

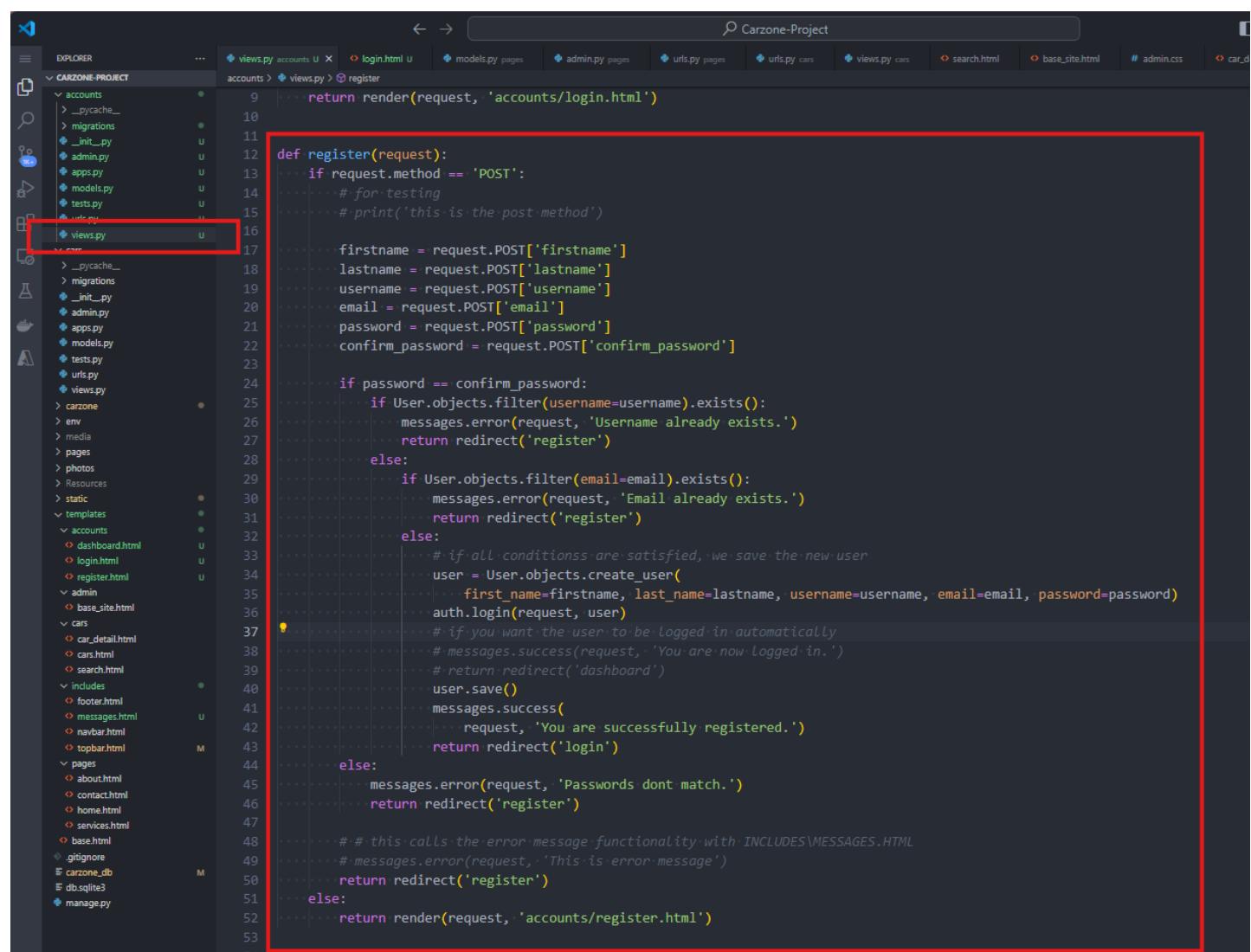
Topic: Car Listing: User Registration, Login, Logout, Dynamic Page Title Part 13

Speaker: Udemy Instructor Rathan Kumar / Notebook: Django Project: Car Listing



This is the continuation of the REGISTER FUNCTIONALITY from Part 12.

1. Go to ACCOUNTS\VIEWS.PY and update



```
def register(request):
    if request.method == 'POST':
        # for testing
        # print('this is the post method')

        firstname = request.POST['firstname']
        lastname = request.POST['lastname']
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        confirm_password = request.POST['confirm_password']

        if password == confirm_password:
            if User.objects.filter(username=username).exists():
                messages.error(request, 'Username already exists.')
                return redirect('register')
            else:
                if User.objects.filter(email=email).exists():
                    messages.error(request, 'Email already exists.')
                    return redirect('register')
                else:
                    # if all conditions are satisfied, we save the new user
                    user = User.objects.create_user(
                        first_name=firstname, last_name=lastname, username=username, email=email, password=password)
                    auth.login(request, user)
                    # if you want the user to be logged in automatically
                    # messages.success(request, 'You are now Logged in.')
                    # return redirect('dashboard')
                    user.save()
                    messages.success(
                        request, 'You are successfully registered.')
                    return redirect('login')
                else:
                    messages.error(request, 'Passwords dont match.')
                    return redirect('register')
            else:
                # this calls the error message functionality with INCLUDES\MESSAGES.HTML
                # messages.error(request, 'This is error message')
                return redirect('register')
        else:
            return render(request, 'accounts/register.html')
```

2. Update your REGISTER.HTML

```

<!-- Contact section start -->
<div class="contact_section">
<div class="container">
<div class="row">
<div class="col-lg-12 col-md-12">
<div class="form-section">
<div class="logo-2">
<a href="{% url 'home' %}">

</a>
</div>
<h3>Create an account</h3>
<!-- This displays our customized error message-->
<% include 'includes/messages.html' %>
<form action="{% url 'register' %}" method="POST">
<!-- this is for security reason that we add the CSRF_TOKEN -->
<% csrf_token %>
<div class="form-group form-box">
<input type="text" name="firstname" class="input-text" placeholder="First Name" required>
<i class="fa fa-user"></i>
</div>
<div class="form-group form-box">
<input type="text" name="lastname" class="input-text" placeholder="Last Name" required>
<i class="fa fa-user"></i>
</div>
<div class="form-group form-box">
<input type="text" name="username" class="input-text" placeholder="Username" required>
<i class="fa fa-user"></i>
</div>
<div class="form-group form-box">
<input type="email" name="email" class="input-text" placeholder="Email Address" required>
<i class="flaticon-mail"></i>
</div>
<div class="form-group form-box">
<input type="password" name="password" class="input-text" placeholder="Password" required>
<i class="flaticon-lock"></i>
</div>
<div class="form-group form-box">
<input type="password" name="confirm_password" class="input-text" placeholder="Confirm Password" required>
<i class="flaticon-lock"></i>
</div>
<div class="form-group mb-0 clearfix">
<button type="submit" class="btn-md btn-theme float-left">Register</button>
</div>

```

3. You can test it. We updated our ViEWS.PY if we want the user to be logged in automatically or not.

4. Update the LOGIN.HTML as

FROM:

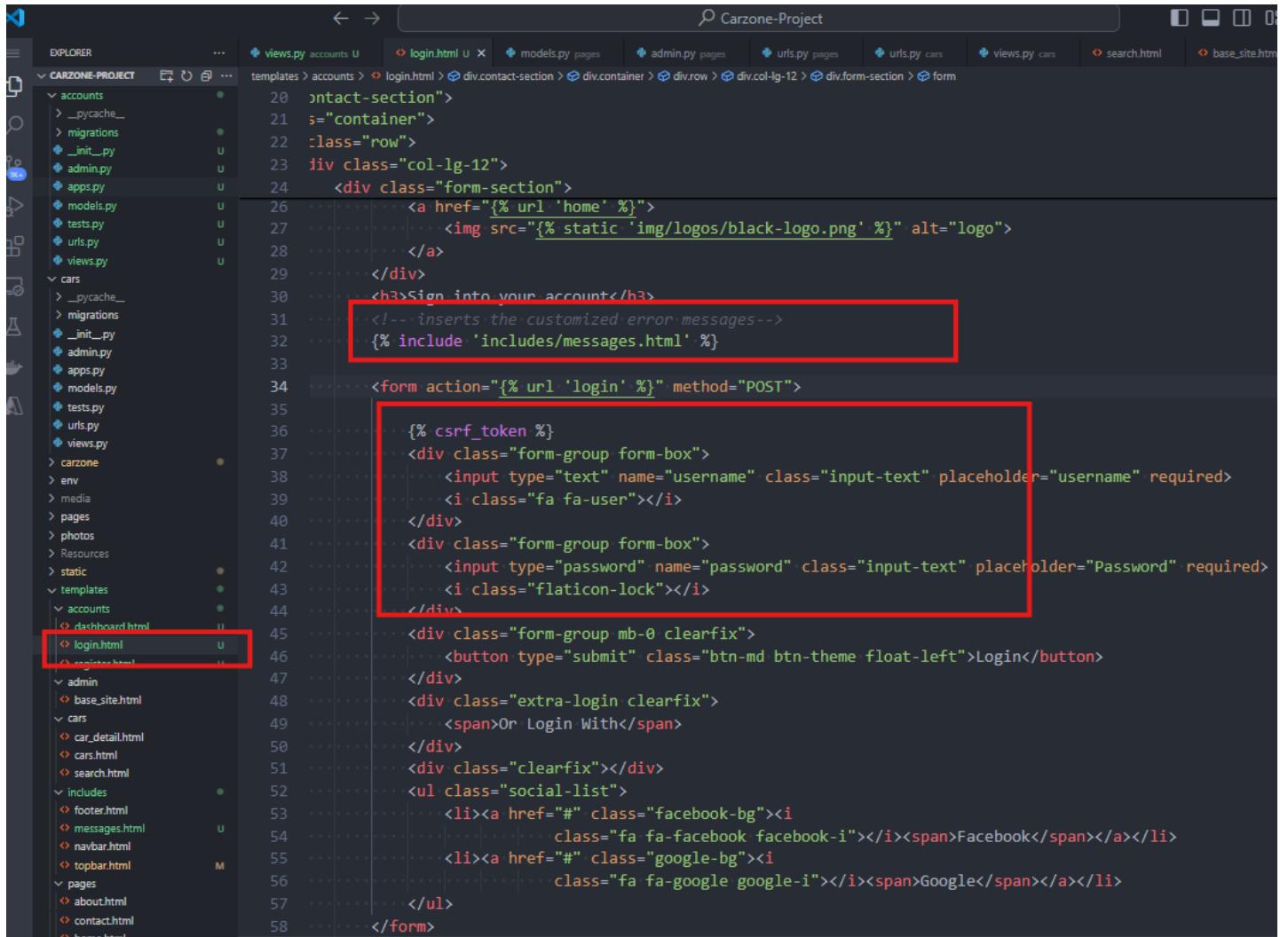
```

<h3>Sign into your account</h3>
<form action="{% url 'login' %}" method="POST">

    {% csrf_token %}
    <div class="form-group form-box">
        <input type="email" name="email" class="input-text" placeholder="Email Address" required>
        <i class="flaticon-mail"></i>
    </div>
    <div class="form-group form-box">
        <input type="password" name="password" class="input-text" placeholder="Password" required>
        <i class="flaticon-lock"></i>
    </div>
    <div class="form-group mb-0 clearfix">
        <button type="submit" class="btn-md btn-theme float-left">Login</button>
    </div>
    <div class="extra-login clearfix">
        <span>Or Login With</span>
    </div>
    <div class="clearfix"></div>
    <ul class="social-list">
        <li><a href="#" class="facebook-bg"><i class="fa fa-facebook facebook-i"></i><span>Facebook</span></a></li>
        <li><a href="#" class="google-bg"><i class="fa fa-google google-i"></i><span>Google</span></a></li>
    </ul>
</form>

```

TO:



The screenshot shows the VS Code interface with the 'Carzone-Project' workspace. The 'EXPLORER' sidebar on the left lists the project structure, including 'CARZONE-PROJECT' (accounts, cars, carzone, env, media, pages, photos, Resources, static, templates), 'accounts' (accounts, migrations, __init__.py, admin.py, apps.py, models.py, tests.py, urls.py, views.py), and 'templates' (accounts, dashboard.html, login.html, response.html). The 'login.html' file is currently selected and open in the main editor area. The code in the editor is as follows:

```

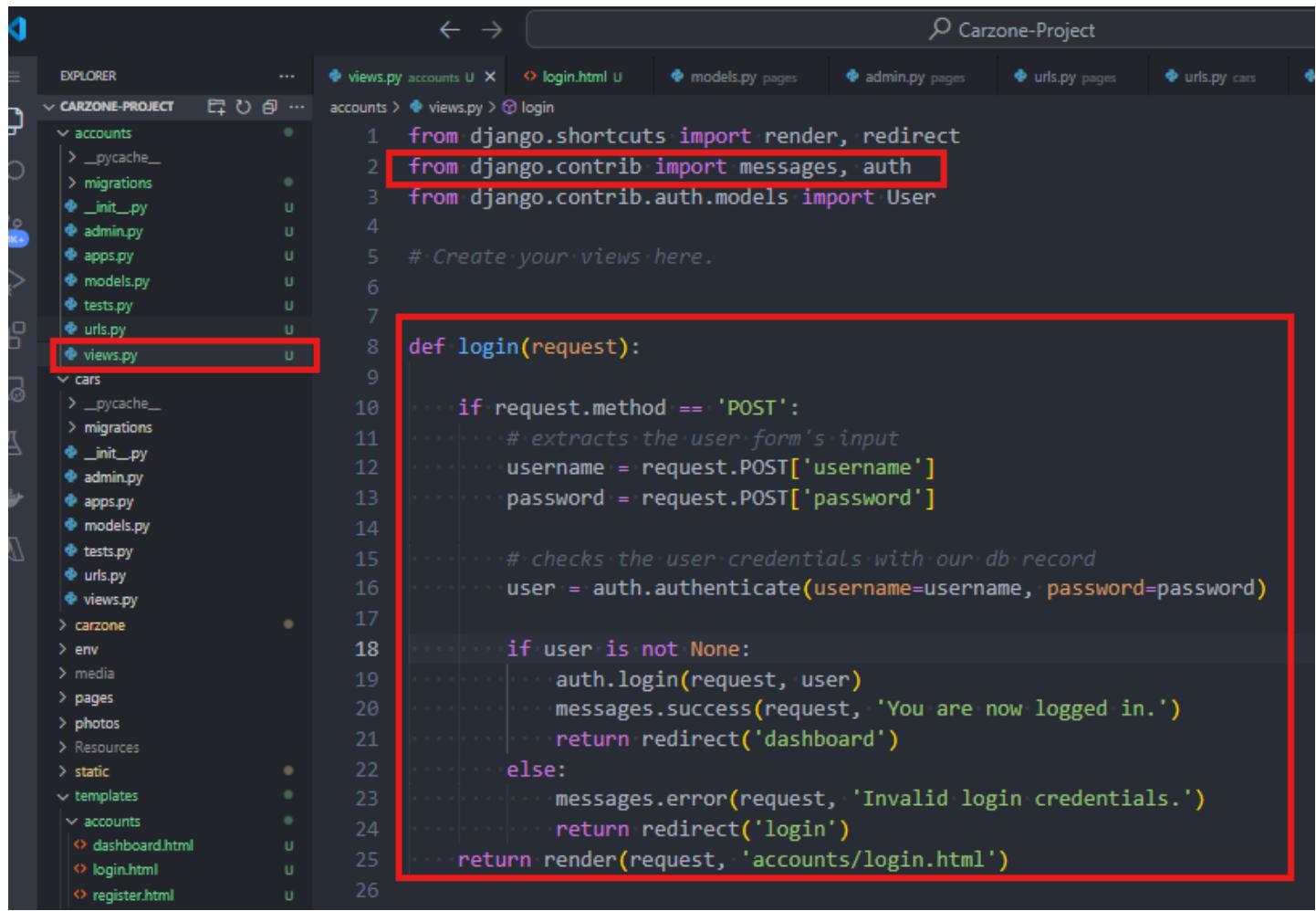
<h3>Sign into your account</h3>
<!-- inserts the customized error messages-->
{% include 'includes/messages.html' %}

<form action="{% url 'login' %}" method="POST">

    {% csrf_token %}
    <div class="form-group form-box">
        <input type="text" name="username" class="input-text" placeholder="username" required>
        <i class="fa fa-user"></i>
    </div>
    <div class="form-group form-box">
        <input type="password" name="password" class="input-text" placeholder="Password" required>
        <i class="flaticon-lock"></i>
    </div>
    <div class="form-group mb-0 clearfix">
        <button type="submit" class="btn-md btn-theme float-left">Login</button>
    </div>
    <div class="extra-login clearfix">
        <span>Or Login With</span>
    </div>
    <div class="clearfix"></div>
    <ul class="social-list">
        <li><a href="#" class="facebook-bg"><i class="fa fa-facebook facebook-i"></i><span>Facebook</span></a></li>
        <li><a href="#" class="google-bg"><i class="fa fa-google google-i"></i><span>Google</span></a></li>
    </ul>
</form>

```

5. Now, for the LOGIN function, update our VIEWS.PY as



Carzone-Project

EXPLORER

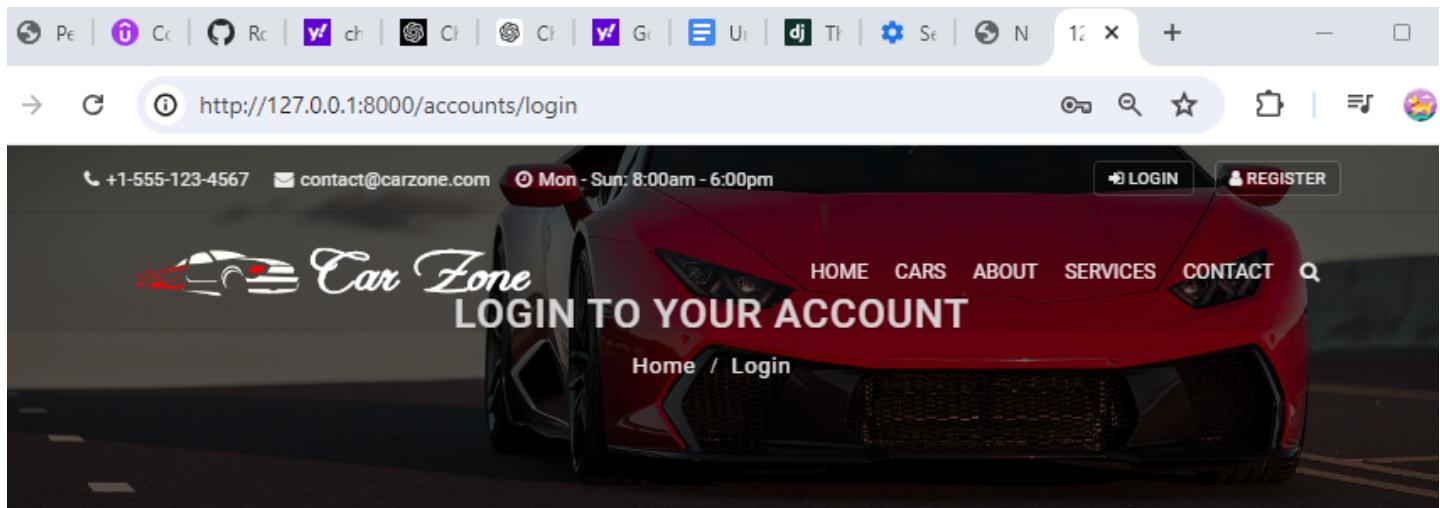
CARZONE-PROJECT

- accounts
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
- cars
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
- carzone
- env
- media
- pages
- photos
- Resources
- static
- templates
 - accounts
 - dashboard.html
 - login.html
 - register.html

views.py accounts U x login.html U models.py pages admin.py pages urls.py pages urls.py cars

```
accounts > views.py > login
1  from django.shortcuts import render, redirect
2  from django.contrib import messages, auth
3  from django.contrib.auth.models import User
4
5  # Create your views here.
6
7
8  def login(request):
9
10     if request.method == 'POST':
11         # extracts the user form's input
12         username = request.POST['username']
13         password = request.POST['password']
14
15         # checks the user credentials with our db record
16         user = auth.authenticate(username=username, password=password)
17
18         if user is not None:
19             auth.login(request, user)
20             messages.success(request, 'You are now logged in.')
21             return redirect('dashboard')
22         else:
23             messages.error(request, 'Invalid login credentials.')
24             return redirect('login')
25
26     return render(request, 'accounts/login.html')
```

6. Run the server and test the LOGIN. You should be able to see the error message if incorrect credentials were used otherwise, we should be logged in to our DASHBOARD.



 *CarZone*
Sign into your account

 tammy



Or Login With

 Facebook  Google

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

7. For LOGOUT, we update our TOPBAR.HTML to show a LOGOUT if the user has logged in.

FROM:

```

<ul class="top-social-media pull-right">
  <li>
    <a href="{% url 'login' %}" class="sign-in"><i class="fa fa-sign-in"></i> Login</a>
  </li>
  <li>
    <a href="{% url 'register' %}" class="sign-in"><i class="fa fa-user"></i> Register</a>
  </li>
</ul>

```

TO:

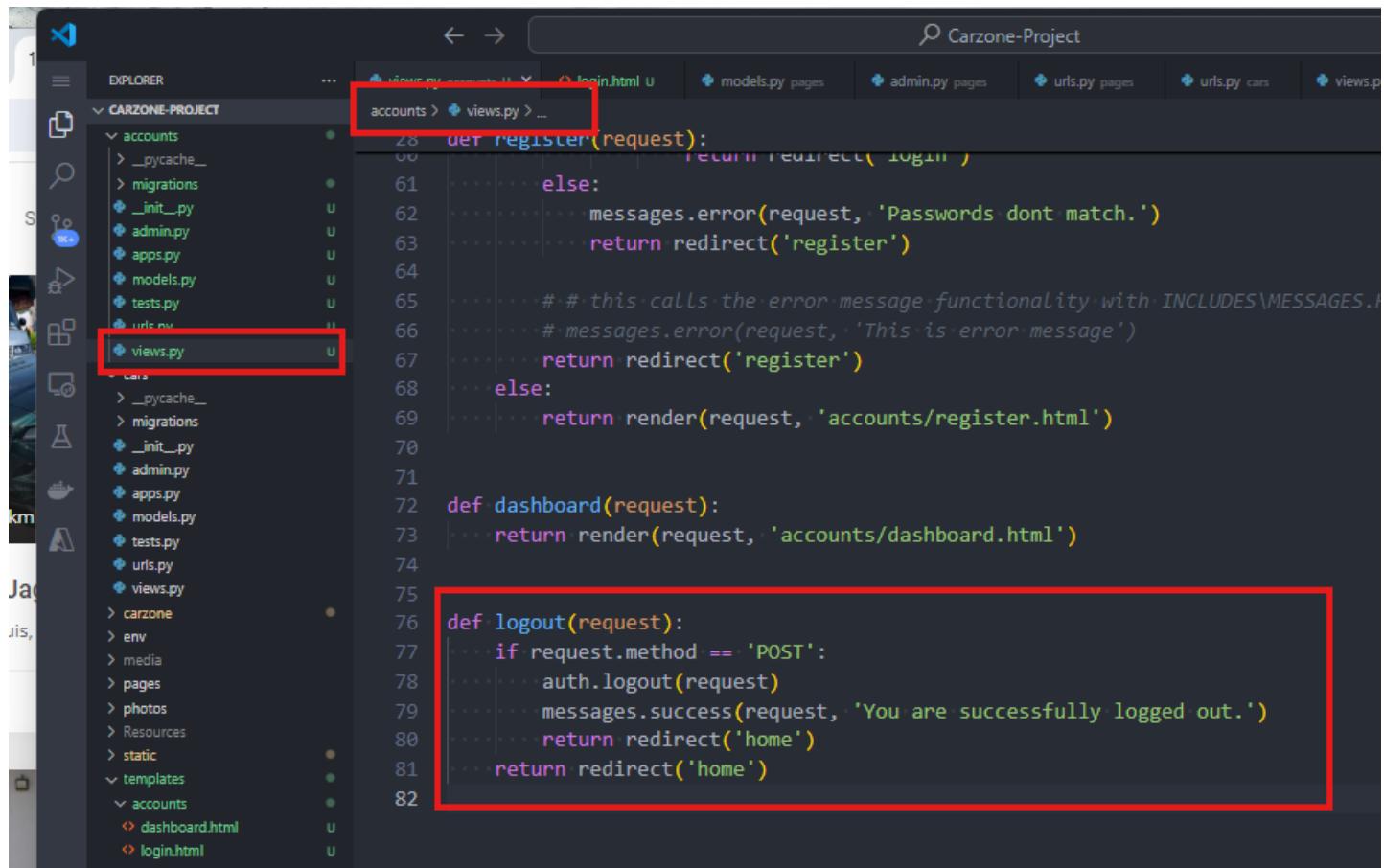
```

<div class="col-lg-4 col-md-4 col-sm-5">
  <ul class="top-social-media pull-right">

    <!-- checks if the user has successfully logged in -->
    {% if user.is_authenticated %}
      <li>
        <a href="{% url 'dashboard' %}" class="sign-in"><i class="fa fa-tachometer"></i> Dashboard</a>
      </li>
      <li>
        <!-- There are other ways to do this Logout, but this is the preferred way-->
        <a href="javascript:(document.getElementById('logout').submit())" class="sign-out"><i class="fa fa-sign-out"></i> Logout</a>
        <form action="{% url 'logout' %}" id="logout" method="POST">
          <!-- this form does not do show anything-->
          {% csrf_token %}
          <input type="hidden">
        </form>
      </li>
    {% else %}
      <li>
        <a href="{% url 'login' %}" class="sign-in"><i class="fa fa-sign-in"></i> Login</a>
      </li>
      <li>
        <a href="{% url 'register' %}" class="sign-in"><i class="fa fa-user"></i> Register</a>
      </li>
    {% endif %}
  </ul>
</div>

```

8. We update our VIEWS.PY to accept the FORM LOGOUT POST METHOD.



The screenshot shows a code editor interface with the following details:

- Project Explorer (Left):** Shows the project structure under "CARZONE-PROJECT". The "accounts" app is expanded, showing files like `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `urls.py`, and `views.py`. The `views.py` file is selected and highlighted with a red box.
- Code Editor (Right):** Displays the content of the `views.py` file. The code includes functions for user registration, dashboard access, and logout. The logout function is highlighted with a red box.
- Toolbar:** Includes standard icons for file operations, search, and navigation.
- Header:** Shows the project name "Carzone-Project" and file navigation buttons.

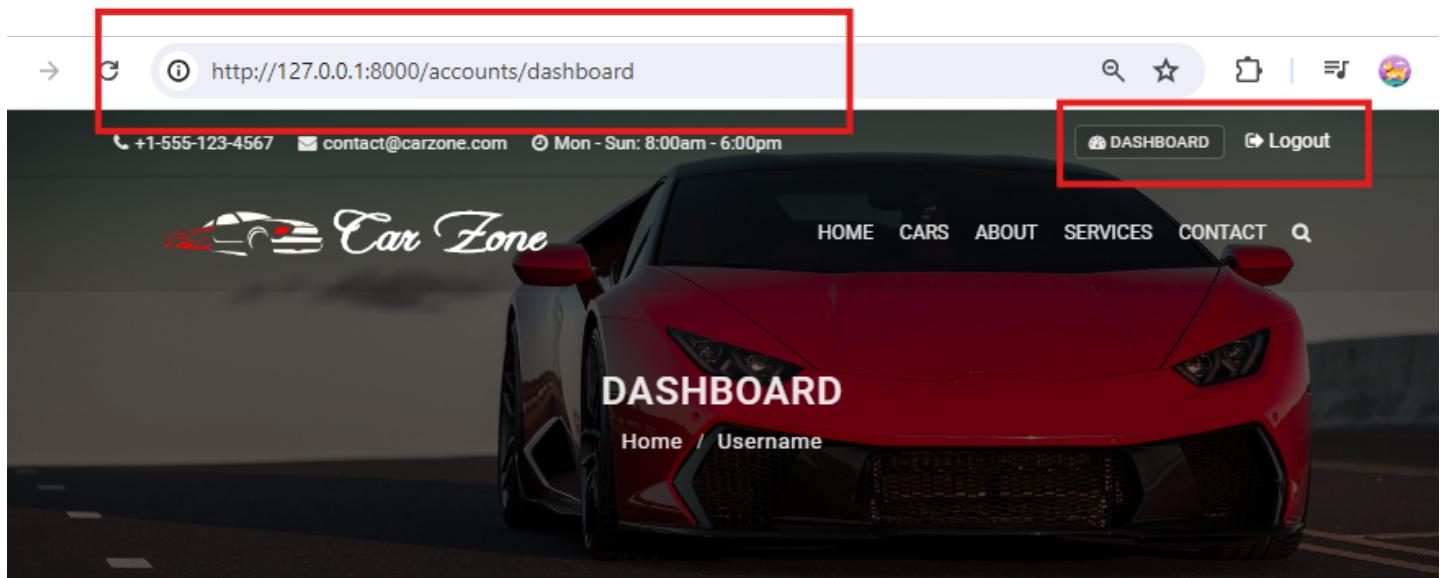
```
def register(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        password2 = request.POST['password2']

        if password == password2:
            if User.objects.filter(username=username).exists():
                messages.error(request, 'This username is already taken.')
            else:
                user = User.objects.create_user(username=username, password=password)
                auth.login(request, user)
                messages.success(request, 'You are successfully registered.')
                return redirect('home')
        else:
            messages.error(request, 'Passwords don't match.')
    else:
        return render(request, 'accounts/register.html')

def dashboard(request):
    return render(request, 'accounts/dashboard.html')

def logout(request):
    if request.method == 'POST':
        auth.logout(request)
        messages.success(request, 'You are successfully logged out.')
        return redirect('home')
    return redirect('home')
```

9. When you are in the dashboard and need to logout, you should be directed to the homepage.



Welcome User

Here are the cars that you have inquired about

#	Car Name	Location	Price	Action
1	Lamborghini Huracan	Austin	\$25000	View Car
2	Lamborghini Huracan	Austin	\$25000	View Car
3	Lamborghini Huracan	Austin	\$25000	View Car

© 2024 Carzone Corp. All Rights Reserved.



10. We want to change the web page title dynamically based on what webpage we are in.

We go to our TEMPLATES\BASE.HTML and find the TITLE SECTION.

FROM:

TO:

11. Then in each webpage like PAGES\HOME.HTML, update it as

So when you run your server. You need to update each of your webpages to write the customized page title.

12. To show the CAR TITLE as part of the PAGE TITLE, we modify CAR_DETAILS.HTML as:

FROM:

TO:

Click on a specific car. The car title should be visible as part of the page title.