1. **Introduction to Django and Setting Up the Environment**

**Learning Outcomes**: Students will understand Django's background, its advantages, and complete project setup using best practices.

**Topics & Add-ons:**

* **History of Django & Its Ecosystem**
* Real-world usage: Instagram, Pinterest, Disqus
* **What is Django & Why Django?**
* Comparisons with Laravel
* **MVT Architecture**
* **Setup Environment**
  + Use VS Code with extensions (Python, Django)
  + Setup using venv, pip, .env and so on
  + Introduce django-admin and manage.py commands
  + Explain folder structure and responsibilities

**Activities:**

* Quick install and setup
* Create and run first Django project
* Live demo with basic homepage view

1. **Django templates and views with URLS**

**Learning Outcomes**: Learn Django templates and views interactions, how URLs can be used to render templates along with views, able to make static template pages along static files.

**Topics & Add-ons:**

* Create app and connect it with project url.py
* Configuring and implement templates
* Understands DTL
* Use of static files in templates
* Use of Bootstrap CDN
* Template inheritance
* Function vs class-based views
* Template rendering with params

**Activities:**

* Live demo of base file (template inheritance), make layout -> base.html> home.html->details.html
* Templates to design like (login, signup, home, etc…)
* Render templates based on views & demonstrate the use of file urls.py
* Navbar using template inheritance
* Pass data to homepage/navbar

1. **Django Models and Database**

**Learning Outcomes**: Learn Django ORM, how models interact with DB, and admin management.

**Topics & Add-ons:**

* Create basic models (Student, Course, etc.)
* Migrations and their purpose
* Intro and setup MySQL
* Database setup and configuration in settings.py
* Implement models and migrations to database
* Understanding Django ORM to interact with database
* Setup Admin panel/super user, add custom display methods into admin interface

**Activities:**

* Create 2–3 related models with ForeignKey
* Customize admin interface

**Supplement: After Creating Project and App — Important Configuration Steps**

**Learning Outcome**: Understanding project–app connection, template configuration, static/media handling, and rendering logic.

**✅ Steps After startproject and startapp**

1. **Create urls.py in the app folder**

# myapp/urls.py

from django.urls import path

from . import views

urlpatterns = [

path('', views.home, name='home'),

]

1. **Connect app URLs to the project-level urls.py**

# myproject/urls.py

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path('', include('myapp.urls')),

]

1. **Template Configuration in settings.py**

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [BASE\_DIR / 'templates'],

...

},

]

1. **Create templates/ folder**

project/

├── templates/

│ └── base.html

│ └── home.html

1. **Template Inheritance**

<!-- base.html -->

<html>

<body>

{% block content %}{% endblock %}

</body>

</html>

<!-- home.html -->

{% extends 'base.html' %}

{% block content %}

<h1>Welcome Home</h1>

{% endblock %}

1. **Static Files Configuration**

# settings.py

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [ BASE\_DIR / 'static' ]

Folder structure:

project/

├── static/

│ └── css/

│ └── style.css

Usage in template:

{% load static %}

<link rel="stylesheet" href="{% static 'css/style.css' %}">

1. **Media Files Configuration**

# settings.py

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / 'media'

# urls.py (project)

from django.conf import settings

from django.conf.urls.static import static

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

For file upload:

# models.py

image = models.ImageField(upload\_to='uploads/')

Use in template:

<img src="{{ object.image.url }}">

1. **Forms, Authentication and CRUD**

**Learning Outcomes**: Implement secure, dynamic form handling and user auth.

**Topics & Add-ons:**

* ModelForm vs Form
* Validating form data (clean\_ methods)
* Create login, register, logout views
* Session usage & flash messages
* Password hashing, authentication backends
* Custom user model (extend AbstractUser)
* CRUD operations (Add/Edit/Delete students/courses)
* SweetAlert2 for UI messages (cdn + JS + message tags)

**Activities:**

* Build Register/Login system with auth
* Form error handling and success message
* Create/update/delete records with feedback

1. **Advanced Concepts**

**Learning Outcomes**: Handle real-life project requirements using advanced features.

**Topics & Add-ons:**

* Edit profile feature
* Role-based access using decorators and middleware
* Send OTP using Email backend (SMTP)
* Upload profile images, timestamp save
* Use pandas and openpyxl for Excel/CSV handling
* Generate downloadable PDFs (with xhtml2pdf or WeasyPrint)

**Activities:**

* Profile page with image upload
* Admin vs Student dashboard using roles
* Password reset via email simulation
* Import/export Excel and generate PDF reports