

```

1
2 from django.contrib.auth import authenticate, login, logout
3 from django.db import IntegrityError
4 from django.http import HttpResponseRedirect, HttpResponseRedirect
5 from django.shortcuts import render
6 from django.urls import reverse
7
8 from .models import User, AuctionListing, Bid, Comment, Category, Info
9
10
11 def index(request):
12     auction_listings = AuctionListing.objects.exclude(listing_open=False)
13
14     for listing in auction_listings:
15         try:
16             Bid.objects.get(bid_identifier=listing.id, watchBidder=request.user)
17             listing.display_button = False
18         except:
19             listing.display_button = True
20
21     return render(request, "auctions/index.html", {
22         "listings": auction_listings,
23         "type": "Active Listings"
24     })
25
26
27 def login_view(request):
28     if request.method == "POST":
29
30         # Attempt to sign user in
31         username = request.POST["username"]
32         password = request.POST["password"]
33         user = authenticate(request, username=username, password=password)
34
35         # Check if authentication successful
36         if user is not None:
37             login(request, user)
38             return HttpResponseRedirect(reverse("index"))
39         else:
40             return render(request, "auctions/login.html", {
41                 "message": "Invalid username and/or password."
42             })
43     else:
44         return render(request, "auctions/login.html")
45
46
47 def logout_view(request):
48     logout(request)
49     return HttpResponseRedirect(reverse("index"))
50
51
52 def register(request):
53     if request.method == "POST":
54         username = request.POST["username"]
55         email = request.POST["email"]
56
57         # Ensure password matches confirmation
58         password = request.POST["password"]
59         confirmation = request.POST["confirmation"]
60         if password != confirmation:
61             return render(request, "auctions/register.html", {
62                 "message": "Passwords must match."
63             })
64
65         # Attempt to create new user
66         try:
67             user = User.objects.create_user(username, email, password)

```

```

68         user.save()
69     except IntegrityError:
70         return render(request, "auctions/register.html", {
71             "message": "Username already taken."
72         })
73     login(request, user)
74     return HttpResponseRedirect(reverse("index"))
75 else:
76     return render(request, "auctions/register.html")
77
78 def create(request):
79     if request.method == "POST":
80         title = request.POST['listing_title']
81         description = request.POST['listing_description']
82         price = request.POST['start_bid']
83         photo = request.POST['listing_image']
84         creator = request.user
85
86         if request.POST['listing_category']:
87             listing_category =
88                 Category.objects.get(pk=int(request.POST['listing_category']))
89         else:
90             listing_category = Category.objects.get(pk=1)
91
92         new_listing = AuctionListing(
93             title=title,
94             description=description,
95             start_price=price,
96             current_bid=price,
97             photo=photo,
98             creator=creator,
99             listing_category=listing_category)
100
101         new_listing.save()
102
103         return HttpResponseRedirect(reverse("index"))
104     return render(request, "auctions/create.html", {
105         "categories": Category.objects.exclude(pk=1)
106     })
107
108 def listing_page(request, listing_id):
109     listing = AuctionListing.objects.get(pk=listing_id)
110     min_bid = listing.current_bid + 5
111     comments = listing.listing_comment.all()
112
113     if listing.listing_open == False:
114         close = "CLOSED"
115         try:
116             info = Info.objects.get(won_listing=listing)
117         except:
118             info = ""
119     else:
120         close = ""
121         info = ""
122
123     try:
124         condition = listing.bids.get(bidder__isnull=False)
125     except:
126         condition = ""
127
128     if not condition:
129         close_button = False
130     else:
131         close_button = True
132
133     try:
134         Bid.objects.get(bid_identifier=listing_id)
135         button = False

```

```

134     except:
135         button = True
136
137     context = {
138         "listing": listing,
139         "status": listing.listing_open,
140         "min_bid": min_bid,
141         "info": info,
142         "closed": close,
143         "button": button,
144         "cButton": close_button,
145         "comments": comments
146     }
147     return render(request, "auctions/listing.html", context)
148
149 def watchlist(request):
150     if request.method == "POST":
151         wList = request.POST['wList']
152         page = request.POST['page']
153         watch_list = AuctionListing.objects.get(pk=wList)
154         Bidder = request.user
155
156         new_bid = Bid(
157             bid_item=watch_list,
158             bid_price=watch_list.current_bid,
159             watchBidder=Bidder,
160             bid_identifier=wList
161         )
162
163         new_bid.save()
164
165         if page == 1:
166             return HttpResponseRedirect(reverse("listing", kwargs={"listing_id":wList}))
167         else:
168             return HttpResponseRedirect(reverse("index"))
169
170
171     else:
172         listings = []
173         bid_listings = request.user.listing_watchBidder.all()
174         for bid in bid_listings:
175             listings.append(AuctionListing.objects.get(pk=bid.bid_identifier))
176
177         for listing in listings:
178             try:
179                 Bid.objects.get(bid_identifier=listing.id, watchBidder=request.user)
180                 listing.display_button = False
181             except:
182                 listing.display_button = True
183
184         return render(request, "auctions/index.html", {
185             "listings": listings,
186             "type": "my Watchlist"
187         })
188
189
190 def bidlist(request):
191     bid_listings = request.user.listing_bidder.all()
192     if request.method == "POST":
193         bItem = request.POST['bItem']
194         bid_list = AuctionListing.objects.get(pk=bItem)
195         bid_list.current_bid = request.POST["bid_price"]
196         bid_list.save()
197         Bidder = request.user
198
199         try:
200             update_bid = Bid.objects.get(bid_item=bid_list, watchBidder=Bidder)

```

```

201         update_bid.bidder = Bidder
202         update_bid.bid_price = bid_list.current_bid
203         update_bid.save()
204     except:
205         new_bid = Bid(
206             bid_item=bid_list,
207             bid_price=bid_list.current_bid,
208             bidder=Bidder,
209             watchBidder=Bidder,
210             bid_identifier=bItem
211         )
212         new_bid.save()
213
214
215         return HttpResponseRedirect(reverse("listing", kwargs={"listing_id":bItem}))
216
217     else:
218         listings = []
219
220         for bid in bid_listings:
221             listings.append(AuctionListing.objects.get(pk=bid.bid_identifier))
222
223         for listing in listings:
224             try:
225                 Bid.objects.get(bid_identifier=listing.id, watchBidder=request.user)
226                 listing.display_button = False
227             except:
228                 listing.display_button = True
229
230         return render(request, "auctions/index.html",{
231             "listings": listings,
232             "type": "my Bids"
233         })
234
235 def close_listing(request):
236     if request.method == "POST":
237         bItem = request.POST['lastBid']
238         listing = AuctionListing.objects.get(pk=bItem)
239         listing.listing_open = False
240         listing.save()
241
242         last_bid = Bid.objects.get(bid_item=listing, bid_price=listing.current_bid)
243         winner = last_bid.bidder
244
245         a_info = User.objects.get(username=listing.creator)
246         w_info = User.objects.get(username=winner)
247
248         closing_info = Info(won_listing=listing,
249                             author_username=listing.creator,author_email = a_info.email, winner_username =
250                             last_bid.bidder,winner_email = w_info.email)
251         closing_info.save()
252
253         return HttpResponseRedirect(reverse("listing", kwargs={"listing_id":bItem}))
254
255 def my_listing(request):
256     listings = request.user.listing_creator.all()
257     return render(request, "auctions/index.html",{
258         "listings": listings,
259         "type": "my Listings"
260     })
261
262 def comments(request):
263     if request.method == "POST":
264         comment = request.POST['comment']
265         listing_id = request.POST['listing_id']
266         item = AuctionListing.objects.get(pk=listing_id)

```

```
266         new_comment = Comment(comment=comment,commentor=request.user, comment_item=item)
267         new_comment.save()
268
269         return HttpResponseRedirect(reverse("listing", kwargs={"listing_id":listing_id}))
270
271 def category(request):
272     if request.method == "POST":
273         listing_category = request.POST['category']
274         category = Category.objects.get(list_category=listing_category)
275
276         listings = category.item_category.exclude(listing_open=False)
277         return render(request, "auctions/index.html",{
278             "listings": listings,
279             "type": f"Listings on: {category}"
280         })
281
282     categories = Category.objects.all()
283     return render(request, "auctions/category.html", {
284         "categories": categories
285     })
286
```