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

1. Installing _Python:

1. **Download _Python:**

- Go to the official Python website: python.org.
- Navigate to the Downloads section and download the latest version of Python for your _operating system (Windows, macOS, or Linux).

1. **Install Python:**

- Once the download is complete, run the installer.
- Make sure to check the box that says “Add Python to PATH” during the installation process.
- Click on “Install Now” to complete the installation.

1. **Verify Installation:**

- Open a command prompt (on Windows) or terminal (on macOS or Linux).
- Type `python --version` or `python3 --version` and press Enter.
- You should see the installed Python version printed, confirming that Python is installed correctly.

2. Installing Django:

1. **Install Django using pip:**

- Open a command prompt or terminal.
- Type `pip install django` and press Enter.
- Pip is the package installer for _Python, and it will download and install Django and its dependencies.

1. **Verify Django Installation:**

- After installation, you can verify Django by typing `django-admin --version` in the command prompt or terminal.

- You should see the installed Django version printed, confirming that Django is installed correctly.

3. Installing Visual Studio Code (VS Code):

1. **Download VS Code:**

- Go to the official Visual Studio Code website: code.visualstudio.com.
- Click on the Download for [Your Operating System] button to download the installer.

1. **Install VS Code:**

- Run the downloaded installer.
- Follow the installation wizard instructions.
- During installation, you can choose options like adding VS Code to the PATH for easy access from the command line.

1. **Open VS Code:**

- After installation, open Visual Studio Code.
- You can open VS Code from the Start Menu (Windows), Applications folder (macOS), or from the installed directory (Linux).

Demonstrating the Installation:

- Once everything is installed:
- Open VS Code.
- Create a new folder for your Django project.
- Open this folder in VS Code.
- Open a terminal in VS Code (`Ctrl +` or `Cmd +`) and start a new Django project by running `django-admin startproject myprojectname`.
- Navigate into the project folder and start the Django development server with `python manage.py runserver`.
- You can now access your Django application by visiting `http://localhost:8000` in your web browser.

2) Creation of virtual environment, Django project and App should be demonstrate.

Step-01: Create a new folder for your project in any location.

Step-02: Open that created folder in the Visual Studio Code.

Step-03: Open the VS Code integrated terminal.

Step-04: Create a virtual environment:-

◉ In the terminal, run the below command to create a new virtual environment.

```
python -m venv env
```

Step-05: Activate the virtual environment:-

◉ In the terminal, run the below command to activate the virtual environment.

```
env\Scripts\activate
```

Step-06: Install Django:-

◉ Run the below command to install Django.

```
pip install django
```

Step-07: Create a new _Django project:-

◉ Run the below command to create _Django project.

```
django-admin startproject project
```

Step-08: Create a new Django app:-

◉ After changing the directory create a Django app using below command.

```
python manage.py startapp firstapp
```

Step-09: Add the app to the installed_apps list:-

◉ locate the settings._py file (usually located in the project directory) and open it.

◉ After then add your app name in INSTALLED_APPS list as per below image.

Step-10: Run Your Project:-

◉ Now setup is completed you can run your project using below command.

```
python manage.py runserver
```

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

Step1: Create a virtual environment:-

```
python -m venv env
```

Step2: Activate Virtual Environment

```
env\Scripts\activate
```

Step3: Install Django (if already installed ignore)

```
pip install Django
```

Step4: Create Django Project

```
django-admin startproject project
```

Step 5: Create Django app in that project

```
python manage.py startapp datetime_app
```

Step 6: Add the datetime_app to the installed_apps list

```
INSTALLED_APPS= [‘ datetime_app’]
```

Step 7: Inside views.py file create a function:-

(datetime_app/views.py).

```
from django.shortcuts import render
import datetime
def datetime_app(request):
    now = datetime.datetime.now ()
    context = {'datetime_app': now}
   return render (request, 'datetime_app.html', context)
```

Step 8: Create a template

Inside the templates folder, create a new file named datetime_app.html.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Current Date and Time</title>
</head>
<body >
   <div >
   <h1>Current Date and Time on the Server:</h1>
   <p>{{ datetime_app }}</p>
   </div>
</body>
</html>
```

Step 9: Include the datetime_app URLs in the project's URL patterns (project/urls._py).

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

Step 10 : Run Your Project

```
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.

Step1: **In the same project folder whatever we made earlier create again one new app name called as datetimeoffset_app using below command.**

```
python manage.py startapp datetimeoffset_app
```

Step2: Add the datetimeoffset_app to the installed_apps list:-

Step3: Inside views.py file create a function:-

◉ Open the views.py file in your Django project directory (datetimeoffset_app/views.py).

```
from django.shortcuts import render
import datetime
def datetimeoffset_app(request):
 now = datetime.datetime.now()
 context = {
  'current_datetime': now,
  'four_hours_ahead': now + datetime.timedelta(hours=4),
  'four_hours_before': now - datetime.timedelta(hours=4),
 }
 return render(request, 'datetimeoffset_app.html', context)
```

Step4: Create a template:-

◉ Right click on datetimeoffset_app folder, create a new folder named templates.

◉ Inside the templates folder, create a new file named datetimeoffset_app.html.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title> Date and Time Offsets</title>
</head>
<body>
```

```
<div class="container">
   <h1>Current Date and Time on the Server</h1>
<p>{{ current_datetime }}</p>
<h2>Four Hours Ahead</h2>
<p>{{ four_hours_ahead }}</p>
<h2>Four Hours Before</h2>
<p>{{ four_hours_before }}</p>
</div>
</body>
</html>
```

Step 5: Include the datetimeoffset_app URLs in the project's URL patterns:-

◉ Open the file in your Django project directory (project/urls._py).

◉ Import the view function at the top of the file.

◉ Add a new URL pattern to the urlpatterns list.

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

from  datetimeoffset_app.views import datetimeoffset_app

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

Step 7 : Run Your Project

```
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

Step1: **Create new folder:-**

◉ In the same project folder whatever we made earlier create again one new app name called as fruitlist_app using below command.

```
python manage.py startapp fruitlist_app
```

Step2: Add the fruitlist_app to the installed_apps list:-

Step3: Inside views. py file create a function:-

◉ Open the views.py file in your _Django project directory (fruitlist_app/views.py).

```
from django.shortcuts import render

def fruitlist_app(request):
 fruits = ['Apple', 'Mango', 'Orange', 'Pineapple','Banana']
 students = ['Braham', 'Bikash', 'Shoaib', 'Aman', 'Shubham']
 context = {
     'fruits': fruits,
    'students': students,
   }
 return render(request, 'fruitlist_app.html', context)
```

Step4: Create a template:-

◉ Right click on fruitlist_app folder, create a new folder named templates.

◉ Inside the templates folder, create a new file named fruitlist_app.html.

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

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title> Fruits and Student List</title>
</head>
```

```
<body>
  <div >
    <div >
      <h1>Fruits</h1>
      <ul >
        {% for fruit in fruits %}
        <li >{{ fruit }}</li>
        {% endfor %}
      </ul>
    </div>

    <div >
      <h1>Selected Student</h1>
      <ol >
        {% for student in students %}
        <li >{{ student }}</li>
        {% endfor %}
      </ol>
    </div>
  </div>
</body>

</html>
```

Step 5: Include the fruitlist_app URLs in the project's URL patterns:-

◉ Open the file in your _Django project directory (project/urls._py).

◉ Import the view function at the top of the file.

◉ Add a new URL pattern to the urlpatterns list.

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

from fruitlist_app.views import fruitlist_app
```

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

Step 7 : Run Your Project

```
python manage.py runserver
```

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.

Step1: Create new folder:-

◉ In the same project folder whatever we made earlier create again one new app name called as mywebsite_app using below command.

```
python manage.py startapp mywebsite_app
```

Step2: Add the mywebsite_app to the installed_apps list:-

Step3: Inside views.py file create a function:-

◉ Open the views.py file in your Django project directory (mywebsite_app/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')
```

Step4: Create a template:-

◉ Right click on mywebsite_app folder, create a new folder named templates.

◉ Inside the templates folder, create a new file named layout.html.

◉ Inside the templates folder, create a new file named home.html.

◉ Inside the templates folder, create a new file named about.html.

◉ Inside the templates folder, create a new file named contact.html.

◉ Copy all the different different html file code and paste into all different html file to show the app.

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

<head>
  <title>Let's create with us LAYOUT.HTML</title>
  <style>
    * {
      padding: 0;
      margin: 0;
      box-sizing: border-box;
      font-family: sans-serif;
    }

    header {
      align-items: center;
      padding: 15px;
      display: flex;
      justify-content: space-between;
    }

    .logo a {
      font-size: 24px;
```

```
            color: blue;
            font-weight: 600;

            text-decoration: none;
        }

        .navbar-item {
            display: flex;
            gap: 40px;
            justify-content: space-between;
        }

        .navbar-item li a {
            font-weight: 600;
            font-size: 17px;
            color: black;
            text-decoration: none;
        }

        .navbar-item li a:hover{
            color:blue;
        }
        .navbar-item li {
            list-style: none;
        }

</style>
    </head>

    <body>
      <header>
```

```
        <div class="logo">
          <a href="{% url 'home' %}">vtucode</a>
        </div>

        <div class="navbar-item">
          <li><a href="{% url 'home' %}">Home</a></li>
          <li><a href="{% url 'about' %}">About</a></li>
          <li><a href="{% url 'contact' %}">Contact</a></li>
        </div>

      </header>


      <main>
        {% block content %}{% endblock %}
      </main>
    </body>

    </html>
```

HOME.html

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

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

    {% block content %}

    <section>
      <h2>This is homepage</h2>
    <p>lorem32<p>
    </section>
    {% endblock %}
```

About.html

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

{% block title %}About Us - My Website{% endblock %}

{% block content %}

<section>
  <h2>This is about us page</h2><br>
  <div>
    <br><p>Welcome to VTUCSE21, one source for all Engineering Notes.<br> We're dedicated to providing you the very best Engineering Notes, PPTs, Model Papers, and previous year question papers with an emphasis on engineering like CSE.

    Here we will provide you only interesting content, which you will like very much. We're dedicated to providing you the best of Educational, with a focus on dependability and VTU study materials. We hope you enjoy our Educational as much as we enjoy offering them to you.

    If you have any questions or comments, please don't hesitate to contact us. I will keep posting more important posts on my Website for all of you. Please give your support and love.</p>
  </div>
</section>
{% endblock %}
```

Contact.html

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

{% block title %}Contact Us - My Website{% endblock %}

{% block content %}
<section>
  <h2>This is contact us page</h2><br>
  <div class="container">
    <form action="#">
      <label for="name">Name</label>
      <input type="text" id="name" name="name" placeholder="Enter your name...">

      <label for="email">Email</label>
      <input type="text" id="email" name="email" placeholder="Enter your email...">

      <label for="subject">Message</label>
      <textarea id="message" name="message" placeholder="Enter your message"
style="height:200px"></textarea>

      <input type="submit" value="Submit">
    </form>
  </div>
</section>
{% endblock %}
```

Step 5: Include the mywebsite_app URLs in the project's URL patterns:-

◉ Open the file in your Django project directory (project/urls._py).

◉ Import the view function at the top of the file.

◉ Add new URL pattern to the urlpatterns list.

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

from mywebsite_app.views import home, about, contact

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', home, name='home'),
    path('about/', about, name='about'),
    path('contact/', contact, name='contact'),]
```

Step 7 : Run Your Project

```
python manage.py runserver
```

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.

Step 1: Create a New App

1. Open your terminal or command prompt.
2. Navigate to your Django project directory.
3. Create a new app named `registration` using the following command:

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

Step 2: Add `registration` to `INSTALLED_APPS`

1. Open the project settings file `school_project/settings.py`.
2. Add `'registration'` to the `INSTALLED_APPS` list:

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

Step 3: **Create Models**

1. Open `registration/models.py`.

2. Define the `Student`, `Course`, and `Enrollment` models:

```
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=200)
    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)
    enrollment_date = models.DateField(auto_now_add=True)

    def __str__(self):
        return f"{self.student} enrolled in {self.course}"
```

Step 4: **Create and Apply Migrations**

1. Run the following commands to create and apply migrations:

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

Step 5: Create Views

1. Open `registration/views.py`.

2. Create views for registering students and displaying the list of students registered for a course:

```
from django.shortcuts import render, redirect
from .models import Student, Course, Enrollment

def register_student(request):
    if request.method == 'POST':
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        email = request.POST['email']
        course_id = request.POST['course']
        student = Student.objects.create(first_name=first_name, last_name=last_name, email=email)
        course = Course.objects.get(id=course_id)
        Enrollment.objects.create(student=student, course=course)
        return redirect('student_list', course_id=course_id)
    else:
        courses = Course.objects.all()
        return render(request, 'registration/register_student.html', {'courses': courses})

def student_list(request, course_id):
    course = Course.objects.get(id=course_id)
    enrollments = Enrollment.objects.filter(course=course)
    students = [enrollment.student for enrollment in enrollments]
    return render(request, 'registration/student_list.html', {'course': course, 'students': students})
```

Step 6: Create Templates

1. Create a folder named `templates` inside the `registration` app directory.

2. Inside the `templates` folder, create another folder named `registration`.

3. Create two HTML files: `register_student.html` and `student_list.html`.

register_student.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register Student</title>
</head>
<body>
    <h1>Register Student</h1>
    <form method="post">
        {% csrf_token %}
        <label for="first_name">First Name:</label>
        <input type="text" id="first_name" name="first_name" required><br>
        <label for="last_name">Last Name:</label>
        <input type="text" id="last_name" name="last_name" required><br>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br>
        <label for="course">Course:</label>
        <select id="course" name="course" required>
            {% for course in courses %}
                <option value="{{ course.id }}">{{ course.name }}</option>
            {% endfor %}
        </select><br>
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

student_list.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Students List</title>
</head>
<body>
    <h1>Students Registered for {{ course.name }}</h1>
    <ul>
        {% for student in students %}
```

```
        <li>{{ student.first_name }} {{ student.last_name }} ({{ student.email }})</li>
    {% endfor %}
  </ul>
  <a href="{% url 'register_student' %}">Register Another Student</a>
</body>
</html>
```

Step 7: Include `registration` URLs in Project URLs

1. Open `school_project/urls.py`.
2. Import the view functions at the top of the file:

```
from registration.views import register_student, student_list
```

3. Add new URL patterns to the `urlpatterns` list:

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

urlpatterns = [
  path('admin/', admin.site.urls),
  path('registration/register/', register_student, name='register_student'),
  path('registration/students/<int:course_id>/', student_list, name='student_list'),
]
```

Step 8: Run Your Project

1. Start the Django development server:

```
python manage.py runserver
```

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

Step 1: Register Models with Admin Site

1. Open `registration/admin.py`.
2. Import the `Student` and `Course` models.
3. Register the models with the admin site:

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

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

Step 2: Create and Apply Migrations

1. Run the following commands to create and apply migrations:

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

Step 3: Create a Superuser

1. Create a superuser to access the Django admin interface:

```sh
python manage.py createsuperuser
```

2. Follow the prompts to enter the username, email, and password for the superuser.

Step 4: Access the Admin Interface

1. Start the Django development server if it's not already running:

```sh
python manage.py runserver
```

2. Open your web browser and navigate to http://localhost:8000/admin/.
3. Log in using the superuser credentials you created.

Step 5: Add Data through Admin Interface

1. In the admin interface, you should see `Students` and `Courses` listed under the app name `Registration`.
2. Click on `Add` next to `Students` to add a new student.
3. Fill out the form with the student's details and save.

4. Click on `Add` next to `Courses` to add a new course.
5. Fill out the form with the course details and save.
6. You can now view and manage students and courses through the admin interface.

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

Step 1: **Update Models**

1. Open `registration/models.py`.
2. Add a `Project` model and update the `Student` model to include a foreign key to the `Project` model:

```
from django.db import models
class Project(models.Model):
    topic = models.CharField(max_length=200)
    languages = models.CharField(max_length=200)
    duration = models.IntegerField(help_text="Duration in months")

    def __str__(self):
        return self.topic

class Student(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    project = models.ForeignKey(Project, on_delete=models.CASCADE, null=True, blank=True)

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

Step 2: **Create and Apply Migrations**

1. Run the following commands to create and apply migrations:

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

Step 3: **Create a Model Form**

1. Open `registration/forms.py` (create this file if it doesn't exist).
2. Create a model form for the `Student` model:

```
from django import forms
from .models import Student


class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = ['first_name', 'last_name', 'email', 'project']
```

Step 4: **Update Views**

1. Open `registration/views.py`.
2. Update the view for registering students to use the model form:

```
from django.shortcuts import render, redirect
from .models import Student, Course, Enrollment, Project
from .forms import StudentForm


def register_student(request):
    if request.method == 'POST':
        form = StudentForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('student_list')
    else:
        form = StudentForm()
    return render(request, 'registration/register_student.html', {'form': form})


def student_list(request):
    students = Student.objects.all()
    return render(request, 'registration/student_list.html', {'students': students})
```

Step 5: **Update Templates**

1. Update `registration/register_student.html` to use the form object:

register_student.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register Student</title>
</head>
<body>
    <h1>Register Student</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

Step 6: **Include Forms in `INSTALLED_APPS`**

1. Ensure that ``'django.forms'`` is included in the `INSTALLED_APPS` list in `school_project/settings.py`:

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

Step 7: **Run Your Project**

1. Start the Django development server:

```
python manage.py runserver
```

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.

Step 1: Update Models

1. Open `registration/models.py`.

2. Ensure you have the `Student` model defined:

```
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}"
```

Step 2: Create and Apply Migrations

1. Run the following commands to create and apply migrations:

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

Step 3: Create Views

1. Open `registration/views.py`.

2. Create generic class views for listing students and displaying student details:

```
from django.views.generic import ListView, DetailView
from .models import Student

class StudentListView(ListView):
    model = Student
    template_name = 'registration/student_list.html'
    context_object_name = 'students'

class StudentDetailView(DetailView):
```

```
    model = Student
    template_name = 'registration/student_detail.html'
    context_object_name = 'student'
```

Step 4: Update URLs

1. Open `registration/urls.py` (create this file if it doesn't exist).

2. Add URL patterns for the new views:

```
from django.urls import path
from .views import StudentListView, StudentDetailView

urlpatterns = [
    path('students/', StudentListView.as_view(), name='student_list'),
    path('students/<int:pk>/', StudentDetailView.as_view(), name='student_detail'),
]
```

Step 5: Update Templates

1. Create `registration/templates/registration/student_list.html` for listing students:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student List</title>
</head>
<body>
    <h1>Student List</h1>
    <ul>
        {% for student in students %}
            <li><a href="{% url 'student_detail' student.pk %}">{{ student.first_name }} {{
student.last_name }}</a></li>
        {% endfor %}
```

```
    </ul>
</body>
</html>
```

2. Create `registration/templates/registration/student_detail.html` for displaying student details:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Detail</title>
</head>
<body>
    <h1>{{ student.first_name }} {{ student.last_name }}</h1>
    <p>Email: {{ student.email }}</p>
    <a href="{% url 'student_list' %}">Back to Student List</a>
</body>
</html>
```

Step 6: Include the App in `INSTALLED_APPS`

1. Ensure that `'registration'` is included in the `INSTALLED_APPS` list in `school_project/settings.py`:

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

Step 7: Run Your Project

1. Start the Django development server:

```
python manage.py runserver
```

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

Step 1: Install Required Libraries

1. Ensure you have reportlab installed for PDF generation:

```
pip install reportlab
```

Step 2: Update Views

1. Open registration/views.py.
2. Add the necessary imports and define the views for CSV and PDF generation:

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

def student_list(request):
    students = Student.objects.all()
    return render(request, 'registration/student_list.html', {'students': students})

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

    writer = csv.writer(response)
    writer.writerow(['First Name', 'Last Name', 'Email'])

    for student in students:
        writer.writerow([student.first_name, student.last_name, student.email])

    return response
```

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

    p = canvas.Canvas(response)
    p.drawString(100, 800, "Student List")

    y = 750
    for student in students:
        p.drawString(100, y, f"{student.first_name} {student.last_name} ({student.email})")
        y -= 20

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

Step 3: Update URLs

1. Open registration/urls.py (create this file if it doesn't exist).
2. Add URL patterns for CSV and PDF generation:

```
from django.urls import path
from .views import student_list, download_csv, download_pdf

urlpatterns = [
    path('students/', student_list, name='student_list'),
    path('students/download/csv/', download_csv, name='download_csv'),
    path('students/download/pdf/', download_pdf, name='download_pdf'),
]
```

Step 4: Update Templates

1. Update registration/templates/registration/student_list.html to include links for downloading CSV and PDF files:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student List</title>
</head>
<body>
    <h1>Student List</h1>
    <ul>
        {% for student in students %}
            <li>{{ student.first_name }} {{ student.last_name }} ({{ student.email }})</li>
        {% endfor %}
    </ul>
    <a href="{% url 'download_csv' %}">Download CSV</a>
    <a href="{% url 'download_pdf' %}">Download PDF</a>
</body>
</html>
```

Step 5: Include the App in INSTALLED_APPS

1. Ensure that 'registration' is included in the INSTALLED_APPS list in school_project/settings.py:

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

Step 6: Run Your Project

1. Start the Django development server:

```
python manage.py runserver
```

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

Step 1: Update Models

1. Ensure you have the Student model defined in 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)

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

Step 2: Update Views

1. Open registration/views.py.
2. Add a view to handle AJAX registration:

```
from django.shortcuts import render
from django.http import JsonResponse
from .models import Student

def register_student(request):
    if request.method == 'POST' and request.is_ajax():
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')
        email = request.POST.get('email')

        if first_name and last_name and email:
            student = Student.objects.create(first_name=first_name, last_name=last_name,
email=email)
            return JsonResponse({'success': True}, status=200)
        else:
```

```
        errors = {}
        if not first_name:
            errors['first_name'] = ['This field is required.']
        if not last_name:
            errors['last_name'] = ['This field is required.']
        if not email:
            errors['email'] = ['This field is required.']
        return JsonResponse({'success': False, 'errors': errors}, status=400)
    return render(request, 'registration/register_student.html')
```

Step 3: Update URLs

1. Open registration/urls.py.
2. Add a URL pattern for the registration view:

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

urlpatterns = [
    path('register/', register_student, name='register_student'),
]
```

Step 4: Update Templates

1. Create or update registration/templates/registration/register_student.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register Student</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <h1>Register Student</h1>
    <form id="studentForm" method="post">
        {% csrf_token %}
```

```
    <p>
        <label for="first_name">First name:</label>
        <input type="text" id="first_name" name="first_name">
    </p>
    <p>
        <label for="last_name">Last name:</label>
        <input type="text" id="last_name" name="last_name">
    </p>
    <p>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email">
    </p>
    <button type="submit">Register</button>
</form>
<div id="message"></div>

<script>
    $(document).ready(function() {
        $('#studentForm').on('submit', function(event) {
            event.preventDefault();
            $.ajax({
                url: "{% url 'register_student' %}",
                type: "POST",
                data: $(this).serialize(),
                success: function(response) {
                    if (response.success) {
                        $('#message').html('<p>Student registered successfully!</p>');
                        $('#studentForm')[0].reset();
                    }
                },
                error: function(response) {
                    let errors = response.responseJSON.errors;
                    let errorMessage = '<ul>';
                    for (let field in errors) {
                        errors[field].forEach(function(error) {
                            errorMessage += '<li>' + error + '</li>';
```

```
                });
            }
            errorMessage += '</ul>';
            $('#message').html(errorMessage);
        }
    });
});
});
</script>
</body>
</html>
```

Step 5: Include the App in INSTALLED_APPS

1. Ensure that 'registration' is included in the INSTALLED_APPS list in school_project/settings.py:

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

Step 6: Run Your Project

1. Start the Django development server:

```
python manage.py runserver
```

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

Step 1: Update Models

Ensure you have the `Student` and `Course` models defined in `registration/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=100)
    last_name = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    courses = models.ManyToManyField(Course, related_name='students')

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

Step 2: Update Views

Open `registration/views.py`.

Add a view to handle AJAX search:

```
from django.shortcuts import render
from django.http import JsonResponse
from .models import Student

def search_student(request):
    query = request.GET.get('query', '')
    if query:
        students = Student.objects.filter(first_name__icontains=query) |
Student.objects.filter(last_name__icontains(query)
        data = []
        for student in students:
            courses = student.courses.all()
            courses_list = [{'name': course.name, 'description': course.description} for course in courses]
            data.append({'student': {'first_name': student.first_name, 'last_name': student.last_name, 'email':
student.email}, 'courses': courses_list})
        return JsonResponse({'students': data}, status=200)
```

```
    return render(request, 'registration/search_student.html')
```

Step 3: Update URLs

Open `registration/urls.py`.

Add a URL pattern for the search view:

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

urlpatterns = [
    path('search/', search_student, name='search_student'),
]
```

Step 4: Update Templates

Create or update `registration/templates/registration/search_student.html`:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Search Student</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <h1>Search Student</h1>
    <input type="text" id="searchQuery" placeholder="Enter student's name">
    <button id="searchButton">Search</button>
    <div id="results"></div>

    <script>
        $(document).ready(function() {
            $('#searchButton').on('click', function() {
                var query = $('#searchQuery').val();
```

```
            $.ajax({
                url: "{% url 'search_student' %}",
                type: "GET",
                data: { 'query': query },
                success: function(response) {
                    $('#results').empty();
                    if (response.students.length > 0) {
                        response.students.forEach(function(student) {
                            var studentInfo = '<h3>' + student.student.first_name + ' ' +
student.student.last_name + '</h3><p>' + student.student.email + '</p><ul>';
                            student.courses.forEach(function(course) {
                                studentInfo += '<li>' + course.name + ': ' + course.description + '</li>';
                            });
                            studentInfo += '</ul>';
                            $('#results').append(studentInfo);
                        });
                    } else {
                        $('#results').html('<p>No students found.</p>');
                    }
                },
                error: function() {
                    $('#results').html('<p>An error occurred.</p>');
                }
            });
        });
    });
    </script>
</body>
</html>
```

Step 5: Include the App in `INSTALLED_APPS`

Ensure that `'registration'` is included in the `INSTALLED_APPS` list in `school_project/settings.py`:

```
INSTALLED_APPS = [
    ...
```

```
    'registration',
]
```

Step 6: Run Your Project

Start the Django development server:

```
python manage.py runserver
```