Django Admin site

Estimated time needed: 20 minutes

In this lab, you will learn to use the admin site provided by Django to manage models, customize the admin site, and create Inline classes for adding related objects on the same page.

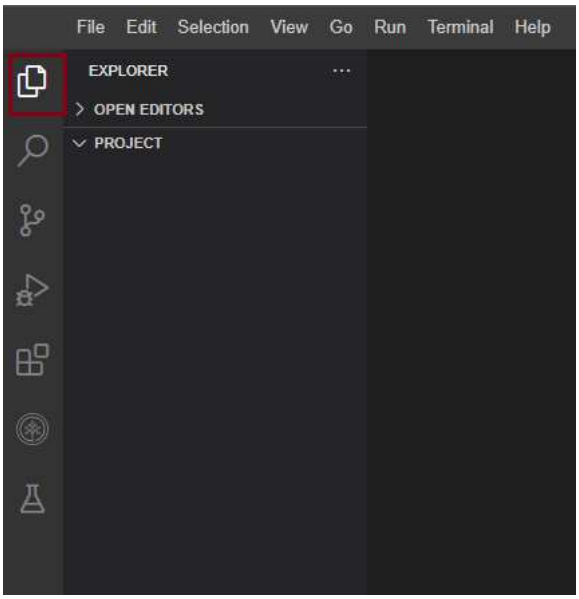
Learning Objectives

- Understand the concepts of Django admin site
- Create and customize your Django admin site to manage the content of an `onlinecourse` app

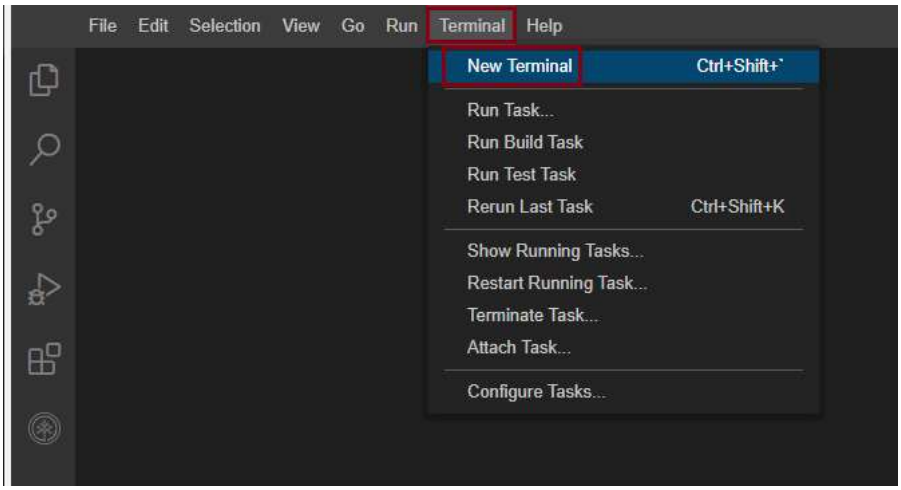
Working with files in Cloud IDE

If you are new to Cloud IDE, this section will show you how to create and edit files, which are part of your project, in Cloud IDE.

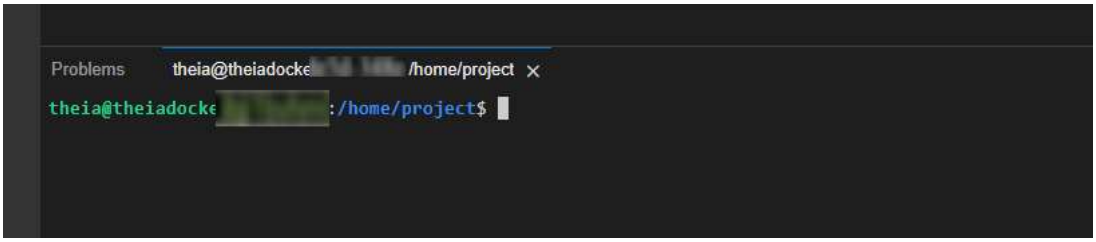
To view your files and directories inside Cloud IDE, click on this files icon to reveal it.



Click on `New`, and then `New Terminal`.



This will open a new terminal where you can run your commands.

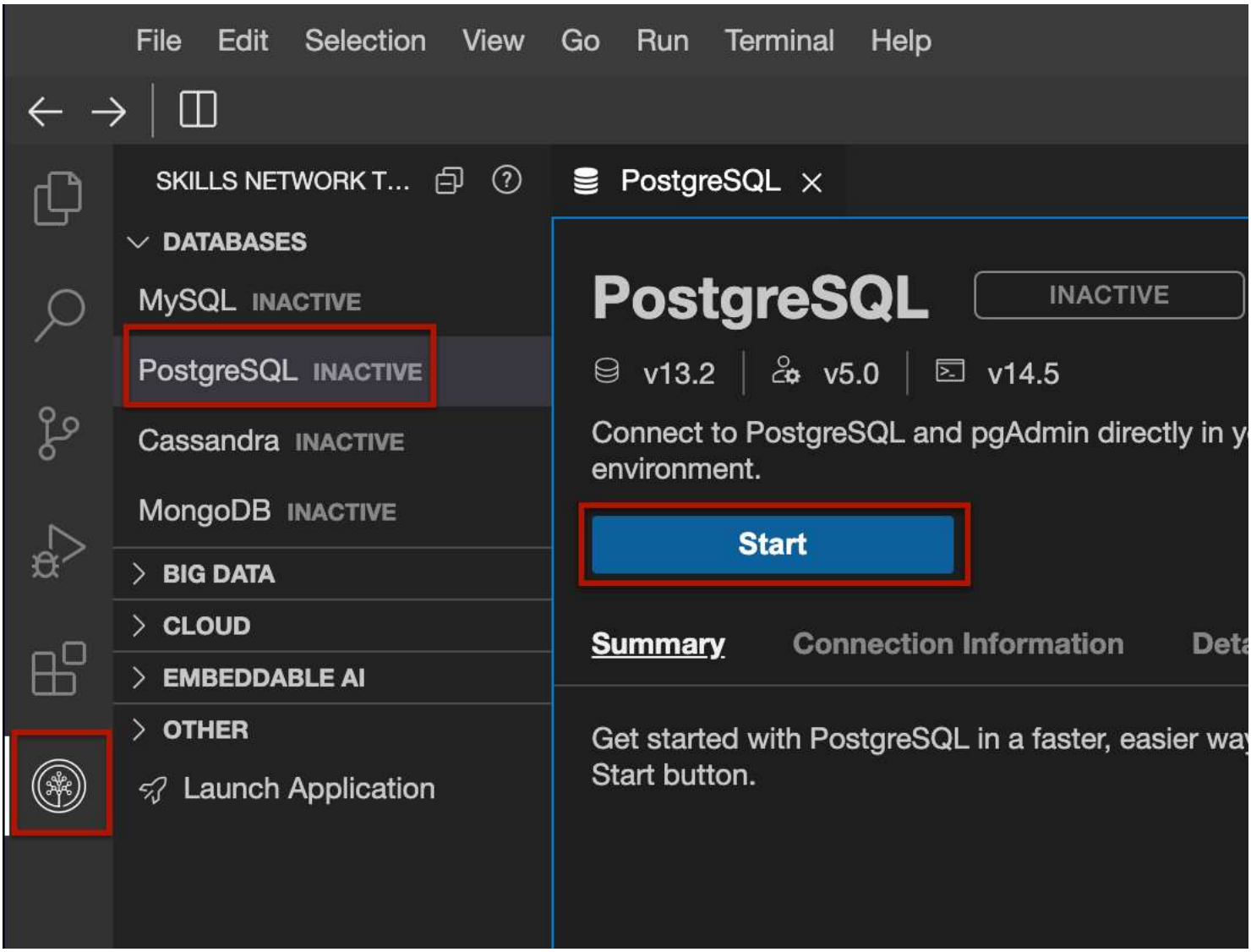


Concepts covered in the lab

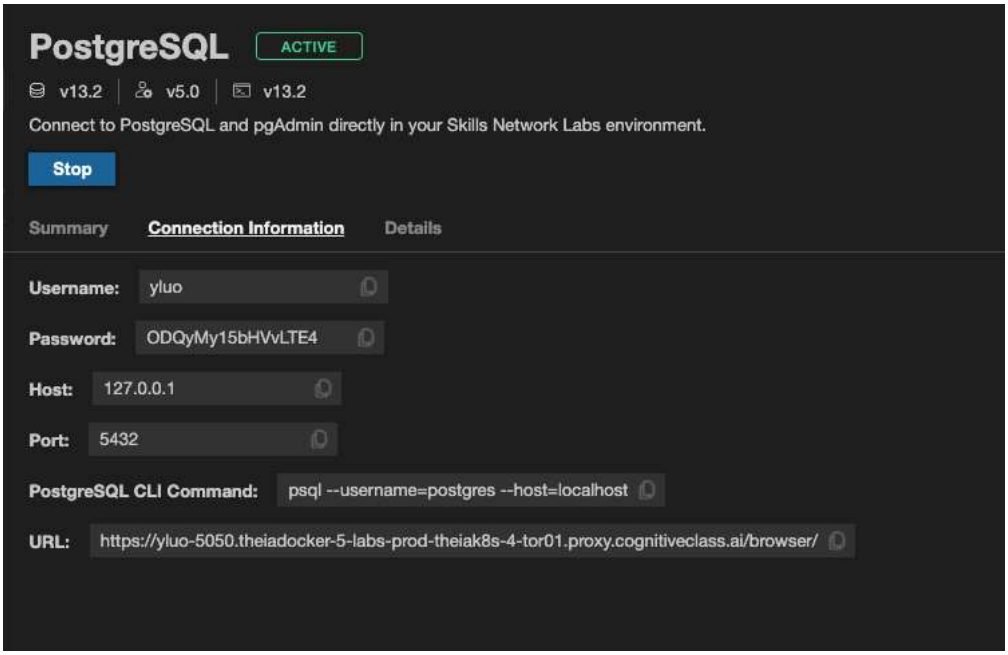
1. `Django admin site`: A built-in feature of the Django web framework that serves as an interface and allows authorized users to perform various management operations on the data stored in the application's database.
2. `Superuser`: A user account with administrative privileges that allows superusers to perform administrative tasks and manage the application's data.
3. `Admin class`: A class that helps fine-tune the behavior, appearance, and functionality of models within the Django admin site.
4. `Inline class`: Allow you to include related model instances in the same page/form as the parent model, instead of switching between different forms or screens.

Start PostgreSQL in Theia

If you are using the Theia environment hosted by Skills Network Labs, you can start the pre-installed `PostgreSQL` instance from UI by finding the SkillsNetwork icon on the left menu bar and selecting PostgreSQL from the DATABASES menu item:



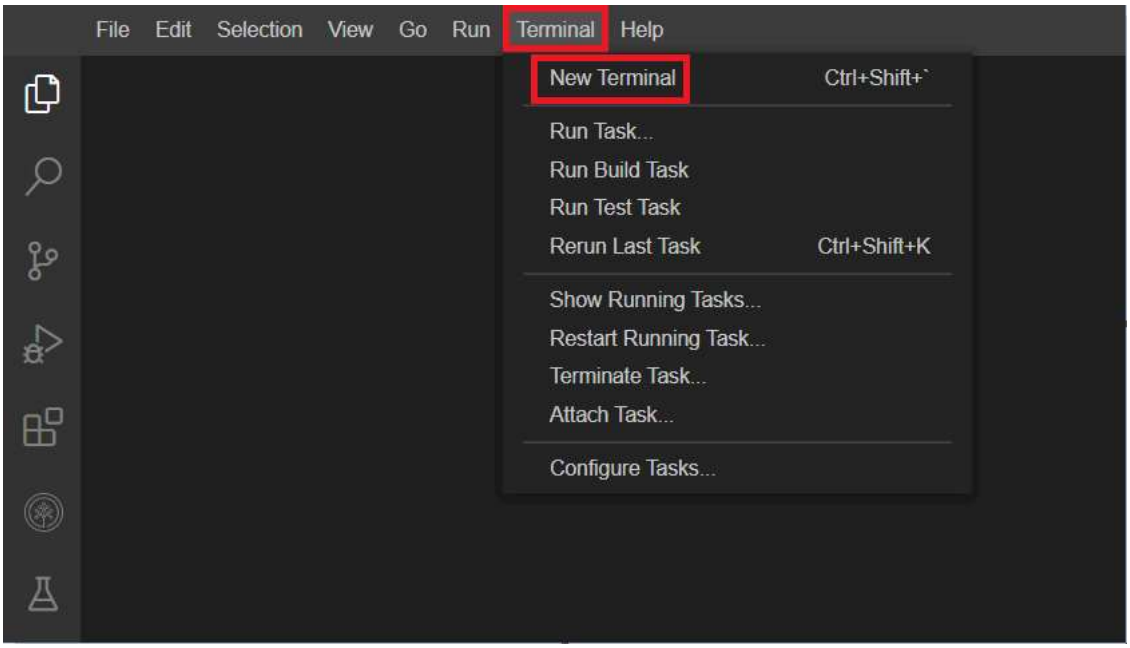
Once the PostgreSQL has been started, you can check the server connection information from the UI.
Please markdown the connection information such as generated `username`, `password`, and `host`, etc, which will be used to configure Django app to connect to this database.



Optional: Start PostgreSQL from terminal

You may also start PostgreSQL from terminal (*if you have not already started it from the UI):*

- If the terminal was not open, go to `Terminal > New Terminal`



- and run:

```
1
```

```
start_postgres
```

Copied!

- Once PostgreSQL is started, you can check the server connection information in the terminal. You need to save the connection information such as generated `username`, `password`, and `host`, etc, which will be used to configure Django app to connect to this database.

```
1
2
3
4
5
6
7
8
9
10
```

```
Starting your Postgres database....
```

```
2. This process can take up to a minute.
3.
4. Postgres database started, waiting for all services to be ready....
5.
6. Your Postgres database is now ready to use and available with username: postgres password: Nzg3Mi15bHVvLTIz
7.
8. You can access your Postgres database via:
9.  ˙.t: https://yluo-5050.theiadocker-1.proxy.cognitiveclass.ai
10.  岸psql --username=postgres --host=localhost
```

Copied!

- Install these must-have packages before you setup the environment to access postgres.

```
1. 1
2. 2
```

```
1. pip install --upgrade distro-info
2. pip3 install --upgrade pip==23.2.1
```

Copied!

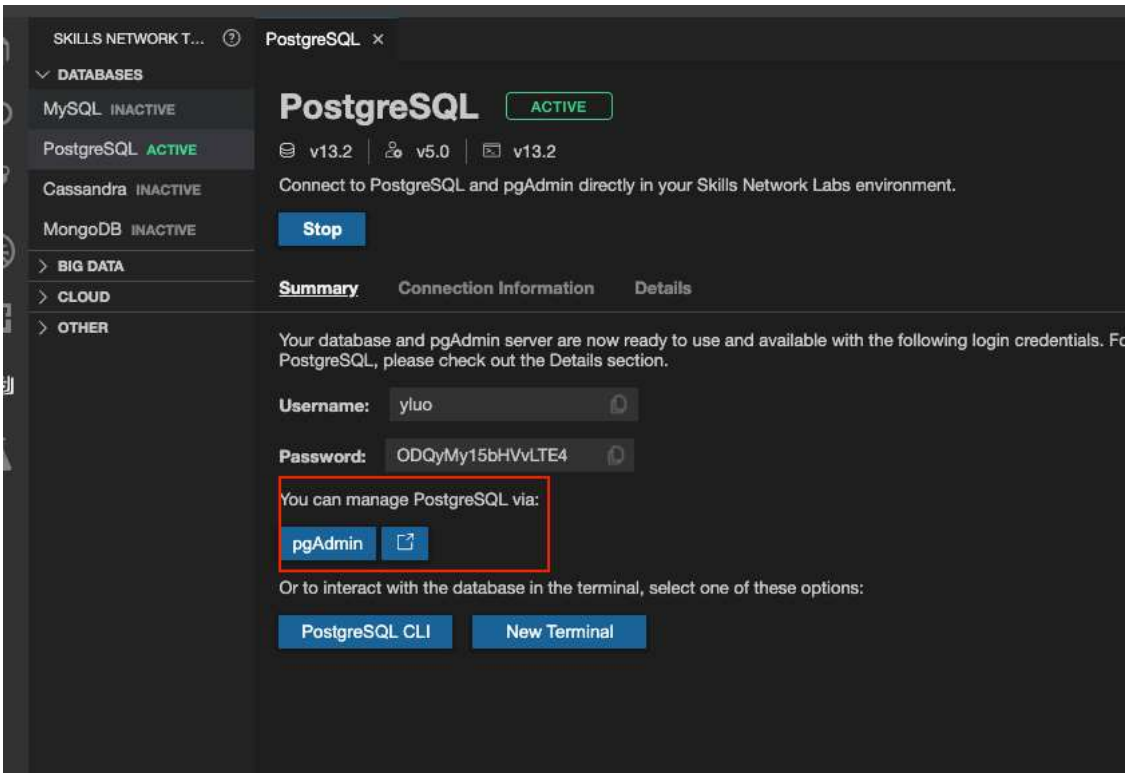
- Install the `Psycopg` adapter:

```
1. 1
```

```
1. pip install psycopg2-binary==2.9.7
```

Copied!

- A `pgAdmin` instance is also installed and started for you.



Import an onlinecourse App Template

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

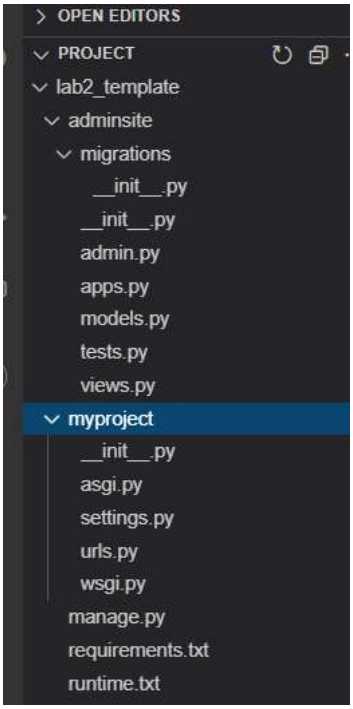
- Run the following command-lines to download a code template for this lab

```
1. 1
2. 2
3. 3
```

```
1. wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m4_django_app/lab2_template.zip"
2. unzip lab2_template.zip
3. rm lab2_template.zip
```

Copied!

Your Django project should look like the following:



Next, we need to set up a proper runtime environment for Django app.

- If the terminal was not open, go to `Terminal > New Terminal` and `cd` to the project folder

```
1
```

```
cd lab2_template
```

Copied!

Let's set up a virtual environment which has all the packages we need.

```
1
2
3
4
5
```

```
pip install virtualenv

virtualenv djangoenv

source djangoenv/bin/activate
```

Copied!

```
1
```

```
pip install -r requirements.txt
```

Copied!

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

- Open `settings.py` file located in `myproject` directory and find `DATABASES` section, and replace the value of `PASSWORD` with the one generated by `PostgreSQL` password.

Next, activate the models for an `onlinecourse` app which will be managed by admin site.

- You need to perform migrations to create necessary tables

```
1
```

```
python3 manage.py makemigrations
```

Copied!

- and run migration

```
1
```

```
python3 manage.py migrate
```

Copied!

Create a Superuser for Admin Site

Before you can access admin site, you will need to create a super user for the admin site.

- Run this command to create a new superuser:

```
1
```

```
python3 manage.py createsuperuser
```

Copied!

- With `Username`, `Email`, `Password` entered, you should see a message indicates the super user is created:

```
1
```

```
Superuser created successfully.
```

Copied!

Let's start our app and login with the super user.

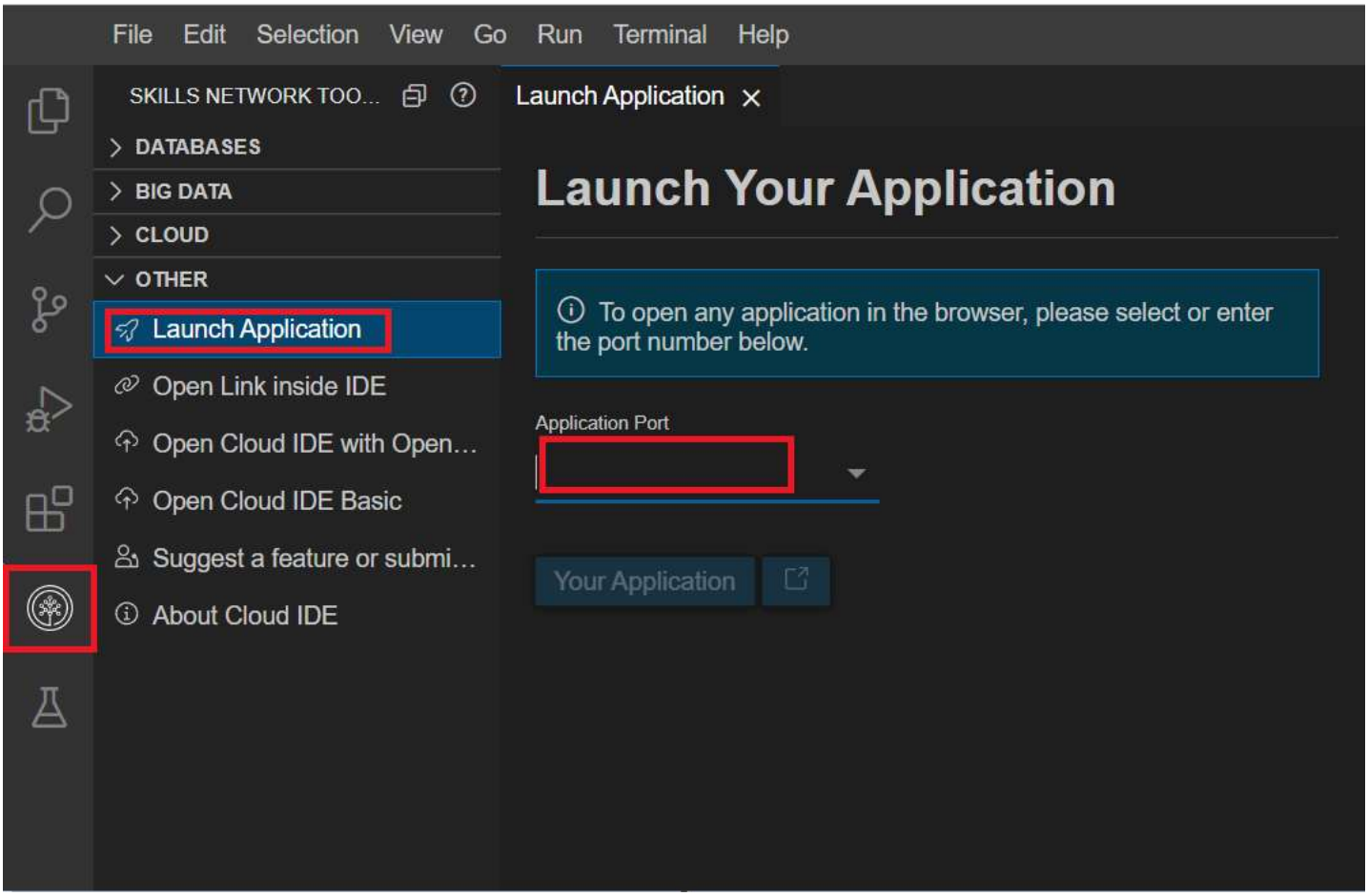
- Start the development server

```
1
```

```
python3 manage.py runserver
```

Copied!

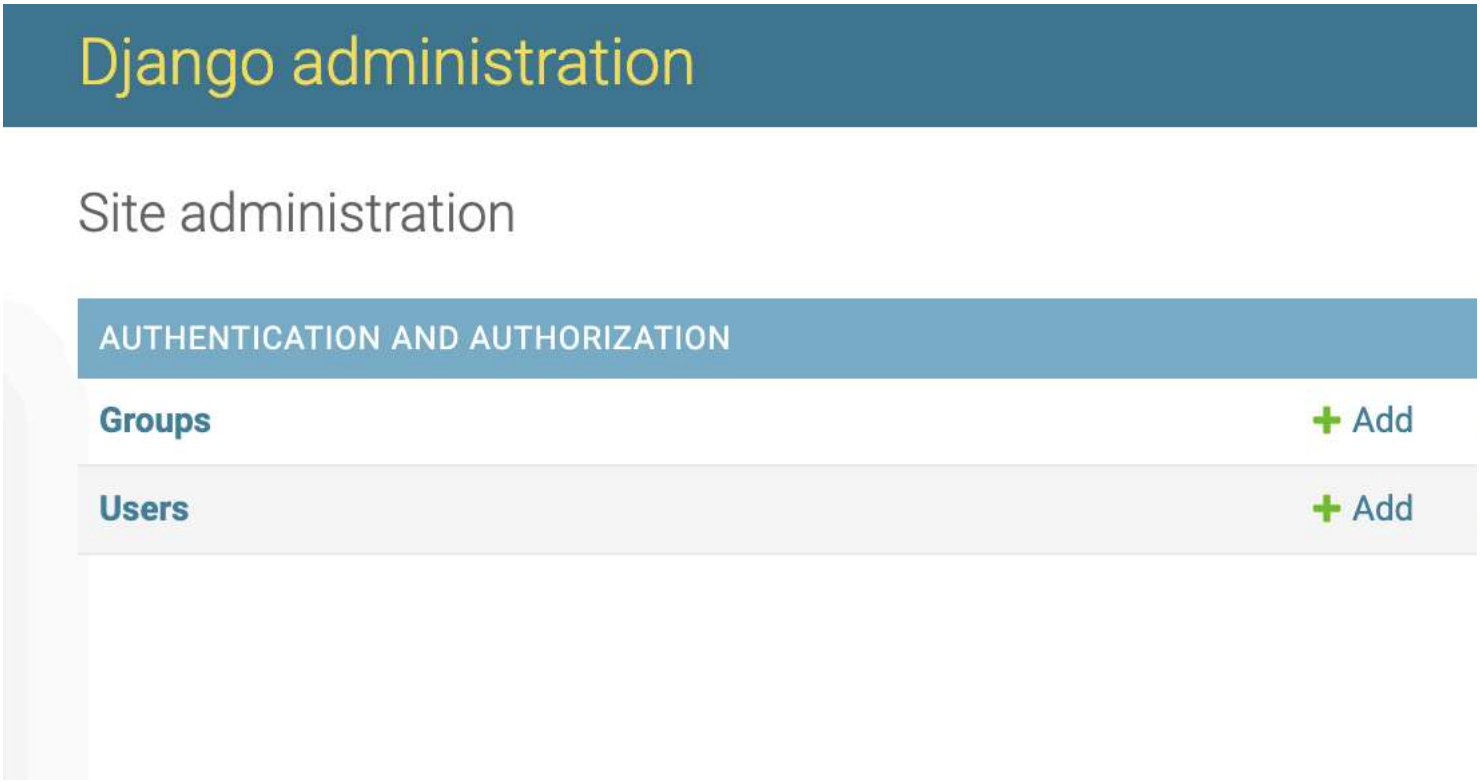
- Click on the Skills Network button on the left, it will open the **"Skills Network Toolbox"**. Then click the **Other** then **Launch Application**. From there you should be able to enter the port `8000` and launch.



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

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

- Login with the user name and password for the super user, then you should see admin site with only `Groups` and `Users` entries created for us. These two tables are created by Django by default for authentication and authorization purposes.



Register Models with Admin site

Once you register models defined in `adminsite/models.py` with admin site, you can then create managing pages for those models.

- Open `adminsite/admin.py`, and register `Course` and `Instructor` models

```
1. 1
2. 2
3. 3
```

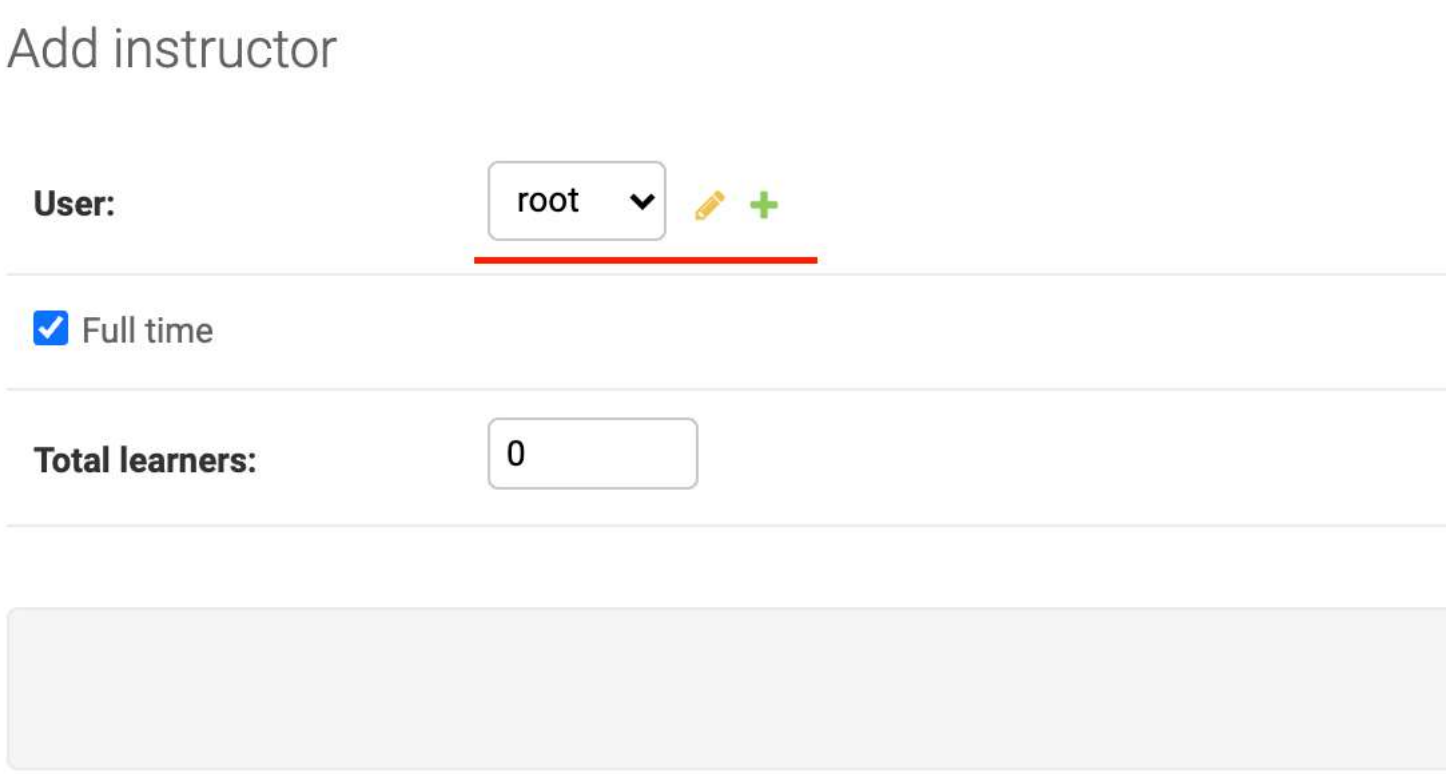
```
1.
2. admin.site.register(Course)
3. admin.site.register(Instructor)
```

Copied!

- Now refresh the admin site, and you should be able to see `Courses` and `Instructors` under `ADMINSITE` section.

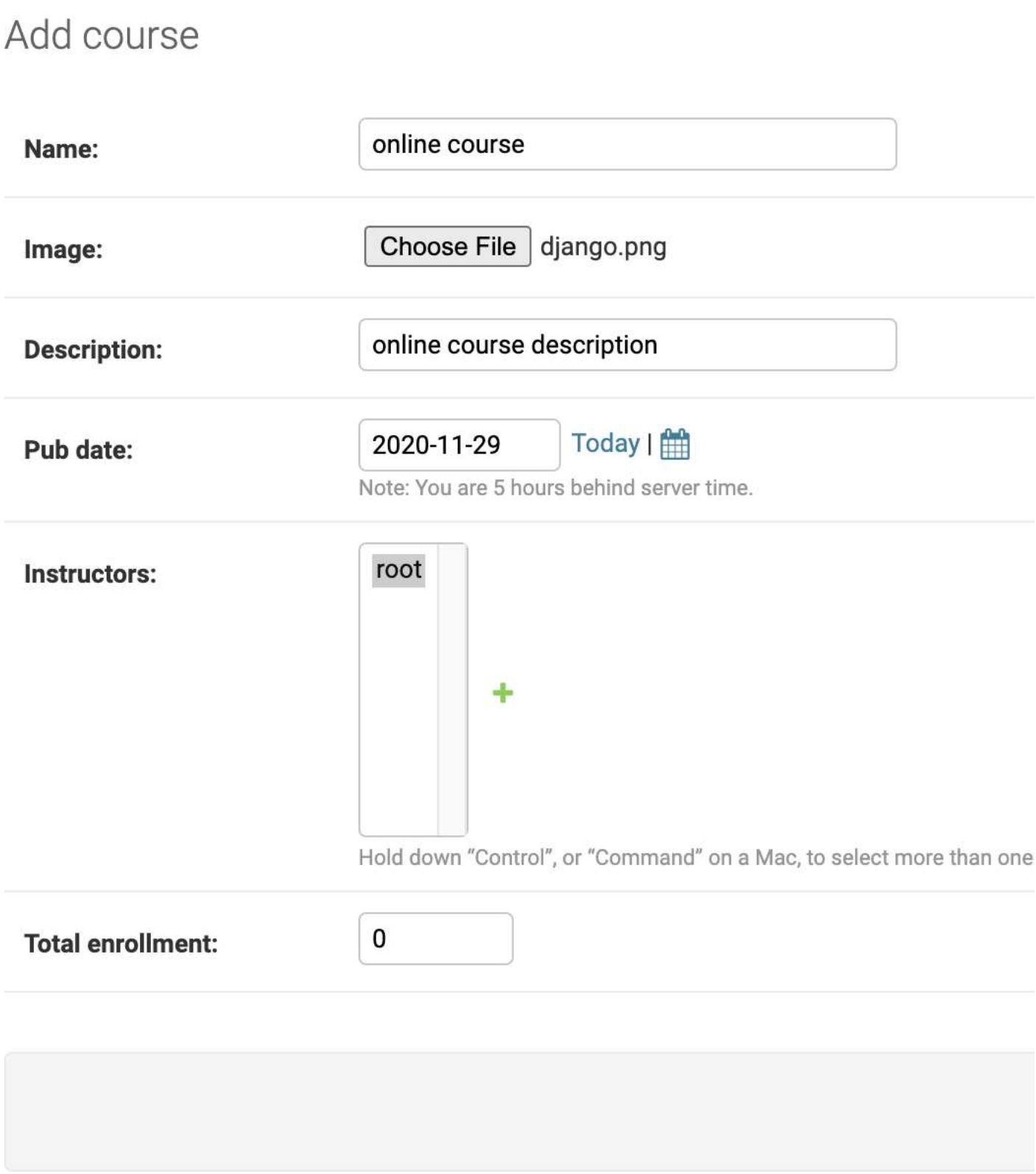
Let's create an instructor first. For simplification, we can assign the super user to be an instructor.

- Click the green `Add` button in the `Instructors` row
- For the `User` field, choose the super user you just created:



For fields in `Instructor` model, Django automatically creates corresponding web widgets for us to receive their values. For example, a checkbox is created for `Full Time` field and numeric input field for `Total learners`.

- Then choose if the instructor is `Full Time` or not and enter `0` for `Total learners`.
- Similarly, you can create a course by entering its `Name`, `Description`, `Publication date`, etc.
- For the `Image` field, we defined an `ImageField` in `Course` model, and then Django admin site app automatically adds image uploading functionality for us.



Now you should have an `Instructor` and `Course` created.

- We leave objects `delete` and `edit` for you to explore as practice. You could also play with the admin site yourself to create more instructors and courses.

Next, let's customize our admin site.

Customize Admin Site

You may only want to include some of the model fields in the admin site.

To select the fields to be included, we create a `ModelAdmin` class and add a fields list with the field names to be included.

- Open `adminsite/admin.py`, add a `CourseAdmin` class:

```
1. 1
2. 2
```

```
1. class CourseAdmin(admin.ModelAdmin):
2.     fields = ['pub_date', 'name', 'description']
```

Copied!

- Update previous course registration `admin.site.register(Course)` with the `CourseAdmin` class:

```
1. 1
2. 2
```

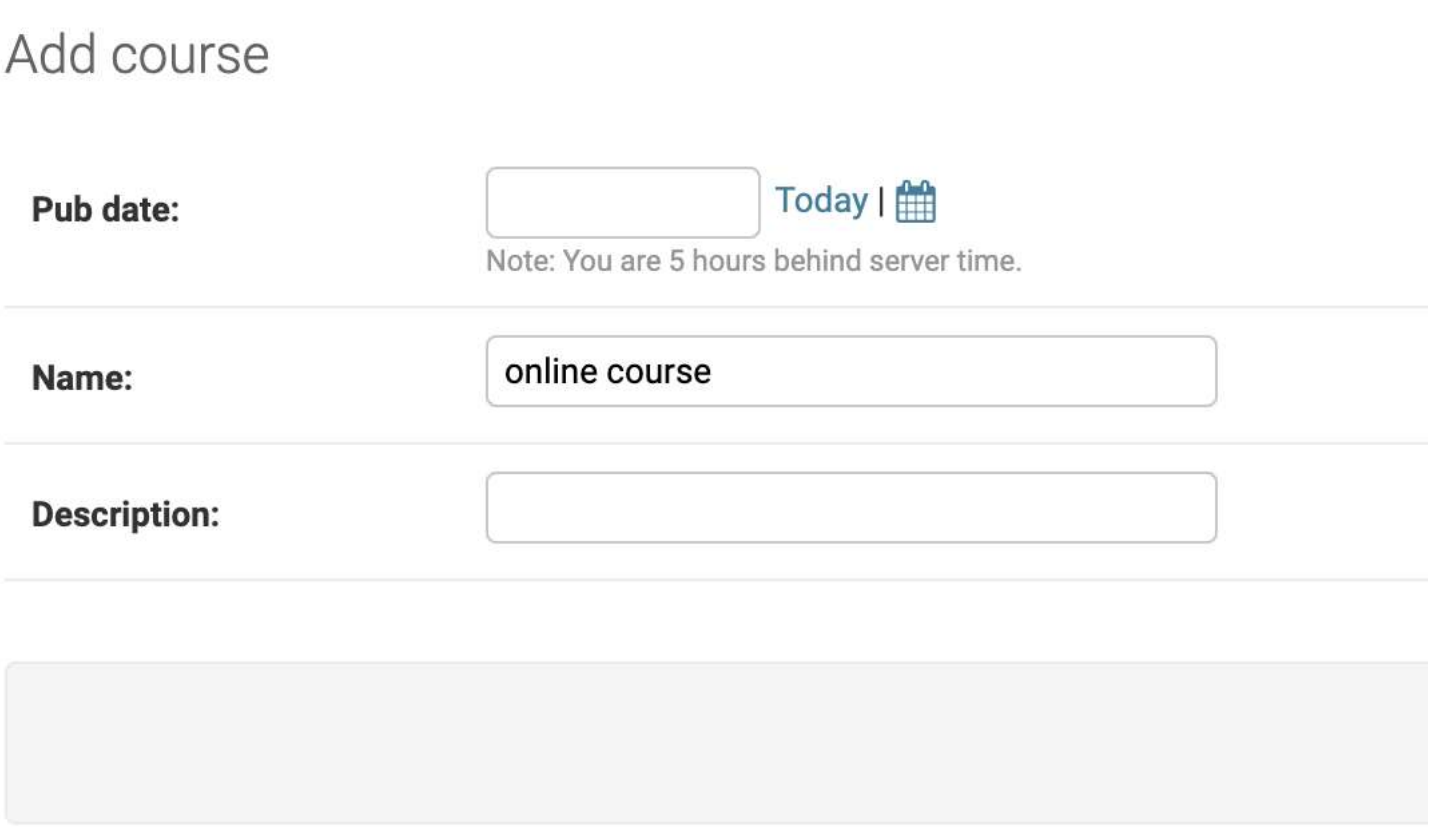
```
1. 
2. admin.site.register(Course, CourseAdmin)
```

Copied!

Now your `adminsite/admin.py` should look like the following:

```
1  from django.contrib import admin
2  from .models import Course, Instructor
3
4  # Register your models here.
5  class CourseAdmin(admin.ModelAdmin):
6      fields = ['pub_date', 'name', 'description']
7  admin.site.register(Course, CourseAdmin)
8  admin.site.register(Instructor)
```

- Refresh the admin page and try adding a course again, and you should see only `Pub date`, `Name`, `Description` fields are included.



Coding Practice: Customize Fields for Instructor Model

Include only `user` and `Full Time` fields for `Instructor` model.

```
1
2
3
4
```

```
class InstructorAdmin(admin.ModelAdmin):
    fields = #<HINT> add user, full_time field names here#

admin.site.register(Instructor, InstructorAdmin)
```

Copied!

▶ Click here to see solution

Associate Related Models

In previous coding practice, you were asked to include `user` model, which acts as a parent model for `instructor`, and you can add such related object while creating an `Instructor` object.

Django admin provides a convenient way to associate related objects on a single model managing page. This can be done by defining `Inline` classes.

Let's try to manage `Lesson` model together with `Course` model on `Course` admin page.

- Open `adminsite/admin.py`, add a `LessonInline` class before `CourseAdmin`:

```
1
2
3
```

```
class LessonInline(admin.StackedInline):
    model = Lesson
    extra = 5
```

Copied!

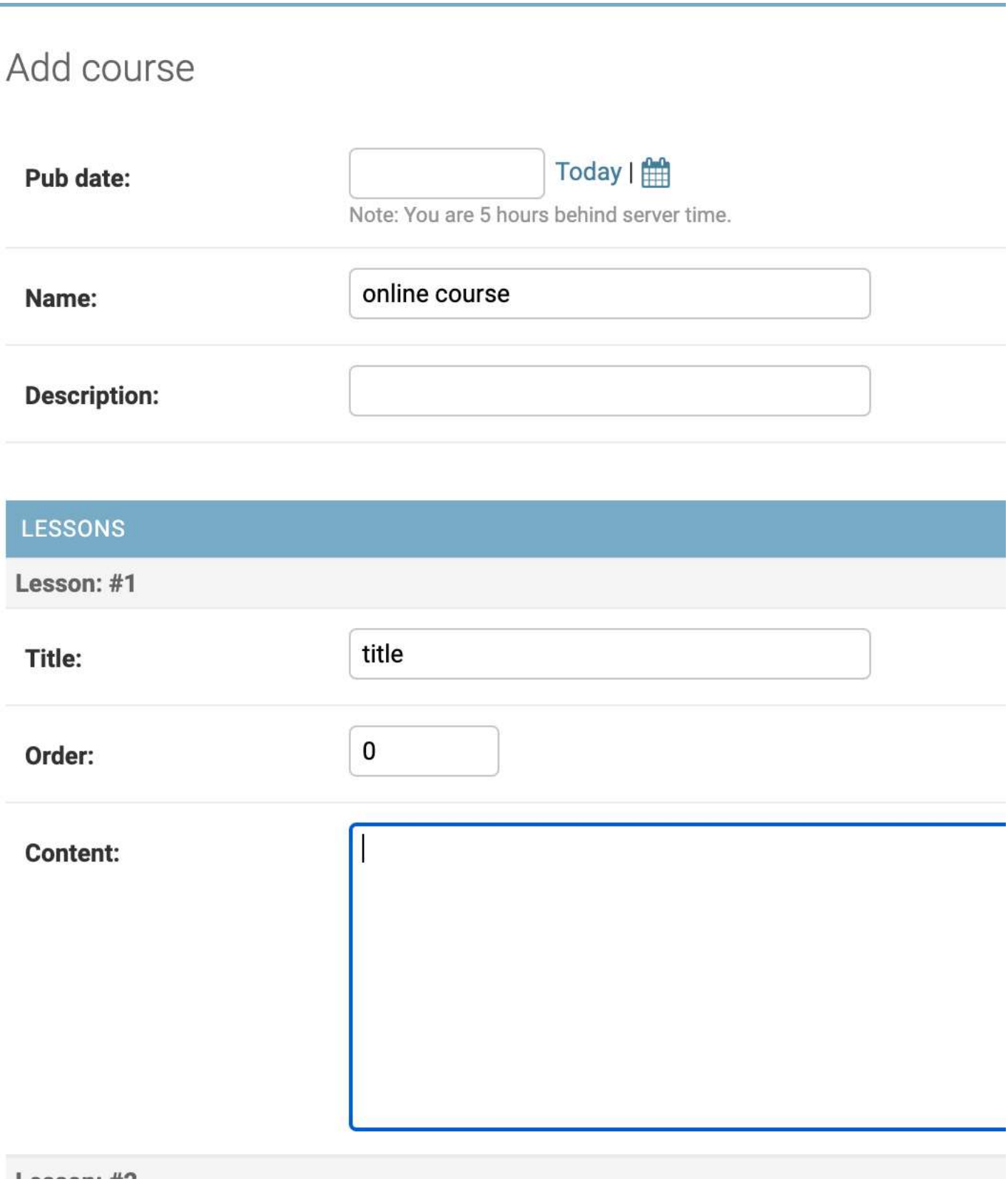
- And update `CourseAdmin` class by adding a `inlines` list

```
1
2
3
```

```
1. class CourseAdmin(admin.ModelAdmin):
2.     fields = ['pub_date', 'name', 'description']
3.     inlines = [LessonInline]
```

Copied!

Now you can see by adding the `inlines` list, you can adding lessons while creating a course



Summary

In this lab, you have learned to use the Django admin site for managing the models defined in the `models.py` file of your Django app, to customize the admin site to include the subset of model fields, and to create `Inline` classes for adding related objects on the same page.

Author(s)

Yan Luo