Module-2

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

   Solution:

   To create a Django app that displays an unordered list of fruits and an ordered list of selected students for an event, follow these steps:

   Step 1a: Set Up Django Project and App. Then, create a new Django project and navigate into the project directory:

   django-admin startproject event_manager
   cd event_manager
   Next, create a new Django app within the project:
   python manage.py startapp events

   Step 1b: Register the App in Settings:
   Open the **settings.py** file inside the **event_manager** directory.
   Find the **INSTALLED_APPS** list and add your app's name (**'events'**) to the list.

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

   Step 1c: Update Settings: Make sure your Django project's settings (**settings.py**) include the **'DIRS'** setting pointing to the templates directory, and define the **year** and **developer** variables for the footer.
   #import necessary libraries

```
import os

#Add the following content to DIRS

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'events','templates')],
        ...
    },
]
```

Step 2:Define Models: Open the **events/models.py** file in your editor and define two models: **Fruit** and **Student**.

```
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)
    event = models.CharField(max_length=100)  # Assuming the event name is a string
    selected = models.BooleanField(default=False)

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

Step 3: Register Models in Admin: Open the events/admin.py file and register the models to make them accessible via the Django admin interface.

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

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

Step 4: Run Migrations: Apply the migrations to create the database tables for your models:

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

Step 5: Create Views and Templates: Create views and templates to display the lists of fruits and students.

In events/views.py, define the view functions:

```
from django.shortcuts import render
from .models import Fruit, Student

def fruit_list(request):
    fruit1=Fruit(name=”mango”)
    fruit1.save()
    fruit2=Fruit(name=”mango”)
    fruit2.save()
```

```
    fruits = Fruit.objects.all()
    return render(request, 'events/fruit_list.html', {'fruits': fruits})

def student_list(request):
    student1=Student(name=”Gaurav”,event=”Anchoring”,selected=True)
    student2=Student(name=”Kanti”,event=”Dance”,selected=True)
    student3=Student(name=”Shalini”,event=”Decoration”,selected=False)
    student1.save()
    student2.save()
    student3.save()
    students = Student.objects.filter(selected=True)
    return render(request, 'events/student_list.html', {'students': students})
```

Create templates in the **events/templates/events/** directory:

fruit_list.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Fruit List</title>
</head>
<body>
    <h1>Available Fruits:</h1>
    <ul>
        {% for fruit in fruits %}
            <li>{{ fruit.name }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

student_list.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Selected Students</title>
</head>
<body>
    <h1>Selected Students for the Event:</h1>
    <ol>
        {% for student in students %}
            <li>{{ student.name }}</li>
        {% endfor %}
```

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

Step 6: Define URLs: Create URL patterns in **events/urls.py** to map the views to URLs.

```
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'),
]
```

Step 7: Include URLs in Project: Include the app's URLs in the project's main **urls.py** file (**event_manager/urls.py**).

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

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

Step 8: Run the Development Server: Start the Django development server:

```
python manage.py runserver
```

Check output in **http://127.0.0.1:8000/events/fruits/** to see the list of fruits and **http://127.0.0.1:8000/events/students/** to see the list of selected students.

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.

Solution:

To create a layout template **layout.html** with a header containing a navigation menu and a footer with copyright and developer information, and then inherit this layout for creating additional pages, follow these steps:

Step 1a: Set Up Django Project and App. Then, create a new Django project and navigate into the project directory:

django-admin startproject event_manager
cd event_manager
Next, create a new Django app within the project:
python manage.py startapp events

Step 1b: Register the App in Settings:
Open the **settings.py** file inside the **event_manager** directory.
Find the **INSTALLED_APPS** list and add your app's name (**'events'**) to the list.

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

Step 1c: Update Settings: Make sure your Django project's settings (**settings.py**) include the **'DIRS'** setting pointing to the templates directory.
#import the necessary libraries as follows:

```
import os

#Add DIRS content as follows:

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        ...
    },
]
```

Step 2: Create layout.html Template: Create a new HTML file named **layout.html** in your Django app's templates directory (e.g., **events/templates/events/layout.html**). This will serve as the base layout for your website.

```
<!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>
   <!-- Add your CSS and JS links here -->
</head>
<body>
   <header>
      <nav>
         <ul>
            <li><a href="{% url 'home' %}">Home</a></li>
            <li><a href="{% url 'about_us' %}">About Us</a></li>
            <li><a href="{% url 'contact_us' %}">Contact Us</a></li>
         </ul>
      </nav>
   </header>

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

   <footer>
      <p>&copy; {{ year }} My Website. All rights reserved. Developed by {{
developer }}</p>
   </footer>
</body>
</html>
```

Step 3: **Create Additional Pages:** Now, create three additional HTML files that will inherit from **layout.html** and define the content for each page.

Home Page (home.html):

```
{% extends 'events/layout.html' %}
{% block title %}Home - My Website{% endblock %}

{% block content %}
```

```
<h1>Welcome to My Website</h1>
<!-- Add home page content here -->
{% endblock %}
```

About Us Page (about_us.html):

```
{% extends 'events/layout.html' %}
{% block title %}About Us - My Website{% endblock %}

{% block content %}
<h1>About Us</h1>
<!-- Add about us page content here -->
{% endblock %}
```

Contact Us Page (contact_us.html):

```
{% extends 'events/layout.html' %}
{% block title %}Contact Us - My Website{% endblock %}

{% block content %}
<h1>Contact Us</h1>
<!-- Add contact us page content here -->
{% endblock %}
```

Step 4: Define URL Patterns: Define URL patterns in your Django app's **urls.py** file (**events/urls.py**) to map these pages to specific URLs.

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

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

Step 5: Create View Functions: Define view functions in your Django app's **views.py** file (**events/views.py**) to render the respective templates for each page.

```
from django.shortcuts import render
developer=”Gaurav”
year=2024
def home(request):
    return render(request, 'events/home.html',{‘developer’:developer,’year’:year})

def about_us(request):
```

```
    return render(request, 'events/about_us.html',{‘developer’:developer,’year’:year})

def contact_us(request):
    return render(request, 'events/contact_us.html',{‘developer’:developer,’year’:year})
```

Step 6: Include URLs in Project: Include the app's URLs in the project's main **urls.py** file (**event_manager/urls.py**).

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('events.urls')),  # Assuming 'events' is your app name
]
```

With these steps, you have created a layout template **layout.html** with a header, footer, and placeholders for content. You can now access the Home, About Us, and Contact Us pages of your 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.

   Solution:

   To create a Django app for student registration to a course and display a list of students registered for a selected course, you can follow these steps:

   Step 1a: Then, create a new Django project:

   django-admin startproject course_registration

   Step 2a: Create the models: Create models for **Student** and **Course** in a new Django app called **registration**:

   cd course_registration
   python manage.py startapp registration

   Step 2b: Register the App in Settings:
   Open the **settings.py** file inside the **course_registration** directory.
   Find the **INSTALLED_APPS** list and add your app's name (**'registration'**) to the list.

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

   Step 2c: Update Settings: Make sure your Django project's settings (**settings.py**) include the **'DIRS'** setting pointing to the templates directory.
   #import the necessary libraries as follows:

```
import os

#Add DIRS content as

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'registration','templates')],
        ...
    },
]
```

Step 3:
In the **registration/models.py** file, define the models as follows:

```
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):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    courses = models.ManyToManyField(Course, related_name='students')

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

Step 4:
Set up the admin interface: Register the models in the **registration/admin.py** file to manage them via the Django admin interface:

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

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

Step 5:

Run migrations: Apply the migrations to create the database tables for the models:

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

Step 6:

Create views and templates: Create views and templates for student registration and course listing. In **registration/views.py**, create view functions for registration and listing:

```
from django.shortcuts import render
from .models import *

def homefun(request):
```

```
    courses = Course.objects.all()
    studentDetails=Student.objects.all()
    return render(request, 'course_list.html' ,{'courses':
courses,'students':studentDetails})

def formfun(request):
    if request.method == 'POST':
        name = request.POST.get('sname')
        email = request.POST.get('semail')
        course_name = request.POST.get('scourse')

        course, created_course = Course.objects.get_or_create(name=course_name)

        student, created_student = Student.objects.get_or_create(name=name,
email=email)

        if course not in student.courses.all():
            student.courses.add(course)
            created = True
        else:
            created = False

        if created:
            message = f'{student.name} registered successfully for {course.name}.'
        else:
            message = f'{student.name} is already registered for {course.name}.'

        return render(request, 'registration_confirmation.html', {'message': message})

    return render(request,'student_registration.html')

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

Create corresponding HTML templates in **registration/templates** folder:
course_list.html to display the list of courses.
student_registration.html for student registration form.
registration_confirmation.html for displaying registration confirmation message.

#Course_list.html- Content

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

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home Page</title>
</head>
<body>
    <h1>Home Page</h1>
    <h1>Click to go to <a href="{% url 'formfunvar' %}">registration page</a></h1>

    <h1>Displaying Courses details</h1>
    <ul>
        {% for course in courses %}
            <li>{{course.name}}</li>
        {% endfor %}
    </ul>

    <h1>Displaying Student details</h1>
    <ul>
        {% for student in students %}
            <li>{{ student.name }} -- {{ student.email }}
                <ul>
                    {% for course in student.courses.all %}
                        <li>{{ course.name }}</li>
                    {% endfor %}
                </ul>
            </li>
        {% endfor %}
    </ul>
</body>
</html>
```

 # student_registration.html- Content

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Page</title>
</head>
<body>
    <h1>Form Page</h1>
    <h1>ENTER the Details</h1>
    <form method="post">
        {% csrf_token %}
        <input type="text" name="sname" placeholder="Student Name">
```

```
        <input type="email" name="semail" placeholder="Enter Email">
        <input type="text" name="scourse" placeholder="Enter Course">
        <input type="submit" placeholder="SUBMIT">
    </form>
</body>
</html>
```

 # registration_confirmation.html – Content

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Thankyou Page</title>
</head>
<body>
    <h1>Thankyou Page</h1>
    <h1>{{message}}</h1>
    <h1><a href="{% url 'homefunvar' %}">GO back to home</a></h1>
</body>
</html>
```

Step 7:

Set up URLs: Configure URLs to route requests to the views. In **registration/urls.py**, define URL patterns:

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

urlpatterns = [
    path('',homefun,name='homefunvar'),
    path('registration/',formfun,name='formfunvar'),
    path('thankyou/',thankfun,name='thankfunvar'),
]
```

Include these URLs in the main project's urls.py file:

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

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

Step 8: Run the development server: Start the Django development server to test the app:

python manage.py runserver