Python Assignment - 10

Aim: Creating web application using Django web framework

Theory:

Django is a high-level Python web framework designed to help developers build web applications quickly and efficiently. It follows the Model-View-Template (MVT) architectural pattern, which separates the application's data, user interface, and control logic into separate components.

Some key features of Django include:

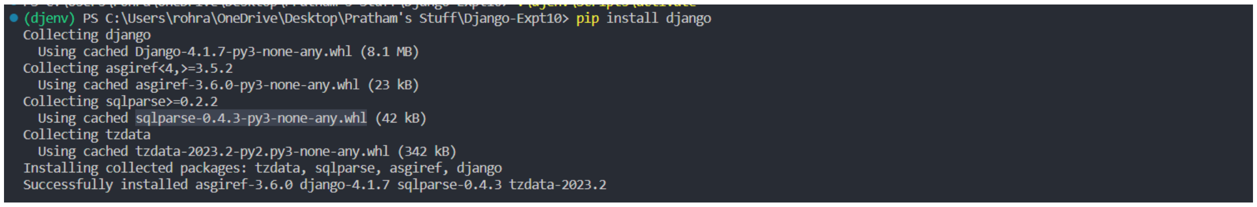
1. Object-relational mapping (ORM): Django provides a high-level ORM that allows developers to interact with the database using Python classes instead of SQL queries. This makes it easier to manage and manipulate data in the application.
2. Built-in admin interface: Django comes with a built-in admin interface that allows developers to easily manage and update the application's data models.
3. URL routing: Django provides a powerful URL routing system that allows developers to map URLs to specific views, making it easy to create complex web applications with many different pages.
4. Template engine: Django's template engine makes it easy to create dynamic HTML pages that can be customized based on user input or other data.
5. Security: Django provides many built-in security features, such as cross-site scripting (XSS) protection, cross-site request forgery (CSRF) protection, and SQL injection prevention.

Overall, Django is a powerful web framework that can help developers build complex web applications quickly and efficiently. Its focus on simplicity, ease of use, and security has made it one of the most popular web frameworks available today.

Installing Django

To install django, type the following command in the terminal:

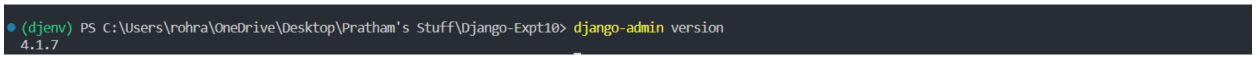
```
(env) $ pip install django
```



```
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10> pip install django
 Collecting django
   Using cached Django-4.1.7-py3-none-any.whl (8.1 MB)
 Collecting asgiref<4,>=3.5.2
   Using cached asgiref-3.6.0-py3-none-any.whl (23 kB)
 Collecting sqlparse>=0.2.2
   Using cached sqlparse-0.4.3-py3-none-any.whl (42 kB)
 Collecting tzdata
   Using cached tzdata-2023.2-py2.py3-none-any.whl (342 kB)
 Installing collected packages: tzdata, sqlparse, asgiref, django
 Successfully installed asgiref-3.6.0 django-4.1.7 sqlparse-0.4.3 tzdata-2023.2
```

To confirm the installation type -

```
(env) $ django-admin version
```



```
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10> django-admin version
4.1.7
```

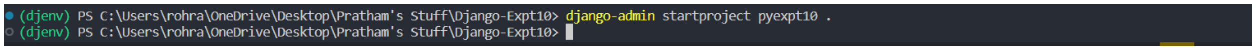
Starting project

After you've successfully installed Django, you're ready to create the scaffolding for your new web application. The Django framework distinguishes between **projects** and **apps**:

- **A Django project** is a high-level unit of organization that contains logic that governs your whole web application. Each project can contain multiple apps.
- **A Django app** is a lower-level unit of your web application. You can have zero to many apps in a project, and you'll usually have at least one app.

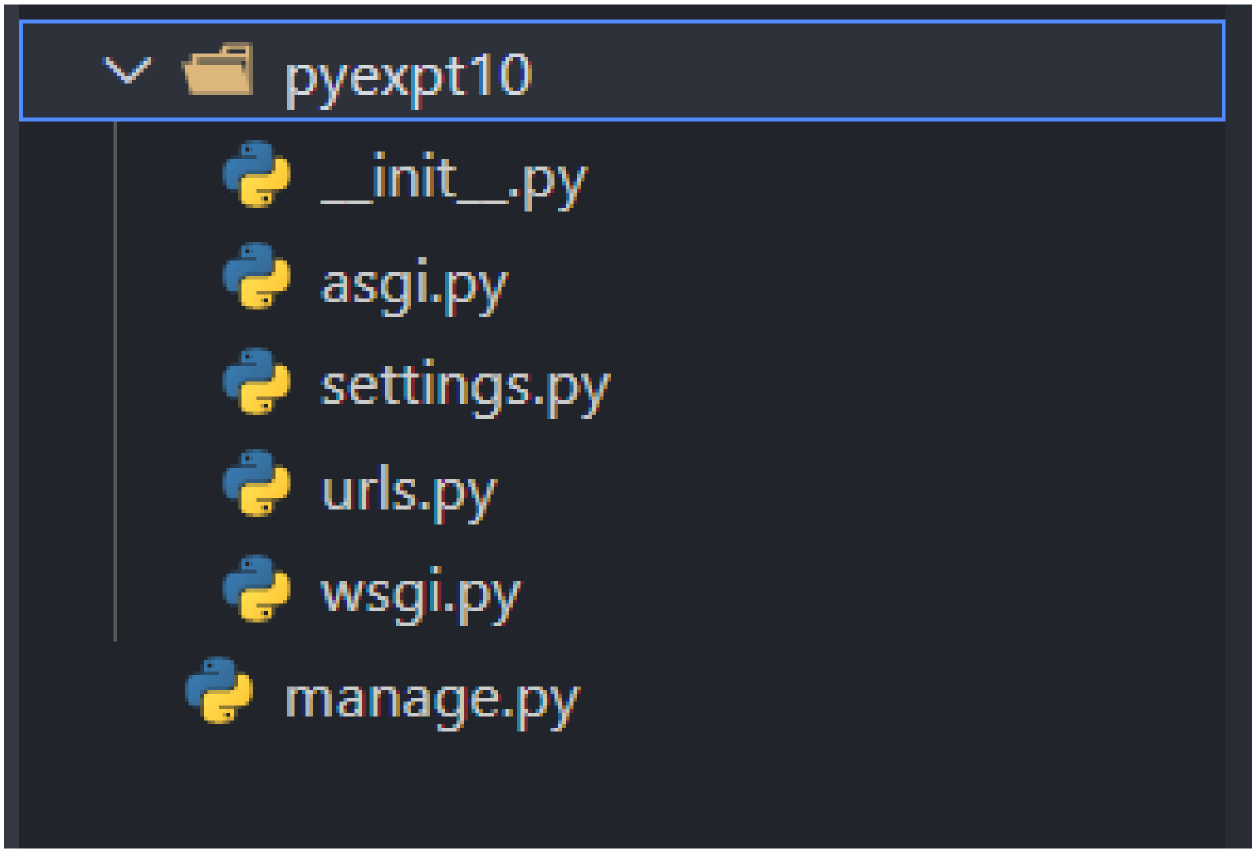
Creating a project -

```
(env) $ django-admin startproject <projectName>
```



```
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10> django-admin startproject pyexpt10 .
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10>
```

Running this command creates a default folder structure, which includes some Python files and your management app that has the same name as your project:


```
pyexpt10
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
manage.py
```

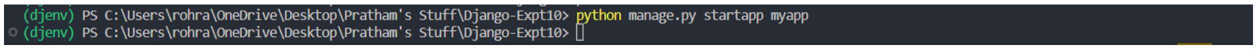
`manage.py` is a Python file that serves as the command center of your project. It does the same as the `django-admin` command-line utility.

Creating an App

Every project you build with Django can contain multiple Django apps. When you ran the `startproject` command in the previous section, you created a management app that you'll need for every default project that you'll build.

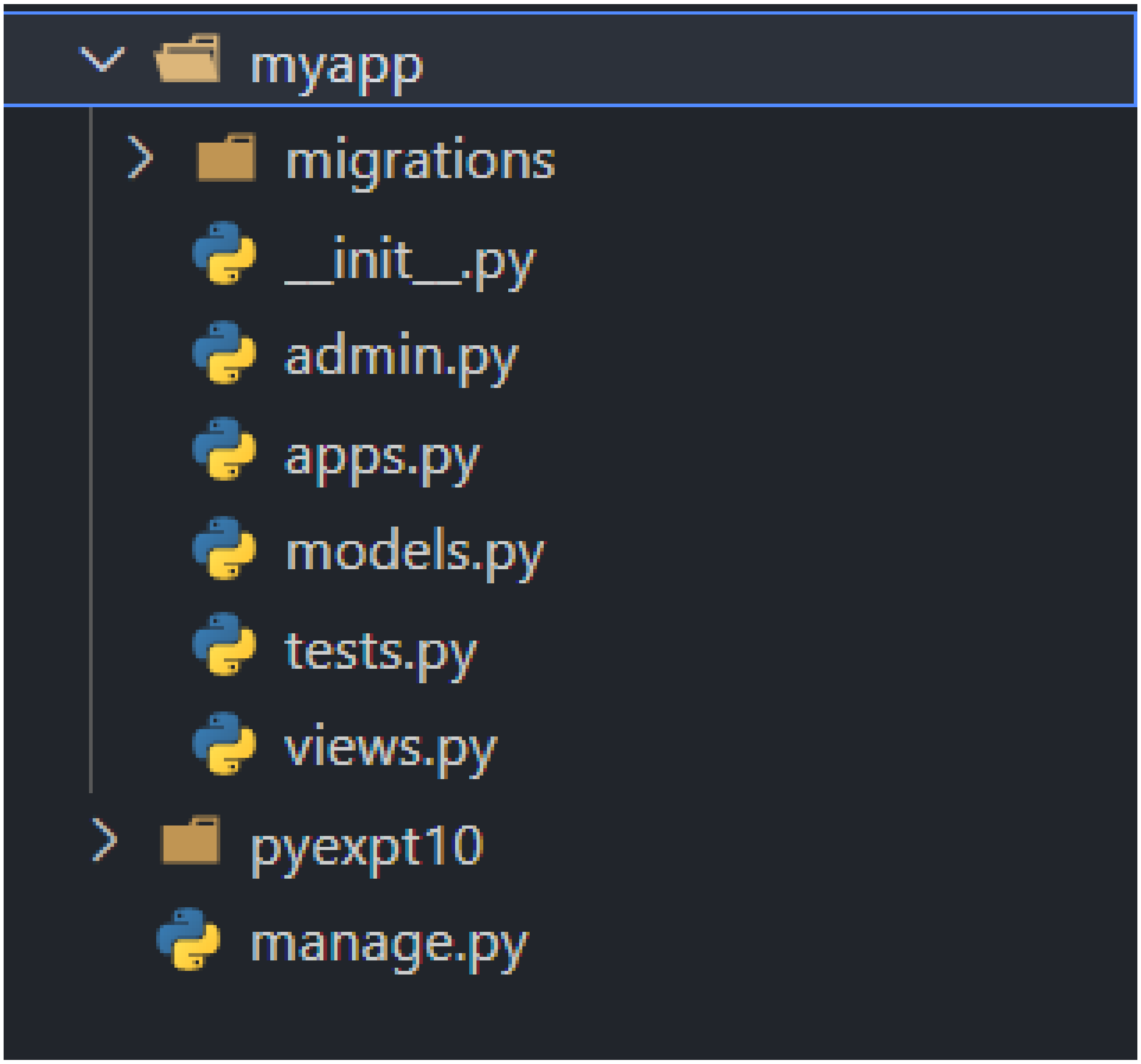
The `startapp` command generates a default folder structure for a Django app. This tutorial uses `example` as the name for the app:

```
(env) $ python manage.py startapp <appname>
```



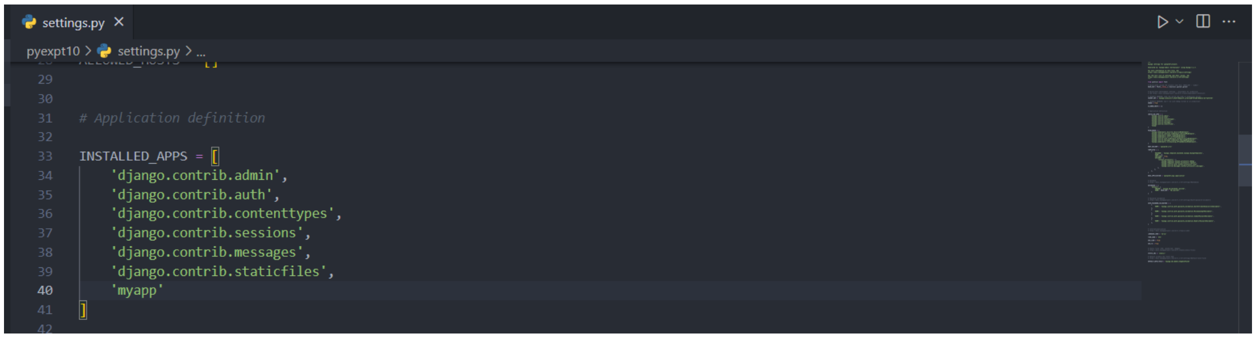
```
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10> python manage.py startapp myapp
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10>
```

Once the `startapp` command has finished execution, you'll see that Django has added another folder to your folder structure:



```
myapp
  migrations
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  views.py
pyexpt10
manage.py
```

To consider the app in your project you need to specify your project name in INSTALLED_APPS list as follows in settings.py:


settings.py

pyexpt10 > settings.py > ...

```python


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp'
]
```

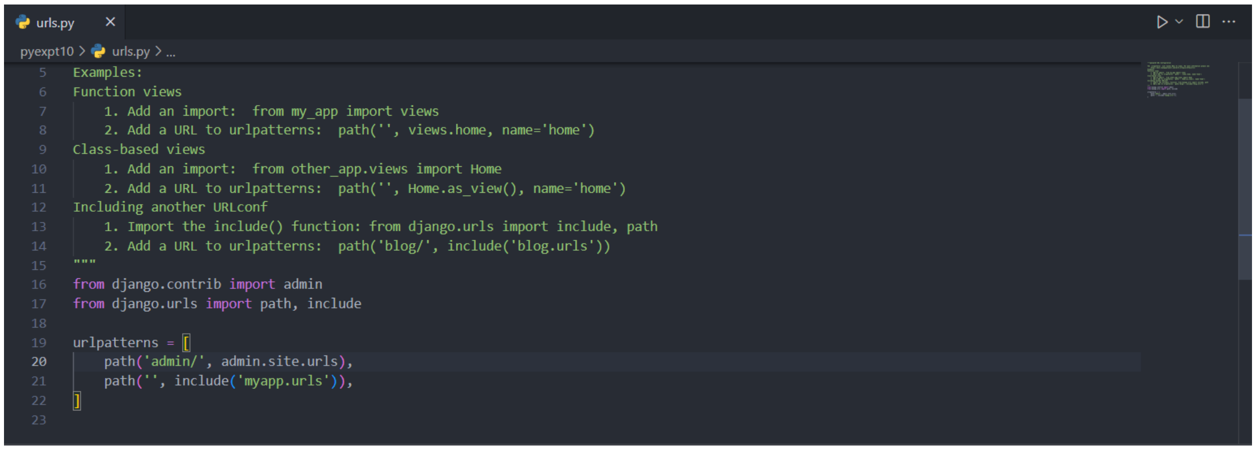
URLs

Django also provides a way to navigate around the different pages in a website.

When a user requests a URL, Django decides which *view* it will send it to.

This is done in a file called `urls.py`.

Now in the list of URL patterns, you need to specify the app name for including your app URLs.

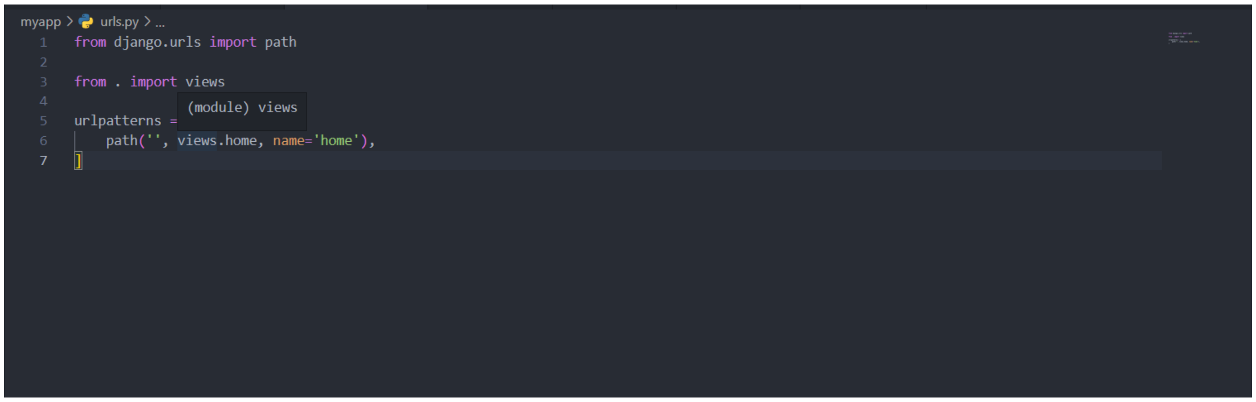

urls.py

pyexpt10 > urls.py > ...

```python
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
]
```

In the app’s url.py file

The below code will call or invoke the function which is defined in the views.py file so that it can be seen properly in the Web browser.

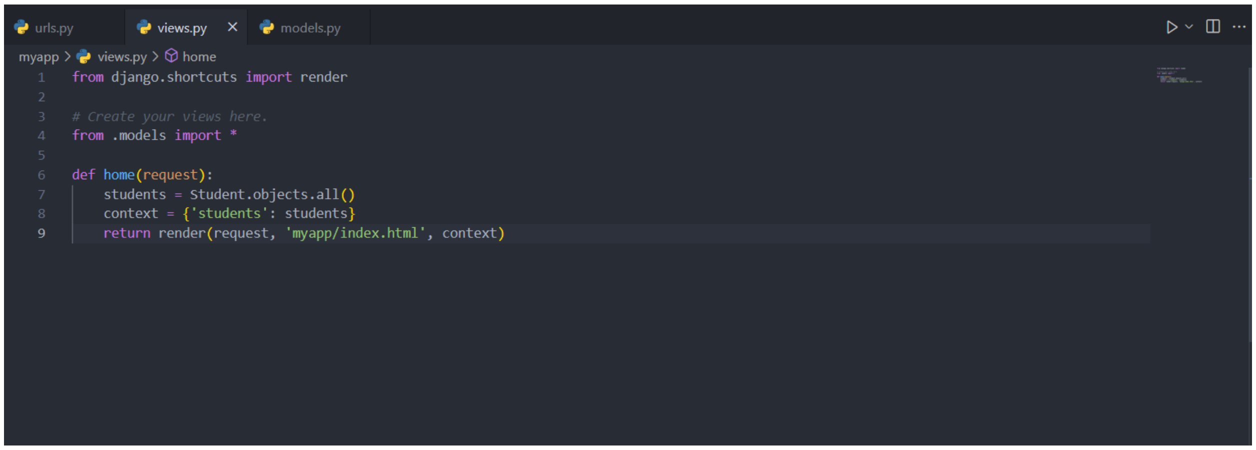


```python
from django.urls import path

from . import views

urlpatterns = [
    path('', views.home, name='home'),
]
```

Views

A view is a function or method that takes http requests as arguments, imports the relevant model(s), and finds out what data to send to the template, and returns the final result.

The views are usually located in a file called `views.py`.



```python
from django.shortcuts import render

# Create your views here.
from .models import *

def home(request):
    students = Student.objects.all()
    context = {'students': students}
    return render(request, 'myapp/index.html', context)
```

Models

The model provides data from the database.

In Django, the data is delivered as an Object Relational Mapping (ORM), which is a technique designed to make it easier to work with databases.

The most common way to extract data from a database is SQL. One problem with SQL is that you have to have a pretty good understanding of the database structure to be able to work with it.

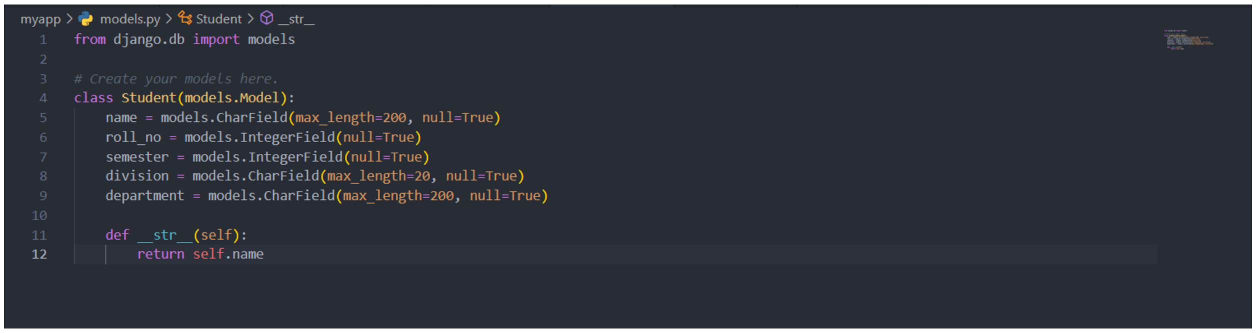
Django, with ORM, makes it easier to communicate with the database, without having to write complex SQL statements.

The models are usually located in a file called `models.py`.

Creating Models - Syntax

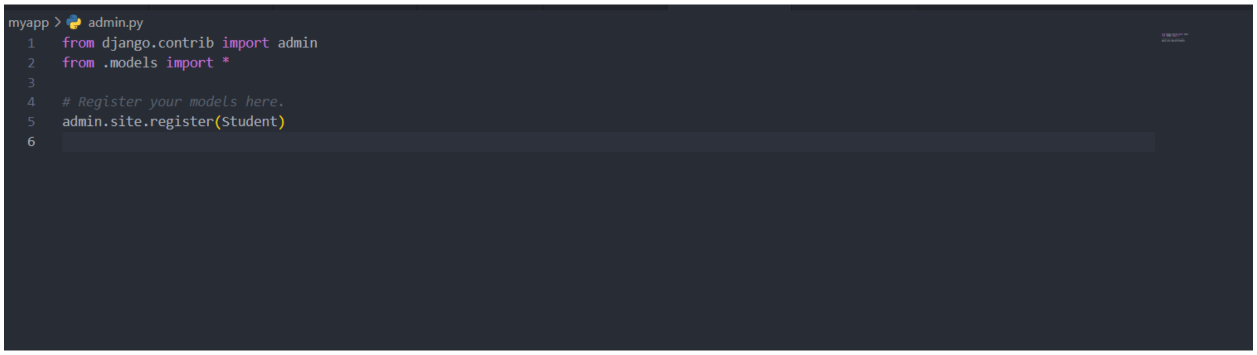
```
from django.db import models

class ModelName(models.Model):
        field_name = models.Field(**options)
```

We are creating a model called students which will store student's name, roll number, etc and we will display it with the help of templates and the data will be added via admin interface.



```
from django.db import models

# Create your models here.
class Student(models.Model):
    name = models.CharField(max_length=200, null=True)
    roll_no = models.IntegerField(null=True)
    semester = models.IntegerField(null=True)
    division = models.CharField(max_length=20, null=True)
    department = models.CharField(max_length=200, null=True)

    def __str__(self):
        return self.name
```

To render a model in Django admin, we need to modify `app/admin.py`. Go to admin.py in geeks app and enter the following code. Import the corresponding model from models.py and register it to the admin interface.



myapp > admin.py

```python
from django.contrib import admin
from .models import *

# Register your models here.
admin.site.register(Student)
```

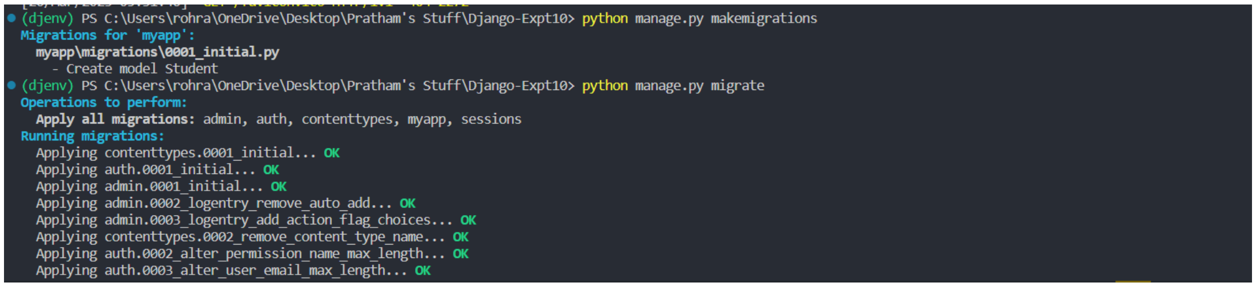
Whenever we create a Model, Delete a Model, or update anything in any of models.py of our project. We need to run two commands makemigrations and migrate. makemigrations basically generates the SQL commands for preinstalled apps (which can be viewed in installed apps in settings.py) and your newly created app's model which you add in installed apps whereas migrate executes those SQL commands in the database file. So when we run,

```
python manage.py makemigrations
```

SQL Query to create above Model as a Table is created and

```
 python manage.py migrate
```

creates the table in the database.



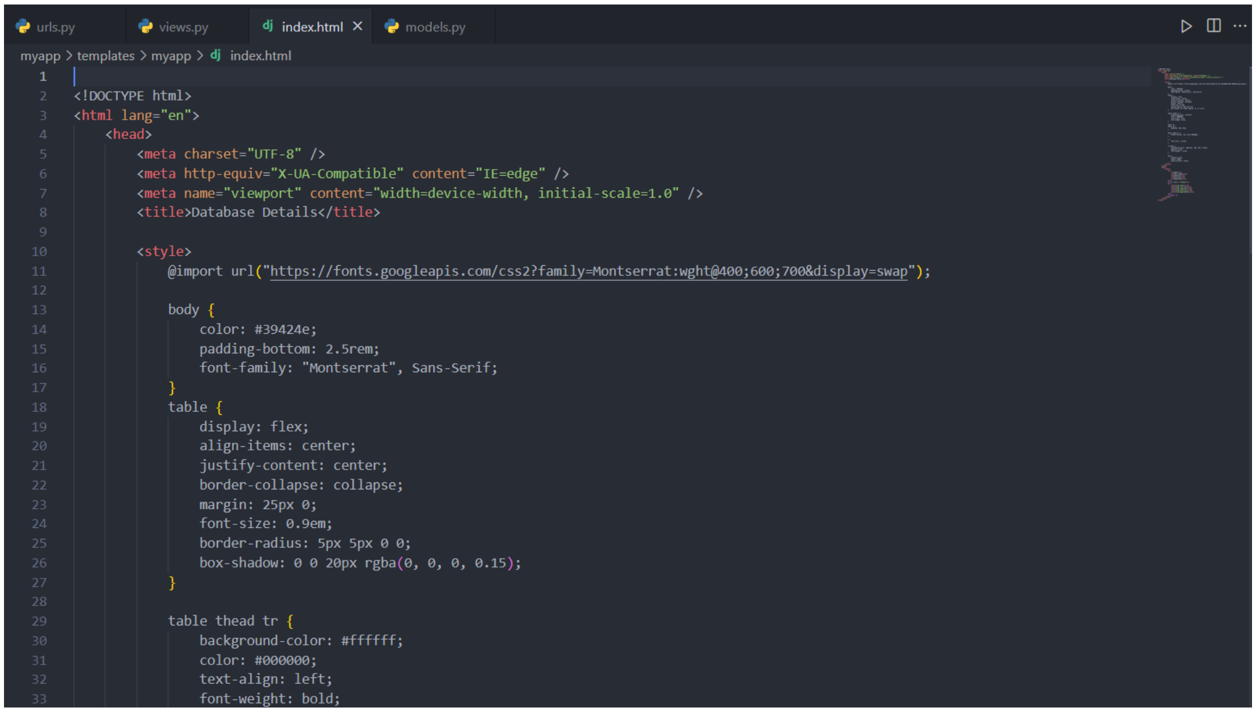
```
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10> python manage.py makemigrations
Migrations for 'myapp':
  myapp\migrations\0001_initial.py
    - Create model Student
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, myapp, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
```

Templates

A template in Django is basically written in HTML, CSS, and Javascript in a .html file. Django framework efficiently handles and generates dynamically HTML web pages that are visible to the end-user. Django mainly functions with a backend so, in order to provide a frontend and provide a layout to our website, we use templates.

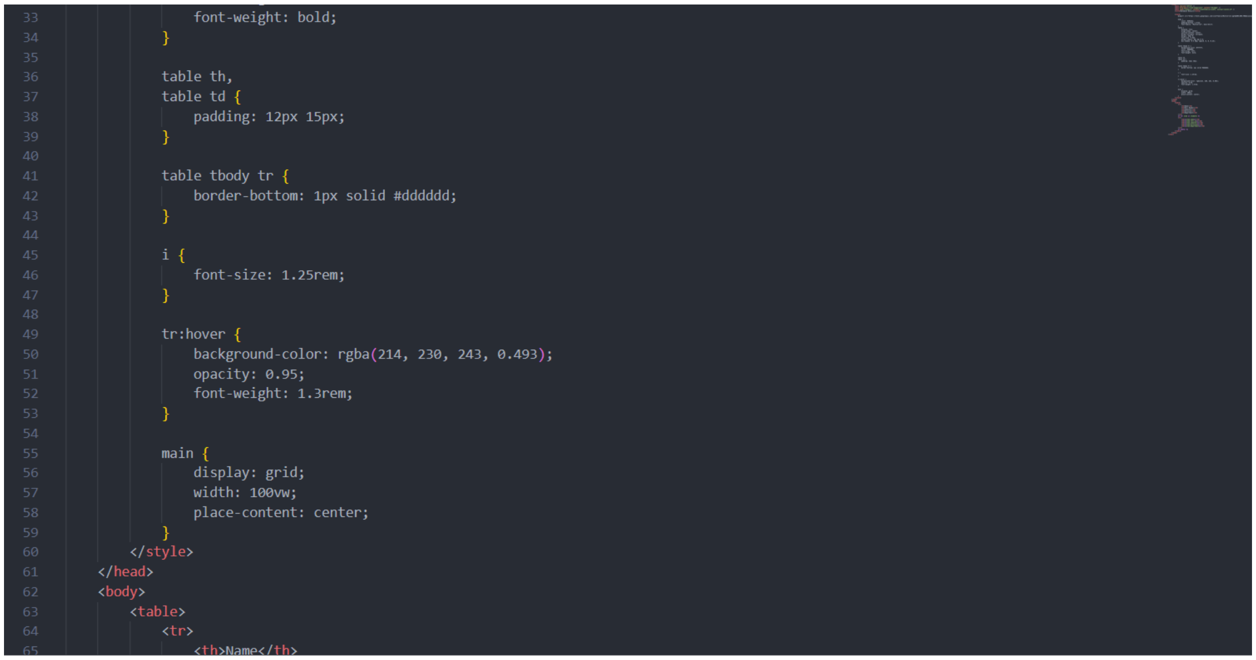
Using Django Templates

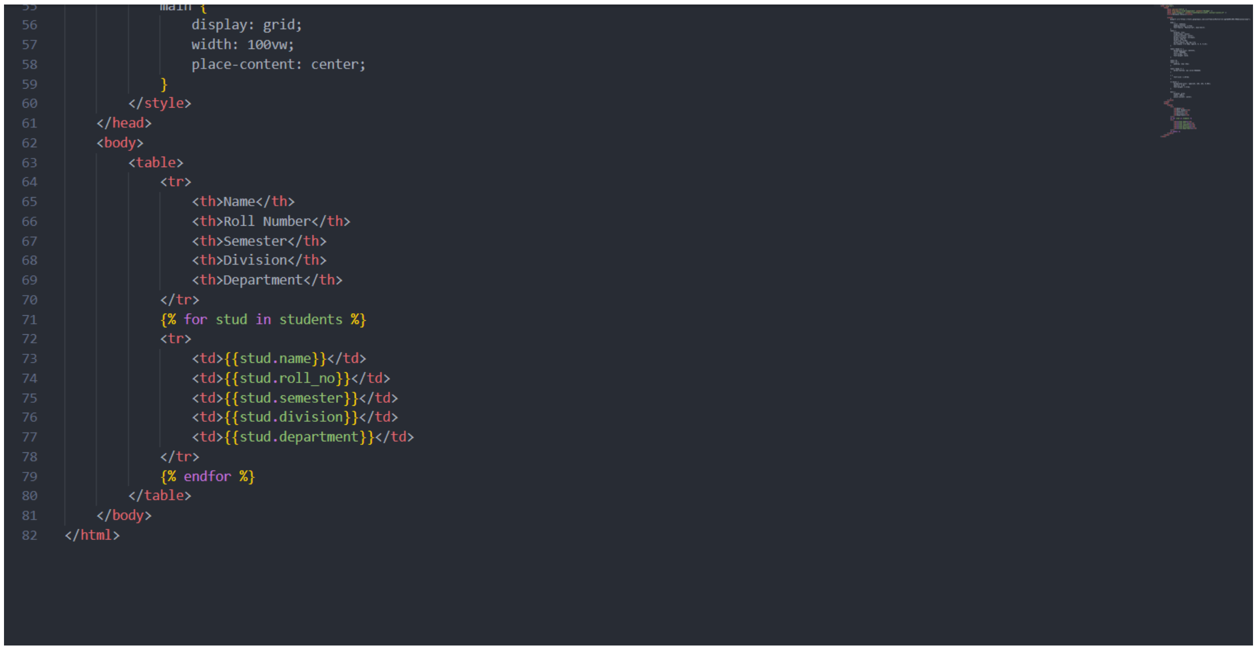
Illustration of How to use templates in Django using an Example Project. Templates not only show static data but also the data from different databases connected to the application through a context dictionary.


urls.py views.py index.html models.py

myapp > templates > myapp > index.html

```html

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Database Details</title>

        <style>
            @import url("https://fonts.googleapis.com/css2?family=Montserrat:wght@400;600;700&display=swap");

            body {
                color: #39424e;
                padding-bottom: 2.5rem;
                font-family: "Montserrat", Sans-Serif;
            }
            table {
                display: flex;
                align-items: center;
                justify-content: center;
                border-collapse: collapse;
                margin: 25px 0;
                font-size: 0.9em;
                border-radius: 5px 5px 0 0;
                box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
            }

            table thead tr {
                background-color: #ffffff;
                color: #000000;
                text-align: left;
                font-weight: bold;
```

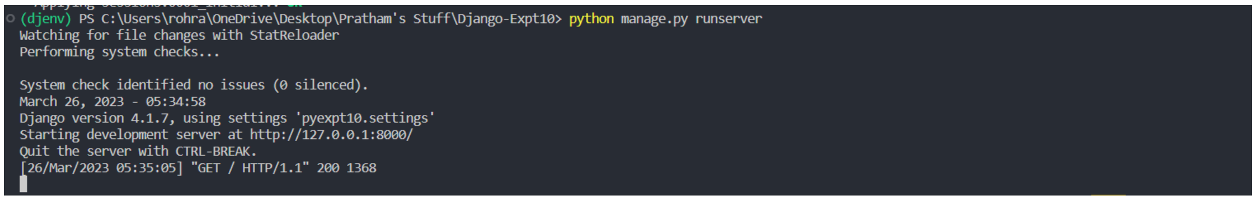

```html
                font-weight: bold;
            }

            table th,
            table td {
                padding: 12px 15px;
            }

            table tbody tr {
                border-bottom: 1px solid #dddddd;
            }

            i {
                font-size: 1.25rem;
            }

            tr:hover {
                background-color: rgba(214, 230, 243, 0.493);
                opacity: 0.95;
                font-weight: 1.3rem;
            }

            main {
                display: grid;
                width: 100vw;
                place-content: center;
            }
        </style>
    </head>
    <body>
        <table>
            <tr>
                <th>Name</th>
```



```
            main {
                display: grid;
                width: 100vw;
                place-content: center;
            }
        </style>
    </head>
    <body>
        <table>
            <tr>
                <th>Name</th>
                <th>Roll Number</th>
                <th>Semester</th>
                <th>Division</th>
                <th>Department</th>
            </tr>
            {% for stud in students %}
            <tr>
                <td>{{stud.name}}</td>
                <td>{{stud.roll_no}}</td>
                <td>{{stud.semester}}</td>
                <td>{{stud.division}}</td>
                <td>{{stud.department}}</td>
            </tr>
            {% endfor %}
        </table>
    </body>
</html>
```

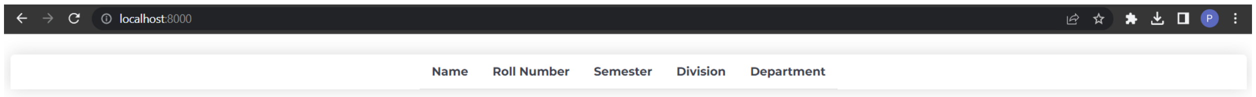
Starting the server

Now, we can finally run our server with the following command



```
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 26, 2023 - 05:34:58
Django version 4.1.7, using settings 'pyexpt10.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[26/Mar/2023 05:35:05] "GET / HTTP/1.1" 200 1368
```

We can see our django server displays the template and views logic on our web server.



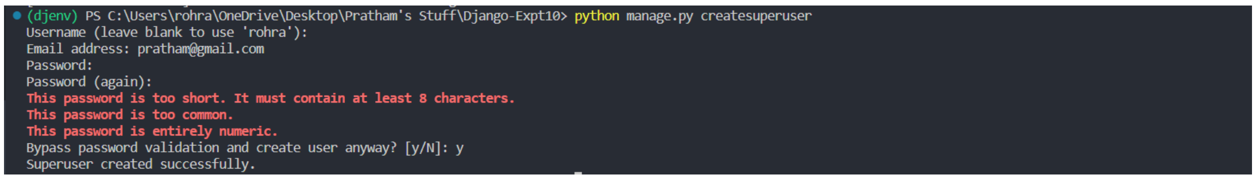
localhost:8000

Name	Roll Number	Semester	Division	Department

Admin Interface

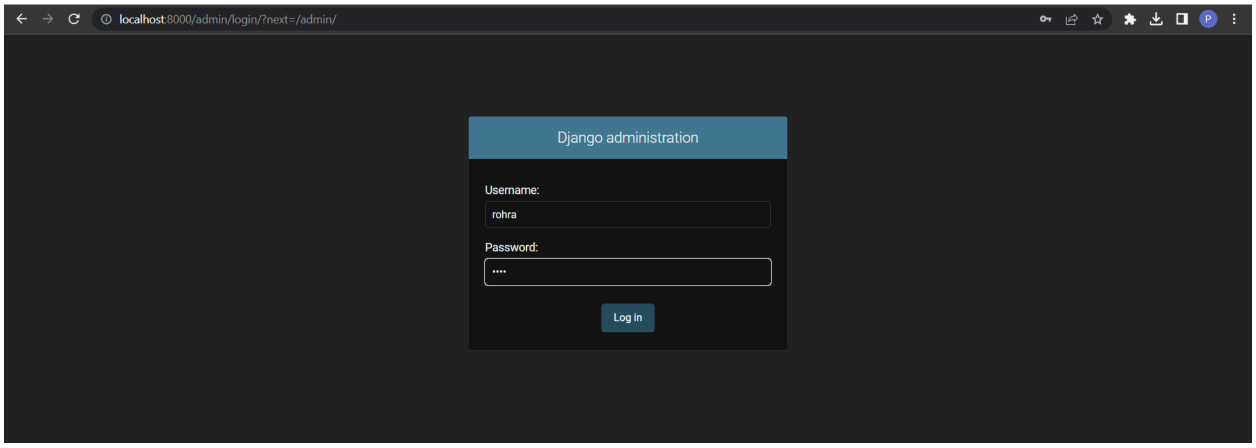
To update the data in the database via admin interface, we need to create a super user so that we can login to the admin interface.

Run the following command and add credentials for the super user.

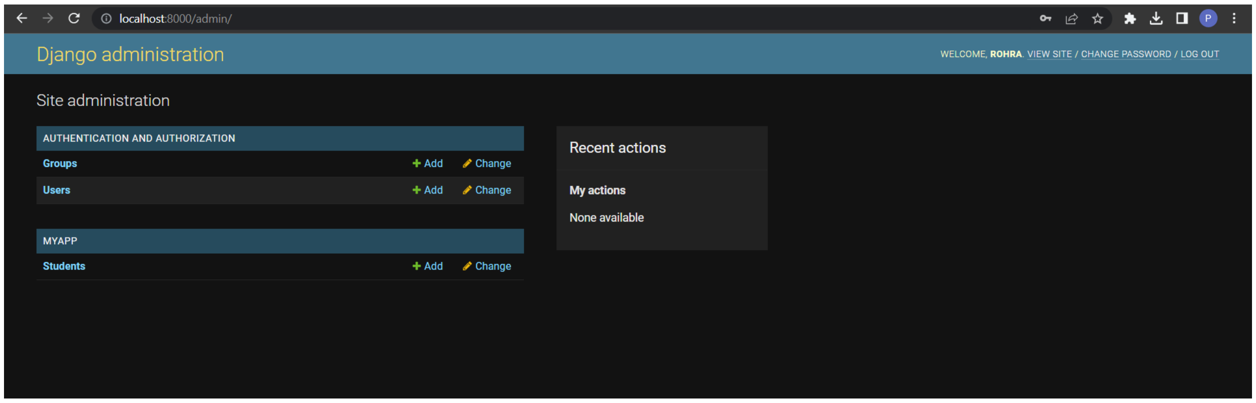


```
(djenv) PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Django-Expt10> python manage.py createsuperuser
Username (leave blank to use 'rohra'):
Email address: pratham@gmail.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

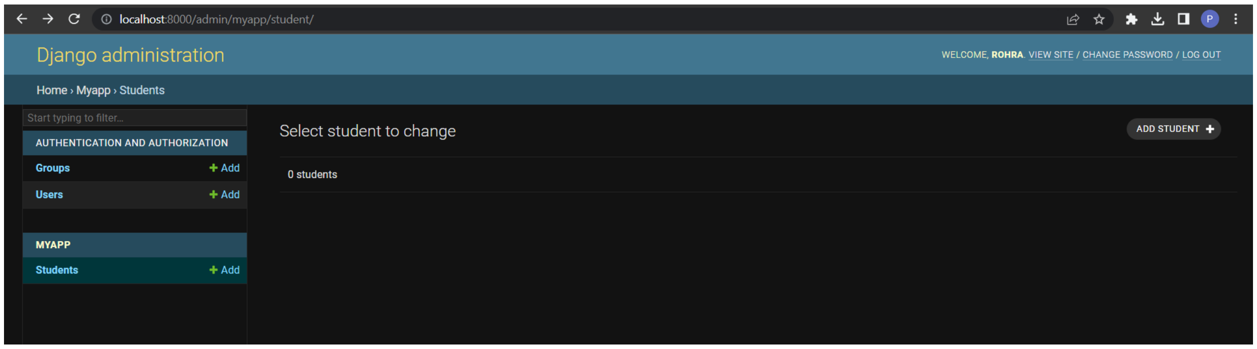
Go to `/admin` on your web browser and enter the credentials of super user and press **Log in**.



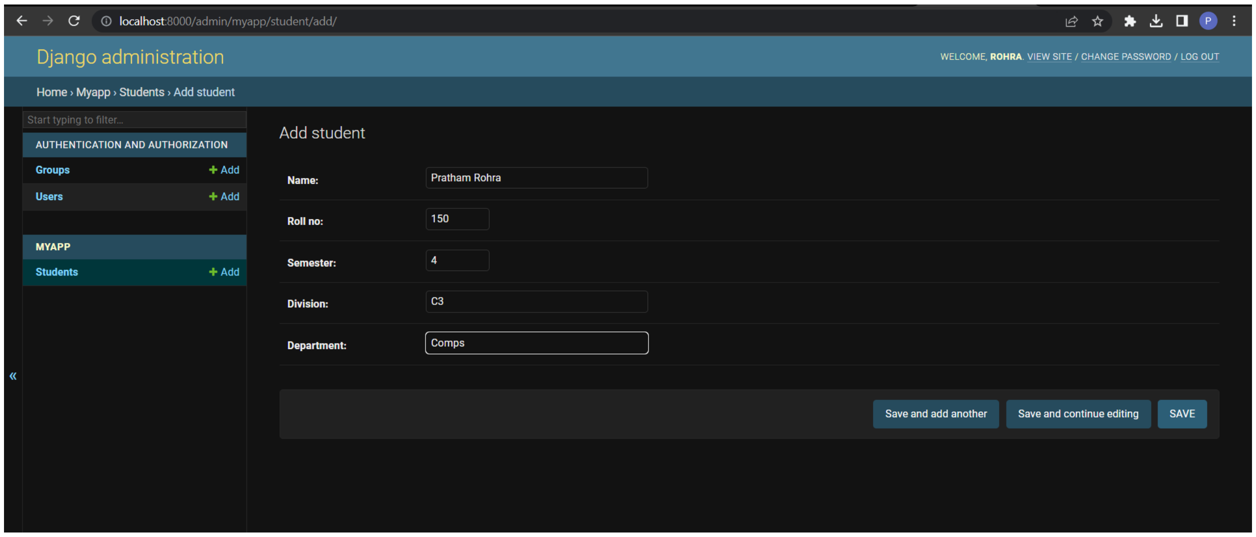
After logging in, you will see the admin interface. You can see that our “Students” model is listed here. Click on **+Add** right next to it.



Now, click on **Add Student+** option in the top right corner.


localhost:8000/admin/myapp/student/
Django administration
WELCOME, ROHRA. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Myapp › Students
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
MYAPP
Students + Add
Select student to change
ADD STUDENT +
0 students

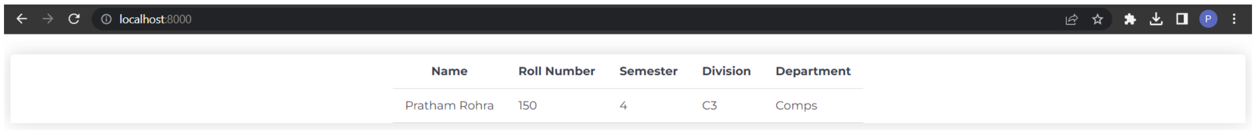
You may see the form with the options that you added in your model-


localhost:8000/admin/myapp/student/add/
Django administration
WELCOME, ROHRA. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Myapp › Students › Add student
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
MYAPP
Students + Add
Add student
Save and add another
Save and continue editing
SAVE

Name:	Pratham Rohra
Roll no:	150
Semester:	4
Division:	C3
Department:	Comps

Enter the details and click on **SAVE**.

Now go to home/index route and you can see our data is displayed -



Name	Roll Number	Semester	Division	Department
Pratham Rohra	150	4	C3	Comps