IBM Developer
SKILLS NETWORK

Bootstrap Integration

Estimated time needed: 30 minutes

Learning Objectives

- Integrate Bootstrap front-end library into your Django templates

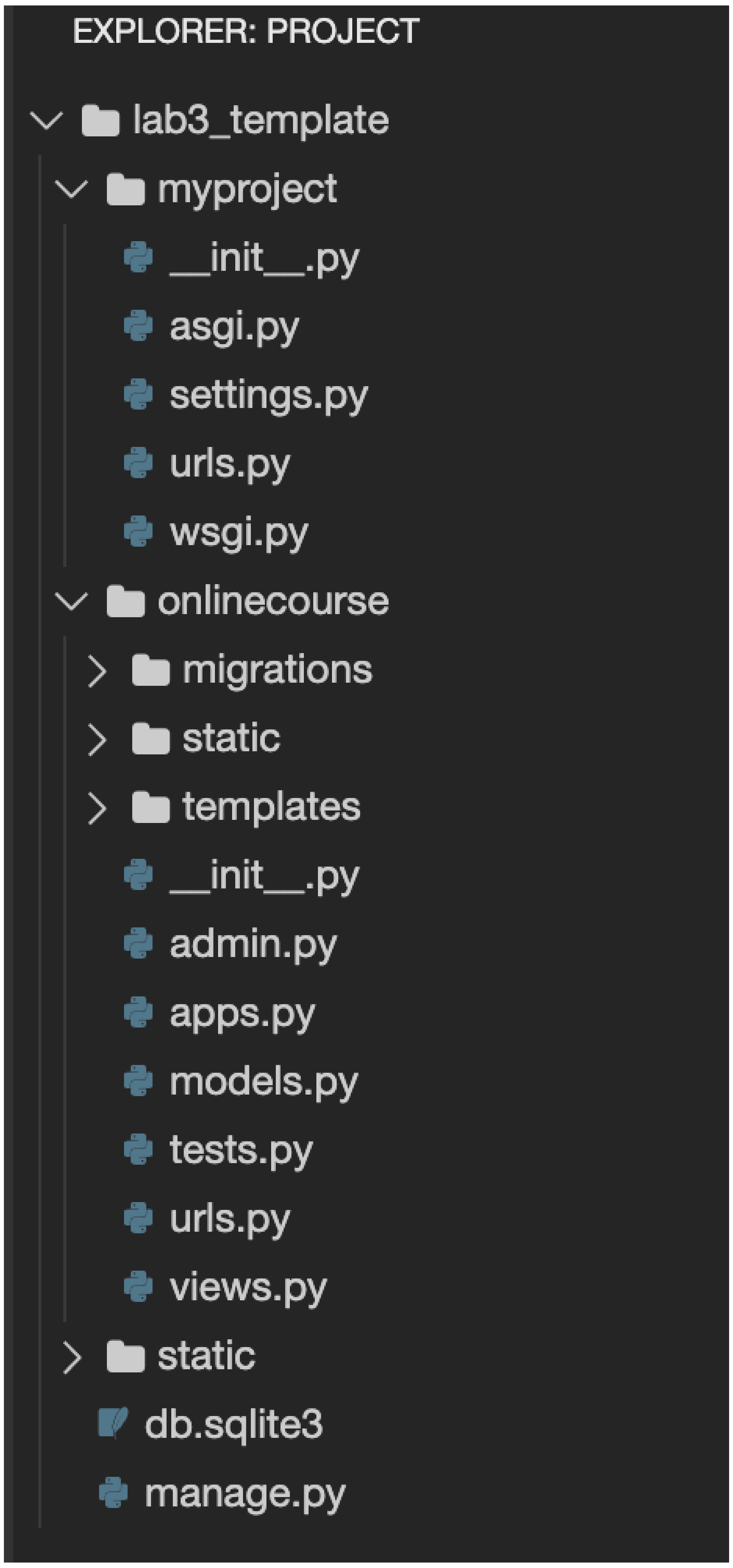
Import an `onlinecourse` App Template and Database

If the terminal was not open, go to `Terminal > New Terminal` and make sure your current Theia directory is `/home/project`.

- Run the following command-lines to download a code template for this lab

```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m5_django_advanced/lab3_template.zip"
unzip lab3_template.zip
rm lab3_template.zip
```

Your Django project should look like the following:


EXPLORER: PROJECT

```
lab3_template
  myproject
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  onlinecourse
    migrations
    static
    templates
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  static
  db.sqlite3
  manage.py
```

- `cd` to the project folder:

```
cd lab3_template
```

- Install the necessary Python packages.

```
pip3 install -r requirements.txt
```

The `requirements.txt` contains all necessary Python packages for you to run this lab.

Next activate the models for an `onlinecourse` app.

- Perform migrations to create necessary tables:

```
python3 manage.py makemigrations
```

- and run migration to activate models for the `onlinecourse` app.

```
python3 manage.py migrate
```

Now let's test the imported `onlinecourse` app.

- Start the development server:

```
python3 manage.py runserver
```

- Click `Launch Application` and enter the port for the development server `8000`

When the browser tab opens, add the `/onlinecourse` path and your full URL should look like the following

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

Now you should see the `onlinecourse` app started, and all the pages are rendered from unpolished Django templates without any CSS.

In this lab, you will be learning how to use Bootstrap to stylize those pages.

Integrate Bootstrap

Adding a minimal Bootstrap CSS file to a Django template is actually very easy.

- Open `onlinecourse/templates/onlinecourse/course_list.html`, link a Bootstrap CSS file `bootstrap.min.css` to

`<head>...</head>` element

```
<head>
    {% load static %}
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <meta charset="UTF-8">
    <title>Online Courses</title>
</head>
```

Here we used `maxcdn` CDN (Content Distribution Network) for delivering `bootstrap.min.css` file to our templates. In addition to `maxcdn`, there are many other popular CDNs you may use such as `StackPath`.

- Repeat the above steps to update the `<head>` of all template files in `onlinecourse/templates/onlinecourse` folder.

Create a Navigation Bar

The first UI improvement is adding a navigation bar to help users accessing information in our app.

- Open `onlinecourse/templates/onlinecourse/course_list.html`, and add an empty light themed navigation bar `<nav>`

with a navigation bar header (as the app brand):

```
<nav class="navbar navbar-light bg-light">
    <div class="container-fluid">
        <div class="navbar-header">
              <a class="navbar-brand" href="#">Online Course</a>
        </div>
    </div>
</nav>
```

The content `<div>` of the navigation bar `<nav>` uses a `.container-fluid` Bootstrap CSS class to make sure the bar content takes the full width of the page.

Then we added a `<div>` with `.navbar-header` class to define header with a simple link styled with `.navbar-brand` to highlight the app name.

Refresh the main course list page and you should see a light theme navigation bar with a header called `Online Course`.

Next, let's move the user sign-in and sign-up area to the navigation bar.

- Remove the existing `<ul>` list of user authentication related elements such as user name and password input fields.
- Under `<div class="navbar-header">`, add a new list styled with classes `nav navbar-nav navbar-right` to

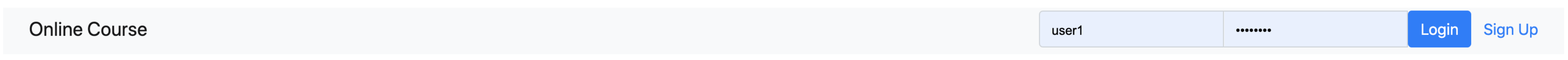
create a navigation bar item and be aligned to the right.

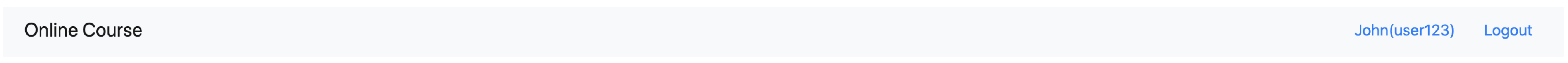
```
<ul class="nav navbar-nav navbar-right">
    {% if user.is_authenticated %}
    <li>
         <a class="btn btn-link" href="#">{{ user.first_name }}({{ user.username }})</a>
         <a class="btn btn-link" href="{% url 'onlinecourse:logout' %}">Logout</a>
    </li>
    {% else %}
    <li>
         <form class="form-inline" action="{% url 'onlinecourse:login' %}" method="post">
            {% csrf_token %}
            <div class="input-group">
                 <input type="text" class="form-control" placeholder="Username" name="username" >
                 <input type="password" class="form-control" placeholder="Password" name="psw" >
                 <button class="btn btn-primary" type="submit">Login</button>
                 <a class="btn btn-link" href="{% url 'onlinecourse:registration' %}">Sign Up</a>
            </div>
         </form>
    </li>
    {% endif %}
</ul>
```

The completed `<nav>` HTML element should look like the following code snippet:

```
<nav class="navbar navbar-light bg-light">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="#">Online Course</a>
        </div>
        <ul class="nav navbar-nav navbar-right">
            {% if user.is_authenticated %}
            <li>
                <a class="btn btn-link" href="#">{{ user.first_name }}({{ user.username }})</a>
                <a class="btn btn-link" href="{% url 'onlinecourse:logout' %}">Logout</a>
            </li>
            {% else %}
            <li>
                <form class="form-inline" action="{% url 'onlinecourse:login' %}" method="post">
                    {% csrf_token %}
                    <div class="input-group">
                        <input type="text" class="form-control" placeholder="Username" name="username" >
                        <input type="password" class="form-control" placeholder="Password" name="psw" >
                        <button class="btn btn-primary" type="submit">Login</button>
                        <a class="btn btn-link" href="{% url 'onlinecourse:registration' %}">Sign Up</a>
                    </div>
                </form>
            </li>
            {% endif %}
        </ul>
    </div>
</nav>
```

Refresh the page and you should see a nice navigation bar with user login and registration elements integrated.





Coding Practice: Create Navigation Bar for All Templates

Copy/paste the navigation bar your created in the previous step to `course_detail.html`, `user_login.html`, `user_registration.html`.

- For all templates other than the course list template, change the navigation bar header link pointing to home page, i.e., page generated by `popular_course_list`

▶ Click here to see solution

Stylize Login and Registration Pages

Next, let's stylize the login and registration pages.

- Open the `onlinecourse/templates/onlinecourse/user_login.html`, find the root `<div>` with

comment `<!--Style root div with .container class -->` and stylize it:

```
<div class="container">
```

Bootstrap `.container` class adds some paddings to the `<div>`.

- Find the second level `<div>` element with comment `<!--Style second div with .form-group class -->` and stylize it:

```
<div class="form-group">
```

Bootstrap `.form-group` class builds the structure for a form.

- Find two `<input>` elements with comment `<!--Style input with .form-control class -->` and stylize it with:

```
<input ... class = "form-control" ...>
```

Bootstrap `.form-control` class stylizes the elements in a form.

- Find a `<div>` with comment `<!--Style the message paragraph with .alert and .alert-warning class -->`.

This `<div>` is designed to show warning messages such as user already exists or user password is not correct. Stylize it with:

```
<div class="alert alert-warning">
```

- Find the submit `<button>` with comment `<!--Style button with .btn and .btn-primary class -->`. Stylize it with

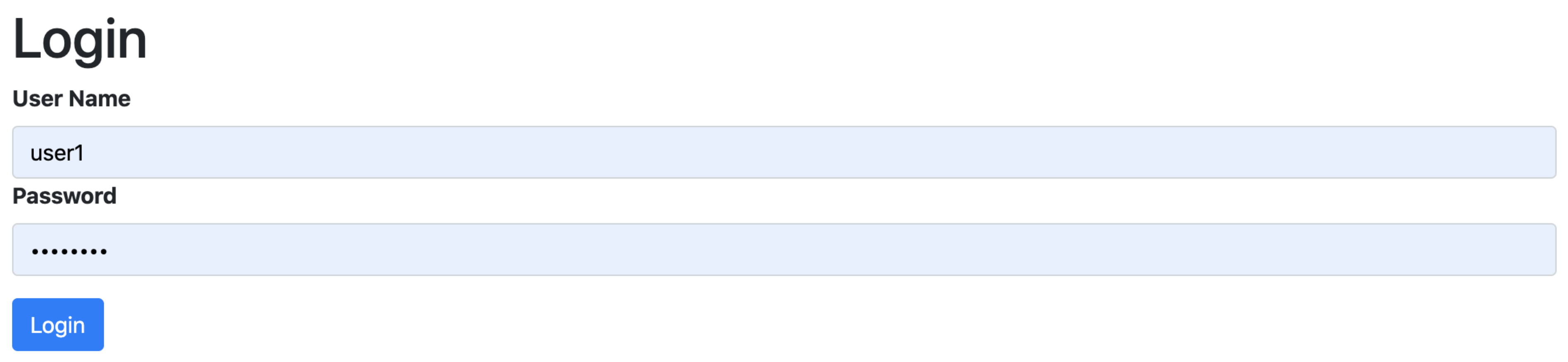
```
<button class="btn btn-primary" ... >
```

Once you have stylized above elements, the login form should look like the following:

```
<form action="{% url 'onlinecourse:login' %}" method="post">
  {% csrf_token %}
  <div class="container"> <!--Style root div with .container class -->
    <h1>Login</h1>
    <div class="form-group"> <!--Style second div with .form-group class -->
      <label for="username"><b>User Name</b></label>
      <input id="username" class = "form-control" type="text" placeholder="Enter User Name: " name="username" required> <!--
Style input with .form-control class -->
      <label for="psw"><b>Password</b></label>
      <input id="psw" class = "form-control" type="password"  placeholder="Enter Password: " name="psw" required> <!--Style
input with .form-control class -->
    </div>
    {% if message %}
      <div class="alert alert-warning"> <!--Style the message paragraph with .alert and .alert-warning class -->
        {{ message }}
      </div>
    {% endif %}
    <button class="btn btn-primary" type="submit">Login</button> <!--Style button with .btn and .btn-primary class -->
  </div>
</form>
```

- Next, let's go to `https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/onlinecourse/login` and check the result:

and your login page should look like the following:


Login

User Name

user1

Password

••••••••

Login

Coding Practice: Stylize the Registration Page

- Stylize the `user_registration.html` by following the examples in `user_login.html` page

and the comments in the template.

▼ Click here to see solution

```
<form action="{% url 'onlinecourse:registration' %}" method="post">
  <div class="container"> <!--Style root div with .container class -->
    {% csrf_token %}
    <h1>Sign Up</h1>
    <hr>
    <div class="form-group"> <!--Style second div with .form-group class -->
        <label for="username"><b>User Name</b></label>
        <input type="text" class="form-control" placeholder="Enter User Name: " name="username" required> <!--Style input with
.form-control class -->
        <label for="firstname"><b>First Name</b></label>
        <input type="text" class="form-control" placeholder="Enter First Name: " name="firstname" required> <!--Style input with
.form-control class -->
        <label for="lastname"><b>Last Name</b></label>
        <input type="text" class="form-control" placeholder="Enter Last Name: " name="lastname" required> <!--Style input with
.form-control class -->
        <label for="psw"><b>Password</b></label>
        <input type="password" class="form-control" placeholder="Enter Password: " name="psw" required> <!--Style input with
.form-control class -->
        {% if message %}
        <div class="alert alert-warning"> <!--Style the message paragraph with .alert and .alert-warning class -->
            {{ message }}
        </div>
        {% endif %}
        <button class="btn btn-primary" type="submit">Sign up</button> <!--Style button with .btn and .btn-primary class -->
    </div>
  </div>
</form>
```

Stylize the Course List using Card and CardDeck classes

Next, let's stylize the main content of course list page.

For the course list, we want to organize each course as a card with an image, title, and description and put them onto a card deck.

- Open `onlinecourse/templates/onlinecourse/course_list.html`, find the content root level

`<div>` with comment `<!--Style root div with .container class -->` and stylize it with:

```
<div class="container">
```

- Find the second level card deck `<div>` with comment `<!--Style second div with .card-deck class -->` and stylize it with:

```
<div class="card-deck">
```

- Find the third level card `<div>` with comment `<!--Style third level div with .card class -->` and stylize it with:

```
<div class="card" ... >
```

- Find card image `<image>` with comment `<!--Style card image with .card-img-left class -->` and stylize it with:

```
<img class="card-img-left" ... >
```

- Find the card body `<div>` class with comment `<!--Style root div with .card-body and .bg-light class -->` and stylize it with:

```
<div class="card-body bg-light">
```

- Find the card title `<h5>` and `<span>` with comment `<!--Style h5 with .card-title and span with .text-success class -->` and stylize them with:

```
<h5 class="card-title"> ... <span class="text-success"> ...
```

- Find the card text `<p>` with comment `<!--Style card description with .card-text class -->` and stylize it with:

```
<p class="card-text">
```

- Find the enrollment submission button `<input>` with comment `<!--Style Enroll button with .btn and .btn-primary class -->` and stylize it with:

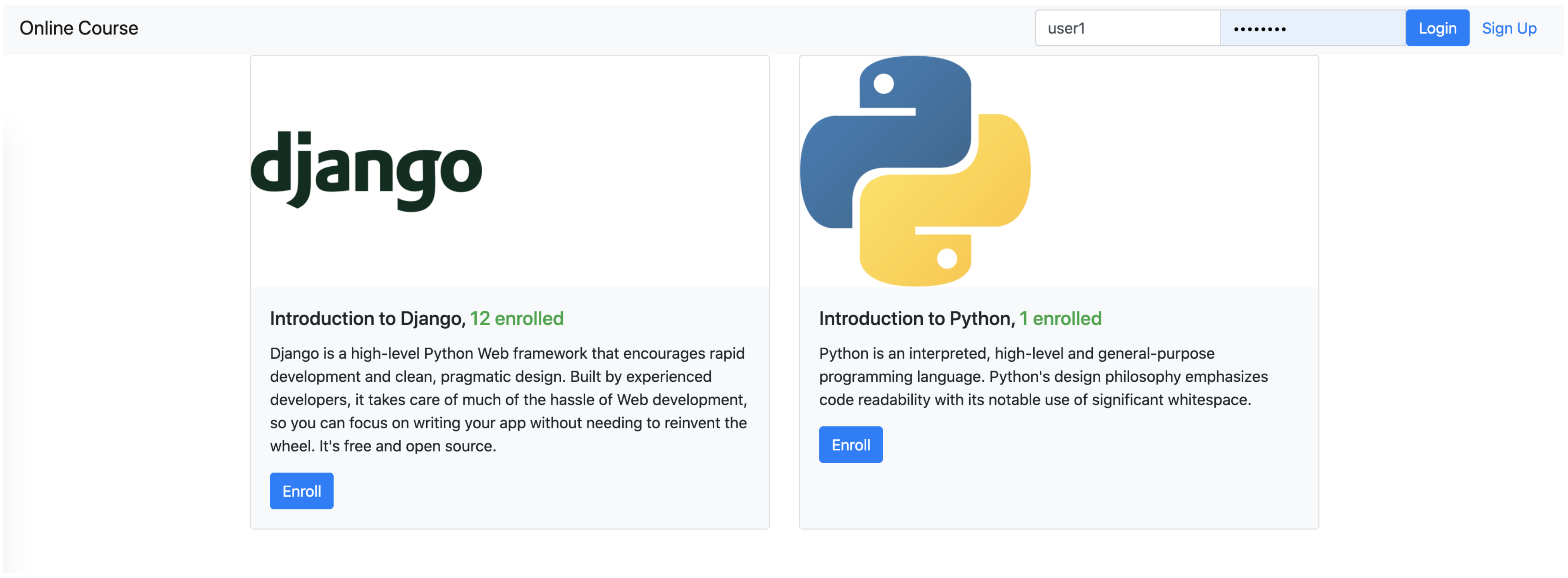
```
<input class="btn btn-primary" >
```

Now your course list is stylized into a course card deck

```
{% if course_list %}
    <div class="container"> <!--Style root div with .container class -->
        <div class="card-deck"> <!--Style second div with .card-deck class -->
            {% for course in course_list %}
            <div class="card" style="width: 36rem;"> <!--Style third level div with .card class -->
                <img class="card-img-left" src="{{MEDIA_URL}}/{{ course.image }}" width="240px" height="240px"
                           alt="Course image"> <!--Style card image with .card-img-left class -->
                <div class="card-body bg-light"> <!--Style root div with .card-body and .bg-light class -->
                    <h5 class="card-title">{{ course.name }}, <span class="text-success"> <!--Style h5 with .card-title and span
with .text-success class -->
                              {{ course.total_enrollment}} enrolled</span></h5>
                    <p class="card-text">{{ course.description}}</p> <!--Style card description with .card-text class -->
                    <form action="{% url 'onlinecourse:enroll' course.id %}" method="post">
                        {% csrf_token %}
                        <input class="btn btn-primary" type="submit"
                               value="Enroll"> <!--Style Enroll button with .btn and .btn-primary class -->
                    </form>
                </div>
            </div>
            {% endfor %}
        </div>
    </div>
{% else %}
    <p>No courses are available.</p>

{% endif %}
```

Let's open `https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/onlinecourse` and check the fully stylized course card deck page:



Introduction to Django, 12 enrolled

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Enroll

Introduction to Python, 1 enrolled

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

Enroll

Coding Practice: Stylize Course Details Page

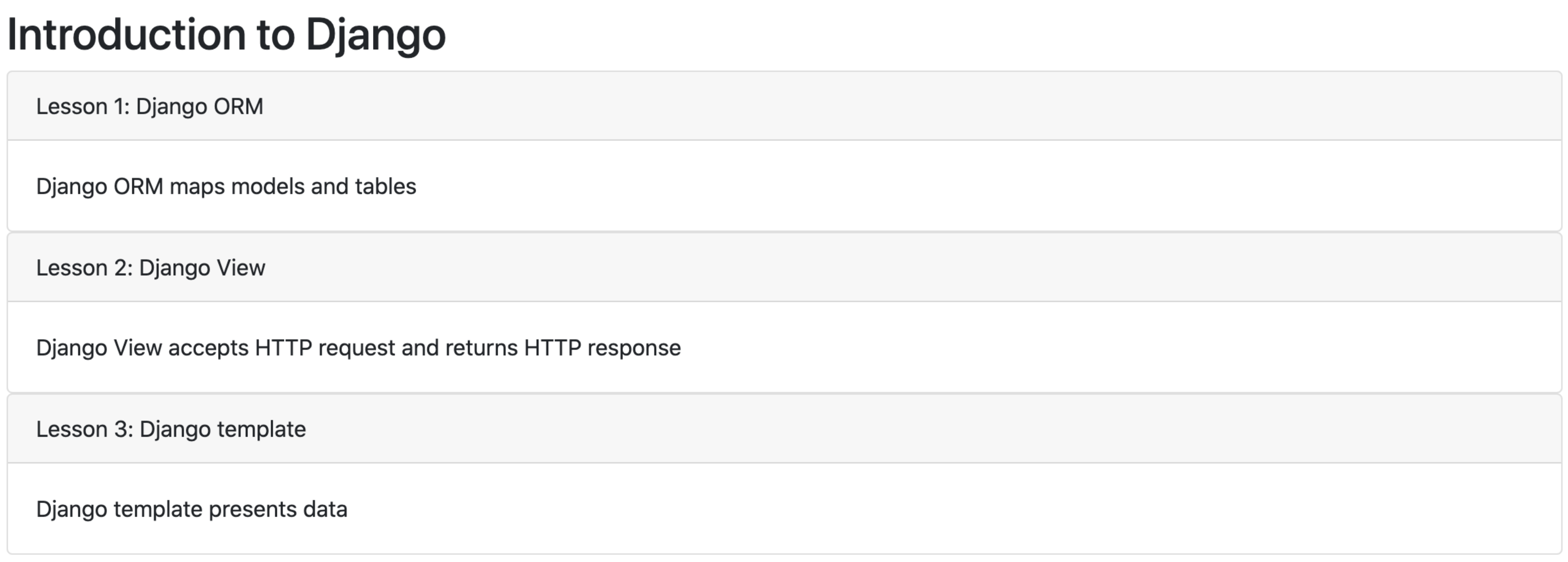
- Stylize `onlinecourse/templates/onlinecourse/course_details.html` by making

each lesson as a card and the lesson list as a vertical card column.

▼ Click here to see solution

```
<div class="container"> <!--Stylize the root div with .container class -->
    <h2>{{ course.name }}</h2>
    <div class="card-columns-vertical"> <!--Stylize the second level card columns div with .card-columns-vertical class-->
        {% for lesson in course.lesson_set.all %}
        <div class="card"> <!--Stylize the third level lesson div with .card class-->
            <div class="card-header">Lesson {{lesson.order|add:1}}: {{lesson.title}}</div> <!--Stylize the fourth level lesson header  div with .card-header class-->
            <div class="card-body">{{lesson.content}}</div>  <!--Stylize the fifth level lesson content div with .card-body class-->
        </div>
        {% endfor %}
    </div>
</div>
```

The stylized lesson columns should look like the following:


Introduction to Django

Lesson 1: Django ORM

Django ORM maps models and tables

Lesson 2: Django View

Django View accepts HTTP request and returns HTTP response

Lesson 3: Django template

Django template presents data

Summary

In this lab, you have learned how to integrate Bootstrap into your Django templates. You also practiced several common Bootstrap CSS classes such as `.nav`, `.container`, `.card-deck`, `.card`, etc.

Author(s)

[Yan Luo](linkedin.com/in/yan-luo-96288783)

Changelog

Date	Version	Changed by	Change Description
14-Dec-2020	1.0	Yan Luo	Initial version created

© IBM Corporation 2020. All rights reserved.