Django Kimlik Doğrulama

Gerekli tahmini süre: 30 dakika

Bu laboratuvar, django kimlik doğrulamasını anlamanızı sağlayacaktır.

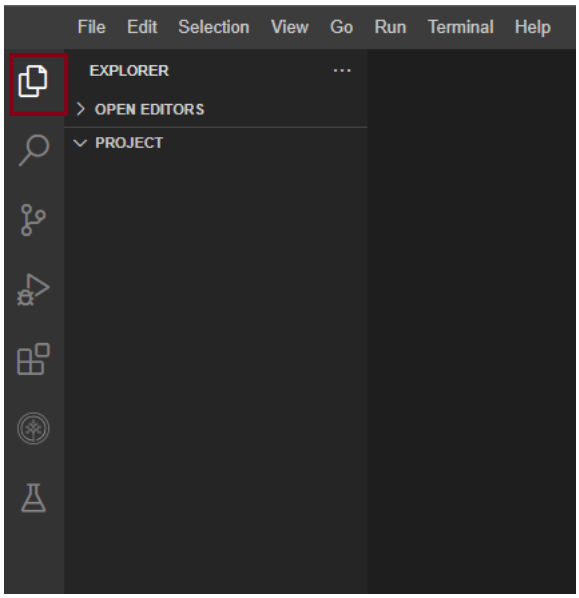
Öğrenme Hedefleri

- Django kimlik doğrulama sistemini anlamak
- Kullanıcı girişi ve çıkışı için görünümler ve şablonlar oluşturmak
- Kullanıcı kaydı için görünümler ve şablonlar oluşturmak

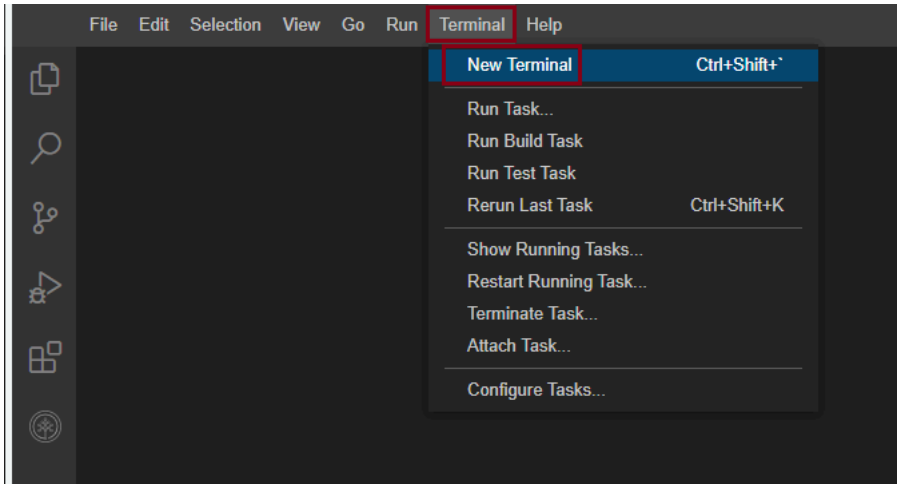
Cloud IDE'de dosyalarla çalışma

Cloud IDE'ye yeniyseniz, bu bölüm size Cloud IDE'deki projenizin bir parçası olan dosyaları nasıl oluşturup düzenleyeceğinizi gösterecektir.

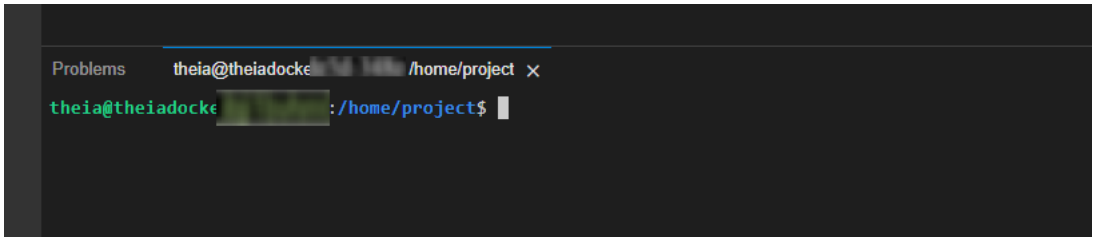
Cloud IDE içindeki dosyalarınızı ve dizinlerinizi görüntülemek için bu dosya simgesine tıklayın.



`Yeni`ye tıklayın, ardından `Yeni Terminal` seçeneğine tıklayın.



Bu, komutlarınızı çalıştırabileceğiniz yeni bir terminal açacaktır.



Laboratuvarda Kapsanan Kavramlar

1. `Authentication`: Bir kullanıcının veya sistemin kimliğini doğrulama süreci.
2. `User`: Django uygulamasıyla etkileşimde bulunan bireyi temsil eder. Genellikle kullanıcı adı, şifre ve e-posta gibi bilgileri içerir.
3. `Registration`: Uygulamada yeni bir kullanıcı hesabı oluşturma süreci.
4. `Login`: Bir kullanıcının korunan bir alana erişmek için kimlik bilgilerini (kullanıcı adı ve şifre gibi) sağladığı süreç.
5. `Logout`: Bir kullanıcının oturumunu sonlandırma ve kimlik doğrulama durumunu kaldırma süreci.

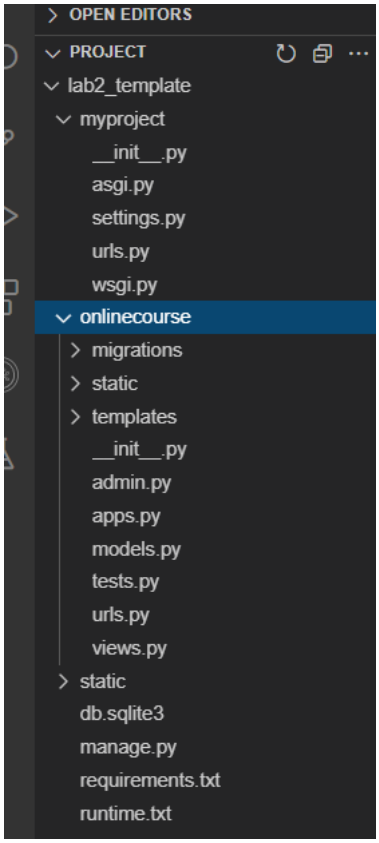
Çevrimiçi Kurs Uygulaması Şablonunu ve Veritabanını İçe Aktarın

Eğer terminal açık değilse, `Terminal > Yeni Terminal` yolunu izleyin ve mevcut Theia dizininizin `/home/project` olduğundan emin olun.

- Bu laboratuvar için bir kod şablonu indirmek üzere aşağıdaki komutları çalıştırın.

```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m5_django_advanced/lab2_template.zip"
unzip lab2_template.zip
rm lab2_template.zip
```

Django projeniz aşağıdaki gibi görünmelidir:



- Proje klasörüne `cd` komutunu kullanarak geçin:

```
cd lab2_template
```

- Sonra, bu laboratuvar için gerekli paketleri yükleyin:

```
pip install --upgrade distro-info
pip3 install --upgrade pip==23.2.1
pip install virtualenv
virtualenv djangoenv
source djangoenv/bin/activate
pip install -U -r requirements.txt
```

Sonra bir `onlinecourse` uygulaması için modelleri etkinleştir.

- Gerekli tabloları oluşturmak için göçleri gerçekleştirin:

  ```
  python3 manage.py makemigrations
  ```

- ve `onlinecourse` uygulaması için modelleri etkinleştirmek üzere göç işlemini çalıştırın.

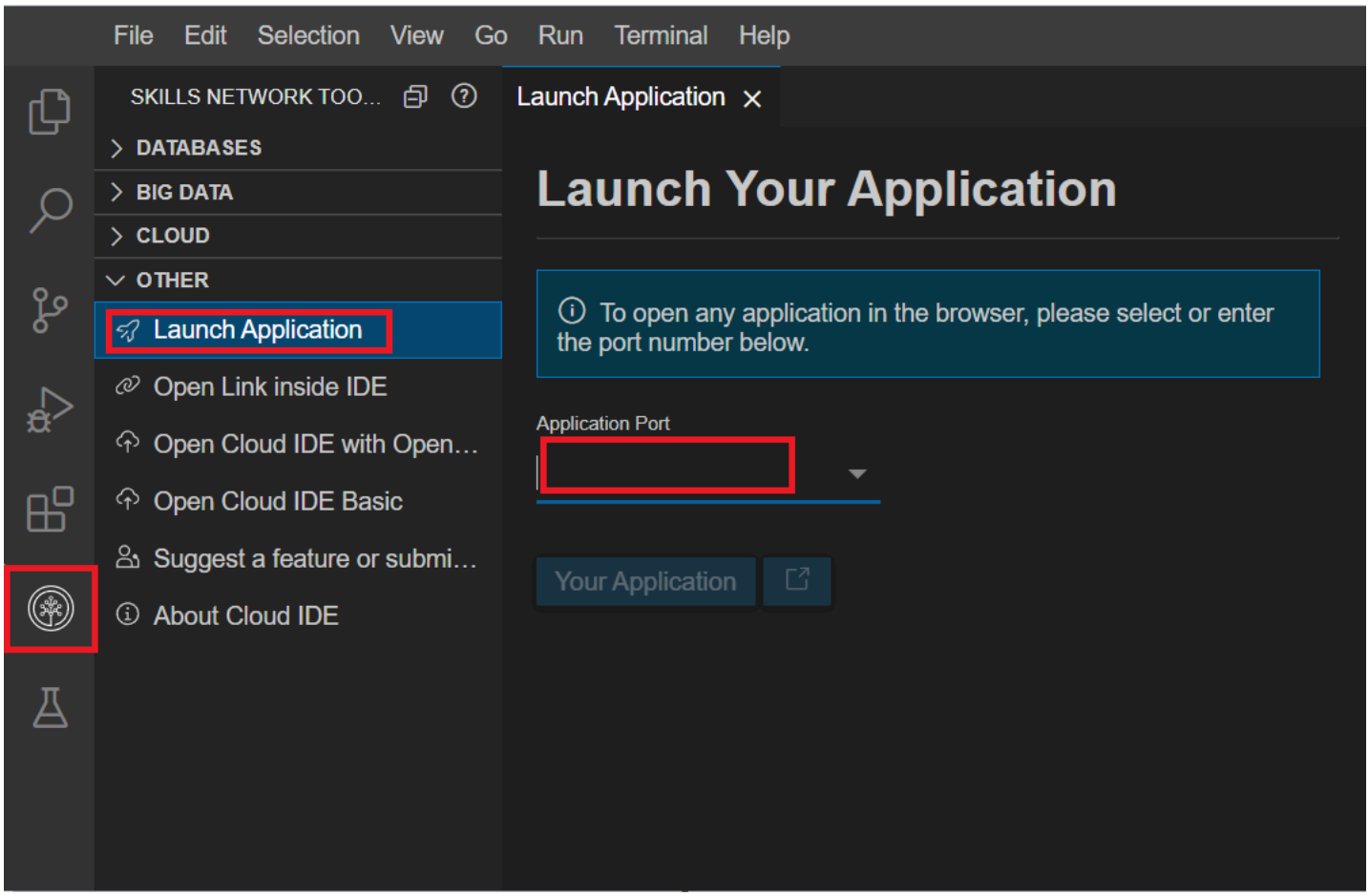
  ```
  python3 manage.py migrate
  ```

Şimdi içe aktarılan `onlinecourse` uygulamasını test edelim.

- Geliştirme sunucusunu başlatın

  ```
  python3 manage.py runserver
  ```

- Sol taraftaki Beceriler Ağı butonuna tıklayın, bu **"Beceriler Ağı Araç Kutusu"**'nu açacaktır. Ardından **Diğer**'e tıklayın ve sonra **Uygulamayı Başlat**'a tıklayın. Buradan, `8000` portunu girmeli ve başlatmalısınız.



Tarayıcı sekmesi açıldığında, `/onlinecourse` yolunu ekleyin ve tam URL'niz aşağıdaki gibi görünmelidir:

`https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/onlinecourse`

Artık ana sayfa olarak bir kurs listesi ile `onlinecourse` uygulamasının başladığını görmelisiniz.

Kullanıcı Bilgilerini Göster

Django kimlik doğrulama sistemi, kullanıcı kimlik doğrulaması ve yetkilendirmesini yönetmek için kullanılan Django yerleşik bir uygulamasıdır.

Django kimlik doğrulamasını öğrenmeye, hızlıca bir süper kullanıcı oluşturarak ve ana sayfada kullanıcı bilgilerini göstererek başlayalım.

- Sunucuyu durdurun eğer başlatıldıysa ve çalıştırın:

```
python3 manage.py createsuperuser
```

`Kullanıcı Adı`, `E-posta`, `Şifre` girildiğinde, süper kullanıcının oluşturulduğunu belirten bir mesaj görmelisiniz:

```
Superuser created successfully.
```

Sunucuyu tekrar başlatalım ve süper kullanıcı ile giriş yapalım.

```
python3 manage.py runserver
```

- `Uygulamayı Başlat` butonuna tıklayın ve geliştirme sunucusu için `8000` portunu girin.

Tarayıcı sekmesi açıldıktan sonra `/admin` yolunu ekleyin ve tam URL'niz aşağıdaki gibi görünmelidir:

`https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/admin`

- Oluşturduğunuz süper kullanıcı için kimlik bilgileriyle admin sitesine giriş yapın.
- `AUTHENTICATION AND AUTHORIZATION` bölümünde `Kullanıcılar` kısmına tıklayın ve oluşturduğunuz süper kullanıcıyı bulun.
- Ana sayfada gösterilecek `Ad`, `Soyad` gibi bazı profil bilgilerini eklemeyi deneyin.
- Daha aşağı kaydırın ve profil bilgilerini saklamak için `Kaydet` butonuna tıklayın.

Şimdi süper kullanıcıyı almayı ve profilini `onlinecourse/course_list.html` şablonunda göstermeyi deneyelim.

- `onlinecourse/templates/onlinecourse/course_list.html` dosyasını açın, `<!--Authentication section-->` yorumu altında aşağıdaki kod parçasını ekleyin.

```
{% if user.is_authenticated %}
<p>Username: {{user.username}}, First name: {{user.first_name}}, Last name: {{user.last_name}} </p>
{% endif %}
```

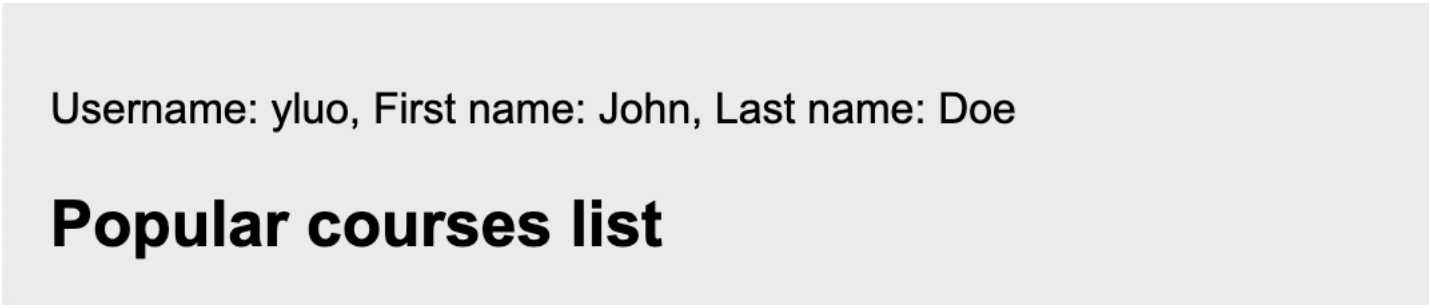
Şablonda, `user` nesnesi Django tarafından otomatik olarak sizin için `session_id`'ye dayalı olarak sorgulanacak ve hem şablonlarda hem de görünümlerde kullanılabilir olacak.

Ardından, kullanıcının kimlik doğrulamasının yapılıp yapılmadığını veya anonim bir kullanıcı olup olmadığını kontrol etmek için `{% if user.is_authenticated %}` ifadesini kullanacağız ve kimlik doğrulaması yapılmışsa ad, soyad, e-posta gibi profil bilgilerini göstereceğiz.

Test etmek için, geliştirme sunucusunun çalıştığından emin olun ve şu adrese gidin:

`https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/onlinecourse`.

Üstte kullanıcı bilgilerini görmelisiniz.



Giriş ve Çıkış

Süper kullanıcı olarak giriş yaptıktan sonra, tarayıcınız `session_id`'yi çerezde saklayacak, böylece süper kullanıcı oturumu süresi dolana kadar açık kalacaktır.

Şimdi, ana sayfadan süper kullanıcıyı manuel olarak çıkış yapmayı deneyelim ve bir çıkış açılır düğmesi ekleyelim.

- `onlinecourse/course_list.html` dosyasını açın, `{% if user.is_authenticated %}` ve `{% endif %}` arasındaki kodu bir açılır `<div>` ile güncelleyin:

```
{% if user.is_authenticated %}
<div class="rightalign">
       <div class="dropdown">
           <button class="dropbtn">{{user.first_name}}</button>
           <div class="dropdown-content">
                <a href="{% url 'onlinecourse:logout' %}">Logout</a>
           </div>
       </div>
</div>
{% endif %}
```

Yukarıdaki kod bloğu, kullanıcının adını görüntülemek için bir açılır düğme ekler. Düğmenin üzerine geldiğinizde, çıkış görünümüne referans veren bir bağlantı açılır.

Sonraki adımda `logout_request` görünümünü oluşturalım.

- `onlinecourse/views.py` dosyasını açın, `# Create authentication related views` yorumu altında fonksiyon tabanlı bir çıkış görünümü ekleyin:

```
def logout_request(request):
    # Get the user object based on session id in request
    print("Log out the user `{}`".format(request.user.username))
    # Logout user in the request
    logout(request)
    # Redirect user back to course list view
    return redirect('onlinecourse:popular_course_list')
```

Yukarıdaki kod parçası, kullanıcıyı çıkış yaptırmak için `request`'i argüman olarak alarak yerleşik bir `logout` yöntemini çağırır.

- `urlpatterns` içinde bir yol girişi ekleyerek `logout_request` görünümü için bir rota yapılandırın:

```
path('logout/', views.logout_request, name='logout'),
```

Artık çıkış işlevselliğini kurs listesi sayfasını yenileyerek test edebilirsiniz. Aşağı açılır menüden `logout` düğmesine tıkladıktan sonra, kullanıcı artık kimlik doğrulaması yapılmadığı için aşağı açılır menü düğmesinin kaybolduğunu görmelisiniz.

Sonraki adımda, süper kullanıcı olarak tekrar giriş yapmayı deneyelim.

- `templates/onlinecourse/course_list.html` dosyasını açın, `{% if user.is_authenticated %}` bloğunu güncelleyin.

```
{% if user.is_authenticated %}
    <div class="rightalign">
        <div class="dropdown">
            <button class="dropbtn">{{user.first_name}}</button>
            <div class="dropdown-content">
                <a href="{% url 'onlinecourse:logout' %}">Logout</a>
            </div>
        </div>
    </div>
    {% else %}
    <div class="rightalign">
        <div class="dropdown">
            <button class="dropbtn">Visitor</button>
            <div class="dropdown-content">
                <a href="{% url 'onlinecourse:login' %}">Login</a>
            </div>
        </div>
    </div>
{% endif %}
```

Burada, kullanıcının kimlik doğrulaması yapılmadığında durumu ele almak için bir `{% else %}` etiketi ekledik ve ayrıca `login` görünümüne işaret eden bir bağlantı ile yeni bir açılır düğme oluşturduk.

`login` görünümü, kullanıcı adı ve şifre gibi kullanıcı kimlik bilgilerini isteyen ortak bir giriş sayfası döndürmelidir.

Şimdi, böyle bir `login` görünümü için bir şablon oluşturalım:

- `templates/onlinecourse/user_login.html` dosyasını açın ve kullanıcı adı ve şifre kabul etmek için basit bir `form` ekleyin.

```
<form action="{% url 'onlinecourse:login' %}"  method="post">
  {% csrf_token %}
  <div class="container">
    <h1>Login</h1>
    <label for="username"><b>User Name</b></label>
    <input type="text" placeholder="Enter User Name: " name="username" required>
    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password: " name="psw" required>
    <div>
      <button class="button" type="submit">Login</button>
    </div>
```

```
  </div>
</form>
```

Bu giriş formunun ana unsurları, kullanıcı adı ve şifre için iki giriş alanıdır. Form gönderildikten sonra, bir `POST` isteği ile giriş görünümüne gönderilir.

Sonraki adımda, giriş isteğini işlemek için bir giriş görünümü oluşturalım.

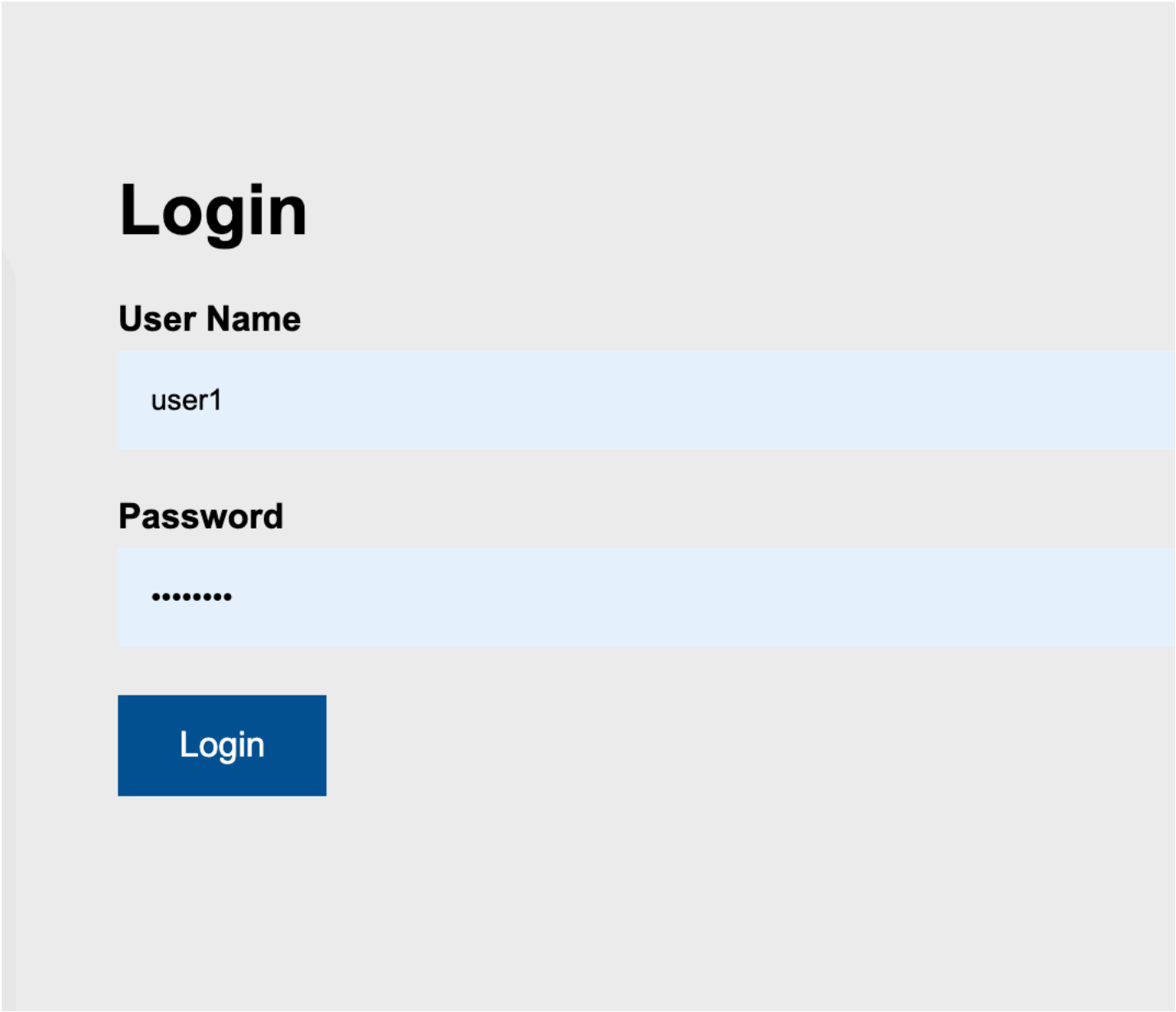
- `onlinecourse/views.py` dosyasını açın, bir `login_request` görünümü ekleyin:

```
def login_request(request):
    context = {}
    # Handles POST request
    if request.method == "POST":
        # Get username and password from request.POST dictionary
        username = request.POST['username']
        password = request.POST['psw']
        # Try to check if provide credential can be authenticated
        user = authenticate(username=username, password=password)
        if user is not None:
            # If user is valid, call login method to login current user
            login(request, user)
            return redirect('onlinecourse:popular_course_list')
        else:
            # If not, return to login page again
            return render(request, 'onlinecourse/user_login.html', context)
    else:
        return render(request, 'onlinecourse/user_login.html', context)
```

- ve `onlinecourse/urls.py` dosyasındaki `urlpatterns` listesine bir yol girişi ekleyerek `login_request` görünümü için bir rota yapılandırın:

```
path('login/', views.login_request, name='login'),
```

Şimdi kurs listesi sayfasını yenileyerek giriş fonksiyonunu test edebiliriz:



Oluşturulan süper kullanıcı ile hem giriş hem de çıkış yapmayı deneyebilirsiniz.

Kodlama Pratiği: Kullanıcı Kaydı

Önceki adımda, bir süper kullanıcı oluşturmak için CLI'yi kullandık. Normal kullanıcılar için, kullanıcı kimlik bilgilerini almak ve kaydetmek için bir kullanıcı kaydı şablonu ve görünümü oluşturmamız gerekecek.

> Not: `onlinecourse/views.py` dosyası üzerinde çalışacaksınız.

Model seviyesinde, kullanıcı kaydını tamamlamak için `auth_user` tablosunda bir kullanıcı nesnesi oluşturulacak. Kullanıcı oluşturulduktan sonra, kullanıcıyı giriş yapabiliriz ve kullanıcıyı kurs listesi sayfasına yönlendirebiliriz.

- `onlinecourse/templates/onlinecourse/course_list.html` dosyasını açın, kimlik doğrulama bölümünü güncelleyerek bir bağlantı ekleyin `<a href="{% url 'onlinecourse:registration' %}">Kaydol</a>` oluşturulacak bir `registration` görünümüne işaret edecek şekilde.

```
{% if user.is_authenticated %}
    <div class="rightalign">
        <div class="dropdown">
            <button class="dropbtn">{{user.first_name}}</button>
            <div class="dropdown-content">
                <a href="{% url 'onlinecourse:logout' %}">Logout</a>
            </div>
        </div>
    </div>
    {% else %}
    <div class="rightalign">
        <div class="dropdown">
            <form action="{% url 'onlinecourse:registration' %}" method="get">
                <input class="dropbtn"  type="submit" value="Visitor">
                <div class="dropdown-content">
                    <a href="{% url 'onlinecourse:registration' %}">Signup</a>
                    <a href="{% url 'onlinecourse:login' %}">Login</a>
                </div>
            </form>
        </div>
    </div>
{% endif %}
```

- Aşağıdaki kod parçasını tamamlayarak bir `registration_request` görünümü oluşturun ve kayıt POST isteğini işleyin:

```
def registration_request(request):
    context = {}
    # If it is a GET request, just render the registration page
    if request.method == 'GET':
        return render(request, 'onlinecourse/user_registration.html', context)
    # If it is a POST request
    elif request.method == 'POST':
        # <HINT> Get user information from request.POST
        # <HINT> username, first_name, last_name, password
        user_exist = False
        try:
            # Check if user already exists
            User.objects.get(username=username)
            user_exist = True
        except:
            # If not, simply log this is a new user
            logger.debug("{} is new user".format(username))
        # If it is a new user
        if not user_exist:
            # Create user in auth_user table
            user = User.objects.create_user(#<HINT> create the user with above info)
            # <HINT> Login the user and
            # redirect to course list page
            return redirect("onlinecourse:popular_course_list")
        else:
            return render(request, 'onlinecourse/user_registration.html', context)
```

▼ Çözümü görmek için buraya tıklayın

```
def registration_request(request):
    context = {}
    # If it is a GET request, just render the registration page
    if request.method == 'GET':
        return render(request, 'onlinecourse/user_registration.html', context)
    # If it is a POST request
    elif request.method == 'POST':
        # Get user information from request.POST
        username = request.POST['username']
        password = request.POST['psw']
        first_name = request.POST['firstname']
        last_name = request.POST['lastname']
        user_exist = False
        try:
            # Check if user already exists
            User.objects.get(username=username)
            user_exist = True
        except:
            # If not, simply log this is a new user
            logger.debug("{} is new user".format(username))
        # If it is a new user
        if not user_exist:
            # Create user in auth_user table
            user = User.objects.create_user(username=username, first_name=first_name, last_name=last_name,
                                            password=password)
            # Login the user and redirect to course list page
            login(request, user)
            return redirect("onlinecourse:popular_course_list")
        else:
            return render(request, 'onlinecourse/user_registration.html', context)
```

- Kullanıcı bilgilerini kabul etmek için bir `registration` şablonu oluşturmak üzere aşağıdaki kod parçasını tamamlayın:

```
<form action="{% url 'onlinecourse:registration' %}" method="post">
  <div class="container">
    <h1>Sign Up</h1>
    <hr>
    <!-- HINT, added inputs for username, firstname, lastname, and password  -->
    <div>
        {% csrf_token %}
```

```
        <button class="button" type="submit">Sign Up</button>
      </div>
    </div>
  </form>
```

▶ Çözümü görmek için buraya tıklayın

- `urls.py` dosyasındaki `urlpatterns` listesine bir yol girişi ekleyerek `registration_request` görünümü için bir yol ekleyin:

  ```
  path('registration/', views.registration_request, name='registration'),
  ```

Ana sayfayı şimdi yenileyin ve yeni kayıt fonksiyonunu giriş/çıkış ile birlikte deneyin.

Özet

Bu laboratuvar sırasında, Django kimlik doğrulamasının nasıl çalıştığını anladınız ve kullanıcı girişi, çıkışı ve kaydı için şablonlar ve görünümler uyguladınız.

Author(s)

Yan Luo