• Steps to create a Django project and individual apps within the project.

1. Setup Virtual Environment

1) pip install virtualenv -> install

2) python -m venv "myenv" -> create

3) cd myenv

4) Scripts\activate

1. Install and create a Django Project

5) pip install django

6) django-admin startproject "myproject"

7) cd myproject

1. Create apps in the project

8) django-admin startapp "myapp"

9) python manage.py migrate

• Understanding the role of manage.py, urls.py, and views.py.

**manage.py**

* **Purpose**: manage.py is a command-line utility that lets you interact with your Django project.
* **Role**: It's a script that provides various commands for managing the Django project. It helps with starting the development server, creating applications, running tests, migrating the database, and more.
* **Common Commands**:
  + python manage.py runserver: Starts the development server.
  + python manage.py startapp <appname>: Creates a new app.
  + python manage.py migrate: Applies database migrations.
  + python manage.py createsuperuser: Creates an admin user.

**urls.py**

* **Purpose**: urls.py is responsible for routing URLs to the appropriate view functions or classes.
* **Role**: Defines URL patterns and maps them to views. This file acts as the "directory" for your web application, determining what users see based on the URL they visit.
* **Structure**: Typically contains a list of urlpatterns that map URLs to views.
* python

# Example urls.py

from django.urls import path

from . import views

urlpatterns = [

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

path('about/', views.about, name='about'),

]

**views.py**

* **Purpose**: views.py contains the logic behind what data is presented to the user and how it is displayed.
* **Role**: Views process user requests, interact with the models to retrieve data, and render templates to create a user response. Essentially, views are the bridge between the models and templates.
* **Types of Views**:
  + **Function-Based Views (FBVs)**: Traditional views implemented as functions.
  + **Class-Based Views (CBVs)**: Views implemented as classes to promote code reuse and organization.
* python

# Example views.py with Function-Based Views

from django.shortcuts import render

def home(request):

return render(request, 'home.html')

def about(request):

return render(request, 'about.html')