



SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY

*(Affiliated to VTU, Recognized by AICTE and Accredited by NBA, NAAC
and an ISO 9001-2008 Certified Institution)*

Bengaluru – 562157



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FULLSTACK DEVELOPMENT

21CS62 - VI Semester B.E

(Academic Year 2023-2024)

Compiled and Prepared by:

Mrs. Savita B & Suraj Kumar B P
Assistant Professor
Dept. of CSE

Under the Guidance of:

Dr. Anitha T N
Professor & Head
Dept. of CSE

Department Vision and Mission

VISION

To build a center for imparting quality technical education and carrying out research activity to meet the current and future challenges in the domain of Computer Science and Engineering.

MISSION

- The Computer Science and Engineering department strives for excellence in teaching, applying, promoting and imparting knowledge through comprehensive academic curricula.
- Train students to effectively apply the knowledge to solve real-world problems, thus enhance their potential for life-long high-quality career and give them a competitive advantage in the ever-changing and fast paced computing.
- Prepare students to demonstrate a sense of societal and ethical responsibilities in their professional endeavors.
- Creating amongst students and faculty a collaborative environment open to the free exchange of ideas, which leads to research activity and fuels innovation thinking.

PROGRAM OUTCOMES

PO's	PO Description
P01	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
P02	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.
P03	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.
P04	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.
P05	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.
P06	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.
P07	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.
P08	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
P09	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
P010	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.
P011	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.
P012	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.

PROGRAM SPECIFIC OUTCOMES

PSO's	PSO Description
PSO1	An ability to design and analyze algorithms by applying theoretical concepts to build complex and computer- based systems in the domain of System Software, Computer Networks & Security, Web technologies, Data Science and Analytics.
PSO2	Be able to develop various software solutions by applying the techniques of Data Base Management, Complex Mathematical Models, Software Engineering practices and Machine Learning with Artificial Intelligence.

FULLSTACK DEVELOPMENT			
Course Code	21CS62	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 T + 20 P	Total Marks	100
Credits	04	Exam Hours	03
Course Learning Objectives: CLO 1.Explain the use of learning full stack web development. CLO 2.Make use of rapid application development in the design of responsive web pages. CLO 3.Illustrate Models, Views and Templates with their connectivity in Django for full stack web development. CLO 4.Demonstrate the use of state management and admin interfaces automation in Django. CLO 5.Design and implement Django apps containing dynamic pages with SQL databases.			
Teaching-Learning Process (General Instructions) These are sample Strategies, which teachers can use to accelerate the attainment of the various course outcomes. <div><div>1.</div><div>Lecturer method (L) does not mean only traditional lecture method, but different type of teaching methods may be adopted to develop the outcomes.</div></div> <div><div>2.</div><div>Show Video/animation films to explain functioning of various concepts.</div></div> <div><div>3.</div><div>Encourage collaborative (Group Learning) Learning in the class.</div></div> <div><div>4.</div><div>Ask at least three HOT (Higher order Thinking) questions in the class, which promotes critical thinking.</div></div> <div><div>5.</div><div>Adopt Problem Based Learning (PBL), which fosters students' Analytical skills, develop thinking skills such as the ability to evaluate, generalize, and analyze information rather than simply recall it.</div></div> <div><div>6.</div><div>Topics will be introduced in a multiple representation.</div></div> <div><div>7.</div><div>Show the different ways to solve the same problem and encourage the students to come up with their own creative ways to solve them.</div></div> <div><div>8.</div><div>Discuss how every concept can be applied to the real world - and when that's possible, it helps improve the students' understanding.</div></div>			
Module-1: MVC based Web Designing			
Web framework, MVC Design Pattern, Django Evolution, Views, Mapping URL to Views, Working of Django URL Confs and Loose Coupling, Errors in Django, Wild Card patterns in URLs.			
Textbook 1: Chapter 1 and Chapter 3			
Laboratory Component: <div><div>1.</div><div>Installation of Python, Django and Visual Studio code editors can be demonstrated.</div></div> <div><div>2.</div><div>Creation of virtual environment, Django project and App should be demonstrated</div></div> <div><div>3.</div><div>Develop a Django app that displays current date and time in server</div></div> <div><div>4.</div><div>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.</div></div>			
Teaching-Learning Process	<div><div>1.</div><div>Demonstration using Visual Studio Code</div></div> <div><div>2.</div><div>PPT/Prezi Presentation for Architecture and Design Patterns</div></div> <div><div>3.</div><div>Live coding of all concepts with simple examples</div></div>		
Module-2: Django Templates and Models			
Template System Basics, Using Django Template System, Basic Template Tags and Filters, MVT Development Pattern, Template Loading, Template Inheritance, MVT Development Pattern.			

Configuring Databases, Defining and Implementing Models, Basic Data Access, Adding Model String Representations, Inserting/Updating data, Selecting and deleting objects, Schema Evolution

Textbook 1: Chapter 4 and Chapter 5

Laboratory Component:

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.

Teaching-Learning Process

1. Demonstration using Visual Studio Code
2. PPT/Prezi Presentation for Architecture and Design Patterns
3. Live coding of all concepts with simple examples
4. Case Study: Apply concepts learnt for an Online Ticket Booking System

Module-3: Django Admin Interfaces and Model Forms

Activating Admin Interfaces, Using Admin Interfaces, Customizing Admin Interfaces, Reasons to use Admin Interfaces.

Form Processing, Creating Feedback forms, Form submissions, custom validation, creating Model Forms, URLConf Ticks, Including Other URLConfs.

Textbook 1: Chapters 6, 7 and 8

Laboratory Component:

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.

Teaching-Learning Process

1. Demonstration using Visual Studio Code
2. PPT/Prezi Presentation for Architecture and Design Patterns
3. Live coding of all concepts with simple examples

Module-4: Generic Views and Django State Persistence

Using Generic Views, Generic Views of Objects, Extending Generic Views of objects, Extending Generic Views.

MIME Types, Generating Non-HTML contents like CSV and PDF, Syndication Feed Framework, Sitemap framework, Cookies, Sessions, Users and Authentication.

Textbook 1: Chapters 9, 11 and 12

Laboratory Component:

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.

Teaching-Learning Process

1. Demonstration using Visual Studio Code
2. PPT/Prezi Presentation for Architecture and Design Patterns

	<ol style="list-style-type: none"> Live coding of all concepts with simple examples Project Work: Implement all concepts learnt for Student Admission Management.
Module-5: jQuery and AJAX Integration in Django	
Ajax Solution, Java Script, XMLHttpRequest and Response, HTML, CSS, JSON, iFrames, Settings of Java Script in Django, jQuery and Basic AJAX, jQuery AJAX Facilities, Using jQuery UI Autocomplete in Django	
Textbook 2: Chapters 1, 2 and 7.	
Laboratory Component: <ol style="list-style-type: none"> Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX. Develop a search application in Django using AJAX that displays courses enrolled by a student being searched. 	
Teaching-Learning Process	<ol style="list-style-type: none"> Demonstration using Visual Studio Code PPT/Prezi Presentation for Architecture and Design Patterns Live coding of all concepts with simple examples Case Study: Apply the use of AJAX and jQuery for development of EMI calculator.
Course outcome (Course Skill Set) At the end of the course the student will be able to: CO 1. Understand the working of MVT based full stack web development with Django. CO 2. Designing of Models and Forms for rapid development of web pages. CO 3. Analyze the role of Template Inheritance and Generic views for developing full stack web applications. CO 4. Apply the Django framework libraries to render nonHTML contents like CSV and PDF. CO 5. Perform jQuery based AJAX integration to Django Apps to build responsive full stack web applications,	
Assessment Details (both CIE and SEE) The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination (SEE), and a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together Continuous Internal Evaluation: Three Unit Tests each of 20 Marks (duration 01 hour) <ol style="list-style-type: none"> First test at the end of 5th week of the semester Second test at the end of the 10th week of the semester Third test at the end of the 15th week of the semester Two assignments each of 10 Marks <ol style="list-style-type: none"> First assignment at the end of 4th week of the semester Second assignment at the end of 9th week of the semester 	

Practical Sessions need to be assessed by appropriate rubrics and viva-voce method. This will contribute to **20 marks**.

- Rubrics for each Experiment taken average for all Lab components – 15 Marks.
- Viva-Voce– 5 Marks (more emphasized on demonstration topics)

The sum of three tests, two assignments, and practical sessions will be out of 100 marks and will be **scaled down to 50 marks**

(to have a less stressed CIE, the portion of the syllabus should not be common /repeated for any of the methods of the CIE. Each method of CIE should have a different syllabus portion of the course).

CIE methods /question paper has to be designed to attain the different levels of Bloom's taxonomy as per the outcome defined for the course.

Semester End Examination:

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the subject (**duration 03 hours**)

1. The question paper will have ten questions. Each question is set for 20 marks. Marks scored shall be proportionally reduced to 50 marks
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.

The students have to answer 5 full questions, selecting one full question from each module

Suggested Learning Resources:

Textbooks

1. Adrian Holovaty, Jacob Kaplan Moss, The Definitive Guide to Django: Web Development Done Right, Second Edition, Springer-Verlag Berlin and Heidelberg GmbH & Co. KG Publishers, 2009
2. Jonathan Hayward, Django Java Script Integration: AJAX and jQuery, First Edition, Pack Publishing, 2011

Reference Books

1. Aidas Bendroraitis, Jake Kronika, Django 3 Web Development Cookbook, Fourth Edition, Packt Publishing, 2020
2. William Vincent, Django for Beginners: Build websites with Python and Django, First Edition, Amazon Digital Services, 2018
3. Antonio Mele, Django3 by Example, 3rd Edition, Pack Publishers, 2020
4. Arun Ravindran, Django Design Patterns and Best Practices, 2nd Edition, Pack Publishers, 2020.
5. Julia Elman, Mark Lavin, Light weight Django, David A. Bell, 1st Edition, Oreily Publications, 2014

Weblinks and Video Lectures (e-Resources):

1. MVT architecture with Django: <https://freevideolectures.com/course/3700/django-tutorials>
2. Using Python in Django: <https://www.youtube.com/watch?v=2BqoLiMT3Ao>
3. Model Forms with Django: <https://www.youtube.com/watch?v=gMM1rtTwKxE>
4. Real time Interactions in Django: <https://www.youtube.com/watch?v=3gHmfoeZ45k>
5. AJAX with Django for beginners: <https://www.youtube.com/watch?v=3VaKNyjlxAU>

Activity Based Learning (Suggested Activities in Class)/ Practical Based learning

1. Real world problem solving - applying the Django framework concepts and its integration with AJAX to develop any shopping website with admin and user dashboards.

Short Preamble on Full Stack Web Development:

Website development is a way to make people aware of the services and/or products they are offering, understand why the products are relevant and even necessary for them to buy or use, and highlight the striking qualities that set it apart from competitors. Other than commercial reasons, a website is also needed for quick and dynamic information delivery for any domain. Development of a well-designed, informative, responsive and dynamic website is need of the hour from any computer science and related engineering graduates. Hence, they need to be augmented with skills to use technology and framework which can help them to develop elegant websites. Full Stack developers are in need by many companies, who knows and can develop all pieces of web application (Front End, Back End and business logic). MVT based development with Django is the cutting-edge framework for Full Stack Web Development. Python has become an easier language to use for many applications. Django based framework in Python helps a web developer to utilize framework and develop rapidly responsive and secure web applications.

SUBJECT: FULLSTACK DEVELOPMENT (21CS62)

LAB COMPONENT SOLUTIONS

Laboratory Component:

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

Python download and installation Link:

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

Visual Studio Code download and installation link:

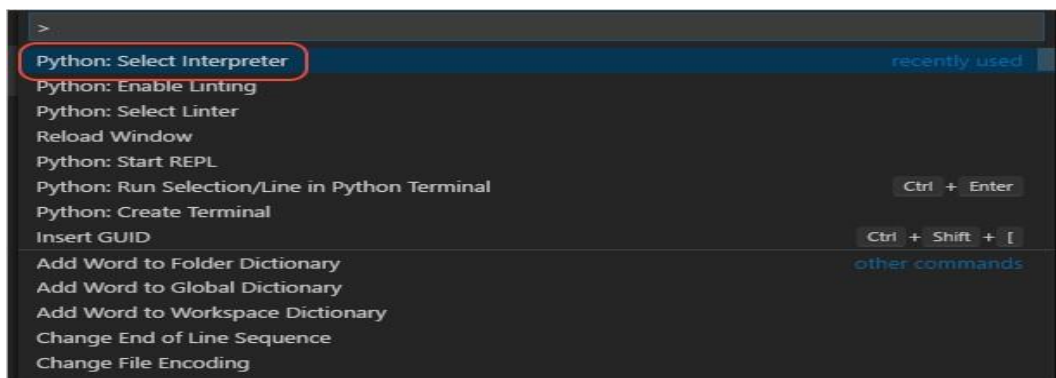
<https://code.visualstudio.com/>

2. Creation of virtual environment, Django project and App should be demonstrated Follow these steps

1. Install the [Python extension](#).- Open VS Code IDE and click extensions there automatically u will be shown Python extension (Make sure you are connected to Internet)
2. On your file system, create a project folder
3. In that folder, use the following command (as appropriate to your computer) to create a virtual environment named `env` based on your current interpreter:

```
# Windows  
python -m venv env
```

4. Open the project folder in VS Code by running `code .`, or by running VS Code and using the **File > Open Folder** command.
5. In VS Code, open the Command Palette (**View > Command Palette** or (**Ctrl+Shift+P**)). Then select the **Python: Select Interpreter** command:



6. The command presents a list of available interpreters that VS Code can locate automatically (your list will vary; if you don't see the desired interpreter, see [Configuring Python environments](#)). From the list, select the virtual environment in your project folder that starts with `./env` or `.\env`:
7. Create a New Terminal : In Menu Terminal -> New Terminal option

Creating project:

1. Django installation:
Open a command prompt and type following command:
`pip install django`
2. Create a django project -

Type following command in the terminal opened:

`django-admin startproject lab .`

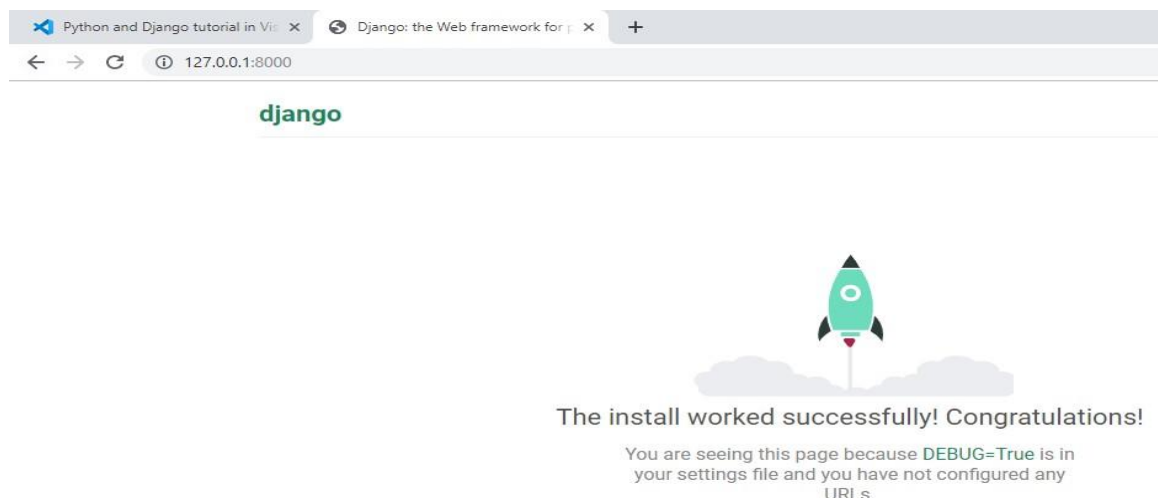
(dot following project name is important which refers to current directory)

This `startproject` command assumes (by use of `.` at the end) that the current folder is your project folder, and creates the following within it:

- `manage.py`: The Django command-line administrative utility for the project. You run administrative commands for the project using `python manage.py <command> [options]`.
- A subfolder named `lab` which contains the following files:
 - `__init__.py`: an empty file that tells Python that this folder is a Python package.
 - `wsgi.py`: an entry point for WSGI-compatible web servers to serve your project. You typically leave this file as-is as it provides the hooks for production web servers.
 - `settings.py`: contains settings for Django project, which you modify in the course of developing a web app.
 - `urls.py`: contains a table of contents for the Django project, which you also modify in the course of development.
- 3. To verify the Django project, make sure your virtual environment is activated, then start Django's development server using the command `python manage.py runserver`. The server runs on the default port 8000, and you see output like the following output in the terminal window:

Verify server by typing:

`python manage.py runserver`



When you run the server the first time, it creates a default SQLite database in the file `db.sqlite3`, which is intended for development purposes but can be used in production for low-volume web

apps. Also, Django's built-in web server is intended *only* for local development

purposes. When you deploy to a web host, however, Django uses the host's web server instead. The `wsgi.py` module in the Django project takes care of hooking into the production servers.

If you want to use a different port than the default 8000, specify the port number on the command line, such as `python manage.py runserver 5000`.

4. When you're done, close the browser window and stop the server in VS Code using `Ctrl+C` as indicated in the terminal output window.
5. In the VS Code Terminal with your virtual environment activated, run the administrative utility's `startapp` command in your project folder (where `manage.py` resides):

```
python manage.py startapp ap1
```

6. The command creates a folder called `ap1` that contains a number of code files and one subfolder. Of these, you frequently work with `views.py` (that contains the functions that define pages in your web app) and `models.py` (that contains classes defining your data objects). The `migrations` folder is used by Django's administrative utility to manage database versions. There are also the files `apps.py` (app configuration), `admin.py` (for creating an administrative interface), and `tests.py` (for unit tests).

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

In lab1 subfolder, make following changes to views.py:

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

# Create your views here.import
datetime
def current_datetime(request):
    now = datetime.datetime.now()
    html = "<html><body><h1>It is now %s.</h1></body></html>" % nowreturn
    HttpResponseRedirect(html)
```

In project named first, make following changes to urls.py

```
from django.contrib import admin
from django.urls import path
from lab11.views import current_datetime
urlpatterns = [

    path('cdt/', current_datetime),
]
```

Output:



It is now 2024-02-03 18:42:01.631243.

. 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.

In lab11 subfolder, make following changes to views.py:

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

# Create your views here.import
datetime
def current_datetime(request): now =
    datetime.datetime.now()

    html = "<html><body><h1>It is now %s.</h1></body></html>" % nowreturn
    HttpResponse(html)
def four_hours_ahead(request):

    dt = datetime.datetime.now() + datetime.timedelta(hours=4)
    html = "<html><body><h1>After 4hour(s), it will be %s.</h1>"% (dt,)return
    HttpResponse(html)

def four_hours_before(request):

    dt = datetime.datetime.now() + datetime.timedelta(hours=-4)
    html = "<html><body><h1>Before 4 hour(s), it was %s.</h1>"% (dt,)return
    HttpResponse(html)
```

In project named first, make following changes to urls.py

```
from django.contrib import admin
from django.urls import path
from lab11.views import current_datetime,four_hours_ahead,four_hours_beforeurlpatterns = [

    path('cdt/', current_datetime),
    path('fhresa/',four_hours_ahead),
    path('fhrsb/',four_hours_before),
```

```
]
```

Output:

← → ↻ ⓘ 127.0.0.1:8000/fhrs/

After 4hour(s), it will be 2024-02-03 22:43:50.544397.

← → ↻ ⓘ 127.0.0.1:8000/fhrs/

Before 4 hour(s), it was 2024-02-03 14:44:10.994024.

Module-2: Django Templates and Models

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

Views.py

```
from datetime import date
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.template import Context, Template
```

Create your views here.

```
def showlist(request):
    fruits=["Mango","Apple","Bananan","Jackfruits"]
    student_names=["Tony","Mony","Sony","Bob"]
    return render(request,'showlist.html',{'fruits':fruits,"student_names":student_names})
```

URLS.py

```
from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time
from ap1.views import four_hours_after, four_hours_before
from ap1.views import n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode
from ap2.views import template_test, showlist
urlpatterns = [
    path('admin/', admin.site.urls),
    path('cdt/', current_date_time),
    path('fha/', four_hours_after),
```

Views.py

```
from datetime import date
```

```
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.template import Context, Template
```

Create your views here.

```
def showlist(request):
    fruits=["Mango","Apple","Bananan","Jackfruits"]
    student_names=["Tony","Mony","Sony","Bob"]
    return render(request,'showlist.html',{ "fruits":fruits,"student_names":student_names })
```

URLS.py

```
from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time
from ap1.views import four_hours_after, four_hours_before
from ap1.views import n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode
from ap2.views import template_test, showlist

urlpatterns = [
    path('admin/', admin.site.urls),
    path('cdt/', current_date_time),
    path('fha/', four_hours_after),
    path('fhb/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/', check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist),
```

Template HTML file (inside ap2/templates subfolder)

```
showlist.html
<html>
    <style type="text/css">
        #i1 { background-color: lightgreen;color:brown;display:table}#i2
        { background-color: black;color:yellow}
    </style>
    <body>
        <h1 id="i1">Unordered list of fruits</h1>
```

```
<ul>
  {% for fruit in fruits %}
  <li>{{ fruit }}</li>
  {% endfor %}

</ul>
<h1 id="i2">Ordered list of Students</h1>
<ol>
  {% for student in student_names %}
  <li>{{ student }}</li>
  {% endfor %}

</ol>
</body>
</html>
```

Output:

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

Views.py

```
from datetime import date
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.template import Context, Template
def home(request):
    return render(request, 'home.html')

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

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

URLS.py

```
from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time
from ap1.views import four_hours_after, four_hours_before
from ap1.views import n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode
from ap2.views import template_test, showlist, list_of_subjects
from ap2.views import aboutus, home, contactus

urlpatterns = [
    path('admin/', admin.site.urls), path('cdt/',
    current_date_time), path('fha/',
    four_hours_after), path('fhb/',
```



```

four_hours_before), path('nha/<int:num>',
n_hours_after),
path('display_string/<slug:sentence>', display_string),
re_path('check_number/(\d){1,2}/', check_number),

path('cts/<int:s>/<int:n>', create_table_of_squares),
path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
find_mode), path('template_test/', template_test), path('showlist/',
showlist), path('list_of_subjects/', list_of_subjects), path('aboutus/',
aboutus),
path('home/', home), path('contactus/',
contactus),

```

Template files:

layout.html

```

<html>
  <title>{% block title %} {% endblock %} </title>
  <style type="text/css">
    nav { background-color: lightblue;padding:10px }
  </style>
  <body>
    <nav>
      <a href="/home/">Home</a>|
      <a href="/aboutus/">About Us</a>|
      <a href="/contactus/">Contact Us</a>|
    </nav>
    <section>
      {% block content %} {% endblock %}
    </section>
    <footer>
      <hr>
      &copy; AIML, Developed by ABC, Inc.
    </footer>
  </body>
</html>

```

home.html

```

{% extends 'layout.html' %}
{% block title %}
Home
{% endblock %}
{% block content %}
<h2>This is the home page</h2>
{% endblock %}

```

aboutus.html

```
{% extends 'layout.html' %}
{% block title %}
About Us
{% endblock %}
{% block content %}
<h2>We are Django developers</h2>
{% endblock %}
```

contactus.html

```
{% extends 'layout.html' %}
{% block title %}
Contact us
{% endblock %}
{% block content %}
<h2>Out phone: 9900923050 <br>
Address: Navule JNNCE</h2>
{% endblock %}
```

Output:

← → ↻ ⓘ 127.0.0.1:8000/home/

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

This is the home page

© AIML, Developed by ABC, Inc.

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**

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

# Create your models here. class
Course(models.Model):
    course_code=models.CharField(max_length=40)
    course_name=models.CharField(max_length=100)course_credits=models.IntegerField()

class Student(models.Model):
    student_usn=models.CharField(max_length=20)
    student_name=models.CharField(max_length=100)
    student_sem=models.IntegerField()
    enrolment=models.ManyToManyField(Course)
```

eg.html inside templates folder

```
<html>
<body>
<form method="post" action="">
    {% csrf_token %}
    Student Name
    <select name="sname">
        {% for student in students %}
        <option value="{{ student.id }}">{{ student.student_name }}</option>
        {% endfor %}
    </select><br>
    Course Name
    <select name="cname">
        {% for course in courses %}
```

```

        <option value="{{ course.id }}">{{ course.course_name }}</option>
    {% endfor %}

</select><br>
    <input type="submit" value="Enroll">
</form>
</body>
</html>

```

views.py

```

from django.http import HttpResponseRedirect
from django.shortcuts import render

from ap3.models import Course, Meeting, Student

def reg(request):
    if request.method == "POST":
        sid=request.POST.get("sname")
        cid=request.POST.get("cname")
        student=Student.objects.get(id=sid)
        course=Course.objects.get(id=cid)
        res=student.enrolment.filter(id=cid)
        if res:
            return HttpResponseRedirect("<h1>Student already enrolled</h1>")
        student.enrolment.add(course)
        return HttpResponseRedirect("<h1>Student enrolled successfully</h1>")
    else:
        students=Student.objects.all()
        courses=Course.objects.all()
        return render(request,"reg.html",{ "students":students,
        "courses":courses})

```

urls.py

```

from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time
from ap1.views import four_hours_after, four_hours_before
from ap1.views import n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode
from ap2.views import template_test, showlist, list_of_subjects
from ap2.views import aboutus, home, contactus, getpos, stable
from ap3.views import insert_demo, update_demo, delete_demo, retrieve_demo
from ap3.views import reg

urlpatterns = [
    path('admin/', admin.site.urls), path('cdt/',
    current_date_time), path('fha/',
    four_hours_after), path('fha/',
    four_hours_before), path('nha/<int:num>',
    n_hours_after),
    path('display_string/<slug:sentence>', display_string),

```

```

re_path('check_number/(\d){1,2}/', check_number),
path('cts/<int:s>/<int:n>', create_table_of_squares),
path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
find_mode), path('template_test/', template_test), path('showlist/',
showlist),
path('list_of_subjects/', list_of_subjects), path('aboutus/',
aboutus),
path('home/', home), path('contactus/', contactus),
path('getpos/', getpos), path('stable/', stable),
path('insert_demo/', insert_demo),
path('update_demo/', update_demo),
path('delete_demo/', delete_demo),
path('retreive_demo/', retreive_demo), path('reg/',
reg),

```

]

Database input:

Insert student and courses record in phpMyAdmin

The screenshot shows the phpMyAdmin interface for the 'ap3_student' table. The top navigation bar includes 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', and a 'Refresh' icon. A green status bar indicates 'Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)'. The SQL query editor shows 'SELECT * FROM `ap3_student`'. Below the query editor, there are options for 'Show all', 'Number of rows' (set to 25), and a 'Filter rows' search box. An 'Extra options' button is also present. The table data is displayed in a grid with columns: 'id', 'student_usn', 'student_name', and 'student_sem'. Each row has checkboxes for 'Edit', 'Copy', and 'Delete' actions.

	id	student_usn	student_name	student_sem
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	4JN22AI001	Sham	4
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	4JN22AI002	Manish	4
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	4JN22AI003	Suma	4

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#)

✓ Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)

```
SELECT * FROM `ap3_course`
```

☐ Profiling [[Edit inline](#)]

☐ Show all | Number of rows: 25 | Filter rows:

Extra options

				id	course_code	course_name	course_credits
<input type="checkbox"/>				1	21AI42	DAA	4
<input type="checkbox"/>				2	21AI43	MPC	3
<input type="checkbox"/>				3	21AI44	UHV	1

Output:

[←](#) [→](#) [↻](#) [127.0.0.1:8000/reg/](#)

Student Name

Course Name

[←](#) [→](#) [↻](#) [127.0.0.1:8000/reg/](#)

Student enrolled successfully

BackEnd

Browse
 Structure
 SQL
 Search
 Insert

Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)

```
SELECT * FROM `ap3_student_enrolment`
```

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

				id	student_id	course_id
<input type="checkbox"/>	Edit	Copy	Delete	1	1	1
<input type="checkbox"/>	Edit	Copy	Delete	2	1	3
<input type="checkbox"/>	Edit	Copy	Delete	3	2	2
<input type="checkbox"/>	Edit	Copy	Delete	4	2	1
<input type="checkbox"/>	Edit	Copy	Delete	5	1	2

If you try again, you will get

127.0.0.1:8000/reg/

Student already enrolled

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

python manage.py createsuperuser

make following changes to admin.py

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

```
from ap3.models import Course, Student
```

```
# Register your models here.
```

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

```
@admin.register(Student)
class StudentAdmin(admin.ModelAdmin):
    list_display = ('student_name','student_usn','student_sem')
    ordering=('student_name',)
    search_fields = ('student_name',)
admin.site.register(Course)
```

urls.py

```
from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time from ap1.views
import four_hours_after, four_hours_before from ap1.views import
n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode from ap2.views
import template_test, showlist, list_of_subjects from ap2.views import
aboutus, home, contactus, getpos, stable
from ap3.views import insert_demo, update_demo, delete_demo, retrieve_demo from
ap3.views import reg, course_search
```

```
admin.site.site_header="My Site Header" admin.site.site_title="My Site Title"
admin.site.index_title="My Site Index"
```

```
urlpatterns = [
    path('secretadmin/', admin.site.urls), path('cdt/',
    current_date_time), path('fha/', four_hours_after),
    path('fhb/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/', check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist),
    path('list_of_subjects/', list_of_subjects), path('aboutus/',
    aboutus),
    path('home/', home), path('contactus/',
    contactus), path('getpos/', getpos),
    path('stable/', stable), path('insert_demo/',
    insert_demo), path('update_demo/',
    update_demo), path('delete_demo/',
    delete_demo), path('retrieve_demo/',
    retrieve_demo), path('reg/', reg),
    path('course_search/', course_search),
```

```
]
```


Changes to models.py

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

# Create your models here. class
Meeting(models.Model):
    meeting_code=models.CharField(max_length=100)
    meeting_dt=models.DateField(auto_now_add=True)
    meeting_subject=models.CharField(max_length=100)meeting_np=models.IntegerField()

class Course(models.Model): course_code=models.CharField(max_length=40)
    course_name=models.CharField(max_length=100)
    course_credits=models.IntegerField(blank=True, null=True)def __str
(self):
    return self.course_name

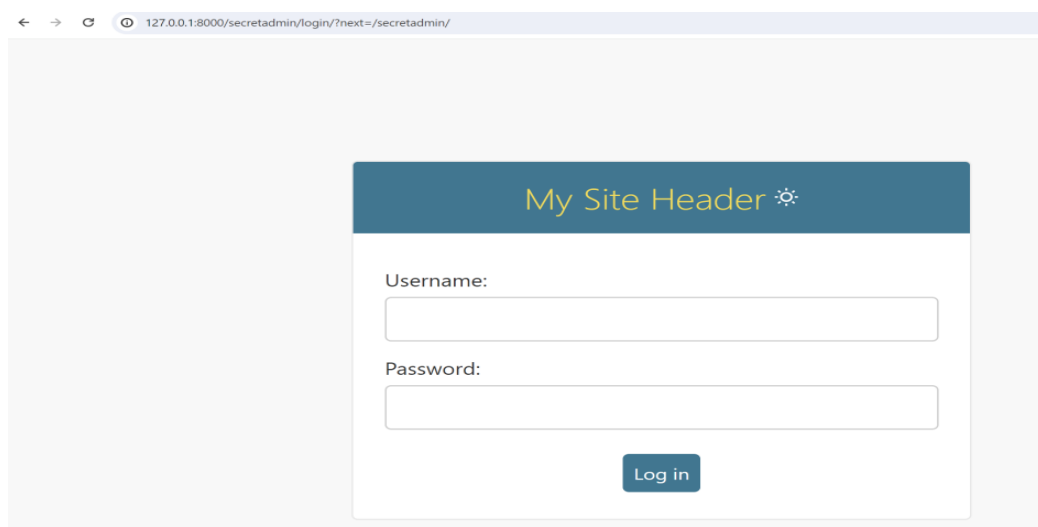
class Student(models.Model):
    student_usn=models.CharField(max_length=20)
    student_name=models.CharField(max_length=100)
    student_sem=models.IntegerField()
    enrolment=models.ManyToManyField(Course)
    def __str__(self):
        return self.student_name+"("+self.student_usn+")"
```

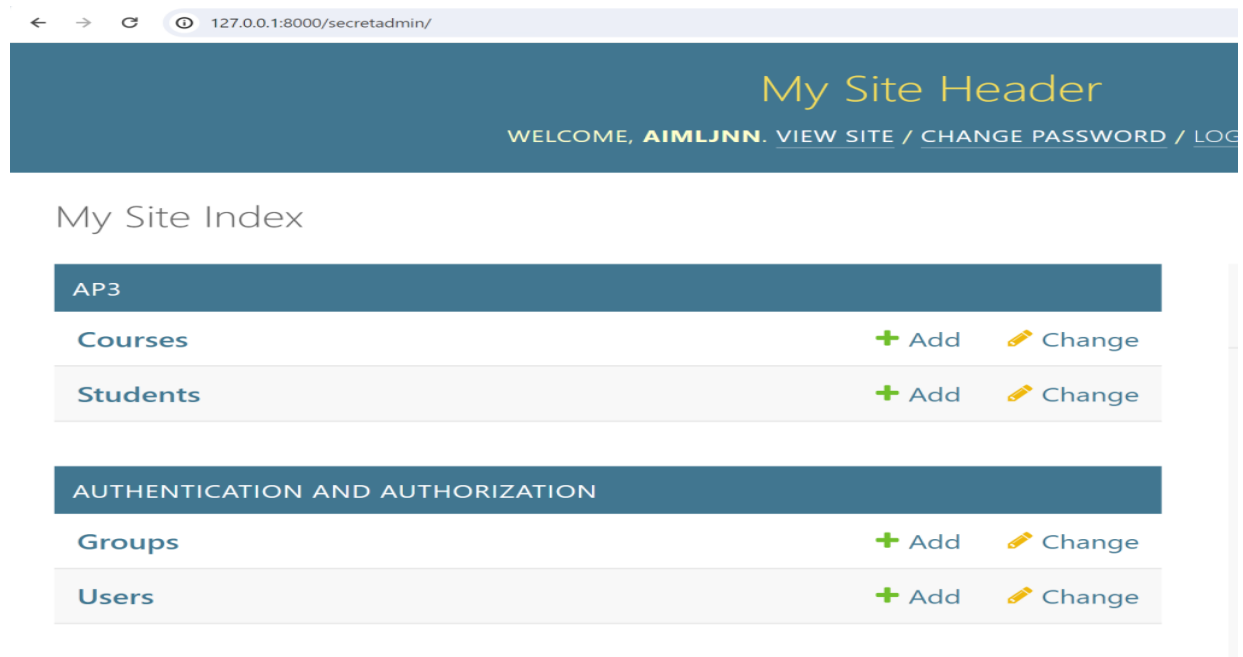
Perform remigrations before running:python

manage.py makemigrations ap3 python

manage.py migrate

Output:





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

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

# Create your models here. class
Meeting(models.Model):
    meeting_code=models.CharField(max_length=100)
    meeting_dt=models.DateField(auto_now_add=True)
    meeting_subject=models.CharField(max_length=100)meeting_np=models.IntegerField()

class Course(models.Model): course_code=models.CharField(max_length=40)
    course_name=models.CharField(max_length=100)
    course_credits=models.IntegerField(blank=True, null=True)def __str
    (self):
        return self.course_name

class Student(models.Model):
    student_usn=models.CharField(max_length=20)
    student_name=models.CharField(max_length=100)
    student_sem=models.IntegerField()
    enrolment=models.ManyToManyField(Course)
    def __str__(self):
        return self.student_name+"("+self.student_usn+")"

class Project(models.Model): student=models.ForeignKey(Student,on_delete=models.CASCADE)
    ptopic=models.CharField(max_length=200) planguages=models.CharField(max_length=200)
```

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

```
class ProjectReg(ModelForm):  
    required_css_class="required" class  
    Meta:  
        model=Project fields=['student','ptopic','plangauges','pduration']
```

models.py

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

```
from ap3.models import Course, Meeting, ProjectReg, Student
```

```
def add_project(request):  
    if request.method=="POST":  
        form=ProjectReg(request.POST)if  
        form.is_valid():  
            form.save()  
            return HttpResponseRedirect("<h1>Record inserted successfully</h1>")else:  
            return HttpResponseRedirect("<h1>Record not inserted</h1>")  
    else:  
        form=ProjectReg()  
        return render(request,"add_project.html",{ "form":form})
```

add_project.html inside templates folder

```
<html>  
    <form method="post" action="">  
        {% csrf_token %}  
        <table>  
            {{ form.as_table }}  
            <tr>  
                <td>  
                    <input type="submit" value="Submit">  
                </td>  
            </tr>  
        </table>  
    </form>  
</html>
```

urls.py

```
from django.contrib import admin  
from django.urls import path, re_path  
from ap1.views import check_number, current_date_time from ap1.views
```

```
import four_hours_after, four_hours_before from ap1.views import
n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode from ap2.views
import template_test, showlist, list_of_subjects from ap2.views import
aboutus, home, contactus, getpos, stable
from ap3.views import insert_demo, update_demo, delete_demo, retrieve_demo from
ap3.views import reg, course_search, add_project
```

```
admin.site.site_header="My Site Header"
admin.site.site_title="My Site Title"
admin.site.index_title="My Site Index" urlpatterns
= [
    path('secretadmin/', admin.site.urls), path('cdt/',
    current_date_time), path('fha/', four_hours_after),
    path('fha/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/', check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist), path('list_of_subjects/', list_of_subjects), path('aboutus/',
    aboutus),
    path('home/', home), path('contactus/', contactus),
    path('getpos/', getpos), path('stable/', stable),
    path('insert_demo/', insert_demo),
    path('update_demo/', update_demo),
    path('delete_demo/', delete_demo),
    path('retrieve_demo/', retrieve_demo), path('reg/',
    reg), path('course_search/', course_search),
    path('add_project/', add_project),
]
```

Perform remigrations before running: python

manage.py makemigrations ap3 python

manage.py migrate

Output:

← → ↻ 127.0.0.1:8000/add_project/

Student: Sham(4JN22AI001) ▾

Ptopic: AI

Plangauges: Python,HTML

Pduration: 22

← → ↻ 127.0.0.1:8000/add_project/

Record inserted successfully

BackEnd

Browse
 Structure
 SQL
 Search
 Insert
 Export
 Refresh

✓ Showing rows 0 - 1 (2 total, Query took 0.0010 seconds.)

`SELECT * FROM `ap3_project``

☐ Profiling [[Edit inline](#)]

☐ Show all |
 Number of rows: 25 ▾ |
 Filter rows:

Extra options

				id	ptopic	plangauges	pduration	student_id
<input type="checkbox"/>				1	IoT	Python,Arduino	20	1
<input type="checkbox"/>				2	AI	Python,HTML	22	1

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

views.py

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

```
class StudentDetailView(generic.DetailView):
    model=Student
    template_name="student_detail.html"
```

student_list.html inside templates folder

```
<html>
  <body>
    {% if student_list %}
      <table border>
        <tr>
          <th>USN</th>
          <th>Courses Enrolled</th>
        </tr>
        {% for student in student_list %}
          <tr>
            <td><a href="/student_detail/{{ student.pk }}">{{
student.student_usn }}</a></td>
              <td>{% for course in student.enrolment.all %}
                <span>{{ course.course_name }}</span>
              {% endfor %}
            </td>
          </tr>
        {% endfor %}
      </table>
    {% else %}
      <h1>No Students Enrolled</h1>
    {% endif %}
  </body>
</html>
```

student_detail.html inside templates folder

```
<h1>Student Name: {{ student.student_name }}</h1>
<h1>Student USN: {{ student.student_usn }}</h1>
<h1>Student Sem: {{ student.student_sem }}</h1>
```

urls.py

```
from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time from ap1.views
import four_hours_after, four_hours_before from ap1.views import
n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode from ap2.views
```

```

import template_test,showlist,list_of_subjects from ap2.views import
aboutus,home,contactus,getpos,stable
from ap3.views import insert_demo,update_demo,delete_demo,retreive_demo from
ap3.views import reg,course_search,add_project
from ap3.views import StudentListView,StudentDetailView

admin.site.site_header="My Site Header"
admin.site.site_title="My Site Title"
admin.site.index_title="My Site Index" urlpatterns
= [
    path('secretadmin/', admin.site.urls),path('cdt/',
    current_date_time), path('fha/', four_hours_after),
    path('fhb/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/',check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist),
    path('list_of_subjects/', list_of_subjects),path('aboutus/',
    aboutus),

    path('home/', home), path('contactus/', contactus),
    path('getpos/', getpos), path('stable/', stable),
    path('insert_demo/', insert_demo),
    path('update_demo/', update_demo),
    path('delete_demo/', delete_demo),
    path('retreive_demo/', retreive_demo),path('reg/',
    reg), path('course_search/', course_search),
    path('add_project/', add_project),
    path('student_list/', StudentListView.as_view()), path('student_detail/<int:pk>',
    StudentDetailView.as_view()),


]

```

Output:

← → ↻ ⓘ 127.0.0.1:8000/student_list/

USN	Courses Enrolled
4JN22AI001	DAA MPC UHV
4JN22AI002	DAA MPC
4JN22AI003	



Student Name: Sham

Student USN: 4JN22AI001

Student Sem: 4

11.Program to illustrate returning a constructed csv file to browser

views.py

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

```
from ap3.models import Course, Meeting, ProjectReg, Student
```

```
import csv
```

```
def construct_csv(request):
```



```

districts=["Shimoga","Bhadravathi","Bangalore","Dharwad","Raichur"]
temperatures=[38,36,34,35,40] response=HttpResponse(content_type="text/csv")
response['Content-Disposition'] = 'attachment;
filename="district_temperature.csv"
writer=csv.writer(response)
writer.writerow(["Districts","Temperatures"])for d,t in
zip(districts,temperatures):
    writer.writerow([d,t])return
response

```

urls.py

```

from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time from ap1.views
import four_hours_after, four_hours_beforefrom ap1.views import
n_hours_after,display_string
from ap2.views import create_table_of_squares,vc,find_mode from ap2.views
import template_test,showlist,list_of_subjectsfrom ap2.views import
aboutus,home,contactus,getpos,stable
from ap3.views import insert_demo,update_demo,delete_demo,retreive_demofrom
ap3.views import reg,course_search,add_project
from ap3.views import StudentListView,StudentDetailView,construct_csv

```

```

admin.site.site_header="My Site Header"
admin.site.site_title="My Site Title"
admin.site.index_title="My Site Index" urlpatterns
= [
    path('secretadmin/', admin.site.urls),path('cdt/',
    current_date_time), path('fha/', four_hours_after),
    path('fhb/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/',check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist),
    path('list_of_subjects/', list_of_subjects),path('aboutus/',
    aboutus),
    path('home/', home), path('contactus/',
    contactus),path('getpos/', getpos),
    path('stable/', stable), path('inser'retreive_demo/',
    retreive_demo),path('reg/', reg),
    path('course_search/', course_search),
    path('add_project/', add_project),
    path('student_list/', StudentListView.as_view()), path('student_detail/<int:pk>',
    StudentDetailView.as_view()),path('construct_csv/', construct_csv),

```

]

Output:

CSV file is generated and downloaded

Develop example Django app that performs CSV

generation for any models created t_demo/',

insert_demo), path('update_demo/',

update_demo), path('delete_demo/',

delete_demo), path(

in previous laboratory component

views.py

from django.http import HttpResponseRedirect

from django.shortcuts import render

from ap3.models import Course, Meeting, ProjectReg, Student import csv

def construct_csv_from_model(request):

courses=Course.objects.all()

response=HttpResponse(content_type="text/csv")

response['Content-Disposition'] = 'attachment;

filename="courses_data.csv"

writer=csv.writer(response)

writer.writerow(["Course Name", "Course Code", "Credits"]) for course in

courses:

writer.writerow([course.course_name, course.course_code, course.course_credits])

return response

urls.py

from django.contrib import admin

from django.urls import path, re_path

from ap1.views import check_number, current_date_time from ap1.views

import four_hours_after, four_hours_before from ap1.views import

n_hours_after, display_string

from ap2.views import create_table_of_squares, vc, find_mode

from ap2.views import template_test, showlist, list_of_subjects from ap2.views

import aboutus, home, contactus, getpos, stable

from ap3.views import insert_demo, update_demo, delete_demo, retrieve_demo from

ap3.views import reg, course_search, add_project

from ap3.views import StudentListView, StudentDetailView, construct_csv from ap3.views

import construct_csv_from_model

admin.site.site_header="My Site Header"

```
admin.site.site_title="My Site Title"
admin.site.index_title="My Site Index" urlpatterns
= [
    path('secretadmin/', admin.site.urls),path('cdt/',
    current_date_time), path('fha/', four_hours_after),
    path('fhb/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/',check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist),
    path('list_of_subjects/', list_of_subjects),path('aboutus/',
    aboutus),
    path('home/', home), path('contactus/', contactus),
    path('getpos/', getpos), path('stable/', stable),
    path('insert_demo/', insert_demo),
    path('update_demo/', update_demo),
    path('delete_demo/', delete_demo),
    path('retreive_demo/', retreive_demo),path('reg/',
    reg), path('course_search/', course_search),
    path('add_project/', add_project),
    path('student_list/', StudentListView.as_view()), path('student_detail/<int:pk>',
    StudentDetailView.as_view()),path('construct_csv/', construct_csv),
    path('construct_csv_from_model/', construct_csv_from_model),
]
```

Output:

CSV file is generated and downloaded

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

views.py

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

from ap3.models import Course, Meeting, ProjectReg, Student
from reportlab.pdfgen import canvas

def construct_pdf_from_model(request):
    courses = Course.objects.all()
    response = HttpResponseRedirect(content_type="application/pdf")
    response['Content-Disposition'] = 'attachment;
filename="courses_data.pdf"'
    c = canvas.Canvas(response)

    c.drawString(70, 720, "Course Name")
    c.drawString(170, 720, "Course Code")
    c.drawString(270, 720, "Credits")
    y = 660
    for course in courses:
        c.drawString(70, y, course.course_name)
        c.drawString(170, y, course.course_code)
        c.drawString(270, y, str(course.course_credits))
        y = y - 60
    c.showPage()
    c.save()
    return response
```

urls.py

```
from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time
from ap1.views import four_hours_after, four_hours_before
from ap1.views import n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode
from ap2.views import template_test, showlist, list_of_subjects
from ap2.views import aboutus, home, contactus, getpos, stable
from ap3.views import insert_demo, update_demo, delete_demo, retrieve_demo
from ap3.views import reg, course_search, add_project
from ap3.views import StudentListView, StudentDetailView, construct_csv
from ap3.views import construct_csv_from_model
from ap3.views import construct_pdf_from_model
```

```

admin.site.site_header="My Site Header"
admin.site.site_title="My Site Title"
admin.site.index_title="My Site Index" urlpatterns
= [
    path('secretadmin/', admin.site.urls),path('cdt/',
    current_date_time), path('fha/', four_hours_after),
    path('fhb/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/',check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist),
    path('list_of_subjects/', list_of_subjects),path('aboutus/',
    aboutus),
    path('home/', home), path('contactus/', contactus),
    path('getpos/', getpos), path('stable/', stable),
    path('insert_demo/', insert_demo),
    path('update_demo/', update_demo),
    path('delete_demo/', delete_demo),
    path('retreive_demo/', retreive_demo),path('reg/',
    reg), path('course_search/', course_search),
    path('add_project/', add_project),
    path('student_list/', StudentListView.as_view()), path('student_detail/<int:pk>',
    StudentDetailView.as_view()),path('construct_csv/', construct_csv),
    path('construct_csv_from_model/', construct_csv_from_model),
    path('construct_pdf_from_model/', construct_pdf_from_model),

]

```

Output:

PDF file is generated and downloaded

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

views.py

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

from ap3.models import Course, Meeting, ProjectReg, Student

def regaj(request):
    if request.method == "POST":
        sid=request.POST.get("sname")
        cid=request.POST.get("cname")
        student=Student.objects.get(id=sid)
```

```

        course=Course.objects.get(id=cid)
        res=student.enrolment.filter(id=cid)if res:
            return HttpResponseRedirect("<h1>Student already enrolled</h1>")
        student.enrolment.add(course)
        return HttpResponseRedirect("<h1>Student enrolled successfully</h1>")

    else:
        students=Student.objects.all()
        courses=Course.objects.all()

        return render(request,"regaj.html",{ "students":students,
"courses":courses})

```

regaj.html

```

{% load static %}
<html>
    <body>
        <form method="post" action="">
            {% csrf_token %}
            Student Name
            <select name="sname" id="sname">
                {%for student in students %}
                <option value="{{ student.id }}">{{ student.student_name }}</option>
                {% endfor %}
            </select><br>
            Course Name
            <select name="cname" id="cname">
                {%for course in courses %}
                <option value="{{ course.id }}">{{ course.course_name }}</option>
                {% endfor %}

            </select><br>
            <span id="ans"></span>
            <input type="button" value="Enroll" id="ebtn">
        </form>
        <script src="{% static 'jquery.min.js' %}"></script>
        <script>
            $(document).ready(function(){
                $("#ebtn").click(function(){
                    var sname = $("#sname").val();var
                    cname = $("#cname").val();
                    $.ajax({
                        type: "POST", url:
                        "/regaj/",
                        data: {sname: sname, cname: cname,
                        csrfmiddlewaretoken:"{{ csrf_token }}"
                    },

```

```

                success: function(response){
                    $("#ans").html(response)
                }
            });
        });
    });
</script>
</body>
</html>

```

urls.py

```

from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time from ap1.views
import four_hours_after, four_hours_before from ap1.views import
n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode from ap2.views
import template_test, showlist, list_of_subjects from ap2.views import
aboutus, home, contactus, getpos, stable
from ap3.views import insert_demo, update_demo, delete_demo, retrieve_demo from
ap3.views import reg, course_search, add_project
from ap3.views import StudentListView, StudentDetailView, construct_csv from ap3.views
import construct_csv_from_model, construct_pdf
from ap3.views import construct_pdf_from_model
from ap3.views import expr, cbox, regaj, course_search_ajax
admin.site.site_header="My Site Header" admin.site.site_title="My Site
Title" admin.site.index_title="My Site Index"
urlpatterns = [
    path('secretadmin/', admin.site.urls), path('cdt/',
    current_date_time), path('fha/', four_hours_after),
    path('fhb/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/', check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist),
    path('list_of_subjects/', list_of_subjects), path('aboutus/',
    aboutus),
    path('home/', home), path('contactus/',
    contactus), path('getpos/', getpos),
    path('stable/', stable), path('insert_demo/',
    insert_demo),

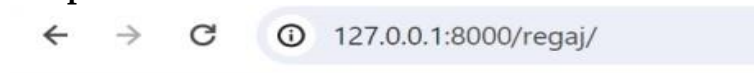
```



```

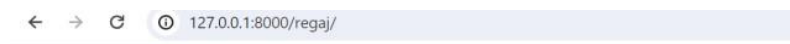
    path('update_demo/', update_demo),
    path('delete_demo/', delete_demo),
    path('retreive_demo/', retreive_demo), path('reg/',
    reg), path('course_search/', course_search),
    path('add_project/', add_project),
    path('student_list/', StudentListView.as_view()), path('student_detail/<int:pk>',
    StudentDetailView.as_view()), path('construct_csv/', construct_csv),
    path('construct_csv_from_model/', construct_csv_from_model), path('construct_pdf/',
    construct_pdf), path('construct_pdf_from_model/', construct_pdf_from_model),
    path('expr/', expr),
    path('cbox/', cbox),
    path('regaj/', regaj),
]

```

Output:

Student Name

Course Name



Student Name

Course Name

Student already enrolled



Student Name

Course Name

Student enrolled successfully

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

views.py

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

from ap3.models import Course, Meeting, ProjectReg, Studentdef

course_search_ajax(request):
    if request.method=="POST":
        cid=request.POST.get("cname")
        s=Student.objects.all() student_list=list()
        for student in s:
            if student.enrolment.filter(id=cid):
                student_list.append(student)
        if len(student_list)==0:
            return HttpResponseRedirect("<h1>No Students enrolled</h1>")return
    render(request,"selected_students.html",{ "student_list":student_list})

    else:
        courses=Course.objects.all()
        return render(request,"course_search_aj.html",{ "courses":courses})
```

course_search_aj.html

```
{% load static %}
<html>
  <body>
    <form method="POST" action="">
      Courses
      {% csrf_token %}
      <select name="cname" id="cname">
        {%for course in courses %}
          <option value="{{ course.id }}">{{ course.course_name }}</option>
        {% endfor %}
      </select>
      <input type="button" value="Search" id="serbtn">
      <span id="result"></span>
    </form>
  </body>
  <script src="{% static 'jquery.min.js' %}"></script>
  <script>
    $(document).ready(function(){
      $("#serbtn").click(function(){
        var cname = $("#cname").val();
        $.ajax({
```

```
url: "/course_search_ajax/",type:
"POST",
data: { cname:cname,csrfmiddlewaretoken:"{ {csrf_token } }"},success:
function(response){
    $("#result").html(response);
```

```

        }
    });
});
});

```

```

</script>
</html>

```

urls.py

```

from django.contrib import admin
from django.urls import path, re_path
from ap1.views import check_number, current_date_time from ap1.views
import four_hours_after, four_hours_before from ap1.views import
n_hours_after, display_string
from ap2.views import create_table_of_squares, vc, find_mode from ap2.views
import template_test, showlist, list_of_subjects from ap2.views import
aboutus, home, contactus, getpos, stable
from ap3.views import insert_demo, update_demo, delete_demo, retrieve_demo from
ap3.views import reg, course_search, add_project
from ap3.views import StudentListView, StudentDetailView, construct_csv from ap3.views
import construct_csv_from_model, construct_pdf
from ap3.views import construct_pdf_from_model
from ap3.views import expr, cbx, regaj, course_search_ajax, cookie_demo
admin.site.site_header="My Site Header"
admin.site.site_title="My Site Title"
admin.site.index_title="My Site Index"
urlpatterns = [

    path('secretadmin/', admin.site.urls), path('cdt/',
    current_date_time), path('fha/', four_hours_after),
    path('fhb/', four_hours_before),
    path('nha/<int:num>', n_hours_after),
    path('display_string/<slug:sentence>', display_string),
    re_path('check_number/(\d){1,2}/', check_number),
    path('cts/<int:s>/<int:n>', create_table_of_squares),
    path('vc/<str:sentence>', vc), path('find_mode/<str:listofnum>',
    find_mode), path('template_test/', template_test), path('showlist/',
    showlist),
    path('list_of_subjects/', list_of_subjects), path('aboutus/',
    aboutus),
    path('home/', home), path('contactus/',
    contactus), path('getpos/', getpos),
    path('stable/', stable), path('insert_demo/',
    insert_demo), path('update_demo/',
    update_demo), path('delete_demo/',
    delete_demo),

```

```

path('retrieve_demo/', retrieve_demo), path('reg', SHIVAMOGGA
reg), path('course_search/', course_search),
path('add_project/', add_project),
path('student_list/', StudentListView.as_view()), path('student_detail/<int:pk>/',
StudentDetailView.as_view()), path('construct_csv/', construct_csv),
path('construct_csv_from_model/', construct_csv_from_model), path('construct_pdf/',
construct_pdf), path('construct_pdf_from_model/', construct_pdf_from_model),
path('expr/', expr),
path('cbox/', cbox), path('course_search_ajax/',
course_search_ajax),
]

```

Output:

← → ↻ ⓘ 127.0.0.1:8000/course_search_ajax/

Courses

← → ↻ ⓘ 127.0.0.1:8000/course_search_ajax/

Courses

Student Name	Student USN	Sem
Sham	4JN22AI001	4
Manish	4JN22AI002	4