



ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ, ಬೆಳಗಾವಿ  
VISVESVARAYA TECHNOLOGICAL UNIVERSITY - BELAGAVI

## **JAIN COLLEGE OF ENGINEERING, BELAGAVI**



**Department of Computer Science and Engineering**

**Full Stack Development Laboratory Manual (21CS62)**

**VI- Semester**

**Prepared by**

**Prof. Sandeep Chavan  
Dept of CSE, JCE**

### **Vision**

Establish and develop the Institute as the Centre of higher learning, ever abreast with expanding horizon of knowledge in the field of Engineering and Technology with entrepreneurial thinking, leadership excellence for life-long success and solve societal problems.

### **Mission**

1. Provide high quality education in the Engineering disciplines from the undergraduate through doctoral levels with creative academic and professional programs.
2. Develop the Institute as a leader in Science, Engineering, Technology, Management and Research and apply knowledge for the benefit of society.
3. Establish mutual beneficial partnerships with Industry, Alumni, Local, State and Central Governments by Public Service Assistance and Collaborative Research.
4. Inculcate personality development through sports, cultural and extracurricular activities and engage in social, economic and professional challenges.

### **VISION:**

To be a center of excellence in computer engineering education, empowering graduates as highly skilled professionals.

### **MISSION:**

1. To provide a platform for effective learning with emphasis on technical excellence.
2. To train the students to meet current industrial standards and adapt to emerging technologies.
3. To instill the drive for higher learning and research initiatives.
4. To inculcate the qualities of leadership and Entrepreneurship.

### **PROGRAM EDUCATIONAL OBJECTIVES (PEO)**

1. Graduates will apply fundamental and advanced concepts of Computer Science and Engineering for solving real world problems.
2. Graduates will build successful professional careers in various sectors to facilitate societal needs.
3. Graduates will have the ability to strengthen the level of expertise through higher studies and research.
4. Graduates will adhere to professional ethics and exhibit leadership qualities to become an Entrepreneur.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

1. The graduates of the program will have the ability to build software products by applying theoretical concepts and programming skills.
2. The graduates of the program will have the ability to pursue higher

## **Program Outcomes**

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### COURSE LEARNING OBJECTIVES (CLO)

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.

### COURSE OUTCOMES (CO)

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,

## CO TO PO & PSO MAPPING

**1/2/3**

[illegible]

## Full Stack Development Laboratory

Subject Code: 21CS62

Hours/Week: 02

### LIST OF PROGRAMS

Expt. No	Name of Experiment
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 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 list of students registered for any selected course. Create students and course 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 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 example Django app that performs CSV and PDF generation for any models created in previous laboratory component
10	Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX
11	Develop a search application in Django using AJAX that displays courses enrolled by a student being searched
12	Creation of virtual environment, Django project and App should be demonstrated

**Full Stack Development Laboratory****Subject Code: 21CS62****Hours/Week: 02****SCHEDULE OF EXPERIMENTS**

<b>Sl. No</b>	<b>Name of Experiment</b>	<b>WEEK</b>
1	Sample programs	Week1
2	Sample programs	Week2
3	Installation of python,Django and VS code, Creation of virtual environment Django Project and App.	Week3
4	Display Current Date and Time in Server.	Week4
5	Display Current Date and Time four hours ahead and four hours before in server.	Week5
6	Django app that displays an unordered list of fruits and ordered list of selected students for an event	Week6
7	Inherit this layout.html and create 3 additional pages: contact us, About Us and Home page of any website.	Week7
8	Django app that performs student registration to a course. It should also display list of students registered for any selected course. Create students and course as models with enrolment as ManyToMany field.	Week8
9	a. For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms b. Develop a Model form for student that contains his topic chosen for project, languages used and duration with a model called project.	Week9
10	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.	Week10
11	Develop example Django app that performs CSV and PDF generation for any models created in previous laboratory component.	Week11

12	Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX.	Week12
13	Develop a search application in Django using AJAX that displays courses enrolled by a student being searched.	Week13



## **LABORATORY**

### **General Lab Guidelines:**

1. Conduct yourself in a responsible manner at all times in the laboratory. Intentional misconduct will lead to exclusion from the lab.
2. Do not wander around, or distract other students, or interfere with the laboratory experiments of other students.
3. Read the handout and procedures before starting the experiments. Follow all written and verbal instructions carefully.
4. If you do not understand the procedures, ask the instructor or teaching assistant. Attendance in all the labs is mandatory, absence permitted only with prior permission from the Class teacher.
5. The workplace has to be tidy before, during and after the experiment.
6. Do not eat food, drink beverages or chew gum in the laboratory.
7. Every student should know the location and operating procedures of all Safety equipment including First Aid Kit and Fire extinguisher.

### **DO'S:-**

1. An ID card is a must.
2. Keep your belongings in a designated area.
3. Sign the log book when you enter/leave the laboratory.
4. Records have to be submitted every week for evaluation.
5. The program to be executed in the respective lab session has to be written in the lab observation copy beforehand.
6. After the lab session, shut down the computers.
7. Report any problem in system (if any) to the person in-charge

### **DON'TS:-**

1. Do not insert metal objects such as clips, pins and needles into the computer casings(They may cause fire) and should not attempt to repair, open, tamper or interfere with any of the computer, printing, cabling, or other equipment in the laboratory.
2. Do not change the system settings and keyboard keys.
3. Do not upload, delete or alter any software/ system files on laboratory computers.
4. No additional material should be carried by the students during regular labs.

5. Do not open any irrelevant websites in labs.
6. Do not use a flash drive on lab computers without the consent of the lab instructor.
7. Students are not allowed to work in the Laboratory alone or without the presence of the instructor/teaching assistant.

## **FULL STACK DEVELOPMENT**

- Full Stack Development refers to the practice of developing both the front-end (client-side) and back-end (server-side) portions of web applications.
- A full stack developer is proficient in working with both the front-end and back-end technologies, allowing them to build complete web applications independently or as part of a team.

Technologies used in full stack development

### **Front-End Technologies:**

- HTML (Hypertext Markup Language): Used for structuring web pages.
- CSS (Cascading Style Sheets): Used for styling the appearance of web pages.
- JavaScript: A programming language used for adding interactivity and dynamic behavior to web pages.
- Front-end frameworks/libraries such as React.js, AngularJS, or Vue.js: These provide tools and utilities for building user interfaces and managing application state.

### **Back-End Technologies:**

- Server-side languages like JavaScript (Node.js), Python (Django, Flask), Ruby (Ruby on Rails), Java (Spring Boot), or PHP (Laravel), C#(.Net Framework), java(Servlets)
- Databases such as MySQL, PostgreSQL, MongoDB, or Firebase for storing and managing data.
- Web servers like Apache or Nginx or IIS or Tomcat or Caddy(with built in support for https) for handling HTTP requests.

### **Development Tools and Environment:**

- Version control systems like Git for managing code changes.
- Integrated Development Environments (IDEs) such as Visual Studio Code, Sublime Text, or Atom.

- Command-line tools for tasks like package management (npm for Node.js, pip for Python, nugget for .Net), running servers, and deployment.

### Important STACKS

- MEAN Stack
- MERN Stack
- Django Stack
- LAMP
- WAMP

### Why Django?

- Rapid Development:- DRY (Don't repeat Yourself)
- Saves Time and Money
- Rounded Solution
- Great Exposure
- Complete Ownership
- Greater opportunity with more learning
- Security (Prevention of threats such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking by providing { % csrf\_token % } django tag to be included inside the form.

This token generates a hidden input field containing a unique CSRF token. This token is then validated by Django when the form is submitted, ensuring that the request originated from the same site and protecting against CSRF attacks.

- Batteries Included Philosophy (Model Forms)

- Built in database (Sqlite DB)

## **What is Django**

- Django is a free and open source web application framework which offers fast and effective dynamic website development.
- It is written using python.
- It follows MVT (model, view and template architecture)

## **Framework**

A framework is a pre-built collection of libraries, modules, and tools that provides a structured approach to developing software applications.

- Django is a web development framework.
- AngularJS is a Web Frontend development framework
- React is a Web frontend development library.

## **Django Evolution**

1. Write a Web application from scratch.
2. Write another Web application from scratch.
3. Realize the application from step 1 shares much in common with the application from step 2.
4. Refactor the code so that application 1 shares code with application 2.
5. Repeat steps 2–4 several times.
6. Realize you've invented a framework

## **Django MVT**

The MVT is a software design pattern which includes three important components Model, View and Template.

- The **Model** helps to handle database. It is a data access layer which handles the data.
- The **Template** is a presentation layer which handles User Interface part completely.
- The **View** is used to execute the business logic and interact with a model to carry data and renders a template.

### **Characteristics of Django**

- Loosely Coupled – Django helps you to make each element of its stack independent of the others.
- Less code - Ensures effective development
- Not repeated- Everything should be developed in precisely one place instead of repeating it again
- Fast development- Django's offers fast and reliable application development.
- Consistent design - Django maintains a clean design and makes it easy to follow the best web development practices.

### **Python Virtual Environment**

- A Python Virtual Environment is an isolated space where you can work on your Python projects, separately from your system-installed Python.
- You can set up your own libraries and dependencies without affecting the system Python.
- There are no limits to the number of virtual environments
- It allows you to have multiple Python environments with different versions of Python and different sets of installed packages on the same system.
- It is generally good to have one new virtual environment for every Python-based project you work on
- You can change the system python version, django version and other dependencies without affecting the project python version, django versions and dependencies

**Installation of Python and Visual Studio code editors can be demonstrated.**

- Python download Link:

<https://www.python.org/downloads/>

- Visual Studio Code download and installation link:

<https://code.visualstudio.com/>

**Creating system root folder, project root project folder, creating virtual env inside project root folder.**

Create a system root folder with the name **JCE\_CSE\_FD** in the file system (inside any preferred drive).

- Create a project root folder inside **JCE\_CSE\_FD** with the name “**hello\_world**” (assuming we are doing simple hello world program)
- Open **cmd** inside “**hello\_world**”
- Create virtual env inside “**hello\_world**” with the name “**hello\_world\_venv**”

```
python -m venv <name_of_virtual_env>
```

```
python -m venv hello_world_venv
```

**Open project root folder in VS code**

- Run “code .” in the cmd prompt to launch the project folder “hello\_world” in VS code.
- or
- Launch VS code from the task bar, goto file menu navigate and open “hello\_world”

**Command palette (select your venv as python interpreter for your project root folder)**

- In VS Code, open the Command Palette (**View > Command Palette** or (Ctrl+Shift+P)). Then select the **Python: Select Interpreter** command
- The command presents a list of available interpreters that VS Code can locate automatically. From the list, select the virtual environment in your project folder that starts with ./env or .\env:
- Select the virtual environment that is created “hello\_world\_venv”. Look for recommended
- Open VS code terminal and install django framework.

```
pip install django
```

- Check whether django installation is correct

```
python manage.py runserver
```

- Create the django project with the name “hello\_world\_proj” inside the project root folder “hello\_world”

```
django-admin startproject proj .
```

- Create the django app with the name “hello\_world\_app”

```
python manage.py startapp hello_world_app
```



## **SAMPLE PROGRAMS**

### **Helloworld**

#### **views.py**

```
from django.http import HttpResponse
from django.shortcuts import render
```

```
# Create your views here.
```

```
def hello(request):
    resp="<h1>Hello World!!!!</h1>"
    return HttpResponse(resp)
```

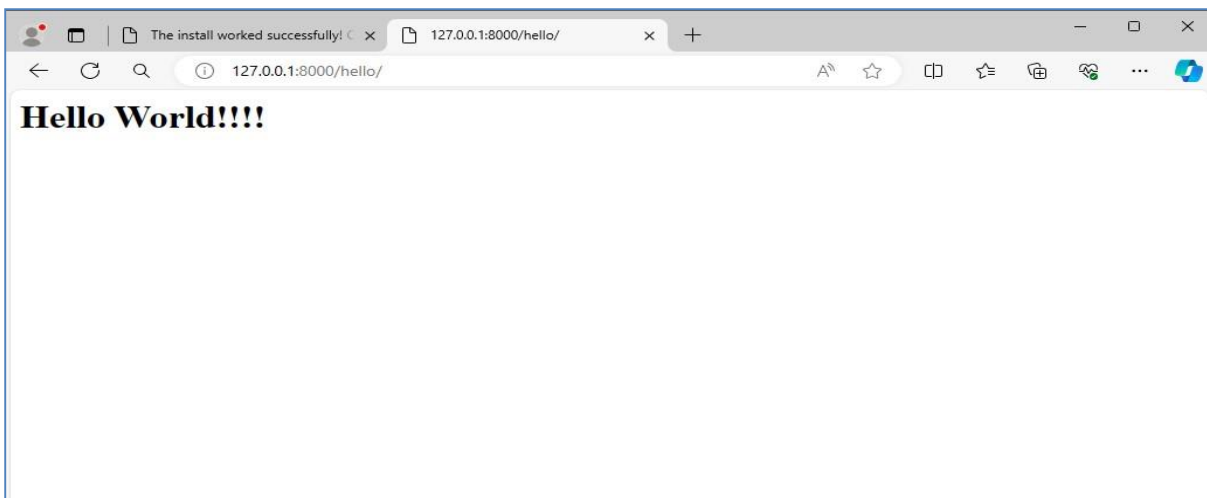
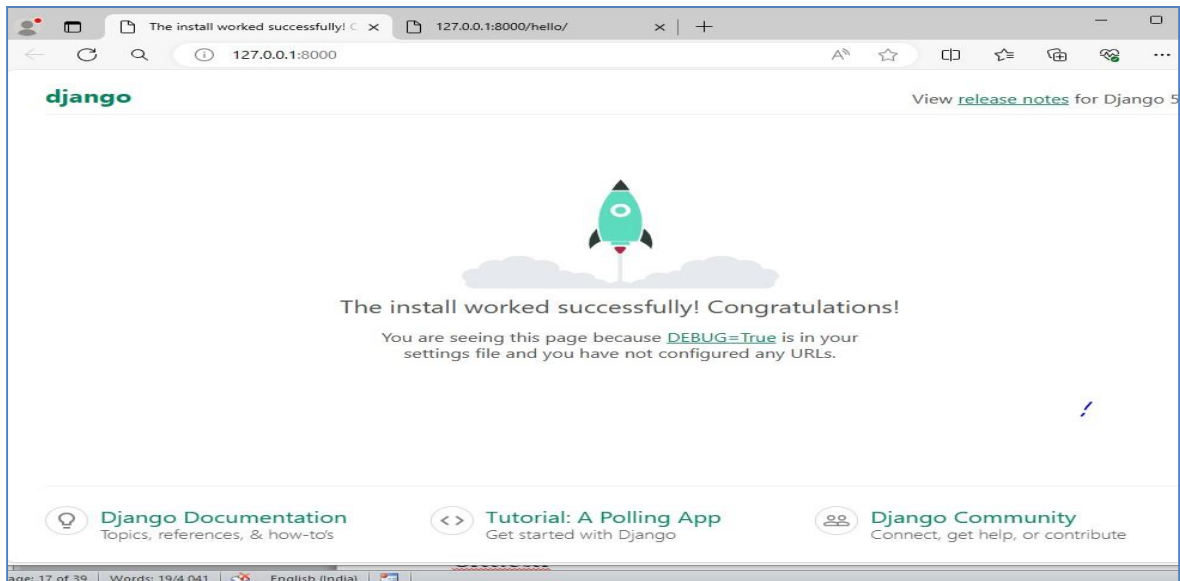
#### **urls.py**

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

from hello_world_app.views import hello

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

## **OUTPUT**



**GreetUser****greet.html**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Greet App</title>

  </head>

  <body>
    <h1>{{ user }} Good afternoon! Welcome to FDP on Full stack development</h1>
  </body>
</html>
```

**views.py**

```
from django.shortcuts import render

# Create your views here.

def greet(request,user):
    uname=user
    d={'user':user}
    return render(request,'greet.html',d)
```

**urls.py**

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

from greetUserApp.views import greet

urlpatterns = [
    path('admin/', admin.site.urls),
    path('greet/<str:user>/',greet),
```

]

## OUTPUT



## Vowel Count

### StudentlistApp

#### views.py

```
from django.http import HttpResponse
from django.shortcuts import render

# Create your views here.

def studentList(request,studentList): # rama,krishna,harish,chetan
    studList=studentList.split(',')    # [rama, krishna, harish, chetan]
    resp="<h1>Student List in Alphabetical order</h1>"
    #resp+="<h2>"+str(sorted(studList))+ "</h2>"
    for s in sorted(studList):
        resp+="<li>"+s+"</li>"
    resp+="<h1>Student List in reverse order</h1>"
    #resp+="<h2>"+str(sorted(studList,reverse=True))+ "</h2>"
    for s in sorted(studList,reverse=True):
        resp+="<li>"+s+"</li>"
    resp+="<h1>Student List in order of their name length</h1>"
    #resp+="<h2>"+str(sorted(studList,key=len))+ "</h2>"
    for s in sorted(studList,key=len):
        resp+="<li>"+s+"</li>"
    return HttpResponse(resp)
```

#### views.py

```
from django.http import HttpResponse
from django.shortcuts import render
```

```
# Create your views here.
```

```
def VC(request,s):
    v_cnt=0
    c_cnt=0
    v_dict={ } #v_dict=dict()
```

```

c_dict={ }

for letter in s:
    if letter.isalpha():
        if letter in "aeiouAEIOU":
            v_cnt=v_cnt+1
            v_dict[letter]=v_dict.get(letter,0)+1
        else:
            c_cnt=c_cnt+1
            c_dict[letter]=c_dict.get(letter,0)+1
resp="<h1>Vowel Count=%d</h1><h1>Cons Count=%d</h1>"%(v_cnt,c_cnt)
resp+="<h1>Vowel Count Frequency</h1>"
for key,val in v_dict.items():
    resp+="<h3>"+key+": "+str(val)+"</h3>"
resp+="<h1>Cons Count Frequency</h1>"
for key,val in c_dict.items():
    resp+="<h3>"+key+": "+str(val)+"</h3>"
return HttpResponse(resp)

```

### urls.py

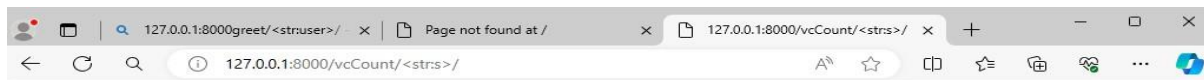
```

from django.contrib import admin
from django.urls import path

from vcCountApp.views import VC
from studentListApp.views import studentList
urlpatterns = [
    path('admin/', admin.site.urls),
    path('vcCount/<str:s>/', VC),
    path('studlist/<str:studentList>/', studentList),
]

```

### OUTPUT



**Vowel Count=0**

**Cons Count=4**

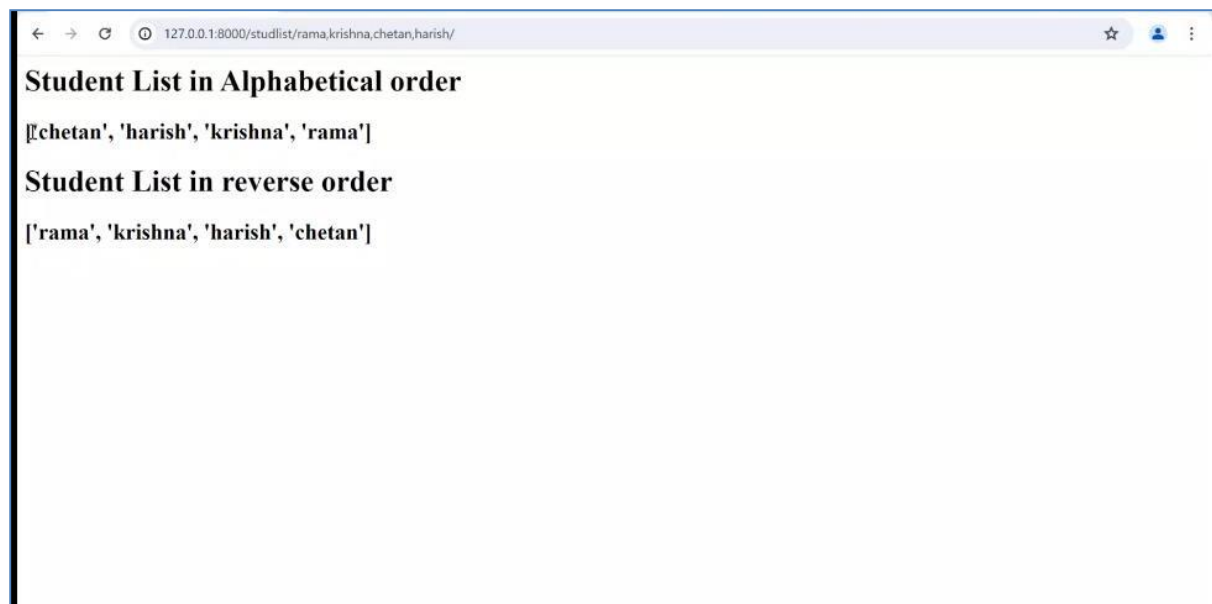
**Vowel Count Frequency**

**Cons Count Frequency**

s:2

t:1

r:1

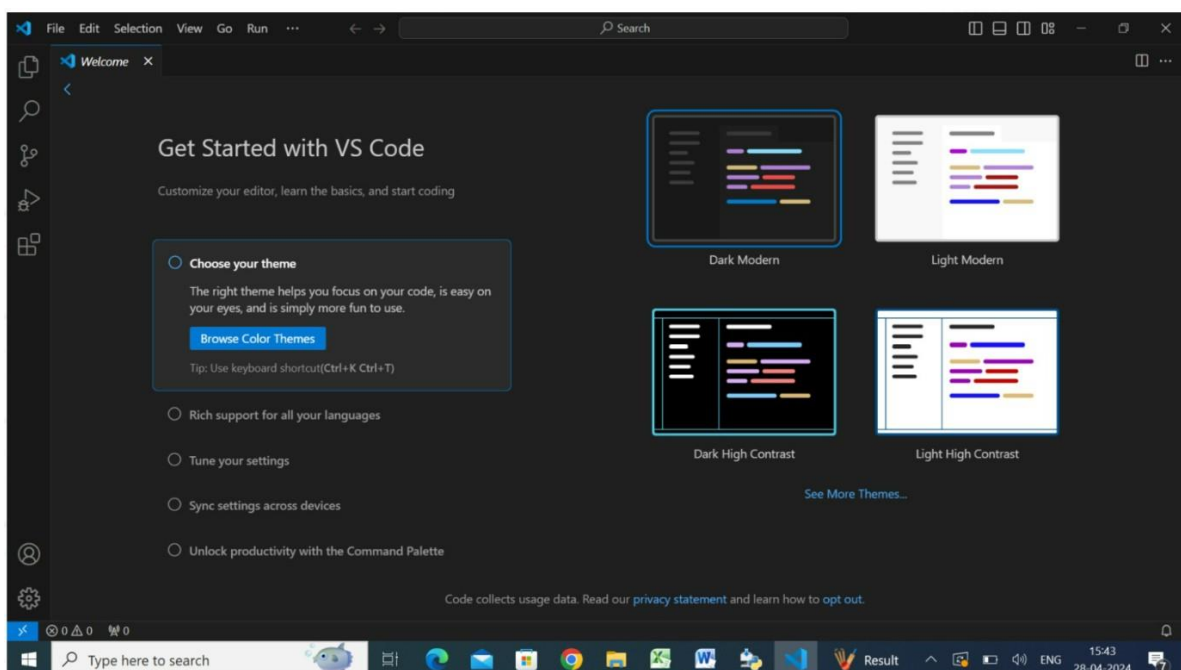


## LAB PROGRAMS

Installation of Python, Django and Visual Studio code editors can be demonstrated.







**PROGRAM . 1**

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

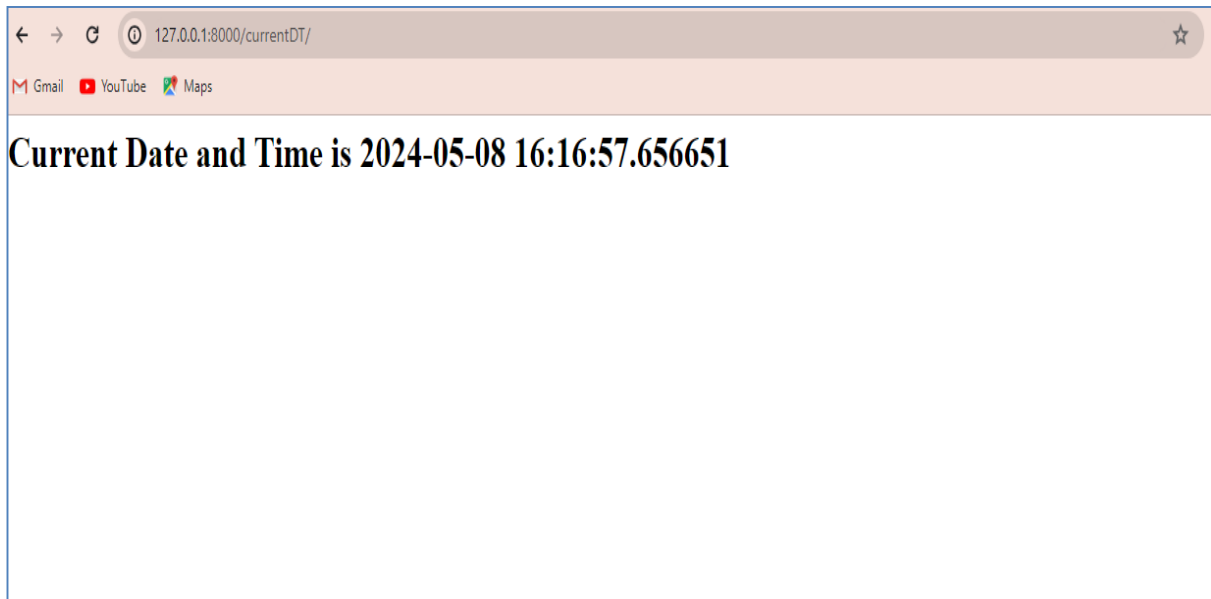
**views.py**

```
from django.shortcuts import render
import datetime
from django.http import HttpResponse
# Create your views here.
def cdt(request):
    dt=datetime.datetime.now()
    resp="<h1>Current Date and Time is %s<h1>"%(dt)
    return HttpResponse(resp)
```

**urls.py**

```
from django.contrib import admin
from django.urls import path
from scdtApp.views import cdt
urlpatterns = [
    path('admin/', admin.site.urls),
    path('currentDT/',cdt),
]
```

**OUTPUT**



---

**PROGRAM . 2**

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 server.

Displays Date & Time Four hours Ahead.

**views.py**

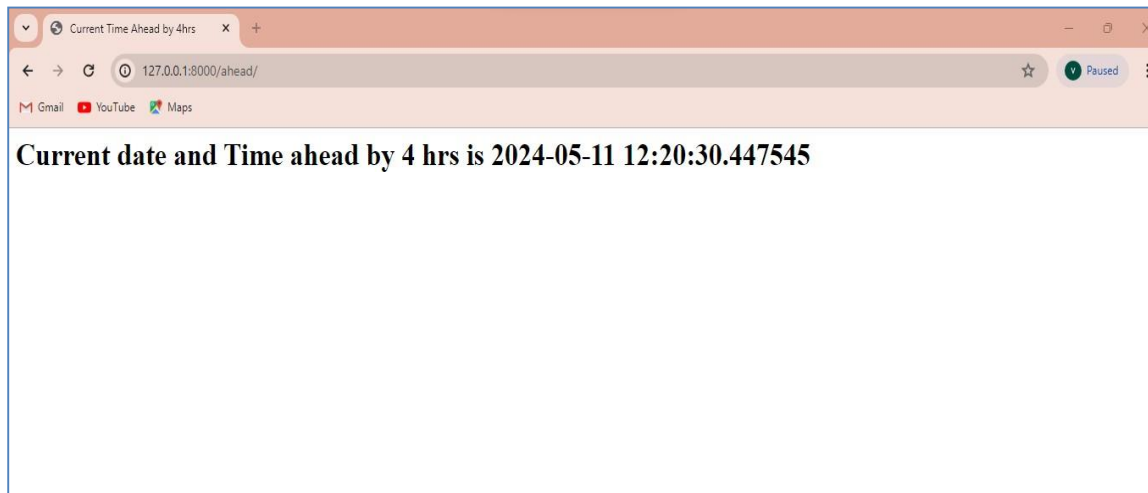
```
from django.shortcuts import render
import datetime
from django.http import HttpResponse
# Create your views here.

def aheadtime(request):
    dt=datetime.datetime.now()+datetime.timedelta(hours=4)
    resp="<html><head><title>Current Time Ahead by
4hrs</title></head><body><h1>Current date and Time ahead by 4 hrs is %s
</h1></body></html>"%(dt)
    return HttpResponse(resp)
```

**urls.py**

```
from django.contrib import admin
from django.urls import path
from scdt_a4App.views import aheadtime
urlpatterns = [
    path('admin/', admin.site.urls),
    path('ahead/',aheadtime),
]
```

## OUTPUT



Displays Date & Time Four hours Before.

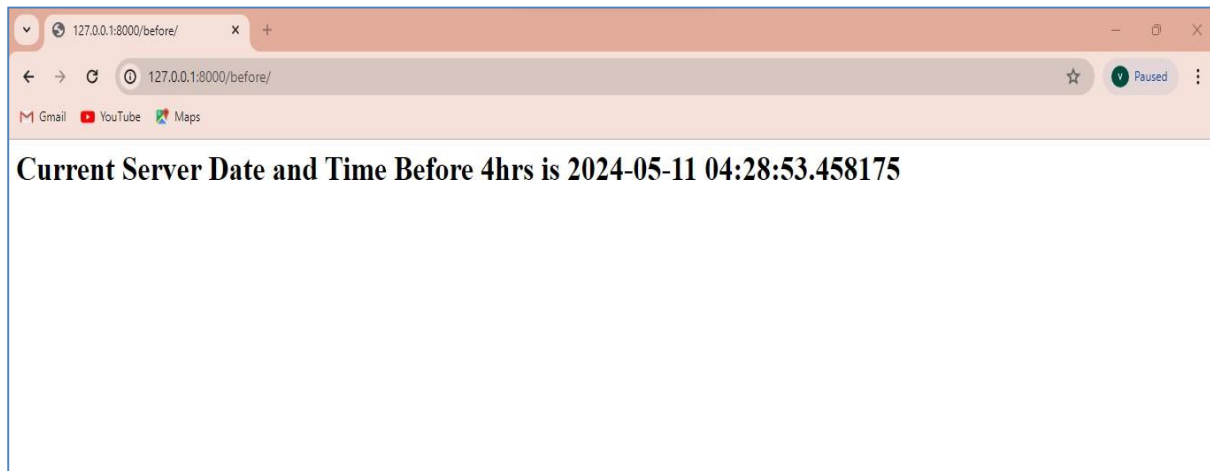
### **views.py**

```
from django.shortcuts import render
import datetime
from django.http import HttpResponse
# Create your views here.
def beforetime(request):
    dt=datetime.datetime.now()+datetime.timedelta(hours=-4)
    resp="<h1>Current Server Date and Time Before 4hrs is %s</h1>"%(dt)
    return HttpResponse(resp)
```

### **urls.py**

```
from django.contrib import admin
from django.urls import path
from scdt_b4App.views import beforetime
urlpatterns = [
    path('admin/', admin.site.urls),
    path('before/',beforetime),
]
```

## OUTPUT



Displays Date & Time Five hours Ahead & Before.

### **views.py**

```
from django.shortcuts import render
import datetime
from django.http import HttpResponse
# Create your views here.
def ahead(request):
    dt=datetime.datetime.now()+datetime.timedelta(hours=5)
    resp="<h1>Current date and time ahead by 5hrs is %s</h1>"%(dt)
    return HttpResponse(resp)

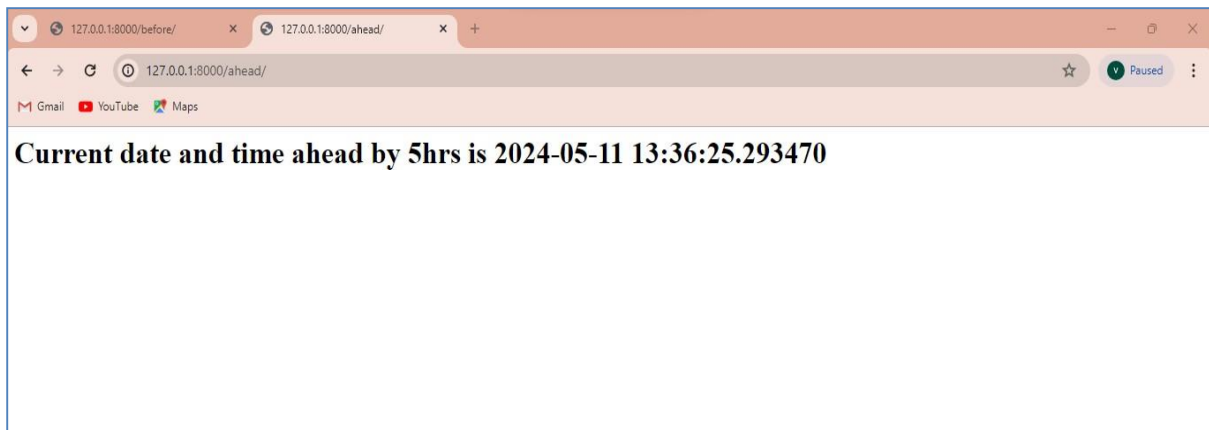
def before(request):
    dt=datetime.datetime.now()+datetime.timedelta(hours=-5)
    resp="<h1>Current date and time before by 5hrs is %s</h1>"%(dt)
    return HttpResponse(resp)
```

### **urls.py**

```
from django.contrib import admin
from django.urls import path
from scdt_abApp.views import ahead, before
```

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('ahead/', ahead),  
    path('before/', before),  
]
```

## OUTPUT



**Displays dynamic date & time.**

### views.py

```
from django.shortcuts import render  
import datetime  
from django.http import HttpResponse
```

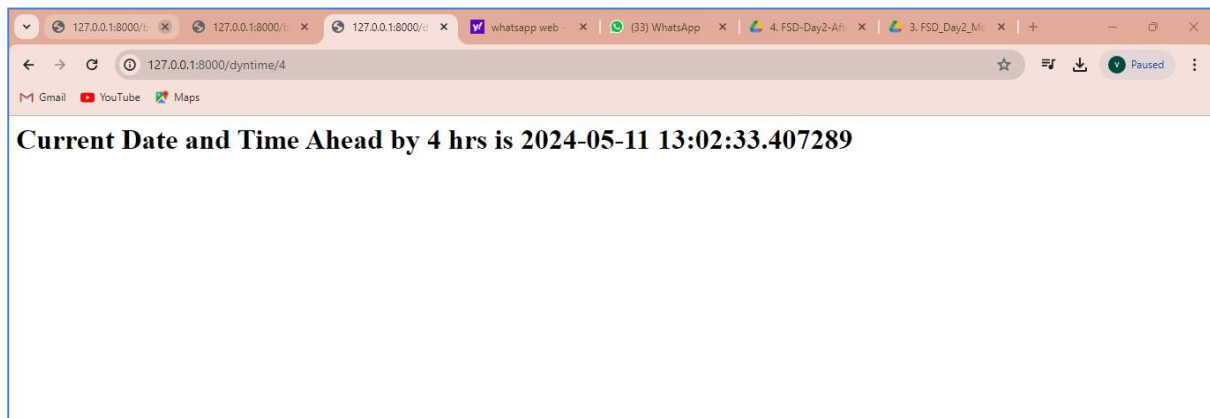
# Create your views here.

```
def scdt_dyn(request,t):  
    dt=datetime.datetime.now()+datetime.timedelta(hours=t)  
    resp="<h1>Current Date and Time Ahead by %d hrs is %s</h1>"%(t,dt)  
    return HttpResponse(resp)
```

### urls.py

```
from django.contrib import admin  
from django.urls import path  
from scdt_dynApp.views import scdt_dyn  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('dyntime/<int:t>',scdt_dyn),  
]
```

### OUTPUT







Displays dynamic date & time. (All Conditions)

### **views.py**

```
import datetime
from django.http import HttpResponse
from django.shortcuts import render

# Create your views here.

def scdt(request,s):
    t=int(s)
    dt=datetime.datetime.now()+datetime.timedelta(hours=t)
    if t<0:
        resp="<h1>Current Date and Time Behind %d hrs is %s</h1>"%(t,dt)
    elif t>0:
        resp="<h1>Current Date and Time ahead by %d hrs is %s</h1>"%(t,dt)
    else:
        resp="<h1>There is no change in current date and time</h1>"
    return HttpResponse(resp)
```

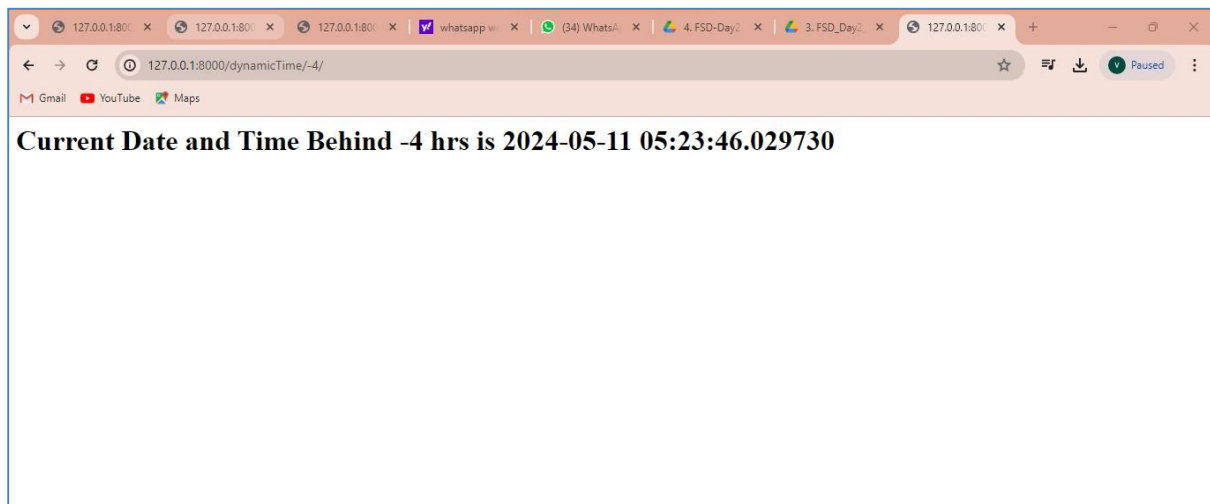
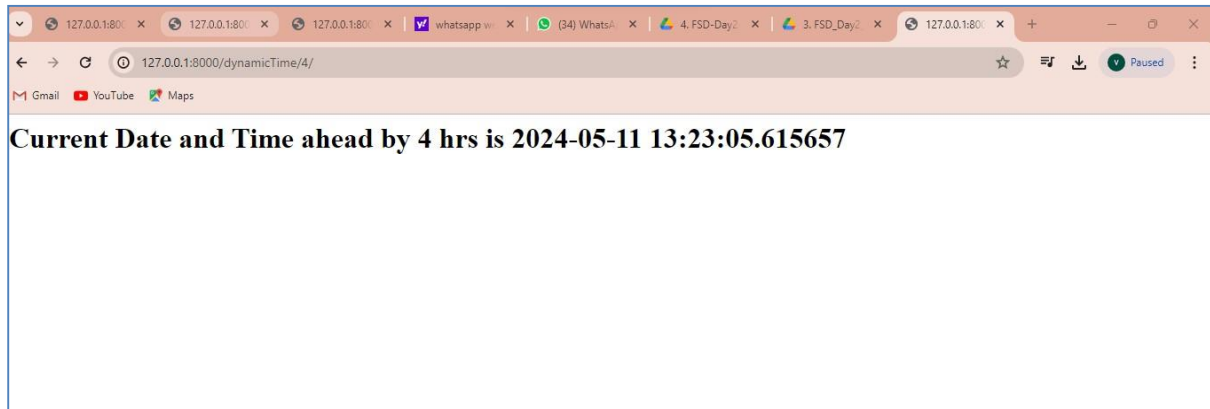
### **urls.py**

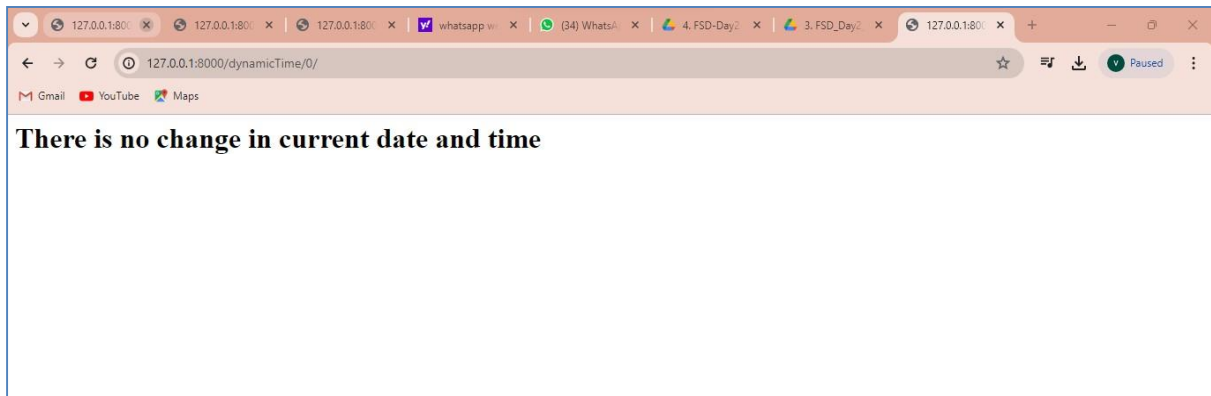
```
from django.contrib import admin
from django.urls import path

from scdt_dynamicApp.views import scdt
```

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('dynamicTime/<str:s>/', scdt),  
]
```

## OUTPUT





**PROGRAM 3**

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

**fruits\_student.html**

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #a1 { background-color: lightblue; color: brown }
      #a2 { background-color: blue; color: yellow }
    </style>
    <title>
      Unordered Fruits and Ordered Students
    </title>
  </head>
  <body>
    <h1 id="a1">Unordered List of Fruits</h1>
    <ul>
      {% for fruit in fruitList %}
        <li>{{ fruit }}</li>
      {% endfor %}
    </ul>
    <h1 id="a2">Ordered List of Students Selected for an Event</h1>
    <ol>
      {% for student in studentList %}
        <li>{{ student }}</li>
      {% endfor %}
    </ol>
  </body>
</html>
```

**views.py**

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

# Create your views here.

```
def fruit_student(request):  
    fruitList=['Mango','Kiwi','Banana','Apple','Grapes']  
    studentList=['Rama','Chetan','Kumar','Harish','Geetha']  
    return  
render(request,'fruit_student.html',{ 'fruitList':fruitList,'studentList':sorted(studentList)})
```

### **urls.py**

```
from django.contrib import admin  
from django.urls import path  
from FruitsApp.views import fruit_student  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('fruits/',fruit_student),  
]
```

### **settings.py (only one change inside installed apps add fruitsapp)**

```
from pathlib import Path  
  
# Build paths inside the project like this: BASE_DIR / 'subdir'.  
BASE_DIR = Path(__file__).resolve().parent.parent  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET_KEY = 'django-insecure-ox% @6r-  
9t3%m$c&8yr(ox$2ktr!dwtl!rwr$g$*tyb6=z=9w+$'
```

```
# SECURITY WARNING: don't run with debug turned on in production!
```

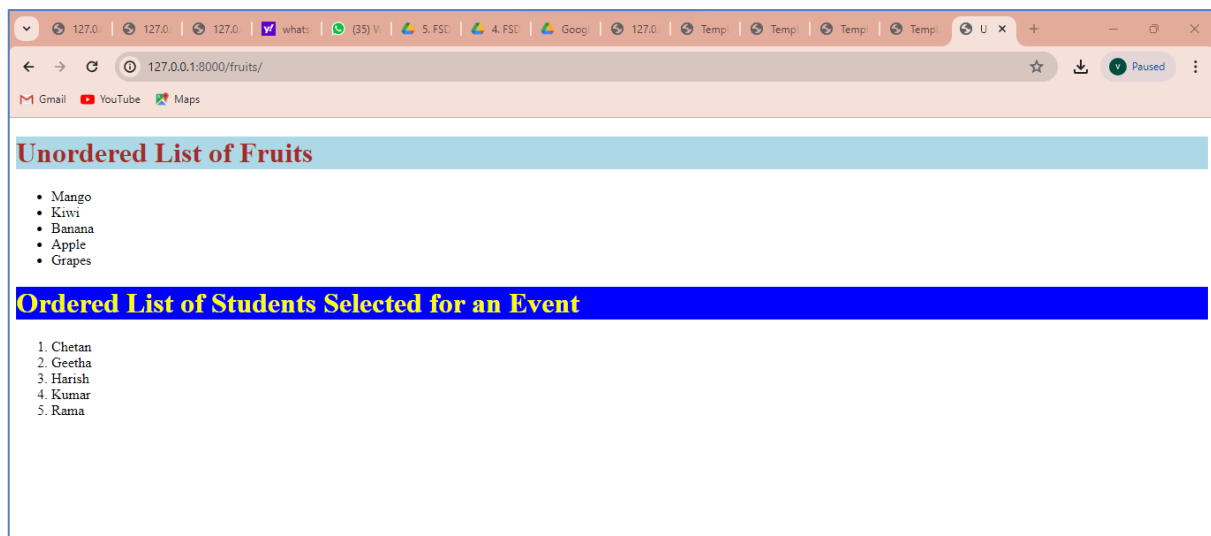
```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

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

## OUTPUT



**PROGRAM . 4**

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.

**layout.html**

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      nav{ background-color: lightblue;padding: 15px;}
    </style>
    <title>
      { % block title % } { % endblock % }
    </title>
  </head>
  <body>
    <nav>
      <a href="/home/">HOME</a>
      <a href="/contactus/">CONTACT US</a>
      <a href="/aboutus/">ABOUT US</a>

    </nav>
    <section>
      { % block content % } { % endblock % }
    </section>
    <footer>
      <hr>
      &copy; Designed and Developed by Mr. Sandeep Chavan, CSE, JCE, BELAGAVI
    </footer>
  </body>
</html>
```

**home.html**

```
{% extends 'layout.html' %}
{% block title %} HOME Page {% endblock %}
{% block content %}
<h1>This is my home page</h1>
{% endblock %}
```

**about.html**

```
{% extends 'layout.html' %}
{% block title %} ABOUT PAGE {% endblock %}
{% block content %}
<h1>About Us</h1>
<p>Mr. Sandeep Chavan, Asso. Prof, Dept of CSE, JCE</p>

{% endblock %}
```

**contactus.html**

```
{% extends 'layout.html' %}
{% block title %} Contact us {% endblock %}
{% block content %}
<h1>Contact us</h1>
<p>Name: Mr. Sandeep Chavan</p>
<p>Designation:Asso. Prof </p>
<p>Mobile: 9591221406</p>
<p>Email: sandeepchavan014@gmail.com</p>
{% endblock %}
```

**views.py**

```
from django.shortcuts import render

# Create your views here.

def home(request):
    return render(request,'home.html')
```



```
def contactus(request):  
    return render(request,'contactus.html')  
def aboutus(request):  
    return render(request,'about.html')
```

**urls.py**

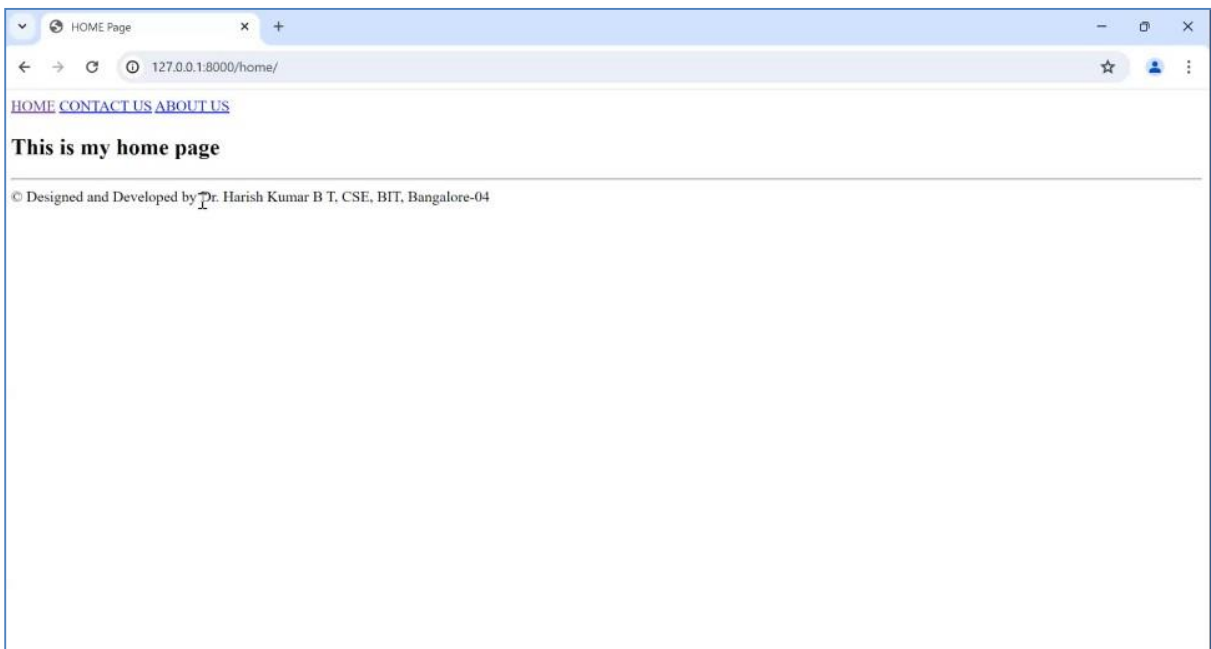
```
from django.contrib import admin  
from django.urls import path  
  
from layoutApp.views import aboutus, contactus, home  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path("",home),  
    path('contactus/',contactus),  
    path('aboutus/',aboutus),  
    path('home/',home),  
]
```

**settings.py (only one change inside installed apps add layoutapp)**

```
# Application definition
```

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

**OUTPUT**





**PROGRAM . 5**

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

**basicTemplate.html**

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      nav{background-color: lightblue;padding: 15px; }
      nav a {
        color: #fff; /* Text color */
        text-decoration: none; /* Remove underline */
        padding: 10px 20px; /* Padding around each link */
        margin: 0px 10px; /* Spacing between links */
        border-radius: 5px; /* Rounded corners */
        background-color: #555;
        flex-wrap: wrap;
      }

      nav a:hover {
        background-color:aqua; /* Background color on hover */
      }

      ul {
        list-style: none;
        margin: 0;
        padding: 0;
        display: flex; /* Use flexbox */
        flex-wrap: wrap; /* Allow items to wrap to the next line */
        flex-direction: row; /* Display items in a column */
      }

      li {
        margin-right: 20px;
```

```

        margin-bottom: 25px;
    }

</style>
<title>
    { % block title % } { % endblock % }
</title>
</head>

<body>
    <center> <h1 style="background-color: blue;color:yellow"> STUDENT COURSE
REGISTRATION PORTAL</h1></center>

    <nav>
        <ul>
            <li> <a href="/home/">HOME</a></li>
            <li> <a href="/studentlist/">STUDENT LIST</a></li>
            <li> <a href="/courselist/">COURSE LIST</a> </li>
            <li> <a href="/register/">REGISTER</a></li>
            <li> <a href="/enrolledlist/">ENROLLED LIST</a></li>
            <li> <a href="/addproject/">ADD PROJECT</a></li>
            <li><a href="/genericlistviewstudent/">GENERIC STUDENT LIST VIEW</a></li>
            <li> <a href="/download_course_table_as_csv/">DOWNLOAD COURSE AS
CSV</a> </li>
            <li> <a href="/download_course_table_as_pdf/">DWONLOAD COURSE AS
PDF</a></li>
        </ul>
    </nav>

    <section>
        { % block content % } { % endblock % }
    </section>

    <footer>
        <hr/>
        <center>

```

&copy; Designed and Developed by Mr. Sandeep Chavan, Dept. of CSE, JCE,  
Belagavi

</center>

</footer>

</body>

</html>

### **home.html**

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

```
{% block title %} Home Page {% endblock %}
```

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

```
<li>Click on Student List to get the List of students</li>
```

```
<li> Click on Course List to get the list of courses</li>
```

```
<li>click on register to enroll student to a course</li>
```

```
{% endblock %}
```

### **studentlist.html**

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

```
{% block title %} Student List {% endblock %}
```

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

```
<h1>Student List</h1>
```

```
<table border="1">
```

```
<tr>
```

```
<th>
```

```
USN
```

```
</th>
```

```
<th>
```

```
NAME
```

```
</th>
```

```
<th>
```

SEM

</th>

</tr>

{% for s in student\_list %}

<tr>

<td>{{ s.usn }}</td>

<td>{{ s.name }}</td>

<td>{{ s.sem }}</td>

</tr>

{% endfor %}

</table>

{% endblock %}

### **courselist.html**

{% extends 'basicTemplate.html' %}

{% block title %} Course List {% endblock %}

{% block content %}

<h1> Course List</h1>

<table border="1">

<tr>

<th>

Sub Code

</th>

<th>

Sub Name

</th>

<th>

Credits

</th>

```

</tr>

{% for c in course_list %}
<tr>
  <td>{{c.courseCode}}</td>
  <td>{{c.courseName}}</td>
  <td>{{c.courseCredits}}</td>
</tr>
{% endfor %}

</table>

{% endblock %}

```

### enrolledlist.html

```

{% extends 'basicTemplate.html' %}
{% block title %} Course Registration Details {% endblock %}

{% block content %}

<form method="POST" action="">
  {% csrf_token %}
  Select Course:
  <select name="course">
    {% for c in Course_List %}
    <option value="{{c.id}}">{{c.courseCode}}</option>
    {% endfor %}
  </select>
  <input type="submit" value="Search"/>
  {% if student_list %}
  <h1> List of Students registered of the course {{course.courseCode}}</h1>
  <table border="1">

```



```
<tr>
  <th>
    USN
  </th>
  <th>
    NAME
  </th>
  <th>
    SEM
  </th>
</tr>

{% for s in student_list %}
<tr>
  <td>{{ s.usn }}</td>
  <td>{{ s.name }}</td>
  <td>{{ s.sem }}</td>
</tr>
{% endfor %}
</table>
{% endif %}
</form>

{% endblock %}
```

### **register.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Course Register Page {% endblock %}

{% block content %}
<h1> Student Course Registration</h1>
<form method="POST" action="">
  {% csrf_token %}
  Select USN:
```

```
<select name="student">
    {% for s in student_list %}
    <option value="{{ s.id }}">{{ s.usn }}</option>
    {% endfor %}
</select>
```

Select Course:

```
<select name="course">
    {% for c in course_list %}
    <option value="{{ c.id }}">{{ c.courseCode }}</option>
    {% endfor %}
</select>

<input type="submit" value="ENROLL"/>
</form>

{% endblock %}
```

### **models.py**

```
from django.db import models
from django.forms import ModelForm
```

# Create your models here.

```
class course(models.Model):
    courseCode=models.CharField(max_length=10)
    courseName=models.CharField(max_length=50)
    courseCredits=models.IntegerField()

    def __str__(self):
        return self.courseCode+" "+self.courseName+" "+str(self.courseCredits)
```

```
class student(models.Model):
    usn=models.CharField(max_length=10)
    name=models.CharField(max_length=40)
```

```

sem=models.IntegerField()

courses=models.ManyToManyField(course,related_name='student_set')

def __str__(self):

    return self.usn+" "+self.name+" "+str(self.sem)

```

**After writing models.py run the below commands in VS code terminal.**

**python manage.py makemigrations**

**python manage.py migrate**

Open python interactive console in VS code terminal by giving the following command  
python manage.py shell

```

PS D:\BIT_CSE_FDP\studCourseReg> python manage.py shell
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> 

```

In interactive python console import the model student and course as shown below  
from studCourseRegApp.models import student,course

```

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from studCourseRegApp.models import student,course
>>> 

```

Create student objects as given below in the interactive python console

```

>>>s1=student(usn= '1BI21CS001',name= 'Harish', sem=6)
>>>s2=student(usn= '1BI21CS002',name= 'Kumar', sem=6)
>>>s3=student(usn= '1BI21CS003',name= 'Chetan', sem=6)
>>>s4=student(usn= '1BI21CS004',name= 'Rama', sem=6)
>>>s5=student(usn= '1BI21CS005',name= 'Krishna', sem=6)
>>>s6=student(usn= '1BI21CS007',name= 'XYZ', sem=6)

```

```

>>> s1=student(usn= '1BI21CS001',name= 'Harish', sem=6)
>>> s2=student(usn= '1BI21CS002',name= 'Kumar', sem=6)
>>> s3=student(usn= '1BI21CS003',name= 'Chetan', sem=6)
>>> s4=student(usn= '1BI21CS004',name= 'Rama', sem=6)
>>> s5=student(usn= '1BI21CS005',name= 'Krishna', sem=6)
>>> s6=student(usn= '1BI21CS007',name= 'XYZ', sem=6)

```

Make the list of students write a for loop and save each student object the student table as show below

```

studList=[s1,s2,s3,s4,s5,s6]
for stud in studList:

```

```
... stud.save()

>>> studList=[s1,s2,s3,s4,s5,s6]
>>> for stud in studList:
...     stud.save()
...
>>> █
```

Similarly add the following courses to the course table

```
>>>c1=course(courseCode='21CS61',courseName='SE',courseCredits=3)
>>>c2=course(courseCode='21CS62',courseName='FSD',courseCredits=3)
>>>c3=course(courseCode='21CS63',courseName='CGV',courseCredits=3)
>>>c4=course(courseCode='21CS64',courseName='DBMS',courseCredits=3)
>>>c5=course(courseCode='21CSL62',courseName='FSD Lab',courseCredits=2)
>>> courseList=[c1,c2,c3,c4,c5]
>>> for course in courseList:
...     course.save()
...
>>> █

>>> c1=course(courseCode='21CS61',courseName='SE',courseCredits=3)
>>> c2=course(courseCode='21CS62',courseName='FSD',courseCredits=3)
>>> c3=course(courseCode='21CS63',courseName='CGV',courseCredits=3)
>>> c4=course(courseCode='21CS64',courseName='DBMS',courseCredits=3)
>>> c5=course(courseCode='21CSL62',courseName='FSD Lab',courseCredits=2)
>>> courseList=[c1,c2,c3,c4,c5]
>>> for course in courseList:
...     course.save()
...
>>> █
```

Open Sqlite DB in VS code and check the student and Course Table [Every time you update the table close and open the Sqlite DB to view the updated data]

### views.py

```
from django.http import HttpResponse
from django.shortcuts import render
from studCourseRegApp.models import student,course, projectForm

# Create your views here.
def home(request):
    return render(request,'home.html')

def studentlist(request):
```

```
s=student.objects.all()
return render(request,'studentlist.html',{ 'student_list':s})
```

```
def courselist(request):
    c=course.objects.all()
    return render(request,'courselist.html',{ 'course_list':c})
```

```
def register(request):
    if request.method=="POST":
        sid=request.POST.get("student")
        cid=request.POST.get("course")
        studentobj=student.objects.get(id=sid)
        courseobj=course.objects.get(id=cid)
        res=studentobj.courses.filter(id=cid)
        if res:
            resp="<h1>Student with usn %s has already enrolled for the
%s<h1>"%(studentobj.usn,courseobj.courseCode)
            return HttpResponseRedirect(resp)
            studentobj.courses.add(courseobj)
            resp="<h1>student with usn %s successfully enrolled for the course with sub code
%s</h1>"%(studentobj.usn,courseobj.courseCode)
            return HttpResponseRedirect(resp)
        else:
            studentlist=student.objects.all()
            courselist=course.objects.all()
            return render(request,'register.html',{ 'student_list':studentlist,'course_list':courselist})
```

```
def enrolledStudents(request):
    if request.method=="POST":
        cid=request.POST.get("course")
        courseobj=course.objects.get(id=cid)
        studentlistobj=courseobj.student_set.all()
        return render(request,'enrolledlist.html',{ 'course':courseobj,'student_list':studentlistobj})
```

else:

    courselist=course.objects.all()

    return render(request,'enrolledlist.html',{ 'Course\_List':courselist})

### **urls.py**

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

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

```
from studCourseRegApp.views import home, studentlist,courselist,register,enrolledStudents
```

```
urlpatterns = [
```

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

```
    path("",home),
```

```
    path('home/',home),
```

```
    path('studentlist/',studentlist),
```

```
    path('courselist/',courselist),
```

```
    path('register/',register),
```

```
    path('enrolledlist/',enrolledStudents),
```

```
]
```

### **settings.py(only one change inside installed apps add studCourseRegApp )**

```
# Application definition
```

```
INSTALLED_APPS = [
```

```
    'django.contrib.admin',
```

```
    'django.contrib.auth',
```

```
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
```

```
    'studCourseRegApp',
```

```
]
```

### **OUTPUT**

Student List

127.0.0.1:8000/studentlist/

### STUDENT REGISTRATION PORTAL

HOME STUDENT LIST COURSE LIST REGISTER ENROLLED LIST ADD PROJECT GENERIC STUDENT LIST VIEW

DOWNLOAD COURSE AS CSV DWONLOAD COURSE AS PDF Course register using Ajax Call

#### Student List

USN	Name	Sem
2ji22cs001	sandy	6
2ji22cs002	kumar	6
2ji22cs003	Rama	6
2ji22cs004	Neha	6
2ji22cs005	Praveen	6

@copy;Designed and Developed by Prof. Sandeep Chavan, bgm

Course List

127.0.0.1:8000/courselist/

### STUDENT REGISTRATION PORTAL

HOME STUDENT LIST COURSE LIST REGISTER ENROLLED LIST ADD PROJECT GENERIC STUDENT LIST VIEW

DOWNLOAD COURSE AS CSV DWONLOAD COURSE AS PDF Course register using Ajax Call

#### Course List

Sub Code	Sub Name	Credits
21cs61	SE	3
21cs62	FSD	3
21cs63	CG	3
21cs64	AG	3

@copy;Designed and Developed by Prof. Sandeep Chavan, bgm

127.0.0.1:8000

### STUDENT REGISTRATION PORTAL

HOME STUDENT LIST COURSE LIST REGISTER ENROLLED LIST ADD PROJECT GENERIC STUDENT LIST VIEW

DOWNLOAD COURSE AS CSV DWONLOAD COURSE AS PDF Course register using Ajax Call

- Click on student list to get the list of students
- Click on course list to get the course of courses
- click on register to enroll student to course

@copy;Designed and Developed by Prof. Sandeep Chavan, bgm



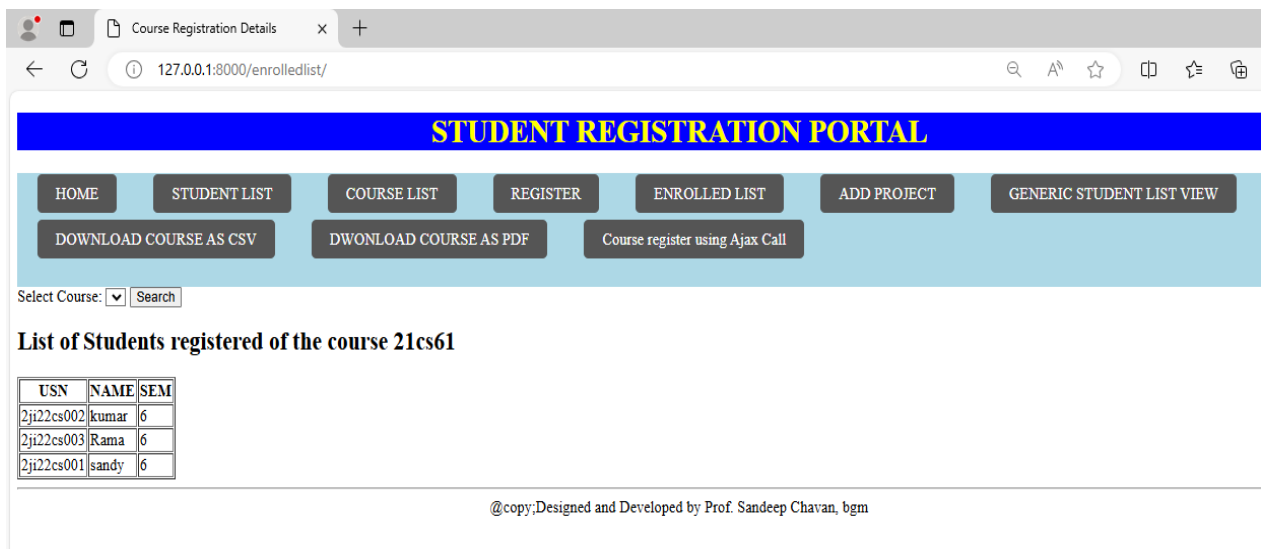
## STUDENT REGISTRATION PORTAL

HOME STUDENT LIST COURSE LIST REGISTER ENROLLED LIST ADD PROJECT GENERIC STUDENT LIST VIEW  
DOWNLOAD COURSE AS CSV DWONLOAD COURSE AS PDF Course register using Ajax Call

### Student Course Registration

Select USN:  Select Course:

@copy;Designed and Developed by Prof. Sandeep Chavan, bgm



@copy;Designed and Developed by Prof. Sandeep Chavan, bgm



**PROGRAM 6**

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

**basicTemplate.html**

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <style>
```

```
      nav{background-color: lightblue;padding: 15px; }
```

```
      nav a {
```

```
        color: #fff; /* Text color */
```

```
        text-decoration: none; /* Remove underline */
```

```
padding: 10px 20px; /* Padding around each link */
margin: 0px 10px; /* Spacing between links */
border-radius: 5px; /* Rounded corners */
background-color: #555;
flex-wrap: wrap;

}

nav a:hover {
    background-color:aqua; /* Background color on hover */
}

ul {
    list-style: none;
    margin: 0;
    padding: 0;
    display: flex; /* Use flexbox */
    flex-wrap: wrap; /* Allow items to wrap to the next line */
    flex-direction: row; /* Display items in a column */
}

li {
    margin-right: 20px;
    margin-bottom: 25px;
}

</style>
<title>
    { % block title % } { % endblock % }
</title>
</head>

<body>
    <center> <h1 style="background-color: blue;color:yellow"> STUDENT COURSE
REGISTRATION PORTAL</h1></center>

    <nav>
```

```

        <ul>
        <li> <a href="/home/">HOME</a></li>
        <li> <a href="/studentlist/">STUDENT LIST</a></li>
        <li> <a href="/courselist/">COURSE LIST</a> </li>
        <li> <a href="/register/">REGISTER</a></li>
        <li> <a href="/enrolledlist/">ENROLLED LIST</a></li>
        <li> <a href="/addproject/">ADD PROJECT</a></li>
        <li><a href="/genericlistviewstudent/">GENERIC STUDENT LIST VIEW</a></li>
        <li> <a href="/download_course_table_as_csv/">DOWNLOAD COURSE AS
CSV</a> </li>
        <li> <a href="/download_course_table_as_pdf/">DWONLOAD COURSE AS
PDF</a></li>
        </ul>
    </nav>
    <section>
        { % block content % } { % endblock % }
    </section>
    <footer>
        <hr/>
        <center>
            &copy; Designed and Developed by Prof. Sandeep Chavan, Dept. of CSE, JCE
        </center>
    </footer>

</body>
</html>

```

### home.html

```

{ % extends 'basicTemplate.html' % }
{ % block title % } Home Page { % endblock % }

{ % block content % }

```

```
<li>Click on Student List to get the List of students</li>
<li> Click on Course List to get the list of courses</li>
<li>click on register to enroll student to a course</li>
```

```
{% endblock % }
```

### **studentlist.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Student List {% endblock %}
{% block content%}
<h1>Student List</h1>
<table border="1">
  <tr>
    <th>
      USN
    </th>
    <th>
      NAME
    </th>
    <th>
      SEM
    </th>
  </tr>

  {% for s in student_list %}
  <tr>
    <td>{{ s.usn }}</td>
    <td>{{ s.name }}</td>
    <td>{{ s.sem }}</td>
  </tr>
  {% endfor %}

</table>
{% endblock % }
```

**courselist.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Course List {% endblock %}
```

```
{% block content %}
<h1> Course List</h1>
<table border="1">
  <tr>
    <th>
      Sub Code
    </th>
    <th>
      Sub Name
    </th>
    <th>
      Credits
    </th>
  </tr>

  {% for c in course_list %}
  <tr>
    <td>{{ c.courseCode }}</td>
    <td>{{ c.courseName }}</td>
    <td>{{ c.courseCredits }}</td>
  </tr>
  {% endfor %}

</table>

{% endblock %}
```

**enrolledlist.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Course Registration Details {% endblock %}

{% block content %}

<form method="POST" action="">
    {% csrf_token %}
    Select Course:
    <select name="course">
        {% for c in Course_List %}
            <option value="{{ c.id }}">{{ c.courseCode }}</option>
        {% endfor %}
    </select>
    <input type="submit" value="Search"/>
    {% if student_list %}
        <h1> List of Students registered of the course {{ course.courseCode }}</h1>
        <table border="1">

            <tr>
                <th>
                    USN
                </th>
                <th>
                    NAME
                </th>
                <th>
                    SEM
                </th>
            </tr>

            {% for s in student_list %}
                <tr>
```

```
<td>{{ s.usn }}</td>
<td>{{ s.name }}</td>
<td>{{ s.sem }}</td>
</tr>
{% endfor %}
</table>
{% endif %}
</form>
```

```
{% endblock %}
```

### **register.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Course Register Page {% endblock %}

{% block content %}
<h1> Student Course Registration</h1>
<form method="POST" action="">
  {% csrf_token %}
  Select USN:
  <select name="student">
    {% for s in student_list %}
    <option value="{{ s.id }}">{{ s.usn }}</option>
    {% endfor %}
  </select>

  Select Course:
  <select name="course">
    {% for c in course_list %}
    <option value="{{ c.id }}">{{ c.courseCode }}</option>
    {% endfor %}
  </select>

  <input type="submit" value="ENROLL"/>
</form>
```

```
{% endblock %}
```

### **projectreg.html**

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

{% block title %} Project Details Registration {% endblock %}

{% block content %}

<form method="POST" action="">

    {% csrf_token %}

    <table border="1">
        {{ form.as_table }}
    <tr>
        <td>
            <input type="submit" value="Add Project"/>
        </td>
    </tr>
    </table>
</form>

{% endblock %}
```

### **admin.py**

```
from django.contrib import admin

from studCourseRegApp.models import student,course


# Register your models here.


#admin.site.register(student)
#admin.site.register(course)


admin.site.site_header='FS ON Django'
admin.site.site_title='FS ON Django'
```



```
@admin.register(student)

class studentAdmin(admin.ModelAdmin):
    list_display=('usn','name')
    ordering=('usn',)
    search_fields=('name',)

@admin.register(course)

class courseAdmin(admin.ModelAdmin):
    list_display=('courseCode','courseName')
    ordering=('courseCode',)
    search_fields=('courseName',)
```

### **models.py**

```
from django.db import models
from django.forms import ModelForm

# Create your models here.

class course(models.Model):
    courseCode=models.CharField(max_length=10)
    courseName=models.CharField(max_length=50)
    courseCredits=models.IntegerField()

    def __str__(self):
        return self.courseCode+" "+self.courseName+" "+str(self.courseCredits)

class student(models.Model):
    usn=models.CharField(max_length=10)
    name=models.CharField(max_length=40)
    sem=models.IntegerField()
    courses=models.ManyToManyField(course,related_name='student_set')

    def __str__(self):
        return self.usn+" "+self.name+" "+str(self.sem)
```

```
class projectReg(models.Model):
    student=models.ForeignKey(student,on_delete=models.CASCADE)
    ptitle=models.CharField(max_length=30)
    planguage=models.CharField(max_length=30)
    pduration=models.IntegerField()
```

```
class projectForm(ModelForm):
    required_css_class="required"
    class Meta:
        model=projectReg
        fields=['student','ptitle','planguage','pduration']
```

**After writing models.py run the below commands in VS code terminal.**

**python manage.py makemigrations**

**python manage.py migrate**

```
import django.db.models.deletion
from django.db import migrations, models
```

```
class Migration(migrations.Migration):

    dependencies = [
        ('studCourseRegApp', '0001_initial'),
    ]

    operations = [
        migrations.CreateModel(
            name='projectReg',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),

```

```
        ('ptitle', models.CharField(max_length=30)),
        ('planguage', models.CharField(max_length=30)),
        ('pduration', models.IntegerField()),
        ('student', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='studCourseRegApp.student')),
    ],
),
]
```

**After writing models.py run the below commands in VS code terminal.**

**python manage.py makemigrations**

**python manage.py migrate**

**views.py**

```
from django.http import HttpResponse
from django.shortcuts import render
from studCourseRegApp.models import student, course, projectForm
```

```
# Create your views here.
```

```
def home(request):
    return render(request, 'home.html')
```

```
def studentlist(request):
    s=student.objects.all()
    return render(request, 'studentlist.html', {'student_list':s})
```

```
def courselist(request):
    c=course.objects.all()
    return render(request, 'courselist.html', {'course_list':c})
```

```
def register(request):
    if request.method=="POST":
        sid=request.POST.get("student")
        cid=request.POST.get("course")
```

```
studentobj=student.objects.get(id=sid)
courseobj=course.objects.get(id=cid)
res=studentobj.courses.filter(id=cid)
if res:
    resp="<h1>Student with usn %s has already enrolled for the
%s<h1>"%(studentobj.usn,courseobj.courseCode)
    return HttpResponse(resp)
studentobj.courses.add(courseobj)
resp="<h1>student with usn %s successfully enrolled for the course with sub code
%s</h1>"%(studentobj.usn,courseobj.courseCode)
    return HttpResponse(resp)
else:
    studentlist=student.objects.all()
    courselist=course.objects.all()
    return render(request,'register.html',{'student_list':studentlist,'course_list':courselist})

def enrolledStudents(request):
    if request.method=="POST":
        cid=request.POST.get("course")
        courseobj=course.objects.get(id=cid)
        studentlistobj=courseobj.student_set.all()
        return render(request,'enrolledlist.html',{'course':courseobj,'student_list':studentlistobj})

    else:
        courselist=course.objects.all()
        return render(request,'enrolledlist.html',{'Course_List':courselist})

def add_project(request):
    if request.method=="POST":
        form=projectForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponse("<h1>Project Data Successfully saved</h1>")
        else:
```

```
        return HttpResponse("<h1>Project details not saved</h1>")
    else:
        form=projectForm()
        return render(request, "projectReg.html",{ 'form':form})
```

**urls.py**

```
from django.contrib import admin
from django.urls import path
from studCourseRegApp.views import enrolledStudentsUsingAjax, generateCSV, home,
registerAjax, studentlist,courselist,register,enrolledStudents,add_project,
urlpatterns = [
    path('secretadmin/', admin.site.urls),
    path('',home),
    path('home/',home),
    path('studentlist/',studentlist),
    path('courselist/',courselist),
    path('register/',register),
    path('enrolledlist/',enrolledStudents),
    path('addproject/',add_project),

]
```

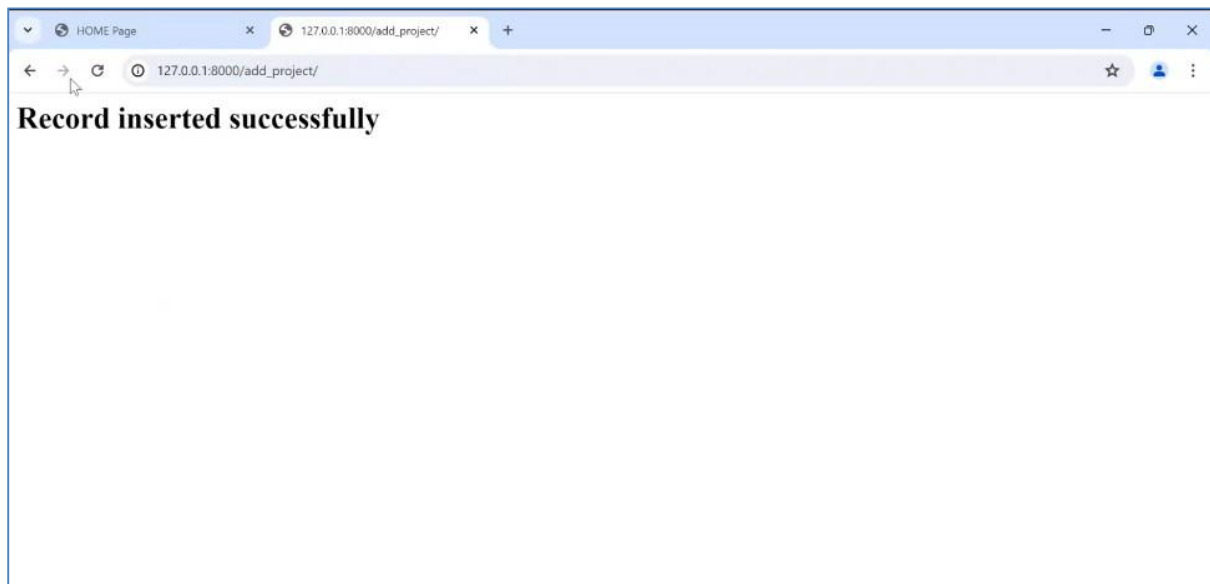
**settings.py(only one change inside installed apps add studCourseRegApp )**

# Application definition

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

## OUTPUT

The screenshot displays a web browser window with two tabs: 'HOME Page' and 'Add Project'. The address bar shows the URL '127.0.0.1:8000/add\_project/'. The page features a blue header with the text 'STUDENT COURSE REGISTRATION PORTAL' in yellow. Below the header is a navigation bar with buttons for 'HOME', 'STUDENT LIST', 'COURSE LIST', 'REGISTER', 'COURSE REGISTERED LIST', and 'ADD PROJECT'. The 'ADD PROJECT' button is highlighted. The main content area contains a registration form with the following fields: 'Student:' (a dropdown menu), 'Propic:' (a text input field), 'Plangauges:' (a text input field), and 'Pduration:' (a text input field). A 'Submit' button is located below the 'Pduration:' field. At the bottom of the page, there is a copyright notice: '© Designed and Developed by Dr. Harish Kumar B T, Dept. of CSE, BIT, Bangalore-04'.



**PROGRAM : 7**

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

**basicTemplate.html**

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      nav{background-color: lightblue;padding: 15px; }
      nav a {
color: #fff; /* Text color */
text-decoration: none; /* Remove underline */
padding: 10px 20px; /* Padding around each link */
margin: 0px 10px; /* Spacing between links */
border-radius: 5px; /* Rounded corners */
background-color: #555;
flex-wrap: wrap;

      }
    </style>
  </head>
</html>
```

```
nav a:hover {
    background-color:aqua; /* Background color on hover */
}
ul {
    list-style: none;
    margin: 0;
    padding: 0;
    display: flex; /* Use flexbox */
    flex-wrap: wrap; /* Allow items to wrap to the next line */
    flex-direction: row; /* Display items in a column */
}
li {
    margin-right: 20px;
    margin-bottom: 25px;
}

</style>
<title>
    { % block title % } { % endblock % }
</title>
</head>

<body>
    <center> <h1 style="background-color: blue;color:yellow"> STUDENT COURSE
REGISTRATION PORTAL</h1></center>
    <nav>
        <ul>
            <li> <a href="/home/">HOME</a></li>
            <li> <a href="/studentlist/">STUDENT LIST</a></li>
            <li> <a href="/courselist/">COURSE LIST</a> </li>
            <li> <a href="/register/">REGISTER</a></li>
            <li> <a href="/enrolledlist/">ENROLLED LIST</a></li>
            <li> <a href="/addproject/">ADD PROJECT</a></li>
```



```

        <li><a href="/genericlistviewstudent/">GENERIC STUDENT LIST VIEW</a></li>
        <li> <a href="/download_course_table_as_csv/">DOWNLOAD COURSE AS
CSV</a> </li>
        <li> <a href="/download_course_table_as_pdf/">DWONLOAD COURSE AS
PDF</a></li>
</ul>
</nav>
<section>
    { % block content % } { % endblock % }
</section>
<footer>
    <hr/>
    <center>
        &copy; Designed and Developeb by Prof. Sandeep Chavan, Dept. of CSE, JCE
    </center>
</footer>

</body>
</html>

```

### home.html

```

{ % extends 'basicTemplate.html' % }
{ % block title % } Home Page { % endblock % }

{ % block content % }

<li>Click on Student List to get the List of students</li>
<li> Click on Course List to get the list of courses</li>
<li>click on register to enroll student to a course</li>

{ % endblock % }

```

**studentlist.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Student List {% endblock %}
{% block content%}
<h1>Student List</h1>
<table border="1">
  <tr>
    <th>
      USN
    </th>
    <th>
      NAME
    </th>
    <th>
      SEM
    </th>
  </tr>

  {% for s in student_list %}
  <tr>
    <td>{{ s.usn }}</td>
    <td>{{ s.name }}</td>
    <td>{{ s.sem }}</td>
  </tr>
  {% endfor %}

</table>
{% endblock %}
```

**courselist.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Course List {% endblock %}
```

```
{% block content%}  
<h1> Course List</h1>  
<table border="1">  
  <tr>  
    <th>  
      Sub Code  
    </th>  
    <th>  
      Sub Name  
    </th>  
    <th>  
      Credits  
    </th>  
  </tr>  
  
  {% for c in course_list %}  
    <tr>  
      <td>{{ c.courseCode }}</td>  
      <td>{{ c.courseName }}</td>  
      <td>{{ c.courseCredits }}</td>  
    </tr>  
  {% endfor %}  
  
</table>  
  
{% endblock %}
```

**enrolledlist.html**

```
{% extends 'basicTemplate.html' %}  
{% block title %} Course Registration Details {% endblock %}  
  
{% block content %}
```

```
<form method="POST" action="">
  {% csrf_token %}
  Select Course:
  <select name="course">
    {% for c in Course_List %}
    <option value="{{ c.id }}">{{ c.courseCode }}</option>
    {% endfor %}
  </select>
  <input type="submit" value="Search"/>
  {% if student_list %}
  <h1> List of Students registered of the course {{ course.courseCode }}</h1>
  <table border="1">

    <tr>
      <th>
        USN
      </th>
      <th>
        NAME
      </th>
      <th>
        SEM
      </th>
    </tr>

    {% for s in student_list %}
    <tr>
      <td>{{ s.usn }}</td>
      <td>{{ s.name }}</td>
      <td>{{ s.sem }}</td>
    </tr>
    {% endfor %}
  </table>
  {% endif %}
```

```
</form>
```

```
{% endblock %}
```

### **register.html**

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

```
{% block title %} Course Register Page {% endblock %}
```

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

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

```
<form method="POST" action="">
```

```
    {% csrf_token %}
```

Select USN:

```
<select name="student">
```

```
    {% for s in student_list %}
```

```
        <option value="{{ s.id }}">{{ s.usn }}</option>
```

```
    {% endfor %}
```

```
</select>
```

Select Course:

```
<select name="course">
```

```
    {% for c in course_list %}
```

```
        <option value="{{ c.id }}">{{ c.courseCode }}</option>
```

```
    {% endfor %}
```

```
</select>
```

```
<input type="submit" value="ENROLL"/>
```

```
</form>
```

```
{% endblock %}
```

### **projectreg.html**

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

```
    {% block title %} Project Details Registration {% endblock %}
```

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

```
        <form method="POST" action="">
```

```
        {% csrf_token %}
<table border="1">
    {{ form.as_table }}
<tr>
    <td>
        <input type="submit" value="Add Project"/>
    </td>
</tr>
</table>
</form>
{% endblock %}
```

### **models.py**

```
from django.db import models
from django.forms import ModelForm
```

# Create your models here.

```
class course(models.Model):
    courseCode=models.CharField(max_length=10)
    courseName=models.CharField(max_length=50)
    courseCredits=models.IntegerField()

    def __str__(self):
        return self.courseCode+" "+self.courseName+" "+str(self.courseCredits)
```

```
class student(models.Model):
    usn=models.CharField(max_length=10)
    name=models.CharField(max_length=40)
    sem=models.IntegerField()
    courses=models.ManyToManyField(course,related_name='student_set')
    def __str__(self):
        return self.usn+" "+self.name+" "+str(self.sem)
```

```
class projectReg(models.Model):
    student=models.ForeignKey(student,on_delete=models.CASCADE)
    ptitle=models.CharField(max_length=30)
    planguage=models.CharField(max_length=30)
    pduration=models.IntegerField()

class projectForm(ModelForm):
    required_css_class="required"
    class Meta:
        model=projectReg
        fields=['student','ptitle','planguage','pduration']
```

**After writing models.py run the below commands in VS code terminal.**

**python manage.py makemigrations**

**python manage.py migrate**

**views.py**

```
from django.http import HttpResponse
from django.shortcuts import render
from studCourseRegApp.models import student,course, projectForm
```

```
# Create your views here.
```

```
def home(request):
    return render(request,'home.html')
```

```
def studentlist(request):
    s=student.objects.all()
    return render(request,'studentlist.html',{'student_list':s})
```

```
def courselist(request):
    c=course.objects.all()
    return render(request,'courselist.html',{'course_list':c})
```

```
def register(request):
    if request.method=="POST":
        sid=request.POST.get("student")
        cid=request.POST.get("course")
        studentobj=student.objects.get(id=sid)
        courseobj=course.objects.get(id=cid)
        res=studentobj.courses.filter(id=cid)
        if res:
            resp="<h1>Student with usn %s has already enrolled for the
%s<h1>"%(studentobj.usn,courseobj.courseCode)
            return HttpResponse(resp)
            studentobj.courses.add(courseobj)
            resp="<h1>student with usn %s successfully enrolled for the course with sub code
%s</h1>"%(studentobj.usn,courseobj.courseCode)
            return HttpResponse(resp)
        else:
            studentlist=student.objects.all()
            courselist=course.objects.all()
            return render(request,'register.html',{'student_list':studentlist,'course_list':courselist})

def enrolledStudents(request):
    if request.method=="POST":
        cid=request.POST.get("course")
        courseobj=course.objects.get(id=cid)
        studentlistobj=courseobj.student_set.all()
        return render(request,'enrolledlist.html',{'course':courseobj,'student_list':studentlistobj})

    else:
        courselist=course.objects.all()
        return render(request,'enrolledlist.html',{'Course_List':courselist})
```



```
def add_project(request):
    if request.method=="POST":
        form=projectForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect("<h1>Project Data Successfully saved</h1>")
        else:
            return HttpResponseRedirect("<h1>Project details not saved</h1>")
    else:
        form=projectForm()
        return render(request, "projectReg.html",{ 'form':form})
```

**urls.py**

```
from django.contrib import admin
from django.urls import path
from studCourseRegApp.views import enrolledStudentsUsingAjax, generateCSV, home,
registerAjax,
studentlist,courselist,register,enrolledStudents,add_project,StudentListView,StudentDetailView,generatePDF
urlpatterns = [
    path('secretadmin/', admin.site.urls),
    path("",home),
    path('home/',home),
    path('studentlist/',studentlist),
    path('courselist/',courselist),
    path('register/',register),
    path('enrolledlist/',enrolledStudents),
    path('addproject/',add_project),
]
```

**settings.py(only one change inside installed apps add studCourseRegApp )**

## # Application definition

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

## OUTPUT

Project Details Registration x +

127.0.0.1:8000/addproject/

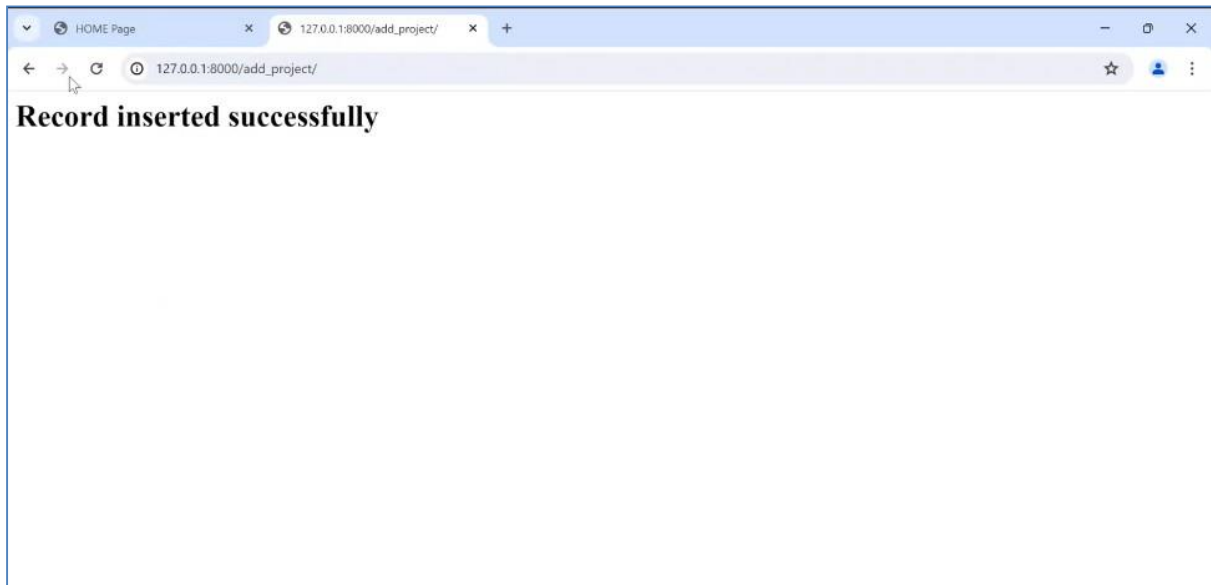
## STUDENT REGISTRATION PORTAL

HOME STUDENT LIST COURSE LIST REGISTER ENROLLED LIST ADD PROJECT GENERIC STUDENT LIST VIEW

DOWNLOAD COURSE AS CSV DWONLOAD COURSE AS PDF Course register using Ajax Call

Student:  Ptitle:  Planguage:  Pduration:  Add Project

@copy;Designed and Developed by Prof. Sandeep Chavan, bgm



**PROGRAM 8**

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

**basicTemplate.html**

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      nav{background-color: lightblue;padding: 15px; }
      nav a {
        color: #fff; /* Text color */
        text-decoration: none; /* Remove underline */
        padding: 10px 20px; /* Padding around each link */
        margin: 0px 10px; /* Spacing between links */
        border-radius: 5px; /* Rounded corners */
        background-color: #555;
        flex-wrap: wrap;
      }

      nav a:hover {
        background-color:aqua; /* Background color on hover */
      }

      ul {
        list-style: none;
        margin: 0;
        padding: 0;
        display: flex; /* Use flexbox */
        flex-wrap: wrap; /* Allow items to wrap to the next line */
      }
```

```

        flex-direction: row; /* Display items in a column */
    }
    li {
        margin-right: 20px;
        margin-bottom: 25px;
    }

</style>
<title>
    { % block title % } { % endblock % }
</title>
</head>

<body>
    <center> <h1 style="background-color: blue;color:yellow"> STUDENT COURSE
REGISTRATION PORTAL</h1></center>

    <nav>
        <ul>
            <li> <a href="/home/">HOME</a></li>
            <li> <a href="/studentlist/">STUDENT LIST</a></li>
            <li> <a href="/courselist/">COURSE LIST</a> </li>
            <li> <a href="/register/">REGISTER</a></li>
            <li> <a href="/enrolledlist/">ENROLLED LIST</a></li>
            <li> <a href="/addproject/">ADD PROJECT</a></li>
            <li><a href="/genericlistviewstudent/">GENERIC STUDENT LIST VIEW</a></li>
            <li> <a href="/download_course_table_as_csv/">DOWNLOAD COURSE AS
CSV</a> </li>
            <li> <a href="/download_course_table_as_pdf/">DWONLOAD COURSE AS
PDF</a></li>
        </ul>
    </nav>

    <section>
        { % block content % } { % endblock % }
    </section>

```

```
<footer>
  <hr/>
  <center>
    &copy; Designed and Developed by Prof. Sandeep Chavan, Dept. of CSE, JCE
  </center>
</footer>

</body>
</html>
```

**home.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Home Page {% endblock %}

{% block content %}

<li>Click on Student List to get the List of students</li>
<li> Click on Course List to get the list of courses</li>
<li>click on register to enroll student to a course</li>

{% endblock %}
```

**studentlist.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Student List {% endblock %}
{% block content %}
<h1>Student List</h1>
<table border="1">
  <tr>
    <th>
      USN
    </th>
```

```
<th>
    NAME
</th>
<th>
    SEM
</th>
</tr>

{% for s in student_list %}
<tr>
    <td>{{ s.usn }}</td>
    <td>{{ s.name }}</td>
    <td>{{ s.sem }}</td>
</tr>
{% endfor %}

</table>
{% endblock %}
```

**courselist.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Course List {% endblock %}
```

```
{% block content %}
<h1> Course List</h1>
<table border="1">
    <tr>
        <th>
            Sub Code
        </th>
        <th>
            Sub Name

```

```

        </th>
        <th>
            Credits
        </th>
    </tr>

    {% for c in course_list %}
    <tr>
        <td>{{ c.courseCode }}</td>
        <td>{{ c.courseName }}</td>
        <td>{{ c.courseCredits }}</td>
    </tr>
    {% endfor %}

</table>

{% endblock %}

enrolledlist.html
{% extends 'basicTemplate.html' %}
{% block title %} Course Registration Details {% endblock %}

{% block content %}

```

```

<form method="POST" action="">
    {% csrf_token %}
    Select Course:
    <select name="course">
        {% for c in Course_List %}
        <option value="{{ c.id }}">{{ c.courseCode }}</option>
        {% endfor %}
    </select>
    <input type="submit" value="Search"/>

```



```
{% if student_list %}
<h1> List of Students registered of the course {{ course.courseCode }}</h1>
<table border="1">

<tr>
  <th>
    USN
  </th>
  <th>
    NAME
  </th>
  <th>
    SEM
  </th>
</tr>

{% for s in student_list %}
<tr>
  <td>{{ s.usn }}</td>
  <td>{{ s.name }}</td>
  <td>{{ s.sem }}</td>
</tr>
{% endfor %}
</table>
{% endif %}
</form>

{% endblock %}
```

**register.html**

```
{% extends 'basicTemplate.html' %}
{% block title %} Course Register Page {% endblock %}

{% block content %}
```

```

<h1> Student Course Registration</h1>
<form method="POST" action="">
  {% csrf_token %}
  Select USN:
  <select name="student">
    {% for s in student_list %}
      <option value="{{ s.id }}">{{ s.usn }}</option>
    {% endfor %}
  </select>

  Select Course:
  <select name="course">
    {% for c in course_list %}
      <option value="{{ c.id }}">{{ c.courseCode }}</option>
    {% endfor %}
  </select>

  <input type="submit" value="ENROLL"/>
</form>
{% endblock %}

```

### GenericListViewStudent.html

```

{% extends 'basicTemplate.html' %}
{% block title %} Generic Student List View {% endblock %}
{% block content %}
{% if student_list %}
<table border="1">
  <tr>
    <th>USN</th>
    <th>Courses Enrolled</th>
  </tr>

  {% for student in student_list %}
  <tr>

```

```

        <td>
            <a href="/genericdetailedviewstudent/{{ student.pk }}">{{ student.usn }}</a>
        </td>
        <td>
            {% for course in student.courses.all %}
            <span>{{ course.courseName }}</span>
            {% endfor %}
        </td>
    </tr>
    {% endfor %}
</table>
{% else %}
<h1>No Students Enrolled</h1>
{% endif %}
{% endblock %}

```

### GenericDetailedViewStudent.html

```

{% extends 'basicTemplate.html' %}
{% block title %} Detailed Student View {% endblock %}
{% block content %}

<h1> Student Name: {{ student.name }}</h1>
<h1>Student USN: {{ student.usn }}</h1>
<h1> Student Sem: {{ student.sem }}</h1>

{% endblock %}

```

### models.py

```

from django.db import models
from django.forms import ModelForm

# Create your models here.

class course(models.Model):

```

```
courseCode=models.CharField(max_length=10)
courseName=models.CharField(max_length=50)
courseCredits=models.IntegerField()

def __str__(self):
    return self.courseCode+" "+self.courseName+" "+str(self.courseCredits)
```

```
class student(models.Model):
    usn=models.CharField(max_length=10)
    name=models.CharField(max_length=40)
    sem=models.IntegerField()
    courses=models.ManyToManyField(course,related_name='student_set')
    def __str__(self):
        return self.usn+" "+self.name+" "+str(self.sem)
```

```
class projectReg(models.Model):
    student=models.ForeignKey(student,on_delete=models.CASCADE)
    ptitle=models.CharField(max_length=30)
    planguage=models.CharField(max_length=30)
    pduration=models.IntegerField()
```

```
class projectForm(ModelForm):
    required_css_class="required"
    class Meta:
        model=projectReg
        fields=['student','ptitle','planguage','pduration']
```

**After writing models.py run the below commands in VS code terminal.**

**python manage.py makemigrations**

**python manage.py migrate**

**views.py**

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

```
from django.shortcuts import render
from studCourseRegApp.models import student,course, projectForm

# Create your views here.
def home(request):
    return render(request,'home.html')

def studentlist(request):
    s=student.objects.all()
    return render(request,'studentlist.html',{ 'student_list':s})

def courselist(request):
    c=course.objects.all()
    return render(request,'courselist.html',{ 'course_list':c})

def register(request):
    if request.method=="POST":
        sid=request.POST.get("student")
        cid=request.POST.get("course")
        studentobj=student.objects.get(id=sid)
        courseobj=course.objects.get(id=cid)
        res=studentobj.courses.filter(id=cid)
        if res:
            resp="<h1>Student with usn %s has already enrolled for the
%s<h1>"%(studentobj.usn,courseobj.courseCode)
            return HttpResponse(resp)
            studentobj.courses.add(courseobj)
            resp="<h1>student with usn %s successfully enrolled for the course with sub code
%s</h1>"%(studentobj.usn,courseobj.courseCode)
            return HttpResponse(resp)
        else:
            studentlist=student.objects.all()
            courselist=course.objects.all()
            return render(request,'register.html',{ 'student_list':studentlist,'course_list':courselist})
```

```
def enrolledStudents(request):
    if request.method=="POST":
        cid=request.POST.get("course")
        courseobj=course.objects.get(id=cid)
        studentlistobj=courseobj.student_set.all()
        return render(request,'enrolledlist.html',{ 'course':courseobj,'student_list':studentlistobj})

    else:
        courselist=course.objects.all()
        return render(request,'enrolledlist.html',{ 'Course_List':courselist})
```

```
def add_project(request):
    if request.method=="POST":
        form=projectForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponse("<h1>Project Data Successfully saved</h1>")
        else:
            return HttpResponse("<h1>Project details not saved</h1>")
    else:
        form=projectForm()
        return render(request, "projectReg.html",{ 'form':form})
```

```
from django.views import generic
class StudentListView(generic.ListView):
    model=student
    template_name="GenericListViewStudent.html"

class StudentDetailView(generic.DetailView):
    model=student
```

```
template_name="GenericDetailedViewStudent.html"
```

**urls.py**

```
from django.contrib import admin
from django.urls import path
from studCourseRegApp.views import enrolledStudentsUsingAjax, generateCSV, home,
registerAjax,
studentlist,courselist,register,enrolledStudents,add_project,StudentListView,StudentDetailView
urlpatterns = [
    path('secretadmin/', admin.site.urls),
    path('',home),
    path('home/',home),
    path('studentlist/',studentlist),
    path('courselist/',courselist),
    path('register/',register),
    path('enrolledlist/',enrolledStudents),
    path('addproject/',add_project),
    path('genericlistviewstudent/',StudentListView.as_view()),
    path('genericdetailedviewstudent/<int:pk>/',StudentDetailView.as_view()),
]
```

**settings.py(only one change inside installed apps add studCourseRegApp )**

```
# Application definition
```

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

```
'studCourseRegApp',  
]
```

## OUTPUT

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/genericlistviewstudent/'. The page title is 'Generic Student List View'. The main content area features a blue header with the text 'STUDENT REGISTRATION PORTAL' in yellow. Below the header is a navigation bar with buttons: HOME, STUDENT LIST, COURSE LIST, REGISTER, ENROLLED LIST, ADD PROJECT, and GENERIC STUDENT LIST VIEW. Underneath the navigation bar are three buttons: DOWNLOAD COURSE AS CSV, DWONLOAD COURSE AS PDF, and Course register using Ajax Call. The main content area displays a table with two columns: USN and Courses Enrolled. The table contains five rows of data. At the bottom of the page, there is a footer that reads '@copy;Designed and Developed by Prof. Sandeep Chavan, bgm'.

USN	Courses Enrolled
2ji22cs001	SE
2ji22cs002	SE
2ji22cs003	SE FSD CG AG
2ji22cs004	FSD CG AG
2ji22cs005	CG

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/genericdetailedviewstudent/3/'. The page title is 'Detailed Student View'. The main content area features a blue header with the text 'STUDENT REGISTRATION PORTAL' in yellow. Below the header is a navigation bar with buttons: HOME, STUDENT LIST, COURSE LIST, REGISTER, ENROLLED LIST, ADD PROJECT, and GENERIC STUDENT LIST VIEW. Underneath the navigation bar are three buttons: DOWNLOAD COURSE AS CSV, DWONLOAD COURSE AS PDF, and Course register using Ajax Call. The main content area displays the following information:  
**Student Name: Rama**  
**Student USN: 2ji22cs003**  
**Student Sem: 6**  
At the bottom of the page, there is a footer that reads '@copy;Designed and Developed by Prof. Sandeep Chavan, bgm'.





**PROGRAM : 9**

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

**basicTemplate.html**

```
<!DOCTYPE html>

<html>

  <head>

    <style>

      nav{background-color: lightblue;padding: 15px; }

      nav a {

color: #fff; /* Text color */

text-decoration: none; /* Remove underline */

padding: 10px 20px; /* Padding around each link */

margin: 0px 10px; /* Spacing between links */

border-radius: 5px; /* Rounded corners */

background-color: #555;

flex-wrap: wrap;

}

      nav a:hover {

        background-color:aqua; /* Background color on hover */

      }

      ul {

        list-style: none;

        margin: 0;

        padding: 0;

        display: flex; /* Use flexbox */

        flex-wrap: wrap; /* Allow items to wrap to the next line */

        flex-direction: row; /* Display items in a column */

      }

      li {

        margin-right: 20px;

        margin-bottom: 25px;

      }

    }

  }

}
```

```

</style>
<title>
    { % block title % } { % endblock % }
</title>
</head>

<body>
    <center> <h1 style="background-color: blue;color:yellow"> STUDENT COURSE
REGISTRATION PORTAL</h1></center>
    <nav>
        <ul>
            <li> <a href="/home/">HOME</a></li>
            <li> <a href="/studentlist/">STUDENT LIST</a></li>
            <li> <a href="/courselist/">COURSE LIST</a> </li>
            <li> <a href="/register/">REGISTER</a></li>
            <li> <a href="/enrolledlist/">ENROLLED LIST</a></li>
            <li> <a href="/addproject/">ADD PROJECT</a></li>
            <li><a href="/genericlistviewstudent/">GENERIC STUDENT LIST VIEW</a></li>
            <li> <a href="/download_course_table_as_csv/">DOWNLOAD COURSE AS
CSV</a> </li>
            <li> <a href="/download_course_table_as_pdf/">DWONLOAD COURSE AS
PDF</a></li>
        </ul>
    </nav>
    <section>
        { % block content % } { % endblock % }
    </section>
    <footer>
        <hr/>
        <center>
            &copy; Designed and Developeb by Prof. Sandeep Chavan, Dept. of CSE, JCE
        </center>

```

```
</footer>
```

```
</body>
```

```
</html>
```

**models.py**

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

```
from django.forms import ModelForm
```

```
# Create your models here.
```

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

```
    courseCode=models.CharField(max_length=10)
```

```
    courseName=models.CharField(max_length=50)
```

```
    courseCredits=models.IntegerField()
```

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

```
        return self.courseCode+" "+self.courseName+" "+str(self.courseCredits)
```

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

```
    usn=models.CharField(max_length=10)
```

```
    name=models.CharField(max_length=40)
```

```
    sem=models.IntegerField()
```

```
    courses=models.ManyToManyField(course,related_name='student_set')
```

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

```
        return self.usn+" "+self.name+" "+str(self.sem)
```

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

```
    student=models.ForeignKey(student,on_delete=models.CASCADE)
```

```
    ptitle=models.CharField(max_length=30)
```

```
    planguage=models.CharField(max_length=30)
```

```
    pduration=models.IntegerField()
```

```
class projectForm(ModelForm):
    required_css_class="required"
    class Meta:
        model=projectReg
        fields=['student','ptitle','planguage','pduration']
```

**After writing models.py run the below commands in VS code terminal.**

**python manage.py makemigrations**

**python manage.py migrate**

**views.py**

```
from django.http import HttpResponse
from django.shortcuts import render
from studCourseRegApp.models import student,course, projectForm
```

# Create your views here.

```
def home(request):
    return render(request,'home.html')
```

```
def studentlist(request):
    s=student.objects.all()
    return render(request,'studentlist.html',{'student_list':s})
```

```
def courselist(request):
    c=course.objects.all()
    return render(request,'courselist.html',{'course_list':c})
```

```
def register(request):
    if request.method=="POST":
        sid=request.POST.get("student")
        cid=request.POST.get("course")
        studentobj=student.objects.get(id=sid)
        courseobj=course.objects.get(id=cid)
        res=studentobj.courses.filter(id=cid)
```

```
        if res:
            resp="<h1>Student with usn %s has already enrolled for the
%s<h1>"%(studentobj.usn,courseobj.courseCode)
            return HttpResponse(resp)
            studentobj.courses.add(courseobj)
            resp="<h1>student with usn %s successfully enrolled for the course with sub code
%s</h1>"%(studentobj.usn,courseobj.courseCode)
            return HttpResponse(resp)
        else:
            studentlist=student.objects.all()
            courselist=course.objects.all()
            return render(request,'register.html',{ 'student_list':studentlist,'course_list':courselist})

def enrolledStudents(request):
    if request.method=="POST":
        cid=request.POST.get("course")
        courseobj=course.objects.get(id=cid)
        studentlistobj=courseobj.student_set.all()
        return render(request,'enrolledlist.html',{ 'course':courseobj,'student_list':studentlistobj})

    else:
        courselist=course.objects.all()
        return render(request,'enrolledlist.html',{ 'Course_List':courselist})

def add_project(request):
    if request.method=="POST":
        form=projectForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponse("<h1>Project Data Successfully saved</h1>")
        else:
            return HttpResponse("<h1>Project details not saved</h1>")
```

```
else:
```

```
    form=projectForm()
```

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

```
from django.views import generic
```

```
class StudentListView(generic.ListView):
```

```
    model=student
```

```
    template_name="GenericListViewStudent.html"
```

```
class StudentDetailView(generic.DetailView):
```

```
    model=student
```

```
    template_name="GenericDetailedViewStudent.html"
```

```
import csv
```

```
def generateCSV(request):
```

```
    courses=course.objects.all()
```

```
    resp=HttpResponse(content_type="text/csv")
```

```
    resp['Content-Disposition']='attachment; filename=course_data.csv'
```

```
    writer=csv.writer(resp)
```

```
    writer.writerow(['Course Code','Course Name','Course Credits'])
```

```
    for c in courses:
```

```
        writer.writerow([c.courseCode,c.courseName,c.courseCredits])
```

```
    return resp
```

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

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

```
def generatePDF(request):
```

```
    courses=course.objects.all()
```

```
    resp=HttpResponse(content_type="text/pdf")
```

```
    resp['Content-Disposition']='attachment; filename=course_data.pdf'
```

```
    pdf=SimpleDocTemplate(resp,pagesize=letter)
```

```
    table_data=[['Course Code','Course Name','Course Credits']]
```

```
for c in courses:
    table_data.append([c.courseCode,c.courseName,str(c.courseCredits)])
table=Table(table_data)
pdf.build([table])
return resp
```

**urls.py**

```
from django.contrib import admin
from django.urls import path
from studCourseRegApp.views import enrolledStudentsUsingAjax, generateCSV, home,
registerAjax,
studentlist,courselist,register,enrolledStudents,add_project,StudentListView,StudentDetailView,generatePDF
urlpatterns = [
    path('secretadmin/', admin.site.urls),
    path("",home),
    path('home/',home),
    path('studentlist/',studentlist),
    path('courselist/',courselist),
    path('register/',register),
    path('enrolledlist/',enrolledStudents),
    path('addproject/',add_project),
    path('genericlistviewstudent/',StudentListView.as_view()),
    path('genericdetailedviewstudent/<int:pk>',StudentDetailView.as_view()),
    path('download_course_table_as_csv/',generateCSV),
    path('download_course_table_as_pdf/',generatePDF),
```



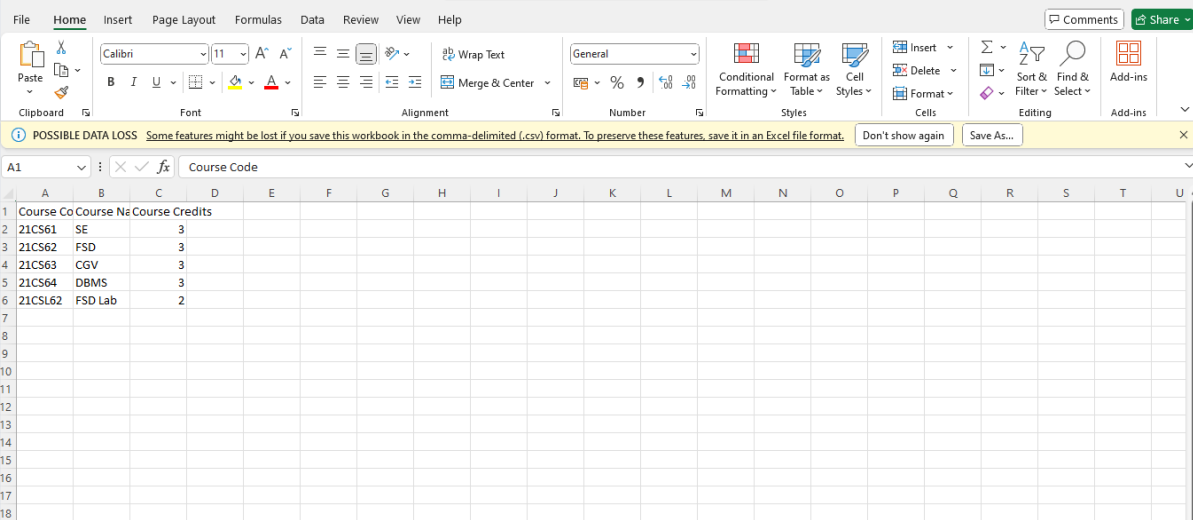
**settings.py(only one change inside installed apps add studCourseRegApp )**

# Application definition

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

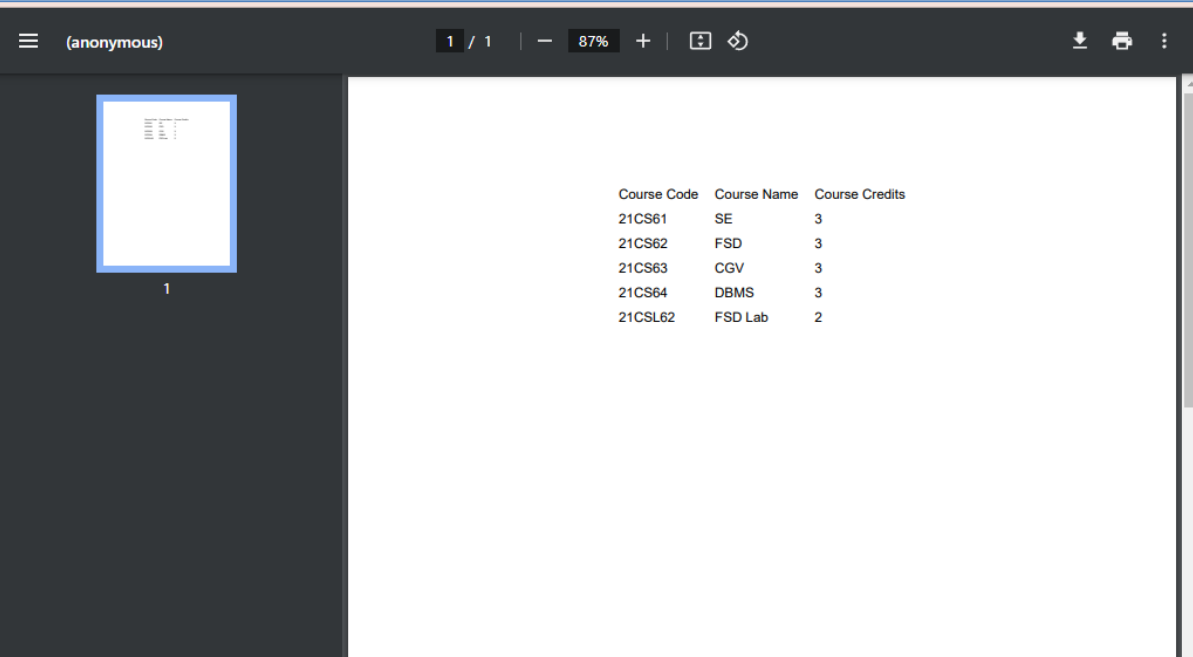
## OUTPUT





POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. Don't show again Save As...

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Course Co	Course Na	Course Credits																		
2	21CS61	SE	3																		
3	21CS62	FSD	3																		
4	21CS63	CGV	3																		
5	21CS64	DBMS	3																		
6	21CSL62	FSD Lab	2																		
7																					
8																					
9																					
10																					
11																					
12																					
13																					
14																					
15																					
16																					
17																					
18																					



(anonymous) 1 / 1 87%

Course Code	Course Name	Course Credits
21CS61	SE	3
21CS62	FSD	3
21CS63	CGV	3
21CS64	DBMS	3
21CSL62	FSD Lab	2

**PROGRAM 10**

**10.** Develop a registration page for student enrolment as done in Module 2 but without pagerefresh using AJAX.

**basicTemplate.html**

```
<!DOCTYPE html>

<html>

  <head>

    <style>

      nav{background-color: lightblue;padding: 15px; }

      nav a {

color: #fff; /* Text color */

text-decoration: none; /* Remove underline */

padding: 10px 20px; /* Padding around each link */

margin: 0px 10px; /* Spacing between links */

border-radius: 5px; /* Rounded corners */

background-color: #555;

flex-wrap: wrap;

}

      nav a:hover {

        background-color:aqua; /* Background color on hover */

      }

      ul {

        list-style: none;

        margin: 0;

        padding: 0;

        display: flex; /* Use flexbox */

        flex-wrap: wrap; /* Allow items to wrap to the next line */

        flex-direction: row; /* Display items in a column */

      }

      li {

        margin-right: 20px;

        margin-bottom: 25px;
```

```

    }

</style>
<title>
    { % block title % } { % endblock % }
</title>
</head>

<body>
    <center> <h1 style="background-color: blue;color:yellow"> STUDENT COURSE
REGISTRATION PORTAL</h1></center>

    <nav>
        <ul>
            <li> <a href="/home/">HOME</a></li>
            <li> <a href="/studentlist/">STUDENT LIST</a></li>
            <li> <a href="/courselist/">COURSE LIST</a> </li>
            <li> <a href="/register/">REGISTER</a></li>
            <li> <a href="/enrolledlist/">ENROLLED LIST</a></li>
            <li> <a href="/addproject/">ADD PROJECT</a></li>
            <li><a href="/genericlistviewstudent/">GENERIC STUDENT LIST VIEW</a></li>
            <li> <a href="/download_course_table_as_csv/">DOWNLOAD COURSE AS
CSV</a> </li>
            <li> <a href="/download_course_table_as_pdf/">DWONLOAD COURSE AS
PDF</a></li>
        </ul>
    </nav>

    <section>
        { % block content % } { % endblock % }
    </section>

    <footer>
        <hr/>
        <center>
            &copy; Designed and Developeb by Prof. Sandeep Chavan, Dept. of CSE, JCE

```

```
</center>

</footer>

</body>
</html>
```

**models.py**

```
from django.db import models
from django.forms import ModelForm

# Create your models here.

class course(models.Model):
    courseCode=models.CharField(max_length=10)
    courseName=models.CharField(max_length=50)
    courseCredits=models.IntegerField()

    def __str__(self):
        return self.courseCode+" "+self.courseName+" "+str(self.courseCredits)

class student(models.Model):
    usn=models.CharField(max_length=10)
    name=models.CharField(max_length=40)
    sem=models.IntegerField()
    courses=models.ManyToManyField(course,related_name='student_set')
    def __str__(self):
        return self.usn+" "+self.name+" "+str(self.sem)

class projectReg(models.Model):
    student=models.ForeignKey(student,on_delete=models.CASCADE)
    ptitle=models.CharField(max_length=30)
    planguage=models.CharField(max_length=30)
    pduration=models.IntegerField()
```

```
class projectForm(ModelForm):
    required_css_class="required"
    class Meta:
        model=projectReg
        fields=['student','ptitle','planguage','pduration']
```

**After writing models.py run the below commands in VS code terminal.**

**python manage.py makemigrations**

**python manage.py migrate**

**views.py**

```
from django.http import HttpResponse
from django.shortcuts import render
from studCourseRegApp.models import student,course, projectForm
```

# Create your views here.

```
def home(request):
    return render(request,'home.html')
```

```
def studentlist(request):
    s=student.objects.all()
    return render(request,'studentlist.html',{'student_list':s})
```

```
def courselist(request):
    c=course.objects.all()
    return render(request,'courselist.html',{'course_list':c})
```

```
def register(request):
    if request.method=="POST":
        sid=request.POST.get("student")
        cid=request.POST.get("course")
        studentobj=student.objects.get(id=sid)
        courseobj=course.objects.get(id=cid)
```

```
res=studentobj.courses.filter(id=cid)
if res:
    resp="<h1>Student with usn %s has already enrolled for the
%s<h1>"%(studentobj.usn,courseobj.courseCode)
    return HttpResponse(resp)
    studentobj.courses.add(courseobj)
    resp="<h1>student with usn %s successfully enrolled for the course with sub code
%s</h1>"%(studentobj.usn,courseobj.courseCode)
    return HttpResponse(resp)
else:
    studentlist=student.objects.all()
    courselist=course.objects.all()
    return render(request,'register.html',{'student_list':studentlist,'course_list':courselist})

def enrolledStudents(request):
    if request.method=="POST":
        cid=request.POST.get("course")
        courseobj=course.objects.get(id=cid)
        studentlistobj=courseobj.student_set.all()
        return render(request,'enrolledlist.html',{'course':courseobj,'student_list':studentlistobj})

    else:
        courselist=course.objects.all()
        return render(request,'enrolledlist.html',{'Course_List':courselist})

def add_project(request):
    if request.method=="POST":
        form=projectForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponse("<h1>Project Data Successfully saved</h1>")
        else:
```

```
        return HttpResponse("<h1>Project details not saved</h1>")
    else:
        form=projectForm()
        return render(request, "projectReg.html",{ 'form':form})


from django.views import generic
class StudentListView(generic.ListView):
    model=student
    template_name="GenericListViewStudent.html"

class StudentDetailView(generic.DetailView):
    model=student
    template_name="GenericDetailedViewStudent.html"

import csv
def generateCSV(request):
    courses=course.objects.all()
    resp=HttpResponse(content_type="text/csv")
    resp['Content-Disposition']='attachment; filename=course_data.csv'
    writer=csv.writer(resp)
    writer.writerow(['Course Code','Course Name','Course Credits'])
    for c in courses:
        writer.writerow([c.courseCode,c.courseName,c.courseCredits])
    return resp

from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Table
def generatePDF(request):
    courses=course.objects.all()
    resp=HttpResponse(content_type="text/pdf")
    resp['Content-Disposition']='attachment; filename=course_data.pdf'
    pdf=SimpleDocTemplate(resp,pagesize=letter)
```



```
table_data=[[ 'Course Code','Course Name','Course Credits']]
for c in courses:
    table_data.append([c.courseCode,c.courseName,str(c.courseCredits)])
table=Table(table_data)
pdf.build([table])
return resp
```

### **urls.py**

```
from django.contrib import admin
from django.urls import path
from studCourseRegApp.views import enrolledStudentsUsingAjax, generateCSV, home,
registerAjax,
studentlist,courselist,register,enrolledStudents,add_project,StudentListView,StudentDetailView,generatePDF
urlpatterns = [
    path('secretadmin/', admin.site.urls),
    path("",home),
    path('home/',home),
    path('studentlist/',studentlist),
    path('courselist/',courselist),
    path('register/',register),
    path('enrolledlist/',enrolledStudents),
    path('addproject/',add_project),
    path('genericlistviewstudent/',StudentListView.as_view()),
    path('genericdetailedviewstudent/<int:pk>',StudentDetailView.as_view()),
    path('download_course_table_as_csv/',generateCSV),
    path('download_course_table_as_pdf/',generatePDF),
    path('courseRegUsingAjax/',registerAjax),
]
```

**settings.py(only one change inside installed apps add studCourseRegApp )**

```
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'studCourseRegApp',
]
```

## OUTPUT

The screenshot shows a web browser window with the title 'Course Reg through AJAX call'. The address bar shows the URL '127.0.0.1:8000/courseRegUsingAjax/'. The page features a blue header with the text 'STUDENT REGISTRATION PORTAL' in yellow. Below the header is a navigation bar with buttons: HOME, STUDENT LIST, COURSE LIST, REGISTER, ENROLLED LIST, ADD PROJECT, and GENERIC STUDENT LIST VIEW. Below the navigation bar is a light blue section with buttons: DOWNLOAD COURSE AS CSV, DWONLOAD COURSE AS PDF, and Course register using Ajax Call. Below this section are two dropdown menus: Student Name (selected: 2ji22cs001) and Course Name (selected: 21cs62). Below the dropdowns is the text 'Student enrolled successfully' and an 'Enroll' button.

Course Reg through AJAX call

127.0.0.1:8000/courseRegUsingAjax/

### STUDENT REGISTRATION PORTAL

HOME STUDENT LIST COURSE LIST REGISTER ENROLLED LIST ADD PROJECT GENERIC STUDENT LIST VIEW

DOWNLOAD COURSE AS CSV DWONLOAD COURSE AS PDF Course register using Ajax Call

Student Name 2ji22cs001 Course Name 21cs62

**Student enrolled successfully**

Enroll

@copy;Designed and Developed by Prof. Sandeep Chavan, bgm



**PROGRAM : 11**

11. Develop a search application in Django using AJAX that displays courses enrolled by a student being searched.

**basicTemplate.html**

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      nav{background-color: lightblue;padding: 15px; }
      nav a {
        color: #fff; /* Text color */
        text-decoration: none; /* Remove underline */
        padding: 10px 20px; /* Padding around each link */
        margin: 0px 10px; /* Spacing between links */
        border-radius: 5px; /* Rounded corners */
        background-color: #555;
        flex-wrap: wrap;
      }

      nav a:hover {
        background-color:aqua; /* Background color on hover */
      }
      ul {
        list-style: none;
        margin: 0;
        padding: 0;
        display: flex; /* Use flexbox */
        flex-wrap: wrap; /* Allow items to wrap to the next line */
        flex-direction: row; /* Display items in a column */
      }
      li {
        margin-right: 20px;
        margin-bottom: 25px;
```

```

    }

</style>
<title>
    { % block title % } { % endblock % }
</title>
</head>

<body>
    <center> <h1 style="background-color: blue;color:yellow"> STUDENT COURSE
REGISTRATION PORTAL</h1></center>

    <nav>
        <ul>
            <li> <a href="/home/">HOME</a></li>
            <li> <a href="/studentlist/">STUDENT LIST</a></li>
            <li> <a href="/courselist/">COURSE LIST</a> </li>
            <li> <a href="/register/">REGISTER</a></li>
            <li> <a href="/enrolledlist/">ENROLLED LIST</a></li>
            <li> <a href="/addproject/">ADD PROJECT</a></li>
            <li><a href="/genericlistviewstudent/">GENERIC STUDENT LIST VIEW</a></li>
            <li> <a href="/download_course_table_as_csv/">DOWNLOAD COURSE AS
CSV</a> </li>
            <li> <a href="/download_course_table_as_pdf/">DWONLOAD COURSE AS
PDF</a></li>
        </ul>
    </nav>

    <section>
        { % block content % } { % endblock % }
    </section>

    <footer>
        <hr/>
        <center>
            &copy; Designed and Developed by Prof. Sandeep Chavan, Dept. of CSE, JCE

```

```
</center>

</footer>

</body>
</html>
```

**models.py**

```
from django.db import models
from django.forms import ModelForm

# Create your models here.

class course(models.Model):
    courseCode=models.CharField(max_length=10)
    courseName=models.CharField(max_length=50)
    courseCredits=models.IntegerField()

    def __str__(self):
        return self.courseCode+" "+self.courseName+" "+str(self.courseCredits)

class student(models.Model):
    usn=models.CharField(max_length=10)
    name=models.CharField(max_length=40)
    sem=models.IntegerField()
    courses=models.ManyToManyField(course,related_name='student_set')
    def __str__(self):
        return self.usn+" "+self.name+" "+str(self.sem)

class projectReg(models.Model):
    student=models.ForeignKey(student,on_delete=models.CASCADE)
    ptitle=models.CharField(max_length=30)
    planguage=models.CharField(max_length=30)
    pduration=models.IntegerField()
```

```
class projectForm(ModelForm):
    required_css_class="required"
    class Meta:
        model=projectReg
        fields=['student','ptitle','planguage','pduration']
```

**After writing models.py run the below commands in VS code terminal.**

**python manage.py makemigrations**

**python manage.py migrate**

**views.py**

```
from django.http import HttpResponse
from django.shortcuts import render
from studCourseRegApp.models import student,course, projectForm
```

# Create your views here.

```
def home(request):
    return render(request,'home.html')
```

```
def studentlist(request):
    s=student.objects.all()
    return render(request,'studentlist.html',{'student_list':s})
```

```
def courselist(request):
    c=course.objects.all()
    return render(request,'courselist.html',{'course_list':c})
```

```
def register(request):
    if request.method=="POST":
        sid=request.POST.get("student")
        cid=request.POST.get("course")
        studentobj=student.objects.get(id=sid)
        courseobj=course.objects.get(id=cid)
```

```
res=studentobj.courses.filter(id=cid)
if res:
    resp="<h1>Student with usn %s has already enrolled for the
%s<h1>"%(studentobj.usn,courseobj.courseCode)
    return HttpResponse(resp)
    studentobj.courses.add(courseobj)
    resp="<h1>student with usn %s successfully enrolled for the course with sub code
%s</h1>"%(studentobj.usn,courseobj.courseCode)
    return HttpResponse(resp)
else:
    studentlist=student.objects.all()
    courselist=course.objects.all()
    return render(request,'register.html',{'student_list':studentlist,'course_list':courselist})

def enrolledStudents(request):
    if request.method=="POST":
        cid=request.POST.get("course")
        courseobj=course.objects.get(id=cid)
        studentlistobj=courseobj.student_set.all()
        return render(request,'enrolledlist.html',{'course':courseobj,'student_list':studentlistobj})

    else:
        courselist=course.objects.all()
        return render(request,'enrolledlist.html',{'Course_List':courselist})

def add_project(request):
    if request.method=="POST":
        form=projectForm(request.POST)
        if form.is_valid():
            form.save()
            return HttpResponse("<h1>Project Data Successfully saved</h1>")
        else:
```



```
        return HttpResponseRedirect("<h1>Project details not saved</h1>")
    else:
        form=projectForm()
        return render(request, "projectReg.html",{ 'form':form})


from django.views import generic
class StudentListView(generic.ListView):
    model=student
    template_name="GenericListViewStudent.html"

class StudentDetailView(generic.DetailView):
    model=student
    template_name="GenericDetailedViewStudent.html"

import csv
def generateCSV(request):
    courses=course.objects.all()
    resp=HttpResponse(content_type="text/csv")
    resp['Content-Disposition']='attachment; filename=course_data.csv'
    writer=csv.writer(resp)
    writer.writerow(['Course Code','Course Name','Course Credits'])
    for c in courses:
        writer.writerow([c.courseCode,c.courseName,c.courseCredits])
    return resp

from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Table
def generatePDF(request):
    courses=course.objects.all()
    resp=HttpResponse(content_type="text/pdf")
    resp['Content-Disposition']='attachment; filename=course_data.pdf'
    pdf=SimpleDocTemplate(resp,pagesize=letter)
```

```
table_data=[[ 'Course Code','Course Name','Course Credits']]
for c in courses:
    table_data.append([c.courseCode,c.courseName,str(c.courseCredits)])
table=Table(table_data)
pdf.build([table])
return resp
```

### **urls.py**

```
from django.contrib import admin
from django.urls import path
from studCourseRegApp.views import enrolledStudentsUsingAjax, generateCSV, home,
registerAjax,
studentlist,courselist,register,enrolledStudents,add_project,StudentListView,StudentDetailView,generatePDF
urlpatterns = [
    path('secretadmin/', admin.site.urls),
    path("",home),
    path('home/',home),
    path('studentlist/',studentlist),
    path('courselist/',courselist),
    path('register/',register),
    path('enrolledlist/',enrolledStudents),
    path('addproject/',add_project),
    path('genericlistviewstudent/',StudentListView.as_view()),
    path('genericdetailedviewstudent/<int:pk>',StudentDetailView.as_view()),
    path('download_course_table_as_csv/',generateCSV),
    path('download_course_table_as_pdf/',generatePDF),
    path('courseRegUsingAjax/',registerAjax),
    path('course_search_ajax/',enrolledStudentsUsingAjax),
]
```

**settings.py(only one change inside installed apps add studCourseRegApp )**

# Application definition

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

## OUTPUT



**STUDENT COURSE REGISTRATION PORTAL**

[HOME](#) [STUDENT LIST](#) [COURSE LIST](#) [REGISTER](#) [ENROLLED LIST](#)  
[ADD PROJECT](#) [GENERIC STUDENT LIST VIEW](#) [DOWNLOAD COURSE AS CSV](#)  
[DOWNLOAD COURSE AS PDF](#) [COURSE REG USING AJAX CALL](#) [COURSE SEARCH USING AJAX](#)

Courses:

**Following is the the list of students enrolled for the course 21CS61**

USN	NAME	SEM
1BI21CS001	Harish	6
1BI21CS002	Kumar	6
1BI21CS003	Chetan	6
1BI21CS004	Rama	6

**PROGRAM : 12**

**12.** Creation of virtual environment, Django project and App should be demonstrated

Create a system root folder with the name **JCE\_CSE\_FD** in the file system (inside any preferred drive).

- Create a project root folder inside **JCE\_CSE\_FD** with the name “**hello\_world**” (assuming we are doing simple hello world program)
- Open **cmd** inside “**hello\_world**”
- Create virtual env inside “**hello\_world**” with the name “**hello\_world\_venv**”

```
python -m venv <name_of_virtual_env>
```

```
python -m venv hello_world_venv
```

**Open project root folder in VS code**

- Run “code .” in the cmd prompt to launch the project folder “hello\_world” in VS code.
- or
- Launch VS code from the task bar, goto file menu navigate and open “hello\_world”

**Command palette (select your venv as python interpreter for your project root folder)**

- In VS Code, open the Command Palette (**View > Command Palette** or (Ctrl+Shift+P)). Then select the **Python: Select Interpreter** command
- The command presents a list of available interpreters that VS Code can locate automatically. From the list, select the virtual environment in your project folder that starts with ./env or .\env:
- Select the virtual environment that is created “hello\_world\_venv”. Look for recommended
- Open VS code terminal and install django framework.

```
pip install django
```

- Check whether django installation is correct

```
python manage.py runserver
```

- Create the django project with the name “hello\_world\_proj” inside the project root folder “hello\_world”

```
django-admin startproject proj .
```

- Create the django app with the name “hello\_world\_app”

```
python manage.py startapp hello_world_app
```

**views.py**

```
from django.http import HttpResponse
from django.shortcuts import render
```

```
# Create your views here.
```

```
def hello(request):
    resp="<h1>Hello World!!!!</h1>"
    return HttpResponse(resp)
```

**urls.py**

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

from hello_world_app.views import hello

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

**OUTPUT**

