

Teaching Learning Plan

Course Code:	21CS62	CIE Marks	50
Teaching Hours/Week	3:0:2:0	SEE Marks	50
Total No. of Hours	40 T + 20 P	Total Marks	100
Credits	04	Exam Hours	03

Course Objectives: The student should be made to:

- CLO 1. Explain the use of learning full stack web development.
- CLO 2. Make use of rapid application development in the design of responsive web pages.
- CLO 3. Illustrate Models, Views and Templates with their connectivity in Django for full stack web development.
- CLO 4. Demonstrate the use of state management and admin interfaces automation in Django.
- CLO 5. Design and implement Django apps containing dynamic pages with SQL databases.

Teaching-Learning Process (General Instructions)

These are sample Strategies, which teachers can use to accelerate the attainment of the various course outcomes.

1. Lecturer method (L) need not to be only a traditional lecture method, but alternative effective teaching methods could be adopted to attain the outcomes.
2. Use of Video/Animation to explain functioning of various concepts.
3. Encourage collaborative (Group Learning) Learning in the class.
4. Ask at least three HOT (Higher order Thinking) questions in the class, which promotes critical thinking.
5. Adopt Problem Based Learning (PBL), which fosters students' Analytical skills, develop design thinking skills such as the ability to design, evaluate, generalize, and analyze information rather than simply recall it.
6. Introduce Topics in manifold representations.
7. Show the different ways to solve the same problem with different logic and encourage the students to come up with their own creative ways to solve them.
8. Discuss how every concept can be applied to the real world - and when that's possible, it helps improve the students' understanding.

Course Outcomes (Course Skill Set) At the end of the course the student will be able to:

- CO 1. Understand the working of MVT based full stack web development with Django.
- CO 2. Designing of Models and Forms for rapid development of web pages.
- CO 3. Analyze the role of Template Inheritance and Generic views for developing full stack web applications.
- CO 4. Apply the Django framework libraries to render nonHTML contents like CSV and PDF.
- CO 5. Perform jQuery based AJAX integration to Django Apps to build responsive full stack web applications,

CONTENT

Exp.No.	List of Programs
Part A (Angular JS)	
	PYTHON SETUP AND DJANGO INSTALLATION
1.	Develop a Django app that displays current date and time in server
2.	Develop a Django app that displays date and time four hours ahead and four hours before as an offset of current date and time in the server.
3.	Develop a simple Django app that displays an unordered list of fruits and ordered list of selected students for an event
4.	Develop a layout.html with a suitable header (containing navigation menu) and footer with copyright and developer information. Inherit this layout.html and create 3 additional pages: contact us, About Us and Home page of any website
5.	Develop a Django app that performs student registration to a course. It should also display a list of students registered for any selected course. Create students and courses as models with enrolment as ManyToMany field.
6.	For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms.
7.	Develop a Model form for a student that contains his topic chosen for project, languages used and duration with a model called project.
8.	For students enrolment developed in Module 2, create a generic class view which displays list of students and detailview that displays student details for any selected student in the list.
9.	Develop an example Django app that performs CSV and PDF generation for any models created in previous laboratory components.

PYTHON SETUP AND DJANGO INSTALLATION

As first step we need to

1. install Python (latest Version)
2. Set up a virtual environment

```
C:\Users\VISHNU>pip install virtualenvwrapper-win
Collecting virtualenvwrapper-win
  Downloading virtualenvwrapper-win-1.2.7-py3-none-any.whl.metadata (10 kB)
Collecting virtualenv (from virtualenvwrapper-win)
  Downloading virtualenv-20.25.1-py3-none-any.whl.metadata (4.4 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv->virtualenvwrapper-win)
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Collecting filelock<4,>=3.12.2 (from virtualenv->virtualenvwrapper-win)
  Downloading filelock-3.13.3-py3-none-any.whl.metadata (2.8 kB)
Requirement already satisfied: platformdirs<5,>=3.9.1 in c:\users\vishnu\appdata\local\programs\python\python38\lib\site-packages (from virtualenv->virtualenvwrapper-win) (3.11.0)
Downloading virtualenvwrapper-win-1.2.7-py3-none-any.whl (18 kB)
Downloading virtualenv-20.25.1-py3-none-any.whl (3.8 MB)
2.8/3.8 MB 302.1 kB/s eta 0:00:04
```

3. Create and Environment

```
C:\Users\VISHNU>mkvirtualenv first
C:\Users\VISHNU\Envs is not a directory, creating
created virtual environment CPython3.8.0.final.0-64 in 7191ms
creator CPython3Windows(dest=C:\Users\VISHNU\Envs\first, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\VISHNU\AppData\Local\pypa\virtualenv)
added seed packages: pip==24.0, setuptools==69.1.0, wheel==0.42.0
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

(first) C:\Users\VISHNU>
```

4. Install Django

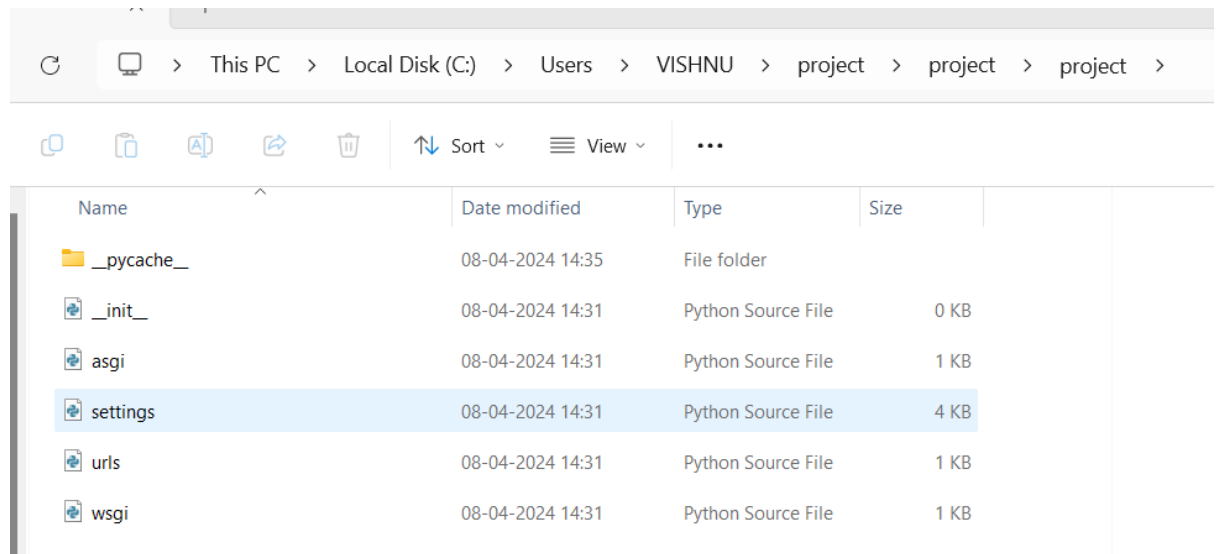
```
(first) C:\Users\VISHNU>pip install django
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'Read TimeoutError("HTTPSConnectionPool(host='pypi.org', port=443): Read timed out. (read timeout=15)')': /simple/django/
WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'Read TimeoutError("HTTPSConnectionPool(host='pypi.org', port=443): Read timed out. (read timeout=15)')': /simple/django/
WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'Read TimeoutError("HTTPSConnectionPool(host='pypi.org', port=443): Read timed out. (read timeout=15)')': /simple/django/
WARNING: Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'Read TimeoutError("HTTPSConnectionPool(host='pypi.org', port=443): Read timed out. (read timeout=15)')': /simple/django/
Collecting django
  Downloading Django-4.2.11-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.6.0 (from django)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Downloading sqlparse-0.4.4-py3-none-any.whl.metadata (4.0 kB)
Collecting backports.zoneinfo (from django)
  Downloading backports.zoneinfo-0.2.1-cp38-cp38-win_amd64.whl.metadata (4.7 kB)
Collecting tzdata (from django)
  Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting typing-extensions>=4 (from asgiref<4,>=3.6.0->django)
  Downloading typing_extensions-4.11.0-py3-none-any.whl.metadata (3.0 kB)
Downloading Django-4.2.11-py3-none-any.whl (8.0 MB)
8.0/8.0 MB 10.0 MB/s eta 0:00:00
Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
41.2/41.2 kB ? eta 0:00:00
```

```
(first) C:\Users\VISHNU>django-admin --version
4.2.11
```

```
(first) C:\Users\VISHNU>
```

```
(first) C:\Users\VISHNU\project>django-admin startproject project  
(first) C:\Users\VISHNU\project>
```

This above command will create a project folder. With some default file and another folder with the same name.

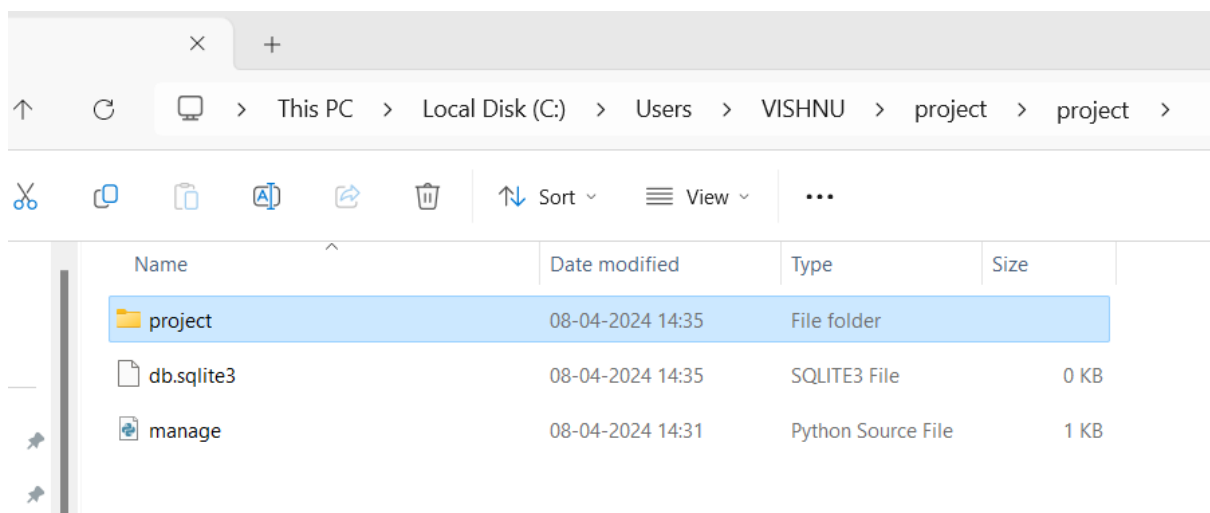


`_init_.py` - used to treat our folder (project) as a package

`wsgi.py` - web server gateway interface - to establish connection between b/w web application and the web server

`url` - used to handle the urls.

`manage.py` - to run the server



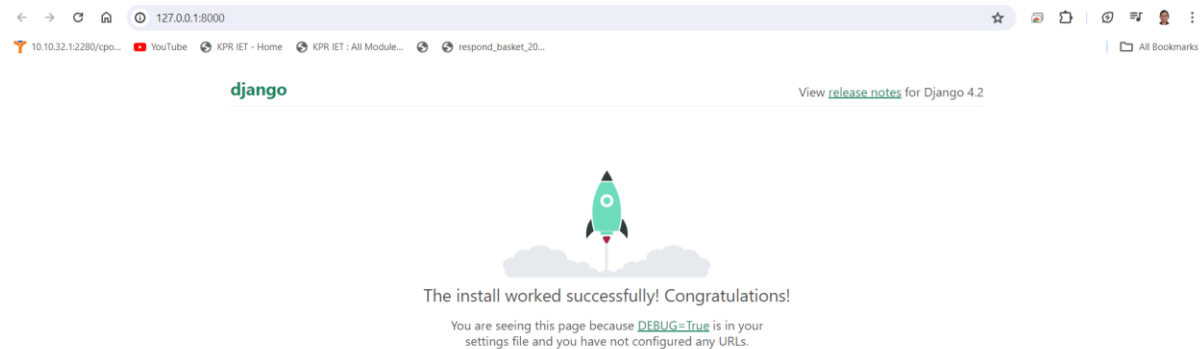
Use the below command to run the manage.py file

```
(first) C:\Users\VISHNU\project\project>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
April 08, 2024 - 14:35:16
Django version 4.2.11, using settings 'project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Click on the url and it will open the below page



PROGRAM 1

Develop a Django app that displays current date and time in server

views.py

```
from django.shortcuts import render
from django.http import HttpResponse
# Create your views here.

def dis_datetime(request):
    import datetime
    x = datetime.datetime.today()
    return HttpResponse(x)
```

Open urls.py in main project folder

```
from django.contrib import admin
from django.urls import path, include

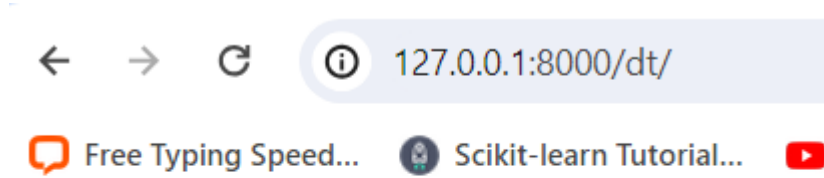
from datetimeapp import views as vd
from initial import views as vi

urlpatterns = [
    #path("",include('initial.urls')),
    path('admin/', admin.site.urls),
    path('wl/', vi.home),
    path('dt/', vd.dis_datetime), #an url to open datetime views.py
]
```

```
PS C:\Users\VISHNU\project\project> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 24, 2024 - 16:26:56
Django version 4.2.11, using settings 'project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Go to the browser and give the link <http://127.0.0.1:8000/dt/>



2024-04-24 15:23:28.941103

PROGRAM 2

Develop a Django app that displays date and time four hours ahead and four hours before as an offset of current date and time in the server.

views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from datetime import datetime, timedelta
# Create your views here.
def datetime_display(request):
    # Get current date and time
    current_datetime = datetime.now()

    # Calculate offsets
    four_hours_before = current_datetime - timedelta(hours=4)
    four_hours_ahead = current_datetime + timedelta(hours=4)

    # Prepare the response content
    response_content = (
        f"Current Date and Time: {current_datetime}\n"
        f"Four Hours Before: {four_hours_before}\n"
        f"Four Hours Ahead: {four_hours_ahead}\n"
    )

    # Return the response
    return HttpResponse(response_content, content_type='text/plain')
```

urls.py

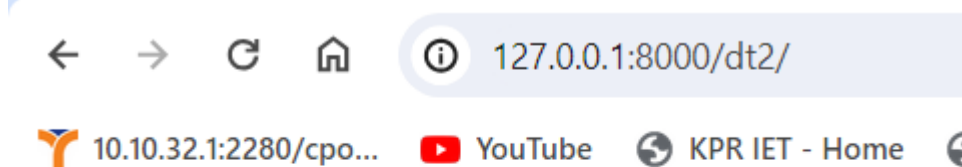
```
from django.contrib import admin
from django.urls import path, include
from datetimeapp import views as vd
from initial import views as vi
urlpatterns = [
    #path("include('initial.urls')"),
    #path('admin/', admin.site.urls),
    #path('wl/', vi.home),
    #path('dt/', vd.dis_datetime),
    path('dt2/', vd.datetime_display),
]
```


Run the command

```
PS C:\Users\VISHNU\project\project> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 24, 2024 - 16:26:56
Django version 4.2.11, using settings 'project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Go to the browser and go to <http://127.0.0.1:8000/dt2/>



Current Date and Time: 2024-04-24 16:24:36.546764
Four Hours Before: 2024-04-24 12:24:36.546764
Four Hours Ahead: 2024-04-24 20:24:36.546764

PROGRAM 3

Develop a simple Django app that displays an unordered list of fruits and ordered list of selected students for an event

Student_tag.html

```
<!DOCTYPE html>

<html>

<head>

  <title>Event Details</title>

</head>

<body>

  <h1>Fruits</h1>

  <ul>

    {% for fruit in fruits %}

      <li>{{ fruit }}</li>

    {% endfor %}

  </ul>

  <h1>Selected Students</h1>

  <ol>

    {% for student in students %}

      <li>{% if student.score > 80 %}{{ student.name }}{% endif %}</li>

    {% endfor %}

  </ol>

</body>

</html>
```

Athlete_list.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <title>Athletes List</title>

</head>

<body>

  <h1>List of Athletes</h1>

  <ul>

    {% for athlete in athlete_list %}

    <li>{{ athlete }}</li>

    {% endfor %}

  </ul>

  {% if athlete_list %}

    <p>Number of athletes: {{ athlete_list|length }}</p>

    {% elif athlete_in_locker_room_list %}

    <p>Athletes should be out of the locker room soon!</p>

    {% else %}

    <p>No athletes.</p>

    {% endif %}

</body>

</html>
```

views.py

```
from django.shortcuts import render

def event_details(request):

    fruits = ['Apple', 'Banana', 'Orange', 'Mango']

    # Sample student data with scores

    students = [

        {'name': 'Harish', 'score': 85},

        {'name': 'Briana', 'score': 70},

        {'name': 'John', 'score': 95},

        {'name': 'Arul', 'score': 60},

    ]

    return render(request, 'student_tag.html', {'fruits': fruits, 'students': students})

def athlete_l(request):

    context = {

        'athlete_list': ['Rajwinder Kaur', 'Chitra Soman', 'Manjeet Kaur '],

        'athlete_in_locker_room_list': []

    }

    return render(request, 'athlete_list.html', context)
```

urls.py

```
from django.urls import path

from . import views

urlpatterns = [

    path('events/', views.event_details, name='event_details'),

    path('athlete/', views.athlete_l, name='athlete_l'),

]
```

urls.py - Main Project

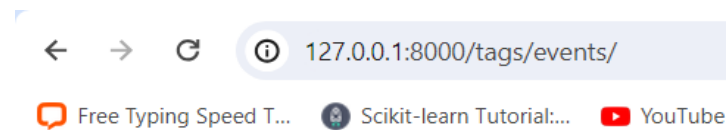
```
from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('tags/', include('Lab_temp_1.urls')),

]
```



Fruits

- Apple
- Banana
- Orange
- Mango

Selected Students

1. Harish
- 2.
3. John
- 4.



List of Athletes

- Rajwinder Kaur
- Chitra Soman
- Manjeet Kaur

Number of athletes: 3

List of Athletes

- Chitra Soman
- Manjeet Kaur

Number of athletes: 2

PROGRAM 4

Develop a layout.html with a suitable header (containing navigation menu) and footer with copyright and developer information. Inherit this layout.html and create 3 additional pages: contact us, About Us and Home page of any website

views.py

```
from django.shortcuts import render

def home(request):

    return render(request, 'home.html')

def about(request):

    return render(request, 'about.html')

def contact(request):

    return render(request, 'contact.html')
```

Home.html

```
{% extends 'layout.html' %}

{% block title %}Home{% endblock %}

{% block content %}

<h1>Welcome to My Website</h1>

<p>This is the home page of My Website.</p>

{% endblock %}
```

contact.html

```
{% extends 'layout.html' %}

{% block title %}Contact Us{% endblock %}
```

```
{% block content %}
```

```
<h1>Contact Us</h1>
```

```
<p>This is the contact page of My Website.</p>
```

```
{% endblock %}
```

About.html

```
{% extends 'layout.html' %}
```

```
{% block title %}About Us{% endblock %}
```

```
{% block content %}
```

```
<h1>About Us</h1>
```

```
<p>This is the about page of My Website.</p>
```

```
{% endblock %}
```

urls.py (App url file)

```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
```

```
    path('lay/', views.home, name='home'),
```

```
    path('about/', views.about, name='about'),
```

```
    path('contact/', views.contact, name='contact'),
```

```
]
```

urls.py (Main project)

```
from django.contrib import admin
```

```

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('lay/', include('Inh_Lab_Prog.urls')),

]

```

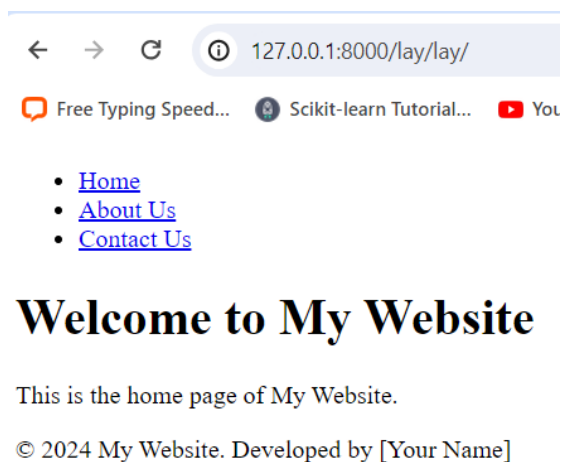
All the HTML files need to be stored in One folder named “Webpages” and the folder needs to be linked. Modify the line in settings.py file

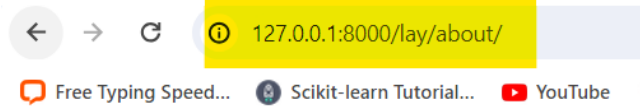
```

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'Webpages')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

Go to the browser and type <http://127.0.0.1:8000/lay/lay/>



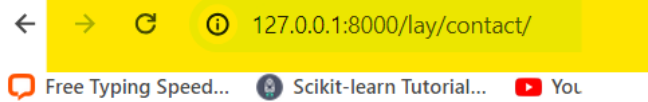


- [Home](#)
- [About Us](#)
- [Contact Us](#)

About Us

This is the about page of My Website.

© 2024 My Website. Developed by [Your Name]



- [Home](#)
- [About Us](#)
- [Contact Us](#)

Contact Us

This is the contact page of My Website.

© 2024 My Website. Developed by [Your Name]

PROGRAM 5

Develop a Django app that performs student registration to a course. It should also display a list of students registered for any selected course. Create students and courses as models with enrolment as ManyToMany field.

Models.py

In your app's models.py

```
from django.db import models
```

```
class Student(models.Model):
```

```
    name = models.CharField(max_length=100)
```

```
    roll_number = models.CharField(max_length=20)
```

```
    def __str__(self):
```

```
        return self.name
```

```
class Course(models.Model):
```

```
    name = models.CharField(max_length=100)
```

```
    students = models.ManyToManyField(Student, related_name='courses')
```

```
    def __str__(self):
```

```
        return self.name
```

admin.py

In your app's admin.py

```
from django.contrib import admin
```

```
from .models import Student, Course
```

```
admin.site.register(Student)
```

`admin.site.register(Course)`

Register student.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Student Registration</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Student Registration</h1>
```

```
    <form method="post">
```

```
        {% csrf_token %}
```

```
        <label for="name">Name:</label><br>
```

```
        <input type="text" id="name" name="name"><br><br>
```

```
        <label for="roll_number">Roll Number:</label><br>
```

```
        <input type="text" id="roll_number" name="roll_number"><br><br>
```

```
        <label for="courses">Select Courses:</label><br>
```

```
        <select id="courses" name="courses" multiple>
```

```
            {% for course in courses %}
```

```
                <option value="{{ course.id }}">{{ course.name }}</option>
```

```
            {% endfor %}
```

```
        </select><br><br>
```

```
<input type="submit" value="Register">

</form>

</body>

</html>
```

Registration success.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Registration Success</title>

</head>

<body>

  <h1>Registration Successful!</h1>

  <p>Thank you for registering.</p>

</body>

</html>
```

Course student.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Course Students</title>

</head>

<body>

<h1>Students Registered for {{ course.name }}</h1>

<ul>

    {% for student in students %}

        <li>{{ student.name }}</li>

    {% endfor %}

</ul>

</body>

</html>

```

urls.py

```

from django.contrib import admin

from django.urls import path

from stud_course.views import register_student, registration_success, course_students

urlpatterns = [

    path('admin/', admin.site.urls),

    path('register/', register_student, name='register_student'),

    path('registration-success/', registration_success, name='registration_success'),

    path('course/<int:course_id>', course_students, name='course_students'),

]

```

Do the following changes in setting.py file

Add your app name “stud_course”

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.staticfiles',
    'stud_course',
]

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'django_p', #give the database name
        'USER': 'postgres',
        'PASSWORD': 'vishnu',
        'HOST': 'localhost', # or the hostname where
    }
}

```

Run the following command

```

PS C:\Users\VISHNU\project\pro6 stud course> python manage.py makemigrations
Migrations for 'stud_course':
  stud_course\migrations\0001_initial.py
    - Create model Student
    - Create model Course

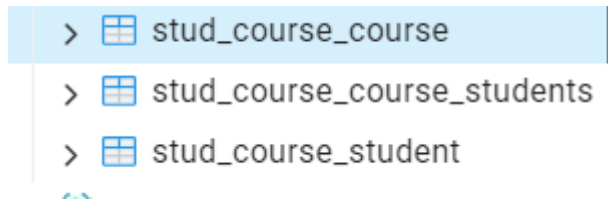
```

```

PS C:\Users\VISHNU\project\pro6 stud course> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, stud_course
Running migrations:
  Applying stud_course.0001_initial... OK

```

After this go to POSTGRESQL and check three tables would have been created



Stud_course_course_students table is created as we have mentioned ManytoMany fields.

Stud_course_course contains course details

Stud_course_student contains students and course registered by student information.

Insert the courses using insert command in POSTGRESQL

Data Output Messages Notifications		
<div> </div>		
	id [PK] bigint	name character varying (100)
1	1	Python
2	2	Java
3	3	DBMS
4	4	Data Structures
5	5	AIML

```
PS C:\Users\VISHNU\project\pro6_stud_course> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

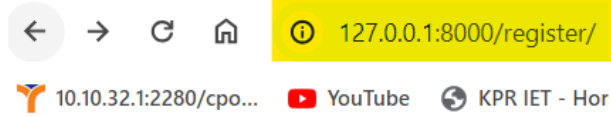
```
System check identified no issues (0 silenced).
```

```
April 24, 2024 - 18:58:01
```

```
Django version 4.2.11, using settings 'pro6_stud_course.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

Go to the browser and give the url <http://127.0.0.1:8000/register/>



Student Registration

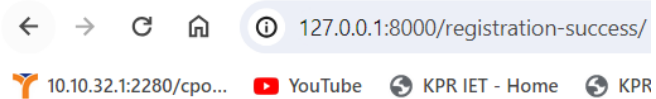
Name:

Roll Number:

Select Courses:

After registering check the table in POSTGRESQL

Data Output Messages Notifications			
	id [PK] bigint	name character varying (100)	roll_number character varying (20)
1	3	Richard	120
2	4	Hari	115
3	5	Bristo	3
4	6	Charley	10
5	7	Saran	122
6	8	YOgi	130



Registration Successful!

Thank you for registering.

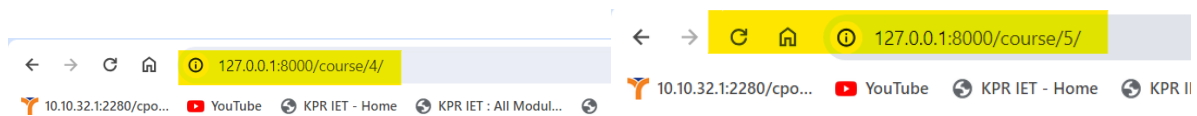
If you check in the link <http://127.0.0.1:8000/course/1/> then you could able to see the students who have registered for the courses



Students Registered for Python Students Registered for DBMS

- Hari
- YOgi

- Richard
- Bristo



Students Registered for Data Structures Students Registered for AIML

- Charley

- Saran

Main student database table

Data Output Messages Notifications			
	id [PK] bigint	name character varying (100)	roll_number character va
1	3	Richard	120
2	4	Hari	115
3	5	Bristo	3
4	6	Charley	10
5	7	Saran	122
6	8	YOgi	130

This table is created as we have used ManyToMany field

Data Output Messages Notifications			
	id [PK] bigint	course_id bigint	student_id bigint
1	1	3	3
2	2	1	4
3	3	3	5
4	4	4	6
5	5	5	7
6	6	1	8

PROGRAM 6

For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms.

Program is similar to the above.

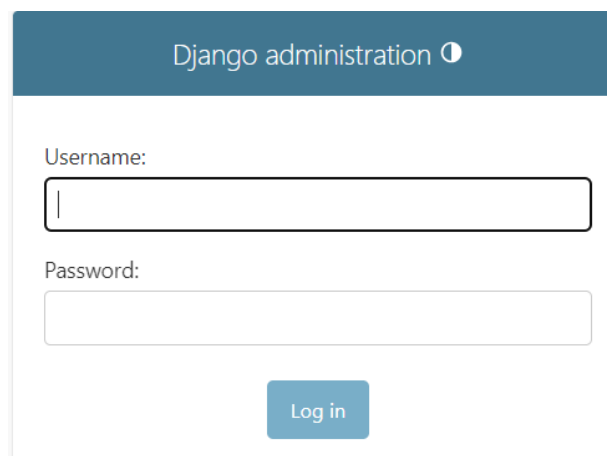
First create a superuser to access the Django Admin page

```
PS C:\Users\VISHNU\project\pro6_stud_course> python manage.py createsuperuser
Username: django_ad
Email address: djangoad@gmail.com
Password:
Password (again):
The password is too similar to the username.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
PS C:\Users\VISHNU\project\pro6_stud_course> █
```

```
PS C:\Users\VISHNU\project\pro6_stud_course> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 24, 2024 - 21:14:41
Django version 4.2.11, using settings 'pro6_stud_course.settings'
Starting development server at http://127.0.0.1:8000/
```

Go to the link <http://127.0.0.1:8000/admin/>



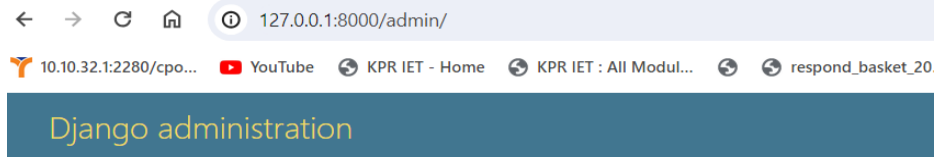
Django administration

Username:

Password:

Log in

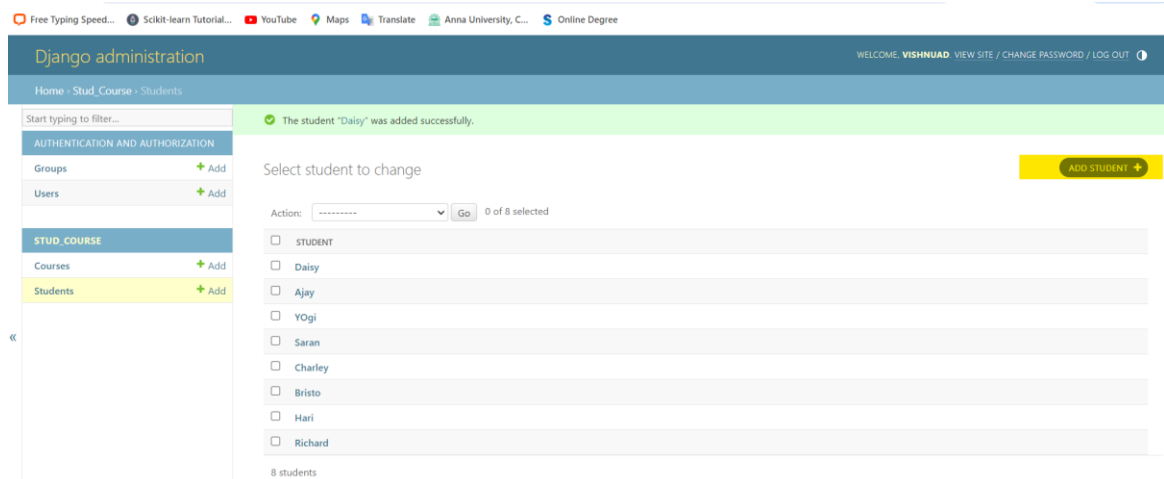
Enter the username and password. Then you will be navigated to the below page



Site administration

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add Change
Users	+ Add Change
STUD_COURSE	
Courses	+ Add Change
Students	+ Add Change

Click on the student to add new students



the Name and Roll Number and save

Give

it

Add student

Name:

Roll number:

SAVE

Save and add another

Save and continue editing

Once you give then student will be added to the stud_course_student table in POSTEGRE Database also

	id [PK] bigint	name character varying (100)	roll_number character varying (20)
1	3	Richard	120
2	4	Hari	115
3	5	Bristo	3
4	6	Charley	10
5	7	Saran	122
6	8	YOgi	130
7	10	Ajay	101
8	11	Daisy	25

Click on the course table, Now you can see the course that we have created in POSTGRESQL. Now try to add the course upon clicking “Add Course” button in top right corner

Django administration
Home > Stud_Course > Courses

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add
Users + Add

STUD_COURSE

Courses + Add
Students + Add

Select course to change

Action: Go 0 of 5 selected

☐ COURSE
☐ AIML
☐ Data Structures
☐ DBMS
☐ Java
☐ Python

5 courses

Add course

Name:

Students:

Richard
Hari
Bristo
Charley
Saran
YOgi

+

Hold down "Control", or "Command" on a Mac, to select more than one.

SAVE

Save and add another

Save and continue editing

Give the course name and click on the add button which will ask you to enter the student information as below

Add student

Name:

Roll number:

Enter the student information and give save

Add course

Name:

Students:

Hold down "Control", or "Command" on a Mac, to select more than one item.

Now you could see the new student added to the table. We are getting this because of ManyToMany field relation,

Action: 0 of 6

<input type="checkbox"/>	COURSE
<input type="checkbox"/>	PQT
<input type="checkbox"/>	AIML
<input type="checkbox"/>	Data Structures
<input type="checkbox"/>	DBMS
<input type="checkbox"/>	Java
<input type="checkbox"/>	Python

Now go and check in the POSTGRESQL Database.

		id [PK] bigint	name character varying (100)	roll_number character varyi
	id [PK] bigint			
	1	Python		
	2	Java		
	3	DBMS		
	4	Data Structures		
	5	AIML		
	6	PQT		
		4	Hari	115
		5	Bristo	3
		6	Charley	10
		7	Saran	122
		8	YOgi	130
		10	Ajay	101
		11	Daisy	25
		12	Rayan	140

id [PK] bigint	course_id bigint	student_id bigint
1	3	3
2	1	4
3	3	5
4	4	6
5	5	7
6	1	8
7	6	12

Student Rayan with ID (12) is mapped to the course PQT

Using the Admin Interface of Django we have add the course and student information.

PROGRAM 7

Develop a Model form for a student that contains his topic chosen for project, languages used and duration with a model called project.

forms.py

```
from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

    class Meta:

        model = Student

        fields = ['name', 'project']
```

Model.py

```
from django.db import models

class Project(models.Model):

    topic = models.CharField(max_length=100)

    languages_used = models.CharField(max_length=100)

    duration = models.CharField(max_length=50)

    def __str__(self):

        return self.topic

class Student(models.Model):

    name = models.CharField(max_length=100)

    project = models.ForeignKey(Project, on_delete=models.CASCADE)

    def __str__(self):
```

```
return self.name
```

views.py

```
from django.shortcuts import render, redirect

from .forms import StudentForm

def student_form(request):

    if request.method == 'POST':

        form = StudentForm(request.POST)

        if form.is_valid():

            form.save()

            return render(request, 'success.html', {'form': form})

    else:

        form = StudentForm()

    return render(request, 'template.html', {'form': form})
```

Template.html (create a templates folder in the project folder and create the HTML files)

```
<!-- template.html -->

<form method="post">

    {% csrf_token %}

    {{ form.as_p }}

    <button type="submit">Submit</button>

</form>
```

Success.html

```
<!DOCTYPE html>

<html lang="en">
```



```
<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Project Allocation</title>

</head>

<body>

  <h1>Project Allocated Successfully!</h1>

  <p>Thank you for registering.</p>

</body>

</html>
```

settings.py

Add you app

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

Add the templates folder path

```

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

Create a database connection

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'django_p', #give the database name
        'USER': 'postgres',
        'PASSWORD': 'vishnu',
        'HOST': 'localhost', # or the hostname wher
    }
}

```

Create a model

```

PS C:\Users\VISHNU\project\pro7_stud_project> python manage.py makemigrations
Migrations for 'stud_project':
  stud_project\migrations\0001_initial.py
    - Create model Project
    - Create model Student

```

Migrate the model

```

PS C:\Users\VISHNU\project\pro7_stud_project> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, stud_project
Running migrations:
  Applying stud_project.0001_initial... OK

```

Upon Migrating the model two tables stud_project_project and stud_project_student will be created. The add few projects to the project table

Data Output Messages Notifications				
	id [PK] bigint	topic character varying (100)	languages_used character varying (100)	duration character varying (50)
1	7	Artificial Intelligence in Healthcare	Python, TensorFlow	8 weeks
2	8	Cybersecurity in Financial Institutions	Java, C++, Python	10 weeks
3	9	Renewable Energy Technologies	C, MATLAB	6 weeks
4	10	Data Analytics for Marketing Optimization	R, Python, SQL	12 weeks
5	11	Blockchain Applications in Supply Chain Management	Solidity, JavaScript	8 weeks
6	12	Augmented Reality for Education Enhancement	Unity, C#	10 weeks

Run the program

```
PS C:\Users\VISHNU\project\pro7_stud_project> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 25, 2024 - 06:45:15
Django version 4.2.11, using settings 'pro7_stud_project.settings'
Starting development server at http://127.0.0.1:8000/
```

Then go to the link <http://127.0.0.1:8000/project/>

← → ↻ 🏠 ⓘ 127.0.0.1:8000/project/

🔍 10.10.32.1:2280/cpo... 📺 YouTube 🌐 KPR IET - Home 🌐 KF

Name:

Project:

Data Output Messages Notifications			
	id [PK] bigint	name character varying (100)	project_id bigint
1	1	Arjun	7
2	2	Vishnu	8

PROGRAM 8

For students enrolment developed in Module 2, create a generic class view which displays list of students and detailview that displays student details for any selected student in the list.

views.py

```
from django.shortcuts import render
```

```
from django.db import connection
```

```
def student_list(request):
```

```
    # Execute a raw SQL query to fetch all students
```

```
    with connection.cursor() as cursor:
```

```
        cursor.execute("SELECT * FROM stud_course_student")
```

```
        rows = cursor.fetchall()
```

```
    # Convert the query results into a list of dictionaries
```

```
    students = [{ 'id': row[0], 'name': row[1], 'roll_number': row[2] } for row in rows]
```

```
    # Render the template with the fetched data
```

```
    return render(request, 'student_list.html', { 'student_list': students })
```

```
def student_detail(request, student_id):
```

```
    with connection.cursor() as cursor:
```

```
        cursor.execute("""
```

```
SELECT s.name AS student_name, c.id AS course_id, c.name AS course_name

FROM stud_course_student s

LEFT JOIN stud_course_course_students sc ON s.id = sc.student_id

LEFT JOIN stud_course_course c ON sc.course_id = c.id

WHERE s.id = %s

""" , [student_id])

rows = cursor.fetchall()


student_name = None

student_courses = []

for row in rows:

    if not student_name:

        student_name = row[0]

    if row[1] and row[2]:

        student_courses.append({'course_id': row[1], 'course_name': row[2]})


return render(request, 'student_detail.html', {'student_name': student_name,
'student_courses': student_courses})
```

Student_list.html

```
<!DOCTYPE html>

<html>

<head>

<title>Student List</title>
```

```
</head>

<body>

  <h1>Student List</h1>

  <ul>

    {% for student in student_list %}

      <!-- Ensure student.id is used to pass the student ID to the student_detail URL -->

      <li><a href="{% url 'student_detail' student.id %}">{{ student.name }}</a></li>

    {% endfor %}

  </ul>

</body>

</html>
```

Student_details.html

```
<!DOCTYPE html>

<html>

<head>

  <title>Student Detail</title>

</head>

<body>

  <h1>Student Detail</h1>

  <p>Name: {{ student_name }}</p>

  <p>Registered Courses:</p>

  <ul>

    {% for course in student_courses %}
```

```

<li>{{ course.course_id }} - {{ course.course_name }}</li>

{% endfor %}

</ul>

</body>

</html>

```

urls.py (Main Project)

```

from django.contrib import admin

from django.urls import path

from stud_gview.views import student_list, student_detail

urlpatterns = [

    path('admin/', admin.site.urls),

    path('students/', student_list, name='student_list'),

    path('students/<int:student_id>', student_detail, name='student_detail'),

]

```

Run the server

```

PS C:\Users\VISHNU\project\pro8_stud_generic_view> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 25, 2024 - 10:23:08
Django version 4.2.11, using settings 'pro8_stud_generic_view.settings'
Starting development server at http://127.0.0.1:8000/

```

Go to the link <http://127.0.0.1:8000/students/>

Student List

- [Richard](#)
- [Hari](#)
- [Bristo](#)
- [Charley](#)
- [Saran](#)
- [YOgi](#)
- [Ajay](#)
- [Daisy](#)
- [Rayan](#)

Click on the student name

Student Detail

Name: Richard

Registered Courses:

- 3 - DBMS

Student Detail

Name: Ajay

Registered Courses:

Student Detail

Name: Saran

Registered Courses:

- 5 - AIML

Student Detail

Name: Charley

Registered Courses:

- 4 - Data Structures

PROGRAM 9

Develop an example Django app that performs CSV and PDF generation for any models created in previous laboratory components.

views.py

```
from django.http import HttpResponse
```

```
import csv
```

```
from reportlab.lib import colors
```

```
from reportlab.lib.pagesizes import letter
```

```
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
```

```
from django.db import connection
```

```
def generate_student_csv(request):
```

```
    response = HttpResponse(content_type='text/csv')
```

```
    response['Content-Disposition'] = 'attachment; filename="students.csv"'
```

```
    writer = csv.writer(response)
```

```
    writer.writerow(['Roll Number', 'Name']) # CSV header
```

```
    with connection.cursor() as cursor:
```

```
        cursor.execute("SELECT roll_number, name FROM stud_course_student")
```

```
        rows = cursor.fetchall()
```

```
        for row in rows:
```

```
            writer.writerow(row)
```

```
    return response
```

```
def generate_student_pdf(request):
```

```
response = HttpResponseRedirect(content_type='application/pdf')

response['Content-Disposition'] = 'attachment; filename="students.pdf"'

students_data = [['Roll Number', 'Name']] # PDF data

with connection.cursor() as cursor:

    cursor.execute("SELECT roll_number, name FROM stud_course_student")

    rows = cursor.fetchall()

    students_data.extend(rows)

generate_pdf(students_data, response) # Call function to generate PDF

return response

def generate_pdf(data, response):

    doc = SimpleDocTemplate(response, pagesize=letter)

    table = Table(data)

    # Add style to table

    style = TableStyle([('BACKGROUND', (0, 0), (-1, 0), colors.grey),

                        ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

                        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),

                        ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

                        ('BOTTOMPADDING', (0, 0), (-1, 0), 12),

                        ('BACKGROUND', (0, 1), (-1, -1), colors.beige),

                        ('GRID', (0, 0), (-1, -1), 1, colors.black)])

    table.setStyle(style)

    doc.build([table])
```

```

from django.urls import path

from p_csv.views import generate_student_pdf, generate_student_csv

urlpatterns = [

    path('generate-student-csv/', generate_student_csv, name='generate_student_csv'),

    path('generate-student-pdf/', generate_student_pdf, name='generate_student_pdf'),

]

```

settings.py

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'django_p', #give the database name
        'USER': 'postgres',
        'PASSWORD': 'vishnu',
        'HOST': 'localhost', # or the hostname where
    }
}

```

Run the server

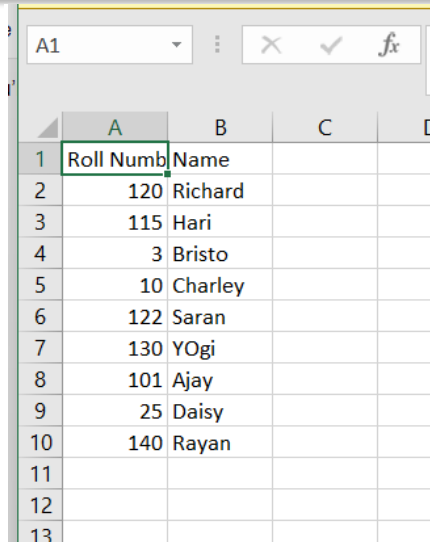
```

PS C:\Users\VISHNU\project\pro9_CSV> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 25, 2024 - 13:15:05
Django version 4.2.11, using settings 'pro9_CSV.settings'
Starting development server at http://127.0.0.1:8000/

```

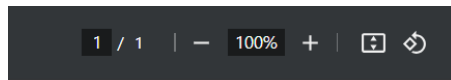
Go to the link <http://127.0.0.1:8000/generate-student-csv/> , it will automatically download the CSV file



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D
1	Roll Number	Name		
2	120	Richard		
3	115	Hari		
4	3	Bristo		
5	10	Charley		
6	122	Saran		
7	130	YOgi		
8	101	Ajay		
9	25	Daisy		
10	140	Rayan		
11				
12				
13				

<http://127.0.0.1:8000/generate-student-pdf/>



Roll Number	Name
120	Richard
115	Hari
3	Bristo
10	Charley
122	Saran
130	YOgi
101	Ajay
25	Daisy
140	Rayan