

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JnanaSangama, Belagavi - 590 018, Karnataka



## Acharya Institute of Technology

Acharya Dr Sarvepalli. Radhakrishnan Road

Acharya P.O Soladevanahalli, Bengaluru-560107



### LABORATORY MANUAL

### FULLSTACK DEVELOPMENT

21CS62

VI Semester

Prepared by:

1. Mr. Vinayak Raju Kage

2. Mr. Gowtham Raj

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Accredited by NBA)

## Table of contents

Vision, Mission, Motto of Institute	I
Vision, Mission of Department	II
Program Educational Objectives (PEOs)	
Program Specific Outcomes (PSOs)	
Program outcomes (POs)	
Course outcomes of course (COs)	
CO-PO matrix	

SL	Name of Program	Page No
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.	
5	Develop a simple Django app that displays an unordered list of fruits and ordered list of selected students for an event	
6	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.	
7	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.	
8	For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms.	
9	Develop a Model form for student that contains his topic chosen for project, languages used and duration with a model called project.	
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.	
11	Develop example Django app that performs CSV and PDF generation for any models created in previous laboratory component.	

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

## **Laboratory Experiments:**

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.
5. Develop a simple Django app that displays an unordered list of fruits and ordered list of selected students for an event
6. 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
7. 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
8. For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms.
9. Develop a Model form for student that contains his topic chosen for project, languages used and duration with a model called project
10. For students enrolment developed in Module 2, create a generic class view which displays list of students and detail view that displays student details for any selected student in the list.
11. Develop example Django app that performs CSV and PDF generation for any models created in previous laboratory component.
12. Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX.
13. Develop a search application in Django using AJAX that displays courses enrolled by a student being searched.

## Evaluation Rubrics for lab Programs (Max marks 20)

### A. Lab write-up and execution rubrics(Max marks 8)

		<b>Good</b>	<b>Average</b>
<b>a.</b>	<b>Understanding of problem (3 marks)</b>	Demonstrate good knowledge of language constructs and programming practice (3)	Moderate understanding of language constructs (1)
<b>b.</b>	<b>Execution and testing (3marks)</b>	Program handles all possible conditions and results with satisfying results. (3)	Partial executions /poor error handling (1)
<b>c.</b>	<b>Result and documentation (2 marks)</b>	Meticulous documentation of changes made and results obtained are in proper format (2)	Moderate formatting of output and average documentation (1)

### B. VIVA Rubrics: (Max marks 2)

		<b>Good</b>	<b>Average</b>
	<b>Conceptual understanding (2 marks)</b>	Explain the complete program with the related concepts.(5)	Adequately provides explanation.(3)

### C. Marks for Lab Record:10

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

Python download link <https://www.python.org/downloads/>

To check out python installed version `py -version`

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

Django installation commands to be typed in cmd

- `py -m pip install Django`

or

`py -m pip install Django==5.0.4 #version`

## 2. Creation of virtual environment, Django project and App should be demonstrated

- Create virtual environment!

```
# Install virtualenv if you haven't already  
py -m pip install --user virtualenv
```

```
# Create a new virtual environment  
py -m venv myenv (myenv is virtual environment name)
```

```
# Activate the virtual environment  
python manage.py runserver
```

- **Installing Django**

Once the virtual environment is activated, you can install Django

```
python -m pip install --upgrade pip  
# Install Django  
pip install Django
```

- **Creating a Django Project**

```
# Navigate to your desired directory  
cd /path/to/your/directory  
# Create a new Django project  
django-admin startproject myproject
```

- **Running the Django Development Server**

```
# Run the development server  
python manage.py runserver
```

- **Creating a Django App**

```
# Ensure you are in the project directory  
cd /path/to/your/directory/myproject
```

```
# Create a new app called 'myapp'  
python manage.py startapp myapp
```

- **Running Migrations**

```
python manage.py makemigrations  
python manage.py migrate
```

- **Creating Superuser (Admin User)**

```
python manage.py createsuperuser  
python manage.py help  
python manage.py runserver
```

### 3. develop a Django app that displays current date and time in server

To create environment → python -m venv myenv

Activate the environment → .\myenv\Scripts\activate

#Instal Django → pip install django #if u created environment newly

---

#### Create a new Django project:

```
django-admin startproject time_project
```

```
cd time_project
```

---

#### Create a new Django app

```
python manage.py startapp time_app
```

---

##### time\_app/views

```
from django.http import HttpResponse
from datetime import datetime
def current_datetime(request):
    now = datetime.now()
    html = "<html><body>Current date and time: {}</body></html>".format(now)
    return HttpResponse(html)
```

##### time\_project/urls.py

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

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

#### #Create a new file called urls.py inside the time\_app directory

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.current_datetime, name='current_datetime'),
]
```

#### to run

```
Python manage.py runserver
```



**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.**

```
django-admin startproject time_project  
cd time_project
```

---

### **Create a new Django app**

```
python manage.py startapp time_app
```

---

#### **time\_app/views**

```
from django.http import HttpResponse  
from datetime import datetime  
def current_datetime(request):  
    now1 = datetime.now() - timedelta(hours=4)  
    now2 = datetime.now() + timedelta(hours=4)  
    html = "<html><body>Current date and time 4hrs ago: {}<br>".format(now1)  
    Current date and time 4hrs after: <br>{}</body></html>".format(now2)  
  
    return HttpResponse(html)
```

#### **time\_project/urls.py**

```
from django.contrib import admin  
from django.urls import path, include  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('time/', include('time_app.urls')),  
]
```

#### **#Create a new file called urls.py inside the time\_app directory**

```
from django.urls import path  
from . import views  
urlpatterns = [  
    path('', views.current_datetime, name='current_datetime'),  
]
```

#### **to run**

```
Python manage.py runserver
```

## 5. Develop a simple django app that displays an unordered list of fruits and ordered list of selected students for an event

```
django-admin startproject listapp
cd listapp
```

### Create a new Django app

```
python manage.py startapp lists
```

---

#### list/model.py

```
from django.db import models
class Fruit(models.Model):
    name = models.CharField(max_length=100)
    def __str__(self):
        return self.name
class Student(models.Model):
    name = models.CharField(max_length=100)
    def __str__(self):
        return self.name
```

#### list/admin.py

```
from django.contrib import admin
from .models import Fruit, Student
admin.site.register(Fruit)
admin.site.register(Student)
```

#### list/views.py

```
from django.shortcuts import render
from .models import Fruit, Student
def fruit_list(request):
    fruits = Fruit.objects.all()
    return render(request, 'lists/fruit_list.html', {'fruits': fruits})
def student_list(request):
    students = Student.objects.all()
    return render(request, 'lists/student_list.html', {'students': students})
```

#### lists/templates/lists/fruit\_list.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Fruit List</title>
</head>
<body>
    <h1>Unordered List of Fruits</h1>
    <ul>
```

```

        {% for fruit in fruits %}
        <li>{{ fruit.name }}</li>
        {% endfor %}
    </ul>
</body>
</html>

```

### **lists/templates/lists/student\_list.html**

```

<!DOCTYPE html>

<html>
<head>
    <title>Student List</title>
</head>
<body>
    <h1>Ordered List of Selected Students</h1>
    <ol>
        {% for student in students %}
        <li>{{ student.name }}</li>
        {% endfor %}
    </ol>
</body>
</html>

```

### **List/urls.py**

```

from django.urls import path
from . import views
urlpatterns = [
    path('fruits/', views.fruit_list, name='fruit_list'),
    path('students/', views.student_list, name='student_list'),
]

```

### **Listsapp/urls.py**

```

from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('lists/', include('lists.urls')),
]

```

### **listapp/settings.py**

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'lists', # Add your app name here
]

```

## **Migration and database changes**

```
python manage.py makemigrations  
python manage.py migrate
```

### **create user in command mode**

```
python manage.py createsuperuser
```

### **restart or run server**

```
python manage.py runserver
```

### **in browser**

<http://127.0.0.1:8000/lists/students/>

<http://127.0.0.1:8000/lists/fruits/>

6. 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

```
django-admin startproject mywebsite  
cd mywebsite
```

```
python manage.py startapp pages
```

### **pages/urls.py**

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

```
urlpatterns = [  
    path("", views.home, name='home'),  
    path('about/', views.about, name='about'),  
    path('contact/', views.contact, name='contact'),  
]
```

### **Pages/views.py**

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

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

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

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

### **pages/templates/layout.html**

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>{ % block title % }My Website{ % endblock % }</title>  
    <style>  
        /* Basic CSS for layout */  
        body {  
            font-family: Arial, sans-serif;  
            margin: 0;  
            padding: 0;  
            box-sizing: border-box;  
        }  
        header {  
            background-color: #333;
```

```

        color: #fff;
        padding: 10px 0;
        text-align: center;
    }
    nav {
        display: flex;
        justify-content: center;
    }
    nav a {
        color: #fff;
        text-decoration: none;
        padding: 10px 20px;
    }
    nav a:hover {
        background-color: #555;
    }
    footer {
        background-color: #333;
        color: #fff;
        text-align: center;
        padding: 10px 0;
        position: absolute;
        bottom: 0;
        width: 100%;
    }
</style>
</head>
<body>
<header>
<h1>My Website</h1>
<nav>
<a href="{ % url 'home' % }">Home</a>
<a href="{ % url 'about' % }">About Us</a>
<a href="{ % url 'contact' % }">Contact Us</a>
</nav>
</header>

<div id="content">
    { % block content % }
    <!-- This block will be overridden by specific content of each page -->
    { % endblock % }
</div>
<footer>
    <p>&copy; { % now 'Y' % } My Website | Developed by Your Name</p>
</footer>
</body>
</html>

```

### **pages/templates/home.html**

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

{% block title %}Home - My Website{% endblock %}

{% block content %}
    <h2>Welcome to Our Home Page!</h2>
    <p>This is the Home Page content of My Website.</p>
    <!-- Add more content specific to the home page here -->
{% endblock %}
```

### **pages/templates/about.html**

```
{% extends 'layout.html' %}
{% block title %}About Us - My Website{% endblock %}
{% block content %}
    <h2>About Us</h2>
    <p>Learn more about our company or organization.</p>
    <!-- Add more content specific to the about page here -->
{% endblock %}
```

### **pages/templates/contact.html**

```
{% extends 'layout.html' %}
{% block title %}Contact Us - My Website{% endblock %}
{% block content %}
    <h2>Contact Us</h2>
    <p>Get in touch with us using the form below or via other contact information.</p>
    <!-- Add more content specific to the contact page here -->
{% endblock %}
```

### **Mywebsite/urls.py**

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('pages.urls')), # Include the URLs from the pages app
]
```

### **mywebsite/settings.py**

```
add--- import os
'DIRS': [os.path.join(BASE_DIR, 'pages/templates')], #replacet this in templates
```

**To run python manage.py runserver**

<http://127.0.0.1:8000/home>

7. 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.

To create environment → `python -m venv myenv`  
Activate the environment → `.\myenv\Scripts\activate`

#Instal Django → if u created environment newly

---

### **Create a new Django project:**

```
django-admin startproject student_registration  
cd student_registration
```

---

### **Create a new Django app**

```
python manage.py startapp registration
```

---

### **Add the new app to your project's settings:**

In `student_registration/settings.py`, add 'registration' to the `INSTALLED_APPS`

```
INSTALLED_APPS = [  
    ...  
    'registration',  
]
```

---

### **Create the models** in `registration/models.py`:

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

```
class Student(models.Model):  
    first_name = models.CharField(max_length=50)  
    last_name = models.CharField(max_length=50)  
    email = models.EmailField(unique=True)
```

```
    def __str__(self):  
        return f'{self.first_name} {self.last_name}'
```

```
class Course(models.Model):  
    name = models.CharField(max_length=100)  
    description = models.TextField()
```

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

```
class Enrollment(models.Model):  
    student = models.ForeignKey(Student, on_delete=models.CASCADE)  
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
```



```
enrolled_at = models.DateTimeField(auto_now_add=True)
```

```
class Meta:
```

```
    unique_together = ('student', 'course')
```

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

```
    return f'{self.student} enrolled in {self.course}'
```

---

**Run migrations** to create the database schema:

```
python manage.py makemigrations
```

```
python manage.py migrate
```

---

**admin interface registration/admin.py**

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

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

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

```
admin.site.register(Course)
```

```
admin.site.register(Enrollment)
```

---

**Create views for registration and displaying enrolled students** in registration/views.py:

```
from django.shortcuts import render, redirect, get_object_or_404
```

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

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

```
def course_list(request):
```

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

```
    return render(request, 'registration/course_list.html', {'courses': courses})
```

```
def enroll_student(request, course_id):
```

```
    course = get_object_or_404(Course, id=course_id)
```

```
    students = Student.objects.all()
```

```
    if request.method == 'POST':
```

```
        student_id = request.POST.get('student')
```

```
        student = get_object_or_404(Student, id=student_id)
```

```
        Enrollment.objects.create(student=student, course=course)
```

```
        return redirect('course_list')
```

```
    return render(request, 'registration/enroll_student.html', {'course': course, 'students': students})
```

```
def enrolled_students(request, course_id):
```

```
    course = get_object_or_404(Course, id=course_id)
```

```
    enrollments = Enrollment.objects.filter(course=course)
```

```
    return render(request, 'registration/enrolled_students.html', {'course': course, 'enrollments': enrollments})
```

---

**Create URLs for these views** in registration/urls.py:

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

```
urlpatterns = [
    path("", views.course_list, name='course_list'),
    path('course/<int:course_id>/enroll/', views.enroll_student, name='enroll_student'),
    path('course/<int:course_id>/students/', views.enrolled_students, name='enrolled_students'),
]
```

---

**Project folder student\_registration/urls.py**

```
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('registration/', include('registration.urls')),
]
```

---

**Create templates** for each view:

**registration/templates/registration/course\_list.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>Courses</title>
</head>
<body>
    <h1>Courses</h1>
    <ul>
        {% for course in courses %}
            <li>
                {{ course.name }} - <a href="{% url 'enroll_student' course.id %}">Enroll</a> -
                <a href="{% url 'enrolled_students' course.id %}">View Students</a>
            </li>
        {% endfor %}
    </ul>
</body>
</html>
```

### registration/templates/registration/enroll\_student.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Enroll Student</title>
</head>
<body>
  <h1>Enroll Student in {{ course.name }}</h1>
  <form method="post">
    {% csrf_token %}
    <select name="student">
      {% for student in students %}
        <option value="{{ student.id }}">{{ student.first_name }} {{ student.last_name }}</option>
      {% endfor %}
    </select>
    <button type="submit">Enroll</button>
  </form>
</body>
</html>
```

### registration/templates/registration/enrolled\_students.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Enrolled Students</title>
</head>
<body>
  <h1>Students Enrolled in {{ course.name }}</h1>
  <ul>
    {% for enrollment in enrollments %}
      <li>{{ enrollment.student.first_name }} {{ enrollment.student.last_name }}</li>
    {% endfor %}
  </ul>
</body>
</html>
```

---

### To create superuser

Python manage.py createsuperuser

---

**To run**

Python manage.py runserver

**Ouput link** → <http://127.0.0.1:8000/admin/> add student name, course, enrolment

**Test app** → <http://127.0.0.1:8000/registration/>

---

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

```
django-admin startproject college
cd college
django-admin startapp students
```

---

#### **students/models.py**

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

```
class Course(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
```

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

```
class Student(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email = models.EmailField(unique=True)
    courses = models.ManyToManyField(Course)
```

```
    def __str__(self):
        return f"{self.first_name} {self.last_name}"
```

---

#### **students/admin.py**

```
from django.contrib import admin
from .models import Student, Course
```

```
class CourseAdmin(admin.ModelAdmin):
    list_display = ('name', 'description')
    search_fields = ('name',)
```

```
class StudentAdmin(admin.ModelAdmin):
    list_display = ('first_name', 'last_name', 'email')
    search_fields = ('first_name', 'last_name', 'email')
    filter_horizontal = ('courses',)
```

```
admin.site.register(Course, CourseAdmin)
admin.site.register(Student, StudentAdmin)
```

---

#### **college/settings.py**

```
INSTALLED_APPS = [
    ...
    'students',
    ...
]
```

---

#### **College/urls.py**

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

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

---

**Ouput link** → <http://127.0.0.1:8000/admin/>

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

```
django-admin startproject collegeportal
cd collegeportal
django-admin startapp studentprojects
```

---

#### **studentprojects/models.py**

```
from django.db import models
class Project(models.Model):
    student_name = models.CharField(max_length=100)
    topic_chosen = models.CharField(max_length=200)
    languages_used = models.CharField(max_length=200)
    duration = models.IntegerField(help_text="Duration in weeks")

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

---

#### **studentprojects/forms.py**

```
from django import forms
from .models import Project

class ProjectForm(forms.ModelForm):
    class Meta:
        model = Project
        fields = ['student_name', 'topic_chosen', 'languages_used', 'duration']
```

---

#### **studentprojects/views.py**

```
from django.shortcuts import render, redirect
from .forms import ProjectForm

def project_create_view(request):
    if request.method == 'POST':
        form = ProjectForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('project_success')
    else:
        form = ProjectForm()
    return render(request, 'studentprojects/project_form.html', {'form': form})

def project_success_view(request):
    return render(request, 'studentprojects/project_success.html')
```

---

#### **studentprojects/templates/studentprojects/ project\_form.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>Create Project</title>
</head>
<body>
    <h2>Create Project</h2>
```

```
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Submit</button>
</form>
</body>
</html>
```

---

### **project\_success.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>Project Success</title>
</head>
<body>
    <h2>Project Submitted Successfully!</h2>
</body>
</html>
```

---

### **studentprojects/urls.py**

```
from django.urls import path
from .views import project_create_view, project_success_view
urlpatterns = [
    path('create/', project_create_view, name='project_create'),
    path('success/', project_success_view, name='project_success'),
]
```

---

### **Collegeportal/Urls.py**

```
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
    path('admin/', admin.site.urls),
    path('projects/', include('studentprojects.urls')),
]
```

---

```
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
```

---

**Ouput link →** <http://127.0.0.1:8000/projects/create/>



10. create a simple blog application with generic views.

```
django-admin startproject mysite
cd mysite
python manage.py startapp blog
```

---

#### **blog/model.py**

```
from django.db import models
from django.urls import reverse

class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('post_detail', kwargs={'pk': self.pk})
```

---

#### **blog/views.py**

```
from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView
from .models import Post
```

```
class PostListView(ListView):
    model = Post
    template_name = 'post_list.html' # default is 'blog/post_list.html'
    context_object_name = 'posts'
```

```
class PostDetailView(DetailView):
    model = Post
    template_name = 'post_detail.html' # default is 'blog/post_detail.html'
```

```
class PostCreateView(CreateView):
    model = Post
    template_name = 'post_form.html' # default is 'blog/post_form.html'
    fields = ['title', 'content']
```

```
class PostUpdateView(UpdateView):
    model = Post
    template_name = 'post_form.html'
    fields = ['title', 'content']
```

```
class PostDeleteView(DeleteView):
    model = Post
    template_name = 'post_confirm_delete.html' # default is 'blog/post_confirm_delete.html'
    success_url = '/'
```

---

#### **blog/urls.py**

```
from django.urls import path
from .views import PostListView, PostDetailView, PostCreateView, PostUpdateView, PostDeleteView
```

```
urlpatterns = [
    path("", PostListView.as_view(), name='post_list'),
    path('post/<int:pk>/', PostDetailView.as_view(), name='post_detail'),
    path('post/new/', PostCreateView.as_view(), name='post_create'),
    path('post/<int:pk>/edit/', PostUpdateView.as_view(), name='post_update'),
    path('post/<int:pk>/delete/', PostDeleteView.as_view(), name='post_delete'),
]
```

---

### **Mysite/urls**

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

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

---

blog/templates/post\_list.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Blog</title>
</head>
<body>
    <h1>Blog Posts</h1>
    <a href="{% url 'post_create' %}">Create new post</a>
    <ul>
        {% for post in posts %}
            <li>
                <a href="{% url 'post_detail' post.pk %}">{{ post.title }}</a>
                <p>{{ post.content }}</p>
            </li>
        {% endfor %}
    </ul>
</body>
</html>
```

---

Post\_detail.html

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ post.title }}</title>
</head>
<body>
    <h1>{{ post.title }}</h1>
    <p>{{ post.content }}</p>
    <a href="{% url 'post_update' post.pk %}">Edit</a>
    <a href="{% url 'post_delete' post.pk %}">Delete</a>
    <a href="{% url 'post_list' %}">Back to list</a>
</body>
</html>
```

---

Post\_form.html

```
<!DOCTYPE html>
```

```
<html>
<head>
  <title>{% if object %}Edit{% else %}New{% endif %} Post</title>
</head>
<body>
  <h1>{% if object %}Edit{% else %}New{% endif %} Post</h1>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save</button>
  </form>
  <a href="{% url 'post_list' %}">Back to list</a>
</body>
</html>
```

---

post\_confirm\_delete.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Delete Post</title>
</head>
<body>
  <h1>Delete Post</h1>
  <p>Are you sure you want to delete "{{ post.title }}"?</p>
  <form method="post">
    {% csrf_token %}
    <button type="submit">Yes, delete</button>
  </form>
  <a href="{% url 'post_detail' post.pk %}">Cancel</a>
</body>
</html>
```

---

Installed apps add blog

---

```
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
```

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

```
django-admin startproject student_management
cd student_management
python manage.py startapp enrollment
```

---

**enrollment/models.py**

```
from django.db import models

class Student(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    enrollment_date = models.DateField()
    def __str__(self):
        return f"{self.first_name} {self.last_name}"
```

---

**student\_management/settings.py**

```
INSTALLED_APPS = [
    ...
    'enrollment',
]
```

---

```
python manage.py makemigrations
```

```
python manage.py migrate
```

---

```
pip install reportlab
```

---

```
from django.urls import include, path
urlpatterns = [
    path('admin/', admin.site.urls),
    path('projects/', include('studentprojects.urls')),
]
```

---

---

**enrollment/views.py**

```
import csv
from django.http import HttpResponse
from reportlab.pdfgen import canvas
from .models import Student
```

```
def export_students_csv(request):
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="students.csv"'
```

```
    writer = csv.writer(response)
    writer.writerow(['First Name', 'Last Name', 'Enrollment Date'])
```

```
    students = Student.objects.all().values_list('first_name', 'last_name', 'enrollment_date')
    for student in students:
        writer.writerow(student)
```

```

return response

def export_students_pdf(request):
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = 'attachment; filename="students.pdf"'

    p = canvas.Canvas(response)

    students = Student.objects.all()
    y = 800
    for student in students:
        p.drawString(100, y, f"{student.first_name} {student.last_name} - {student.enrollment_date}")
        y -= 20

    p.showPage()
    p.save()
    return response

```

---

#### **enrollment/urls.py**

```

from django.urls import path
from .views import export_students_csv, export_students_pdf

urlpatterns = [
    path('export/csv/', export_students_csv, name='export_students_csv'),
    path('export/pdf/', export_students_pdf, name='export_students_pdf'),
]

```

---

#### **student\_management/urls.py**

```

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

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

```

---

#### **enrollment/admin.py**

```

from django.contrib import admin
from .models import Student
@admin.register(Student)
class StudentAdmin(admin.ModelAdmin):
    list_display = ('first_name', 'last_name', 'enrollment_date')

```

---

python manage.py createsuperuser  
python manage.py runserver  
goto this url and add few student details

**Test CSV and PDF Export**

Go to <http://127.0.0.1:8000/enrollment/export/csv/> to download the CSV file.

Go to <http://127.0.0.1:8000/enrollment/export/pdf/> to download the PDF file

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

```
django-admin startproject college_management
cd college_management
python manage.py startapp registration
```

---

**enrollment/models.py**

```
from django.db import models
class Student(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    enrollment_date = models.DateField()
    def __str__(self):
        return f"{self.first_name} {self.last_name}"
```

---

**college\_management/settings.py**

```
INSTALLED_APPS = [
    ...
    'registration',
    ...
]
```

---

**registration/models.py**

```
from django.db import models
class Student(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    date_of_birth = models.DateField()

    def __str__(self):
        return f"{self.first_name} {self.last_name}"
```

---

**registration/forms.py**

```
from django import forms
from .models import Student
class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = ['first_name', 'last_name', 'email', 'date_of_birth']
```

---

**Registration/views.py**

```
from django.shortcuts import render
from django.http import JsonResponse
from .forms import StudentForm
def student_registration(request):
    if request.method == 'POST' and request.headers.get('x-requested-with') == 'XMLHttpRequest':
        form = StudentForm(request.POST)
        if form.is_valid():
```

```

        form.save()
        return JsonResponse({'success': True})
    else:
        return JsonResponse({'success': False, 'errors': form.errors})
    else:
        form = StudentForm()
    return render(request, 'registration/student_registration.html', {'form': form})

```

### registration/templates/registration/student\_registration.html

```

<!DOCTYPE html>
<html>
<head>
    <title>Student Registration</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <h2>Student Registration</h2>
    <form id="registrationForm" method="POST">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Register</button>
    </form>

    <div id="successMessage" style="display:none;">
        <p>Registration successful!</p>
    </div>

    <div id="errorMessage" style="display:none;">
        <p>There were errors in your form:</p>
        <ul id="errorList"></ul>
    </div>

    <script>
    $(document).ready(function() {
        $('#registrationForm').on('submit', function(event) {
            event.preventDefault();
            $.ajax({
                type: 'POST',
                url: "",
                data: $(this).serialize(),
                success: function(response) {
                    if (response.success) {
                        $('#registrationForm')[0].reset();
                        $('#successMessage').show();
                        $('#errorMessage').hide();
                    } else {
                        $('#errorMessage').show();
                    }
                }
            });
        });
    });
    </script>

```



```

        $('#errorList').empty();
        $.each(response.errors, function(key, value) {
            $('#errorList').append('<li>' + key + ': ' + value + '</li>');
        });
    }
}
});
});
});
</script>
</body>
</html>

```

---

### **registration/urls.py**

```

from django.urls import path
from . import views

```

```

urlpatterns = [
    path('register/', views.student_registration, name='student_registration'),
]

```

---

### **college\_management/urls.py**

```

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

```

```

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

```

---

```

python manage.py makemigrations
python manage.py migrate
python manage.py runserver

```

To run → <http://127.0.0.1:8000/registration/register/>

---

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

```
django-admin startproject education_portal
cd education_portal
python manage.py startapp search
```

#### **education\_portal/settings.py**

```
INSTALLED_APPS = [
```

```
...
```

```
'search',
```

```
...
```

```
]
```

---

#### **search/models.py**

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

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

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

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

```
    email = models.EmailField(unique=True)
```

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

```
        return f"{self.first_name} {self.last_name}"
```

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

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

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

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

```
        return self.name
```

---

```
python manage.py makemigrations
```

```
python manage.py migrate
```

---

#### **Search/forms.py**

```
from django import forms
```

```
class StudentSearchForm(forms.Form):
```

```
    email = forms.EmailField(label='Student Email', max_length=100)
```

#### **Search/views.py**

```

from django.shortcuts import render
from django.http import JsonResponse
from .models import Student
from .forms import StudentSearchForm

def search_student_courses(request):
    if request.method == 'GET' and request.is_ajax():
        email = request.GET.get('email')
        student = Student.objects.filter(email=email).first()
        if student:
            courses = student.courses.all()
            course_names = [course.name for course in courses]
            return JsonResponse({'success': True, 'courses': course_names})
        else:
            return JsonResponse({'success': False, 'message': 'Student not found'})
    else:
        form = StudentSearchForm()
        return render(request, 'search/student_search.html', {'form': form})

```

#### ----- search/templates/search/student\_search.html

```

<!DOCTYPE html>
<html>
<head>
    <title>Student Course Search</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <h2>Search for Student Courses</h2>
    <form id="searchForm" method="GET">
        { % csrf_token % }
        { { form.as_p } }
        <button type="submit">Search</button>
    </form>

    <div id="results" style="display:none;">
        <h3>Courses:</h3>
        <ul id="courseList"></ul>
    </div>

    <div id="errorMessage" style="display:none;">
        <p>Student not found</p>
    </div>

```

```

<script>
$(document).ready(function() {
    $('#searchForm').on('submit', function(event) {
        event.preventDefault();
        $.ajax({
            type: 'GET',
            url: "",
            data: $(this).serialize(),
            success: function(response) {
                if (response.success) {
                    $('#courseList').empty();
                    $.each(response.courses, function(index, course) {
                        $('#courseList').append('<li>' + course + '</li>');
                    });
                    $('#results').show();
                    $('#errorMessage').hide();
                } else {
                    $('#results').hide();
                    $('#errorMessage').show();
                }
            }
        });
    });
});
</script>
</body>
</html>

```

---

### **search/urls.py**

```

from django.urls import path
from . import views

urlpatterns = [
    path('search/', views.search_student_courses, name='search_student_courses'),
]

```

---

### **education\_portal/urls.p**

```

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

```

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

---

### **Search/admin.py**

```
from django.contrib import admin  
from .models import Student, Course
```

```
@admin.register(Student)  
class StudentAdmin(admin.ModelAdmin):  
    list_display = ('first_name', 'last_name', 'email')
```

```
@admin.register(Course)  
class CourseAdmin(admin.ModelAdmin):  
    list_display = ('name', 'get_students')  
  
    def get_students(self, obj):  
        return ", ".join([student.first_name + " " + student.last_name for student in obj.students.all()])  
    get_students.short_description = 'Students Enrolled'
```

---

Create super user to add student details  
python manage.py createsuperuser  
and add student details and subject name

---

To run python manage.py runserver  
To test <http://127.0.0.1:8000/search/search/>

---