INSTITUTE OF TECHNOLOGY,

NIRMA UNIVERSITY

2CSDE86 Application Development Framework

PRACTICAL 1

Name : Makwana Sagarkumar

Roll No. : 20BCE512

Semester : 7th

Teacher's Signature :

Aim :

1. Introduction to MVC and MVT architecture. Comparative Study on MVC and MVT by considering various parameters for comparison.
2. Explore the Django application structure.

1. MVC and MVT architecture

Model View Controller (MVC) :

The emphasis is placed on separating data representation from the components that interact with and process the data in this software design pattern, which is used to develop user interfaces.

Each of its three parts has a certain role.

1. The Model, which controls the application's data, logic, and other constraints, is the key element of the design.
2. The View contains numerous data representation components and deals with how the data will be presented to the user.
3. The Controller serves as a link between the Model and the View, manipulating it and rendering the view.

Model View Template (MVT) :

Another design pattern that resembles MVC is this one. It is also used to build web interfaces and applications, but unlike MVC, the framework itself handles the controller portion for us.

Each of its three parts has a certain role.

1. This model, which is essentially the logical framework for the entire web application and is similar to MVC, functions as an interface for your data and is represented by a database like MySql or PostgreSQL.
2. The View interacts with the Model, carries out the business logic, and renders the template. It receives HTTP requests and then replies using HTTP.

3. The element that distinguishes MVT from MVC is the template. In essence, templates are the HTML code that renders the data and serve as the presentation layer. These files' content can either be static or dynamic.

Difference between MVC and MVT design patterns :

Model View Controller (MVC)	Model View Template (MVT)
MVC has controller that drives both Model and View.	MVT has Views for receiving HTTP request and returning HTTP response.
View tells how the user data will be presented.	Templates are used in MVT for that purpose.
In MVC, we have to write all the control specific code.	Controller part is managed by the framework itself.
Highly coupled	Loosely coupled
Modifications are difficult	Modifications are easy
Suitable for development of large applications but not for small applications.	Suitable both small and large applications.
Flow is clearly defined thus easy to understand.	Flow is sometimes harder to understand as compared to MVC.
It doesn't involve mapping of URLs.	URL pattern mapping takes place.
Examples are ASP.NET MVC, Spring MVC etc.	Django uses MVT pattern

2. Django application structure.

A project is the fundamental unit of your Django web application.

To create a Django project you use the following command:

```
$ django-admin startproject [project_name]
```

manage.py:

The `manage.py` file provides a command-line utility for a Django project. You will use this command-line utility to perform various operations related to debugging, running, and deploying a Django web application. For example, to run a Django application in the development server you will use the following command:

```
$ python manage.py runserver
```

The files are present in project:

__init__.py

It is an empty Python file. The `__init__.py` file tells the Python interpreter that the directory `project` is a Python package.

asgi.py & wsgi.py

`asgi` stands for **Asynchronous Server Gateway Interface** and `wsgi` stands for **Web Server Gateway Interface.**

After your development process is completed, you will move to production and hosting. For hosting you will use `asgi` or `wsgi` compatible servers. According to the type of server you use, you have to import middleware accordingly.

`asgi.py` enables ASGI compatible servers and `wsgi.py` enables WSGI compatible servers to serve your Django web app.

settings.py

This is the main configuration file for a Django project. This is the main settings file and here you will configure all the apps and middleware for your project.

This file also handles the database settings. By default Django uses sqlite3. But if you use a different database, which you will most probably do, you will configure it in `settings.py`.

`settings.py` also handles templates, static files, and media files related settings.

urls.py

URLs are different endpoints of your website. `urls.py` contains the URL configurations for your website. By default, `urls.py` comes with the URL pattern for the admin panel. You will create other endpoints for your web app in this file.

Creating Apps

An app in a Django project is a Python package that does a particular job. A Django project contains one or more apps and each of them handles a particular task. For example, a Django blog website will have a list of posts, user authentication, user profiles, etc. The best practice is to create different apps for each one of them.

To create an app you should go to the root directory of your project where the `manage.py` file is. Then you need to run the following command:

```
$ python manage.py startapp [app name]
```

__init__.py

The `__init__.py` file in an app is no different than the `__init__.py` file in a project. This empty Python file is telling the interpreter that `app` is a Python package.

admin.py

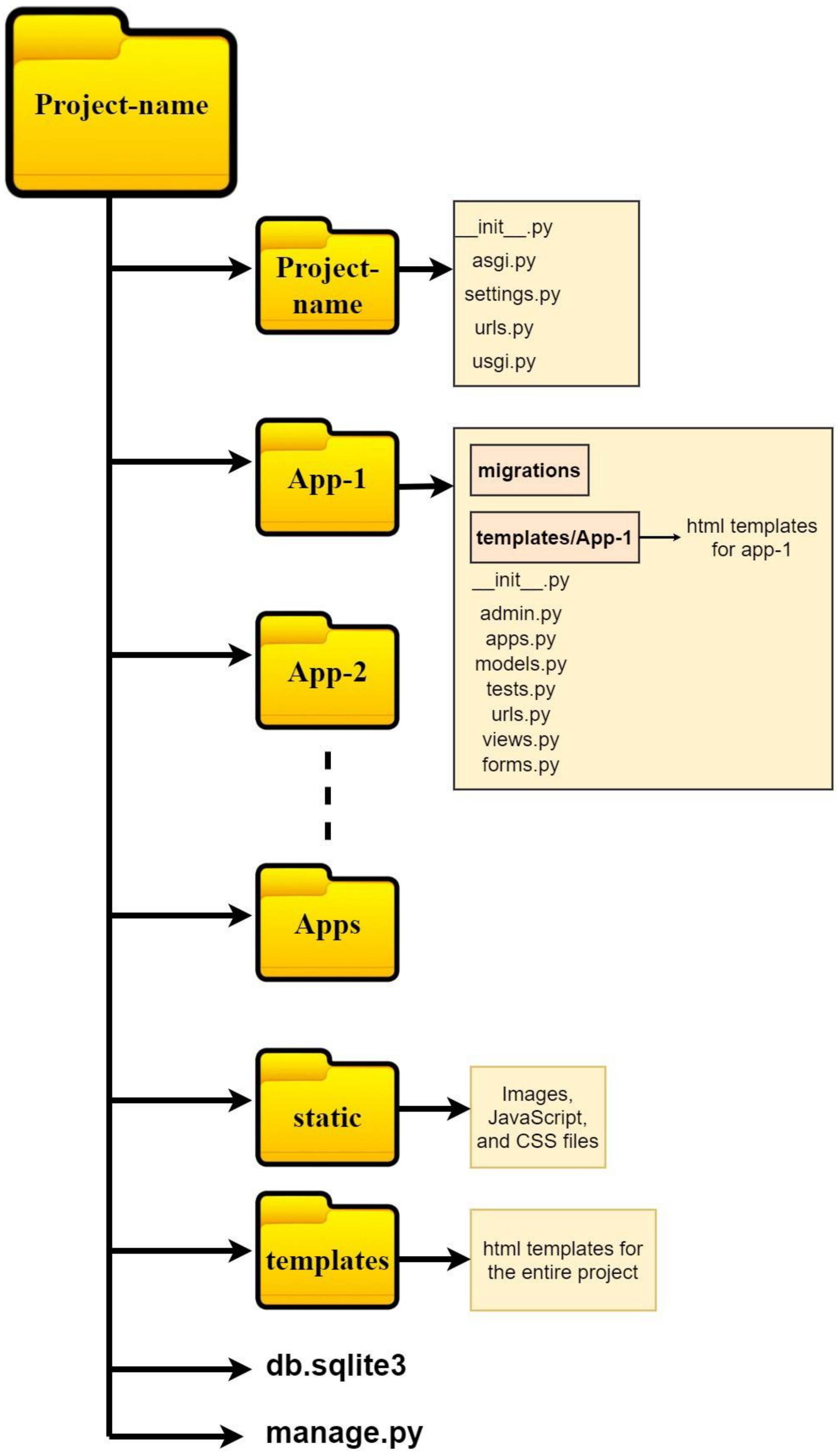
This file is used to register the models in your app in the Django administration. You will use this file to display the models of your app in the admin panel.

apps.py

It is a common configuration file for all Django apps. You can configure the attributes for your app using this file. However, the default configuration is sufficient for most cases. So, adding app configuration is a rare case.

views.py

In this file, you write the business logic for your app. A view can be either function-based or class-based. You decide if you want to write your views using functions or classes.


Project-name
Project-name
__init__.py
asgi.py
settings.py
urls.py
usgi.py
App-1
migrations
templates/App-1
html templates for app-1
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
forms.py
App-2
Apps
static
Images, JavaScript, and CSS files
templates
html templates for the entire project
db.sqlite3
manage.py