**Full Stack Development Laboratory**

# Subject Code: 21CS62 I.A. Marks: 20

**Hours/Week: 02 Total Hours: 20**

# LIST OF PROGRAMS

|  |  |
| --- | --- |
| **Sl.**  **No.** | **Name of Experiment** |
| 1. | 1. Installation of Python, Django and Visual Studio code editors can be demonstrated. 2. Creation of virtual environment, Django project and App should be demonstrated 3. Develop a Django app that displays current date and time in server 4. 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. |
| 2. | 1. Develop a simple Django app that displays an unordered list of fruits and ordered list of selected students for an event 2. 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. 3. 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. |
| 3. | 1. For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms. 2. Develop a Model form for student that contains his topic chosen for project, languages   used and duration with a model called project. |
| 4. | 1. 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. 2. Develop example Django app that performs CSV and PDF generation for any models   created in previous laboratory component. |
| 5. | 1. Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX. 2. Develop a search application in Django using AJAX that displays courses enrolled by a   student being searched. |

# 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 **Mode**l 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 **RNSIT\_CSE\_FDP** in the file system (inside any preferred drive).

 Create a project root folder inside **RNSIT\_CSE\_FDP** 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**” pyton –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

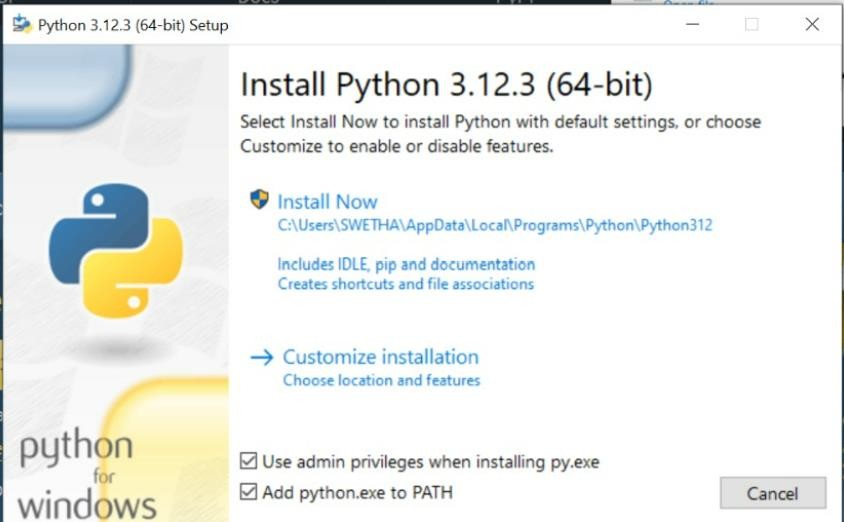
* Check whether django installation is correct python manage.py runserver
* Create the django project with the name “hello\_world\_proj” inside the project root folder “hello\_world”

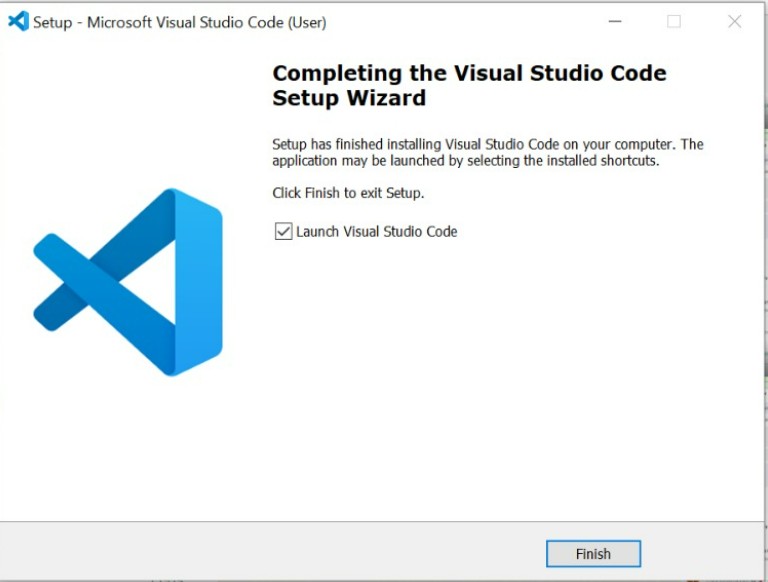
django-admin startproject proj .

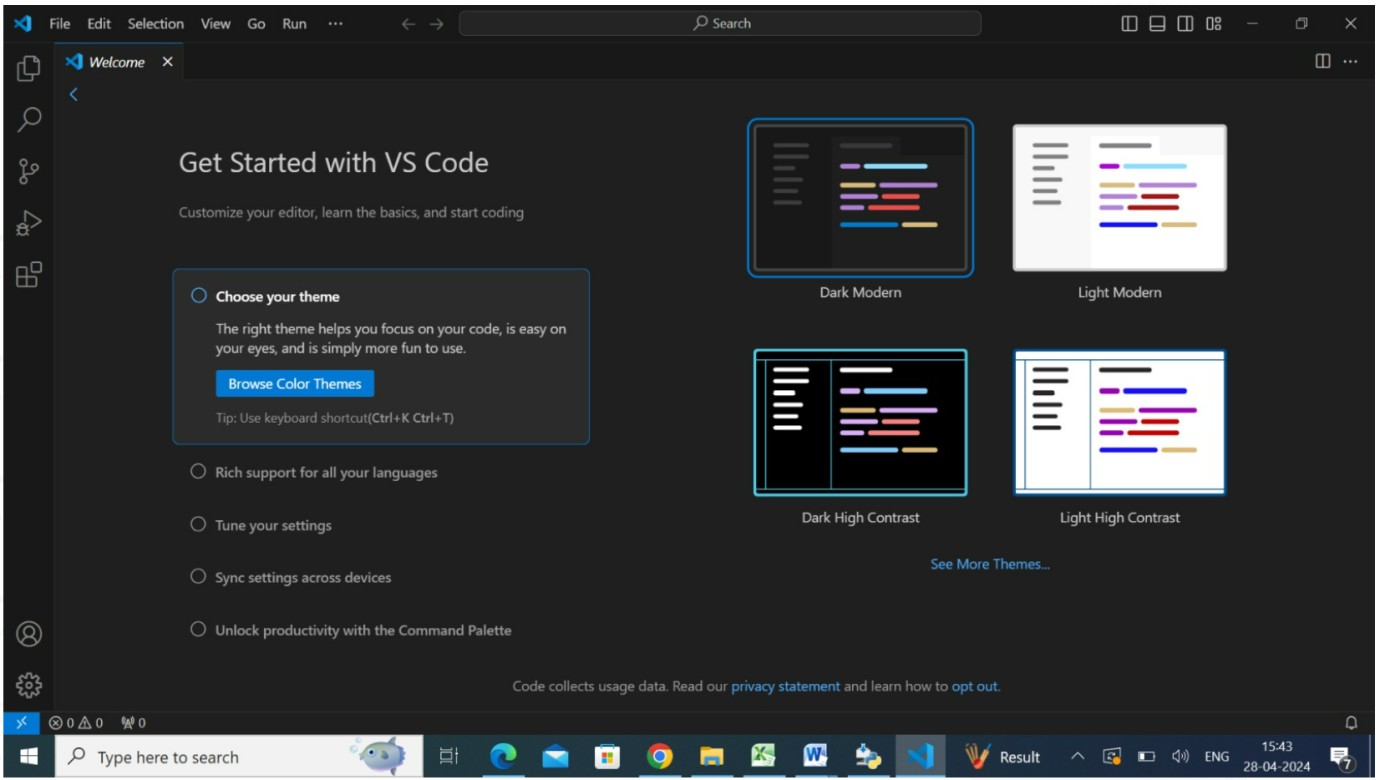
* Create the django app with the name “hello\_world\_app” python manage.py startapp hello\_world\_app

**LAB PROGRAMS PROGRAM 1**

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







* 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



* 1. 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

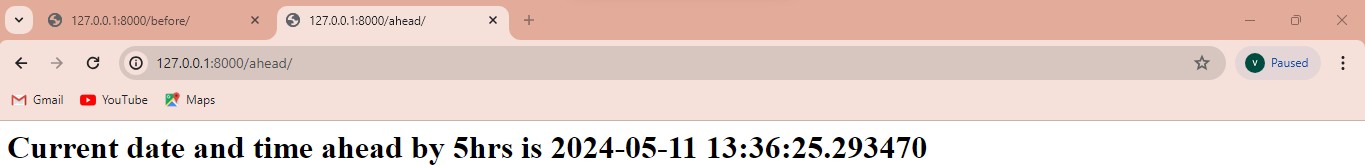
from scdt\_abApp.views import ahead, before

urlpatterns = [

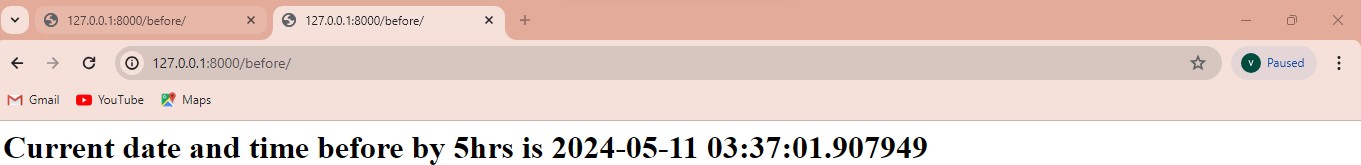
path('admin/', admin.site.urls), path('ahead/',ahead), path('before/',before),

]

# 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

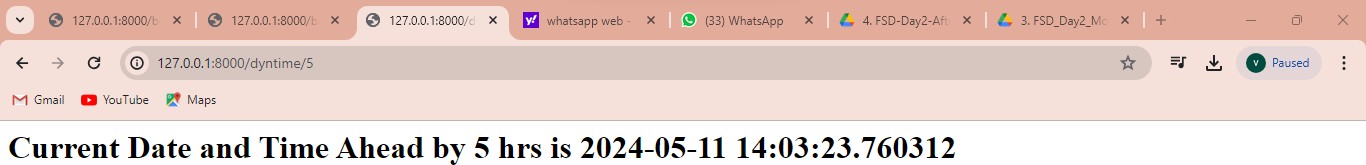
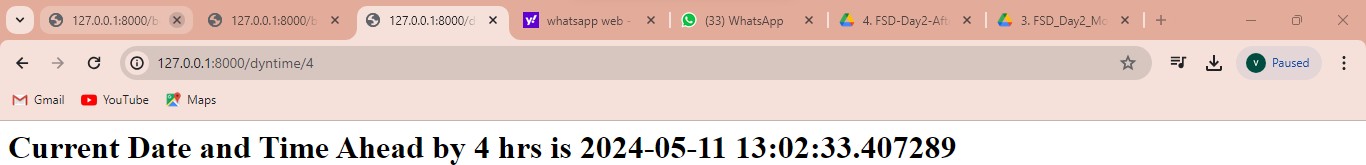
from django.contrib import admin from django.urls import path

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

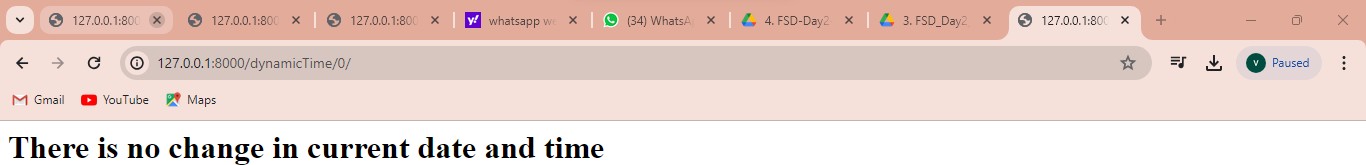
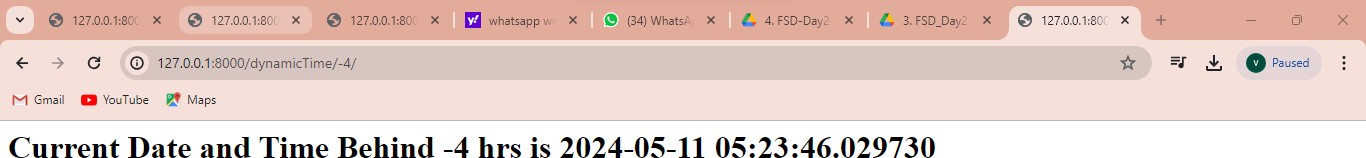
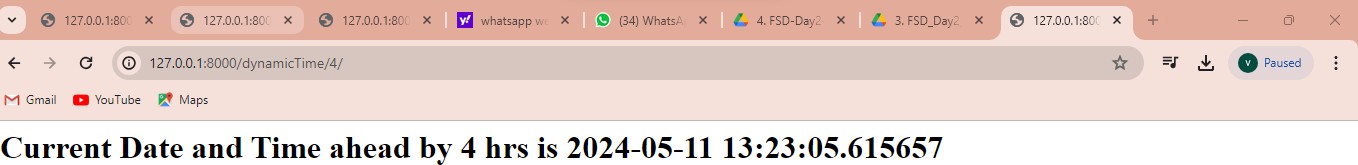
from scdt\_dynamicApp.views import scdt

urlpatterns = [

path('admin/', admin.site.urls), path('dynamicTime/<str:s>/',scdt),

]

# OUTPUT



**PROGRAM 2**

* 1. 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),

]

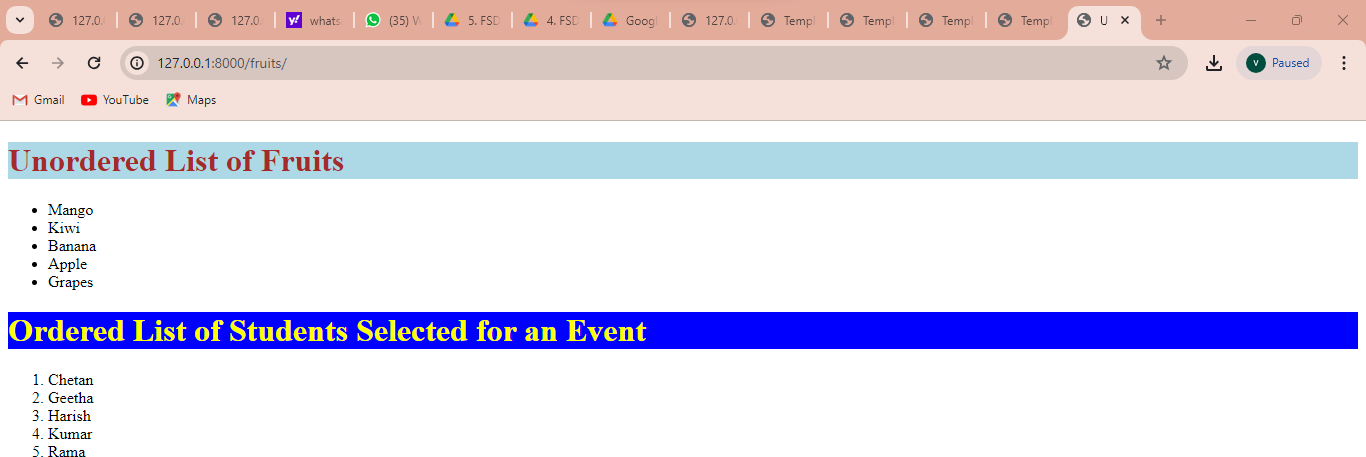
# settings.py (only one change inside installed apps add fruitsapp)

INSTALLED\_APPS = [

'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', 'fruitsapp',

]

# OUTPUT



* 1. **.** 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 Harish , CSE, RNSIT, Bangalore-04

</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>XXX, Asso. Prof, Dept of CSE, RNSIT</p>

{% endblock %}

# contactus.html

{% extends 'layout.html' %}

{% block title %} Contact us {% endblock %}

{% block content %}

<h1>Contact us</h1>

<p>Name: Harish</p>

<p>Designation:Asso. Prof </p>

<p>Mobile: 9988119894</p>

<p>Email: [harish.cse90@gmail.com<](mailto:harish.cse90@gmail.com%3c)/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)

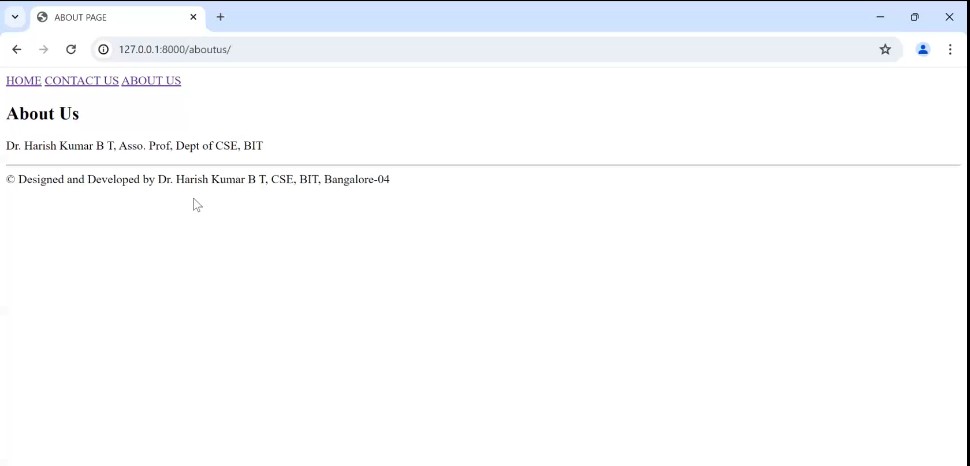
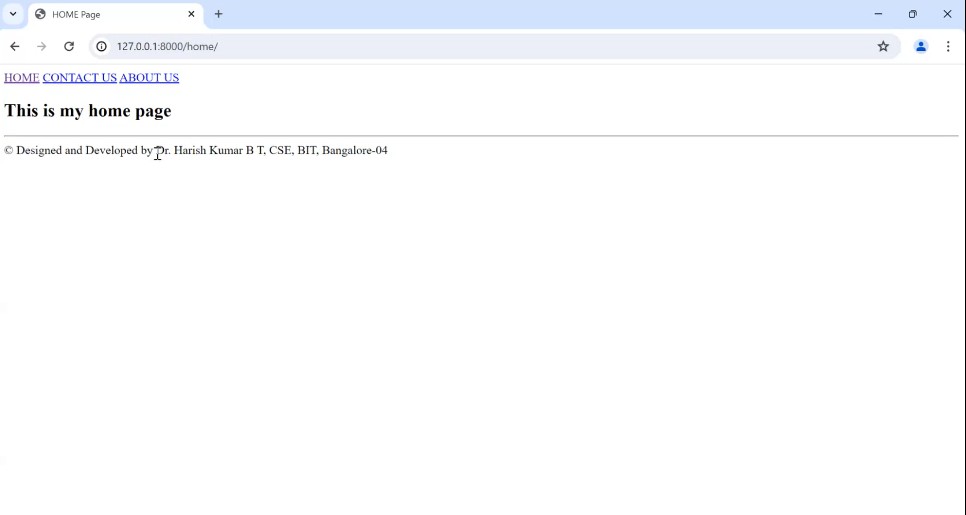
# Application definition

INSTALLED\_APPS = [

'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', 'layoutApp',

]

# OUTPUT



**2.3.** 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 Developeb by RNSIT. of CSE, , Bangalore-04

</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)

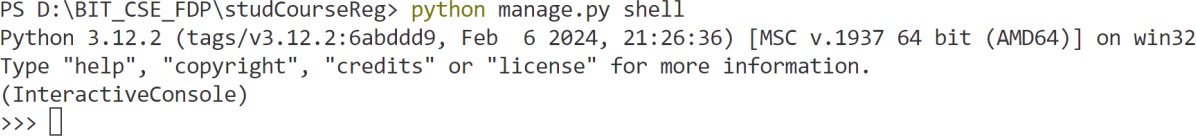
sem=models.IntegerField() courses=models.ManyToManyField(course,related\_name='student\_set') def str (self):

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

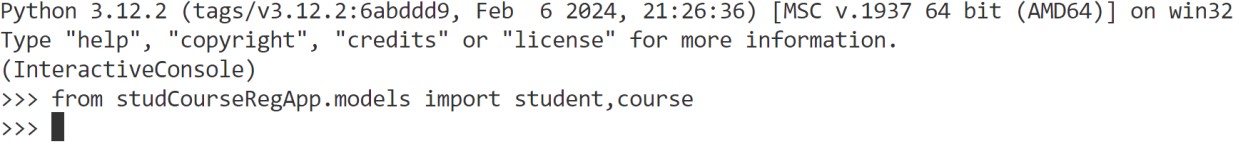
# After writing models.py run the below commands in VS code terminal. python manage.py makemigrations

**python manage.py migrate**

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



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



Create student objects as given below in the interactive python console

>>>s1=student(usn= ‘1RN21CS001’,name= ‘Harish’, sem=6)

>>>s2=student(usn= ‘1RN 21CS002’,name= ‘Kumar’, sem=6)

>>>s3=student(usn= ‘1RN 21CS003’,name= ‘Chetan, sem=6)

>>>s4=student(usn= ‘1RN 21CS004’,name= ‘Rama’, sem=6)

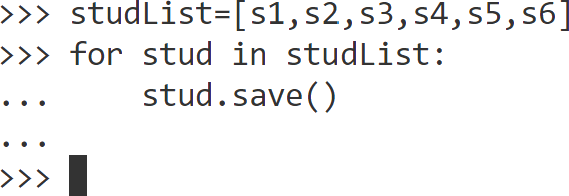
>>>s5=student(usn= ‘1RN 21CS005’,name= ‘Krishna, sem=6)

>>>s6=student(usn= ‘1RN 21CS007’,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()



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)

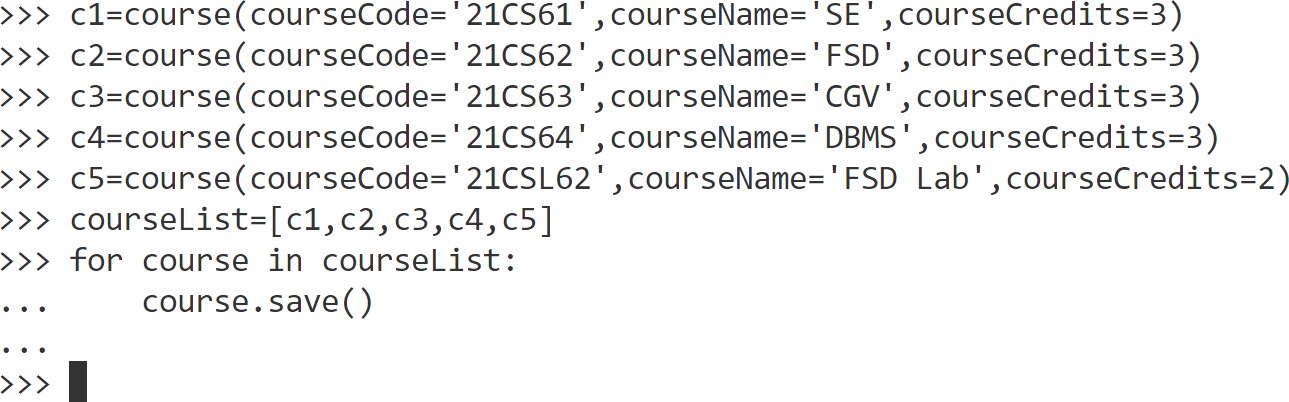
>>>c4=course(courseCode='21CS64',courseName='DBMS',courseCredits=3)

>>>c5=course(courseCode='21CSL62',courseName='FSD Lab',courseCredits=2)

>>> courseList=[c1,c2,c3,c4,c5]

>>> for course in courseList:

... course.save()



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 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})

# 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

