold; padding-right: 10px;">Bid</label>
30            {% if last_bid %}
31              <label style="font-size: 35px; ">${{last_bid}}</label>
32            {% else %}
33              <label style="font-size: 35px; ">${{item.starting_bid}}</label>
34            {% endif %}
35          </div>
36        </div>
37      </div>
38    </div>
39  </div>
40</div>
```

LAYOUT.HTML

The Layout is mainly the navigation bar of the website.

It contains the information of the user.

Also, it contains Log In and Log Out option for the user.

This also contains Add Item, Watchlist, Auction History, and Category.

CODE

```
{% load static %}
```

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<title>{% block title %}Auctions{% endblock %}</title>
```

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
```

```
<link href="{% static 'auctions/styles.css' %}" rel="stylesheet">
```

```
<link href="{% static 'auctions/cards.css' %}" rel="stylesheet">
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
```

```
</head>
```

```
<style>
```

```
body{
```

```
background-color: rgb(173, 233, 230);
```

```
background-image : url("https://i1.wp.com/www.estidia.eu/wp-content/uploads/2018/04/Savin-NY-Website-
Background-Web.jpg");
```

```
background-repeat: no-repeat;
```

```
background-size: cover;
```

```
}
```

```
</style>
```

```
<body>
```

```
<h1>Auctions</h1>
```

```
<div>
```



```

{% if user.is_authenticated %}
    Signed in as <strong>{{ user.username }}</strong>.
{% else %}
    Not signed in.
{% endif %}
</div>
<ul class="nav">
    <li class="nav-item">
        <a class="nav-link" href="{% url 'auctions:index' %}">Active Listings</a>
    </li>
    {% if user.is_authenticated %}
        <li class="nav-item">
            <a class="nav-link" href="{% url 'auctions:logout' %}">Log Out</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{% url 'auctions:add' %}">Add Item</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{% url 'auctions:watchlist' %}">Watchlist</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{% url 'auctions:auctions_history' %}">Auctions History</a>
        </li>
    {% else %}
        <li class="nav-item">
            <a class="nav-link" href="{% url 'auctions:login' %}">Log In</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{% url 'auctions:register' %}">Register</a>
        </li>
    {% endif %}
    <li class="nav-item">
        <a class="nav-link" href="{% url 'auctions:category_list' %}">Categories</a>
    </li>
</ul>
<hr>
{% block body %}
{% endblock %}
</body>
</html>

```

CATEGORY_LISTING.HTML

The Category Listing page contains the information of the products that are active for the bidding process. Once the auction for the product is over, the product is automatically removed from this page. Also, whenever the user add a new product for bidding, that product is displayed on this page.

CODE

```
{% extends 'auctions/layout.html' %}
{% load static %}

{% block body %}
<h1>Active listings for {{ category }}</h1>

{% if items %}
<div class="container">
    <div class="row">
        {% for item in items_bids %}

            <div class="col-lg-4">
                <div class="card" style="width: 18rem;">
                    
                    <div class="card-body">
                        <h5 class="card-title">
                            <span style="display: block; overflow: hidden; white-space: nowrap; height: 23px; text-overflow:
ellipsis;">
                                {{ item.0.title }}
                            </span>
                        </h5>
                        <h6 class="card-subtitle mb-2 text-muted">{{ item.0.user.username }}</h6>
                        <p class="card-text">
                            <span id="item-title" style="display: block; overflow: hidden; white-space: nowrap; height: 23px; text-
overflow: ellipsis;">
                                {{ item.0.description }}
                            </span>
                        </p>

                    <div style="flex-direction: row; " class="flexbox-container">
```

```

        <label style="font-size: 20px; font-weight: bold; padding-right: 10px;">Bid</label>
        <label style="font-size: 35px; ">${ {item.1} }</label>
    </div>
    <div style="text-align: center;">
        <!-- <button onclick="showAlert(' { {item.title} }');" href="#" id="myBtn" class="btn btn-primary" data-
toggle="modal">Open</button> -->
        <form method="POST" action=" { % url 'auctions:listing_details' % } ">
            { % csrf_token % }
            <input hidden type="text" value=" { {item.0.title} }" name="listing_title">
            <button href="#" id="myBtn" class="btn btn-primary" type="submit">Open</button>
        </form>
    </div>

</div>
</div>
</div>
{ % endfor % }
</div>
</div>
{ % else % }
    <h4>There's no listing at the moment</h4>
{ % endif % }

{ % endblock % }

```

URLS.PY

This is the backend part of the website with python framework DJANGO.

This file contains the urls of the website.

Due to this, the user can navigate to the different page dynamically.

CODE

```
from django.urls import path
```

```
from . import views
```

```
app_name = "auctions"
```

```
urlpatterns = [  
    path("", views.index, name="index"),  
    path("login", views.login_view, name="login"),  
    path("logout", views.logout_view, name="logout"),  
    path("register", views.register, name="register"),  
    path("cards", views.cards_view, name="cards"),  
    path("add", views.add, name="add"),  
    path("add_comment", views.add_comment, name="add_comment"),  
    path("listing_details", views.listing_details_view, name="listing_details"),  
    path("watchlist", views.watchlist_view, name="watchlist"),  
    path("category_list", views.category_list, name="category_list"),  
    path("category_list/<str:category>", views.category_list_redirect, name="category_list_redirect"),  
    path("add_to_watchlist", views.add_to_watchlist, name="add_to_watchlist"),  
    path("place_bid", views.place_bid, name="place_bid"),  
    path("end_listing", views.end_listing, name="end_listing"),  
    path("auctions_history", views.auctions_history, name="auctions_history"),  
    path("delete_item_watchlist", views.delete_item_watchlist, name="delete_item_watchlist")  
]
```

MODELS.PY

This is the backend part of the website with python framework DJANGO.
This page contains the models created using python.
It contains the properties of the attributes.

CODE

```
from django.contrib.auth.models import AbstractUser
from django.db import models
```

```
class User(AbstractUser):
    def __str__(self):
        return f"{self.first_name}"
```

```
class Item(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="item_list")
    title = models.CharField(max_length=64)
    description = models.CharField(max_length=256, blank=True)
    img_url = models.CharField(max_length=256, blank=True)
    starting_bid = models.DecimalField(decimal_places=2, max_digits=8)
    category = models.CharField(max_length=24, default='No Category')
    status = models.BooleanField(default=True)
    def __str__(self):
        return f"{self.title}, {self.description}"
```

```
class ItemCategory(models.Model):
    name = models.CharField(max_length=24, blank=False)
    item = models.ForeignKey(Item, blank=True, on_delete=models.CASCADE,
related_name="category_list")
    def __str__(self):
        return f"{self.name}"
```

```

class Bid(models.Model):
    id = models.AutoField(primary_key=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="user_bids")
    amount = models.DecimalField(decimal_places=2, max_digits=8)
    items = models.ForeignKey(Item, on_delete=models.CASCADE, related_name="user_bid_items",
blank=True)
    def __str__(self):
        return f"{self.amount}"

class ItemComment(models.Model):
    text = models.TextField(max_length=512, blank=False)
    item = models.ForeignKey(Item, on_delete=models.CASCADE, related_name="comments_list")
    author = models.ForeignKey(User, on_delete=models.CASCADE)

class Watchlist(models.Model):
    items = models.ForeignKey(Item, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='watchlist_list')

    def __str__(self):
        return f"{self.items}, {self.user}"

class Category(models.Model):
    name = models.CharField(max_length=24, blank=False)
    items = models.ManyToManyField(Item, blank=True, related_name="categories")

    def __str__(self):
        return f"{self.name}"

class AuctionHistory(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="history_user")
    items = models.ForeignKey(Item, on_delete=models.CASCADE, related_name="items_auction_history")

    def __str__(self):
        return f"{self.user.first_name}, {self.items}"

```

CHAPTER 7

ADVANTAGES:

- **Planned approach towards working:** The working in the organization will be well organized. The data will be stored properly in data stores which will help in retrieval of information as well as its storage.
- **Accuracy:** The level of accuracy in the proposed system will be higher. All the operations would be done correctly and it would ensure that the information is accurate
- **Reliability:** The reliability of the proposed system will be high due the above mentioned reasons. The reason for the increased reliability of the system is that now there would be proper storage of information.
- **No Redundancy:** Utmost care would be taken in the proposed system so that no information is repeated anywhere, in storage or otherwise. This would assure economic use of storage space and consistency in the data stored.
- **Easy to Operate:** The system will be easy to operate by the users of many ages as the interface will be user friendly

CHAPTER 8

CONCLUSION

Much of the existing auction software literature is dated and is of limited usefulness to researchers. Furthermore, most existing proposals do not adhere to sound UML standards. We provided a simple and elegant design for an online auction system based on UML. We presented an analysis and design for the auction system illustrating key system components using UML diagrams. uAuction is being used to facilitate our research into real-time shill bidding detection. uAuction provides us the ability to conduct various types of testing using a combination of human users, simulated auctions and/or synthetic data. Future work involves further development of the auction system so that it has more functionality and can potentially undertake different types of auctions. We also intend on implementing our real-time shill bidding detection algorithm as part of the fraud detection subsystem. This will then allow us to undertake some of the aforementioned testing approaches in order to help refine our proposed fraud detection techniques.

FUTURE SCOPE:



In this digital era, companies prefer to fetch business through online platforms. Therefore, they tend to approach their existing and new clients via the internet source.

Now, the point is who will approach the customers to provide the services in an enterprise? Yes! It is the responsibility of an online bidder to search the clients who are interested in availing the services offered by you.

Moreover, while working as an online bidder, you will get an opportunity to upgrade your knowledge about recent technologies and platforms which are used to attract customers and interact with them easily. Furthermore, you will be paid good salary packages which can increase your bank amount.

Once you understand the responsibilities of the bidders, such as making proposals, replying to the leads, strengthen the relationship with the prospects, etc., you will get a wide range of opportunities to expand your bandwidth in the corporate sector. For this you can always utilize online job bidding sites to work more professionally.

Several insights are gained from this study. In particular, some research areas need more investigation, especially bidder behavior in B2B reverse auctions, and online auction design that shows how technology can be incorporated into auction design to affect the price paid by bidders. Also, more seller-level analysis, especially in the context of marketing research, could be performed to show how seller-level effects, such as promotion, market presence, etc., can affect the price paid in online auctions. This study is limited by the vast amount of new working papers, conference proceedings, and journal articles that are being developed or have been published. Though an attempt was made to find and report on these articles, some relevant research is bound to be omitted from this study, despite efforts to be inclusive. Rather, this study should be looked at as an unbiased sample of current research so that researchers can glean levels of interest and areas of future research. Thus, this research should not be considered to be a substitute for a literature review, but instead can be used as a springboard to delve into new research topics in online auctions.

REFERENCE:

- [1] Simone Pigolotti; Sebastian Bernhardsson; Jeppe Juul; Gorm Galster; Pierpaolo Vivo (2012). ["Equilibrium strategy and population-size effects in lowest unique bid auctions"](#). Retrieved 25 October 2012.
- [2] ["Stolen-Property Purchases Leave Ebay Buyers Burned"](#). *San Jose Mercury News*. 11 June 2002. Archived from the original on 25 January 2013.
- [3] ["Is Swoopo Nothing More Than a Well-Designed Gimmick?"](#). Technogizer.com. 17 September 2008. Retrieved 3 January 2013.
- [4] ["Auction of Collectibles on the Internet"](#). *The New York Times*. 23 May 1995. Retrieved 18 September 2019.
- [5] Gimein, Mark (12 July 2009). ["The Big Money: The Pennies Add Up at Swoopo.com"](#). *The Washington Post*. Retrieved 26 April 2010.
- [6] Ockenfels, Axel; Reiley, David; Sadrieh, Abdolkarim (December 2006). ["Online Auctions"](#).
- [7] Amit, Alon. ["What Is a Vickrey Auction, and How Does It Work?"](#). *Slate*. Retrieved 27 January 2017.