# Final Report
# (e-Auction Management System)

**Course Code:** CS254 **Course Title:** DBMS LAB

**Semester:** B. Tech $4^{th}$ Sem **Section:** S1 **Academic Year:** 2020-21

**Course Instructor:** Dr. Annappa B and Mr. Sharath Yaji

**Team Members:**

**1.** Nishant N. Nayak, 191CS141, 8867907525, nishant.191cs141@nitk.edu.in

**2.** Aditya Santhosh, 191CS105, 8129926927, aditya.191cs105@nitk.edu.in

**3.** Reshma Tresa Antony, 191CS149, 9400000804, reshmatantony.191cs149@nitk.edu.in

## 1 Abstract

This application is an eBay-like e-commerce auction website which will enable users to post auction listings, place bids on listings, comment on those listings, and add listings to a "watch-list".

**Key Features of the project:**

1. Create Listing
2. Active Listings Page
3. Listing Page
4. Watch-list
5. Categories
6. Django Admin Interface

**Softwares used in the project:**

- Frontend: HTML, CSS, JS
- Backend: Django, MySQL

# 2 Introduction

Users will be able to visit a page to create a new listing through the **Create Listing page**. They will be able to specify a title for the listing, a text-based description, and what the starting bid ought to be. Users will also optionally be able to give a URL for an image for the listing and/or a category (e.g. Fashion, Toys, Electronics, Home, etc.).

The web application lets users view all of the presently active auction listings. For every active listing, the **Active Listings page** displays the title, description, current price, and photo (if one exists for the listing).

Clicking on a listing takes users to a page specific to that listing. On that **Listing page**, users will be able to view all details concerning the listing, including the current price for the listing.

Users who are signed in will be able to visit a **Watch-list page**, which displays all of the listings that a user has added to their watch-list. Clicking on any of those listings takes the user to that listing's page.

Users will be able to visit a page that displays a list of all listing **categories**. Clicking on the name of any category takes the user to a page that displays all of the active listings in that category.

Via the **Django admin interface**, a site administrator will be able to view, add, edit, and delete any listings, comments, and bids made on the site.

# 3 ER Diagram

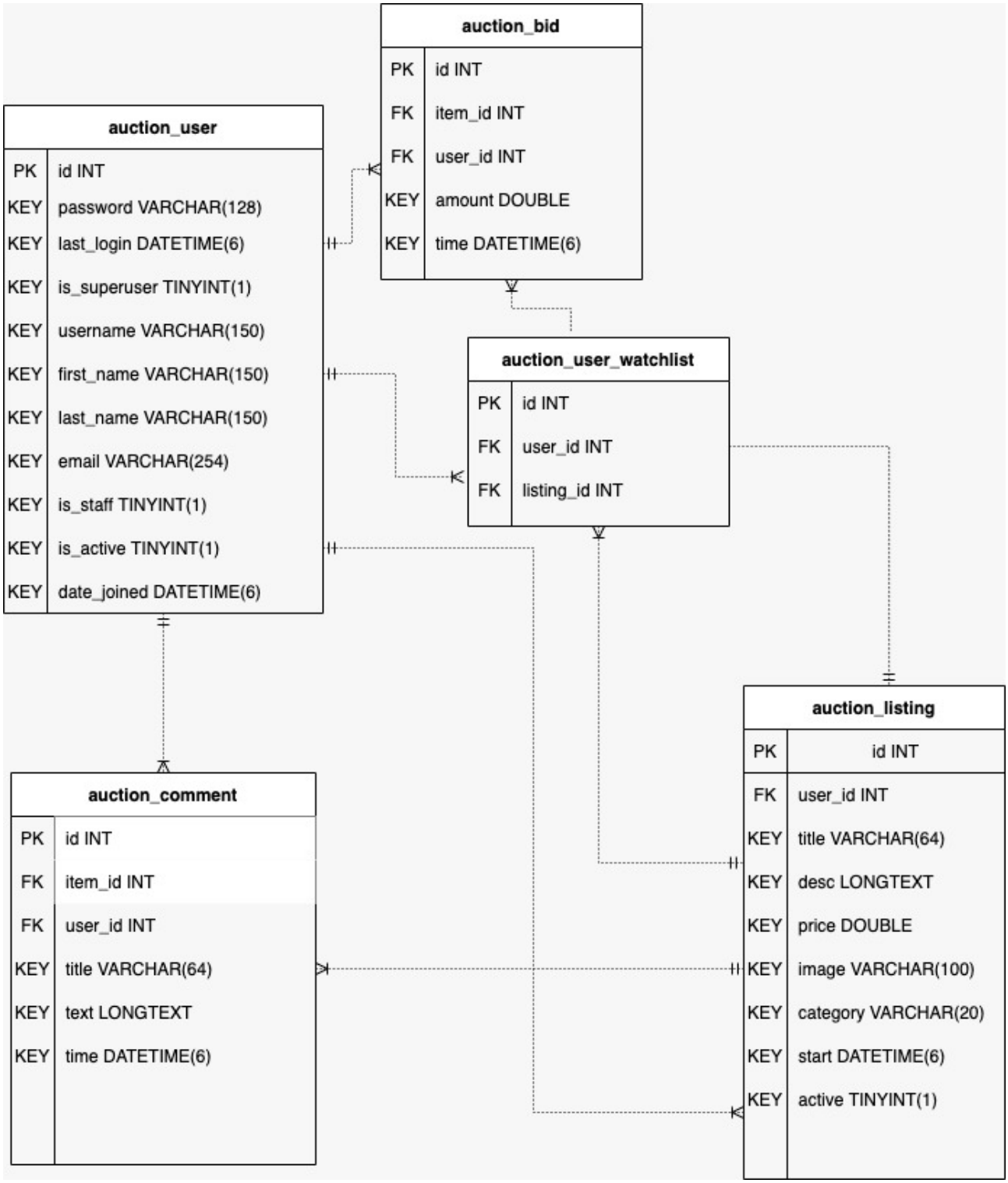


Figure 1: Relation : ER diagram of the database

# 4 Source Code

**Backend :**

cs254-project/auction/views.py :

```
from django.contrib.auth import authenticate, login, logout
from django.shortcuts import render
from django.db import IntegrityError
from django.http import HttpResponseRedirect
from django.urls import reverse
from django.contrib.auth.decorators import login_required

from .models import User, Listing, Bid, Comment
from .forms import ListingForm, BidForm, CommentForm
from datetime import datetime, timezone

# Default view of the website, this is the first page that the user will see
def index(request):
    # Fetch all active listings to display on the main page
    # SQL : SELECT * FROM listings WHERE acitve=True;
    listings = Listing.objects.filter(active=True)
    # Render the webpage with all active listings as context
    return render ( request, 'auction/index.html', { 'listings' : listings } )

# Login view of the web application
def login_view(request):
    # If user tried to access this page by submitting the login form
    if request.method == 'POST':
        # Retrieve the username and password from the POST parameters
        username = request.POST['username']
        password = request.POST['password']
        # Authenticate the user by comparing the username and password to the data available
in the database
```

```
        # SQL : SELECT * FROM user WHERE username=username AND password={hash
(password)}
        user = authenticate(request, username=username, password=password)
        # If the user exists and the password is valid, login the user and redirect them to the
index page
        if user is not None:
            login(request, user)
            return HttpResponseRedirect (reverse(index))
        # If the authenticate function did not return a user, then the input data is invalid
        # Render the login page again with error message
        else:
            return render ( request, 'auction/login.html', { 'message' : 'Invalid Username or Pass-
word.', } )
    # If the user tried to access this page by clicking the login link
        return render(request, 'auction/login.html')

# Logout view of the web application
def logout_view(request):
    # Logout the user, and redirect them back to the index page
    logout(request)
    return HttpResponseRedirect(reverse('index'))

# Register view of the web application
def register(request):
    # If the user tried to access the page by submitting the register form
    if request.method == "POST":
        # Retrieve the data from the POST parameters
        username = request.POST["username"]
        email = request.POST["email"]
        first_name = request.POST["fname"]
        last_name = request.POST["lname"]
        # Ensure password matches confirmation
        # TODO: Move this confirmation to the front-end as well using JS
        password = request.POST["password"]
```

```
        if password != request.POST["confirmation"]:
            return render (request, "auction/register.html", {"message" : "Passwords must match."})
        # Attempt to create new user
        try:
            # SQL : INSERT INTO user VALUES ({username}, {email}, {password}, {first_name},
{last_name}); COMMIT;
            user = User.objects.create_user(username, email, password)
            user.first_name = first_name
            user.last_name = last_name
            user.save()
        # If username already exists, Django will throw an integrity error since username must
be unique
        except IntegrityError:
            return render ( request, "auction/register.html", { "message" : "Username already
taken." } )
        # The user is created and saved to the database, so login the user and redirect them to
the index page
        login(request, user)
        return HttpResponseRedirect(reverse("index"))
    # If the user tried to access the page by clicking the register link
```

.....For the rest of views.py please refer :

https://github.com/nishant-nayak/cs254-project/blob/master/auction/views.py

cs254-project/auction/models.py :

https://github.com/nishant-nayak/cs254-project/blob/master/auction/models.py

cs254-project/auction/forms.py :

https://github.com/nishant-nayak/cs254-project/blob/master/auction/forms.py

cs254-project/auction/urls.py :

https://github.com/nishant-nayak/cs254-project/blob/master/auction/urls.py

cs254-project/auction/admin.py :

https://github.com/nishant-nayak/cs254-project/blob/master/auction/admin.py

cs254-project/cs254_project/settings.py :

https://github.com/nishant-nayak/cs254-project/blob/master/cs254_project/settings.py

cs254-project/cs254_project/urls.py :

https://github.com/nishant-nayak/cs254-project/blob/master/cs254_project/urls.py

**Frontend :**

cs254-project/auction/templates/auction/index.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/index.html
cs254-project/auction/templates/auction/layout.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/layout.html
cs254-project/auction/templates/auction/listing.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/listing.html
cs254-project/auction/templates/auction/userpage.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/userpage.html
cs254-project/auction/templates/auction/register.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/register.html
cs254-project/auction/templates/auction/create.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/create.html
cs254-project/auction/templates/auction/login.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/login.html
cs254-project/auction/templates/auction/category.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/category.html
cs254-project/auction/templates/auction/watchlist.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/watchlist.html
cs254-project/auction/templates/auction/404.html :
https://github.com/nishant-nayak/cs254-project/blob/master/auction/templates/auction/404.html

# 5 Results

```
1    SELECT * FROM auction_user;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| id | password | last_login | is_superuser | username | first_name | last_name | email | is_staff | is_active | date_joined |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | pbkdf2_sha256$216000$xyX6OvxxytcQ$9lGFA... | 2021-03-10 08:48:23.902724 | 1 | admin | | | admin@example.com | 1 | 1 | 2021-02-10 09:52:48.691791 |
| 2 | pbkdf2_sha256$216000$LmqfBtJdGOjB$uHj294... | 2021-03-22 19:37:44.290002 | 0 | foo | Foo | Bar | foo@example.com | 0 | 1 | 2021-02-10 10:01:26.035065 |
| 4 | pbkdf2_sha256$216000$f3zGMRe09PM$k/zfV9... | 2021-03-13 13:58:42.893183 | 0 | bar | Bar | Foo | bar@example.com | 0 | 1 | 2021-03-09 17:15:30.864215 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Figure 2: Relation : auction_user

```
1    SELECT * FROM auction_listing;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| id | title | desc | price | image | category | start | active | user_id |
|---|---|---|---|---|---|---|---|---|
| 1 | Test Item | An item that is used for testing purposes | 1200 | https://pbs.twimg.com/...s3.amazonaws.com/ima... | electronics | 2021-02-10 09:54:12.165219 | 1 | 1 |
| 2 | Test 2 | Another test product | 300000 | | electronics | 2021-02-23 03:34:58.226511 | 0 | 2 |
| 3 | Test 3 | Another test item to be used | 45678 | https://image.shutterstock.com/image-vector/u... | home | 2021-03-06 21:04:16.321827 | 1 | 2 |
| 4 | Test 3 | Test | 100 | | pets | 2021-03-10 08:47:44.353482 | 1 | 2 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Figure 3: Relation : auction_listing

```
1 •  SELECT * FROM auction_bid;
```

Result Grid | Filter Rows: | Edit:

| id | amount | time | item_id | user_id |
|---|---|---|---|---|
| 1 | 3001 | 2021-03-13 13:58:51.414183 | 2 | 4 |
| 2 | 300000 | 2021-03-13 13:58:57.977654 | 2 | 4 |
| 3 | 1200 | 2021-03-22 19:41:18.149219 | 1 | 2 |
| NULL | NULL | NULL | NULL | NULL |

Figure 4: Relation : auction_bid

```
1 •  SELECT * FROM auction_comment;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

| id | title | text | time | item_id | user_id |
|---|---|---|---|---|---|
| 1 | Tip | This product looks great! | 2021-03-13 14:10:54.245718 | 1 | 2 |
| NULL | NULL | NULL | NULL | NULL | NULL |

Figure 5: Relation : auction_comment

```
1  SELECT * FROM auction_user_watchlist;
```

Result Grid | Filter Rows: | Edit:

| id | user_id | listing_id |
|---|---|---|
| 2 | 2 | 1 |
| NULL | NULL | NULL |

Figure 6: Relation : auction_user_watchlist

# 6 References:


1. https://medium.com/@isubhamsr/how-i-made-an-e-commerce-website-with-django-e17f0a65cbd6

2. https://www.djangoproject.com/start/

3. https://www.w3schools.com/


****** END ******