

1. What is Django?

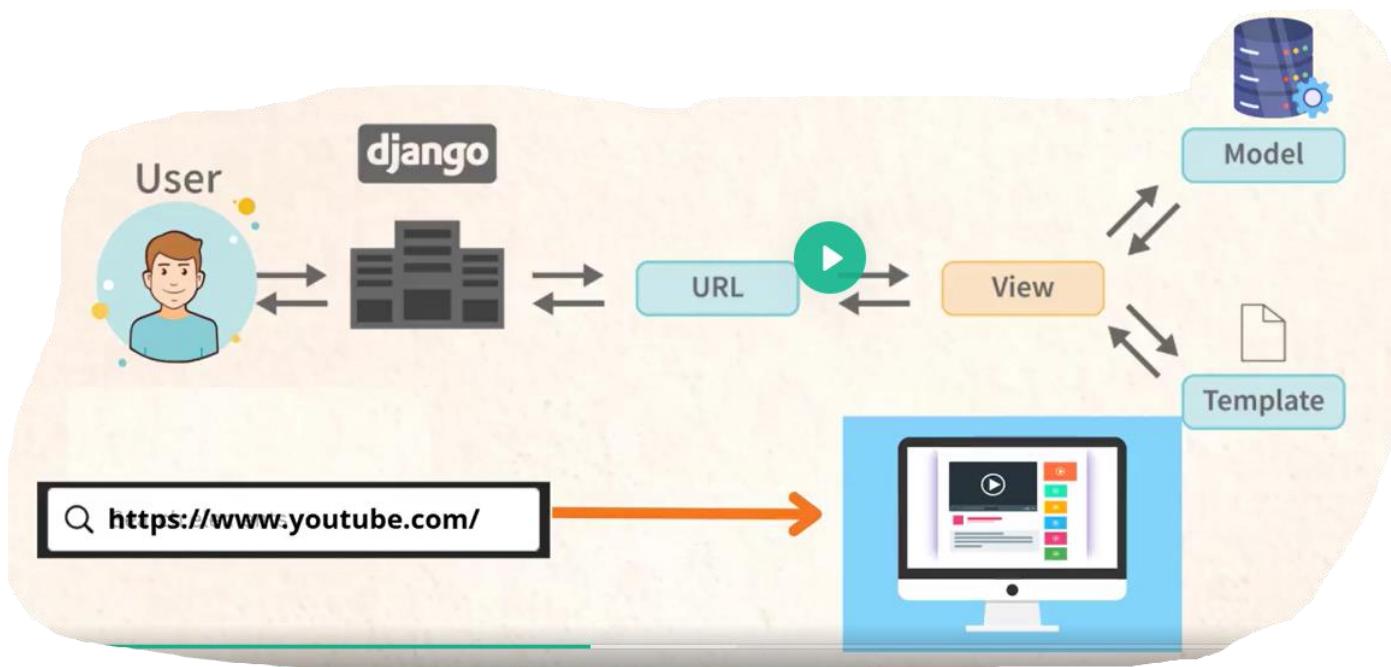
DJANGO IS A WEB APPLICATION FRAMEWORK WRITTEN IN PYTHON PROGRAMMING LANGUAGE. IT IS BASED ON MVT (MODEL VIEW TEMPLATE) DESIGN PATTERN. IT IS ALSO CALLED BATTERIES INCLUDED FRAMEWORK BECAUSE DJANGO PROVIDES BUILT-IN FEATURES FOR EVERYTHING.

2. Why is Django?

- EXCELLENT DOCUMENTATION AND HIGH SCALABILITY.
- USED BY TOP MNCs AND COMPANIES, SUCH AS INSTAGRAM, SPOTIFY, YOUTUBE, BITBUCKET, DROPBOX, ETC. AND THE LIST IS NEVER-ENDING.
- EASIEST FRAMEWORK TO LEARN, RAPID DEVELOPMENT AND BATTERIES FULLY INCLUDED.
- ONE CAN INTEGRATE IT WEB SCRAPING, MACHINE LEARNING, IMAGE PROCESSING, SCIENTIFIC COMPUTING, ETC WITH WEB APPLICATION AND DO LOTS AND LOTS OF ADVANCE STUFF.

3. How Does Django Work?

THE MVT IS AN DESIGN PATTERN THAT SEPARATES AN APPLICATION INTO THREE MAIN LOGICAL COMPONENTS MODEL, VIEW, AND TEMPLATE.



4. What is MVT?

MVT = MODEL – VIEW – TEMPLATE

- MODEL → WORKS WITH DATABASE (STORES & MANAGES DATA)
- VIEW → WORKS WITH LOGIC (GETS DATA FROM MODEL, SENDS TO TEMPLATE)
- TEMPLATE → WORKS WITH DESIGN (HTML/CSS SHOWN TO USER)

FLOW:

USER → VIEW → MODEL → VIEW → TEMPLATE → USE

5. UNIQUE FEATURES OF DJANGO

- ADMIN INTERFACE
- OBJECT-RELATIONAL MAPPING (ORM)
- URL ROUTING
- TEMPLATE SYSTEM
- FORM HANDLING
- SECURITY FEATURES
- SCALABILITY

6. WHAT IS VIRTUAL ENVIRONMENT?

A VIRTUAL ENVIRONMENT IS LIKE A SEPARATE SANDBOX (SMALL ROOM) FOR PYTHON.
INSIDE IT, YOU CAN INSTALL THE LIBRARIES (PACKAGES) YOU NEED ONLY FOR ONE PROJECT
WITHOUT AFFECTING OTHER PROJECTS OR YOUR WHOLE SYSTEM.

7. WHY VIRTUAL ENVIRONMENT?

EXAMPLE:

- PROJECT A NEEDS DJANGO 4.0
- PROJECT B NEEDS DJANGO 3.2

☞ IF YOU INSTALL EVERYTHING GLOBALLY, BOTH VERSIONS WILL CLASH.

WITH VIRTUAL ENVIRONMENTS, YOU CAN KEEP DIFFERENT VERSIONS FOR DIFFERENT PROJECTS.

8. SETUP

How does a virtual environment work?

Step-1 : Installing virtualenv :

\$ pip install virtualenv

Step-2 : Test your installation:

\$ virtualenv --version

Step-3 : Naming Virtual Env:

\$ virtualenv my_env

Step-4 : Activating Virtual Env:

\$ source ./my_env/Scripts/activate

Step-5 : Deactivating Virtual Env:

\$ deactivate



9. WORKING FOR DJANGO SOME TIPS:

1. PYHTON VERSION CHECK:

| \$ python --version

2. DJANGO VERSION CHECK:

| \$python -m django --version

3. INSTALL DJANGO:

```
$ pip install django
```

10. MAKE DJANGO PROJECT

1. FIRST MAKE STARTPROJECT:

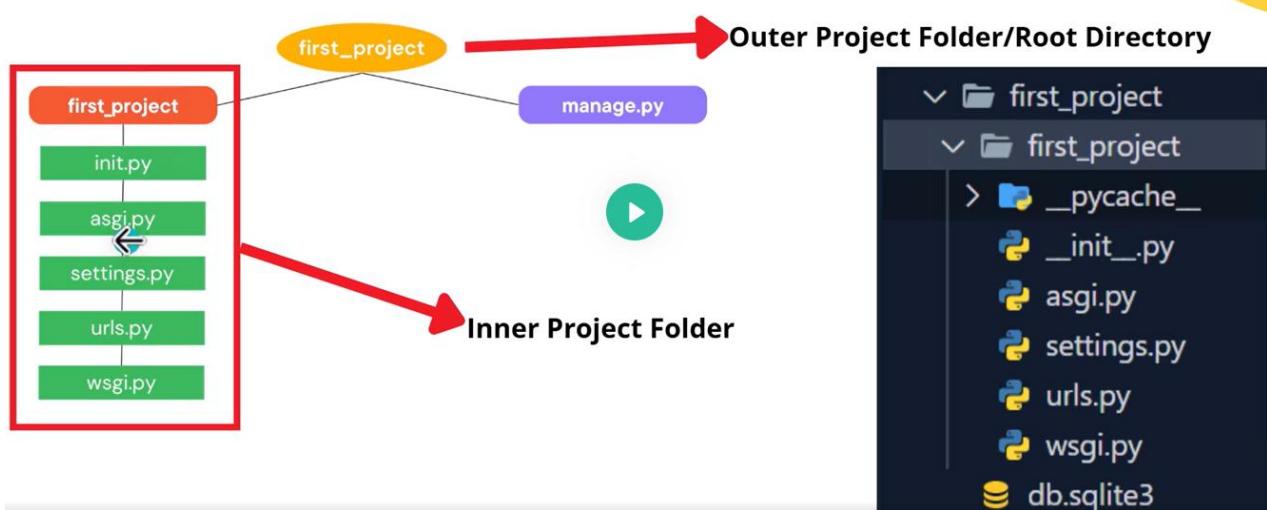
```
$ django-admin startproject first_project[your project name]
```

2. RUN DJANGO SERVER:

Go to the location where `manage.py` is located, then run:

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

11. DJANGO PROJECT FOLDER STRUCTURE



__init__.py :

THE FOLDER WHICH CONTAINS `__init__.py` FILE IS CONSIDERED AS PYTHON PACKAGE.

wsgi.py :

WSGI (WEB SERVER GATEWAY INTERFACE) IS A SPECIFICATION THAT DESCRIBES HOW A WEB SERVER COMMUNICATES WITH WEB APPLICATIONS, SYNCHRONOUS.

asgi.py :

ASYNCHRONOUS SERVER GATEWAY INTERFACE. ASGI PROVIDES STANDARD FOR BOTH ASYNCHRONOUS AND SYNCHRONOUS

settings.py :

THIS FILE CONTAINS ALL THE INFORMATION OR DATA ABOUT PROJECT SETTINGS. E.G.: - DATABASE CONFIG INFORMATION, TEMPLATE, INSTALLED APPLICATION, VALIDATORS ETC.

urls.py :

CONTAINS INFORMATION OF URL ATTACHED WITH APPLICATION.

manage.py

MANAGE.PY IS A PROJECT-SPECIFIC COMMAND-LINE UTILITY

DETAILS FOR: <https://chatgpt.com/share/689f1b87-df44-8010-a5b9-60cfa2c21967>

DJANGO PROJECT IMPORTANT FILES (NOTES)

◊ 1. __INIT__.PY

- ফোল্ডারকে PYTHON PACKAGE হিসেবে চিহ্নিত করে।
- ফলে ওই ফোল্ডারের কোড IMPORT করা যায়।
- নতুন PYTHON ভাস্বনে না থাকলেও চলে, তবে DJANGO ডিফল্টভাবে যোগ করে।

◊ 2. WSGI.PY

- FULL FORM: WEB SERVER GATEWAY INTERFACE
- কাজ: ওয়েব সার্ভার আর DJANGO এর মধ্যে যোগাযোগ করায়।
- প্রকৃতি: SYNCHRONOUS (এক কাজ শেষ হলে আরেকটা শুরু হয়)।
- ব্যবহার: প্রজেক্টকে PRODUCTION SERVER এ DEPLOY করার সময় লাগে।

◊ 3. ASGI.PY

- FULL FORM: ASYNCHRONOUS SERVER GATEWAY INTERFACE
- কাজ: DJANGO কে ASYNCHRONOUS + SYNCHRONOUS দুইভাবে চালাতে পারে।
- ব্যবহার:
 - REAL-TIME CHAT
 - LIVE NOTIFICATION
 - WEB SOCKET CONNECTION
- DJANGO 3.0 থেকে ব্যবহার শুরু হয়েছে।

◊ 4. SETTINGS.PY

- প্রজেক্টের সব কনফিগারেশন এখানে থাকে।
- শুরুত্বপূর্ণ বিষয়গুলো:
 - DATABASE SETTINGS
 - INSTALLED APPS
 - MIDDLEWARE
 - TEMPLATES CONFIG
 - STATIC FILES (CSS, JS, IMAGES)
 - SECURITY SETTINGS (PASSWORD VALIDATORS, ALLOWED HOSTS)
- কে এক কথায় পুরো প্রজেক্টের REMOTE CONTROLLER।

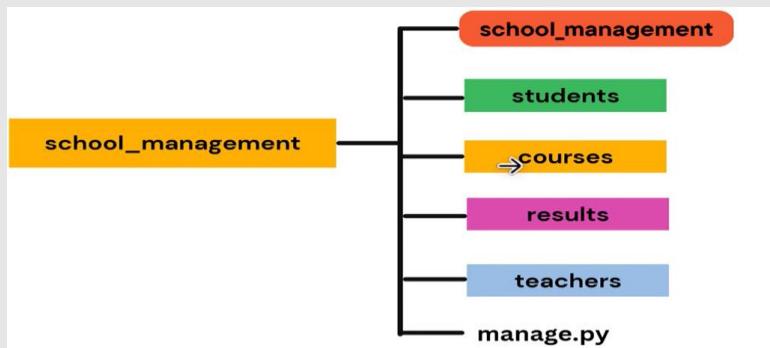
◊ 5. URLs.PY

- ওয়েবসাইটের URL রাউটিং সিস্টেম।
- কোন URL এ গেলে কোন VIEW ফাংশন চালু হবে সেটার লিস্ট।
- উদাহরণ:
- URLPATTERNS = [
 - PATH("HOME/", VIEWS.HOME_PAGE),
 - PATH("ABOUT/", VIEWS.ABOUT_PAGE),
 -]

◊ 6. MANAGE.PY

- DJANGO প্রজেক্টের COMMAND-LINE TOOL।
- সাধারণ কমান্ড:
 - python manage.py runserver → সার্ভার চালানো
 - python manage.py startapp appname → নতুন অ্যাপ তৈরি

- python manage.py makemigrations → ডাটাবেস পরিবর্তন ট্র্যাক
- python manage.py migrate → ডাটাবেস আপডেট
- python manage.py createsuperuser → অ্যাডমিন ইউজার তৈরি



শৈক্ষণিক উদাহরণ (SCHOOL ANALOGY)

- `__INIT__.PY` → ক্লাস রেজিস্টার (এই ক্লাস/ফোল্ডারটাকে একটা দল ধরা হবে)
- `WSGI.PY` → হেডস্যার (শৃঙ্খলাভাবে কাজ করান)
- `ASGI.PY` → ভাইস-প্রিলিপাল (একসাথে অনেক কাজ হ্যান্ডেল করতে পারেন)
- `SETTINGS.PY` → স্কুলের নিয়ম-কানুন (সব সেটিংস এখানে)
- `URLS.PY` → টাইম টেবিল (কোন সময়ে কোন ক্লাস হবে)
- `MANAGE.PY` → ক্লাস মনিটর (কমান্ড দিয়ে সব চালায়)

12. COMMAND START PROJECT:

`django-admin startproject first_project[your project name]`

`python manage.py runserver` → সার্ভার চালানো

`django-admin startapp appname [your app name]/python manage.py startapp appname` →
নতুন অ্যাপ তৈরি

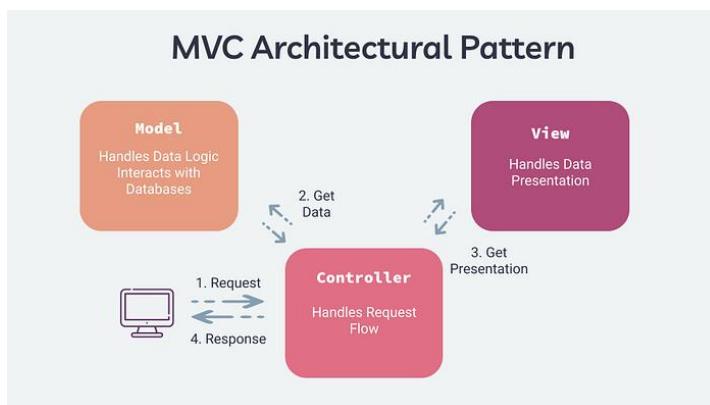
`python manage.py makemigrations` → ডাটাবেস পরিবর্তন ট্র্যাক

`python manage.py migrate` → ডাটাবেস আপডেট

`python manage.py createsuperuser` → অ্যাডমিন ইউজার তৈরি

MVC VS MVT

Mvc = (MODEL-VIEW-CONTROLLER)



MVC (MODEL – VIEW – CONTROLLER) সহজভাবে

- MVC কী?

 MVC হলো একটি DESIGN PATTERN, যেখানে ওয়েব অ্যাপকে ৩ ভাগে ভাগ করা হয় –

- MODEL
- VIEW
- CONTROLLER

শৈলী প্রতিটি অংশের কাজ

1. MODEL

- DATA বা BUSINESS LOGIC রাখে।
- DATABASE থেকে ডেটা আনে বা সংরক্ষণ করে।
- যখন ডেটা পরিবর্তন হয় তখন VIEW কে জানায়।

2. VIEW

- USER যা দেখে (UI) → যেমন HTML, টেক্স্ট, বাটন, ফর্ম ইত্যাদি।
- MODEL থেকে পাওয়া ডেটা ইউজারকে দেখায়।

3. CONTROLLER

- MODEL আর VIEW এর মধ্যে সেতু।
- USER INPUT (যেমন বাটন ক্লিক, ফর্ম সাবমিট) নেয়।
- MODEL আপডেট করে।
- তারপর VIEW কে আপডেট করতে বলে।

◦ MVC FLOW (STEP BY STEP)

1. USER ACTION: ইউজার VIEW এর সাথে কাজ করে (বাটন চাপা/ফর্ম সাবমিট)।
2. CONTROLLER: INPUT নেয়, প্রসেস করে, MODEL কে আপডেট করে।
3. MODEL: DATA পরিবর্তন করে বা নতুন DATA আনে।
4. VIEW: MODEL এর নতুন DATA দেখায়।

◦ MVC EXAMPLE

- MODEL: USER ক্লাস (NAME, EMAIL ফিল্ড সহ DATABASE TABLE)।
- VIEW: HTML ফাইল যেখানে USER এর ডেটা দেখানো হয়।
- CONTROLLER: PYTHON/JAVA ফাংশন → ফর্ম সাবমিশন প্রসেস করে → MODEL আপডেট করে → VIEW কে রিটার্ন করে।

◦ MVC এর সুবিধা

- কোড পরিষ্কারভাবে আলাদা আলাদা ভাগে থাকে (SEPARATION OF CONCERNS)
- কোড পুনঃব্যবহার সহজ
- একই MODEL এর জন্য একাধিক VIEW বানানো যায়

◦ MVC ব্যবহার করে এমন জনপ্রিয় FRAMEWORK

- RUBY ON RAILS
- ASP.NET MVC
- ANGULAR (CLIENT-SIDE)

ক্রে এক লাইনে মনে রাখো:

MODEL → DATA, VIEW → UI, CONTROLLER → LOGIC (BRIDGE BETWEEN MODEL & VIEW).

MVT=(MODEL-VIEW-TEMPLATE)

■ MVT (MODEL – VIEW – TEMPLATE) সহজভাবে

👉 DJANGO (PYTHON WEB FRAMEWORK) এ ব্যবহার হয়। এটা MVC এর মতো, কিন্তু কিছু নাম ও কাজের পার্থক্য আছে।

* প্রতিটি অংশের কাজ

1. MODEL

- DATABASE এর সাথে কাজ করে।
- DATA কীভাবে SAVE, UPDATE, RETRIEVE হবে সেটা ঠিক করে।
- BUSINESS LOGIC হ্যান্ডেল করে।

2. VIEW

- USER এর REQUEST (HTTP REQUEST) নেয়।
- MODEL থেকে DATA আনে বা আপডেট করে।
- DATA কে TEMPLATE এ পাঠায়।
- 👉 MVC এর CONTROLLER এর মতো কাজ করে।

3. TEMPLATE

- HTML ফাইল যেখানে DATA সুন্দরভাবে দেখানো হয়।
- DJANGO TEMPLATE LANGUAGE (DTL) ব্যবহার করে।
- শুধু PRESENTATION (UI) এর কাজ করে।

◆ MVT FLOW (STEP BY STEP)

1. USER REQUEST: ইউজার কোনো URL এ যায় (যেমন /HOME/)।
2. VIEW: REQUEST নেয় → MODEL থেকে ডেটা আনে।
3. MODEL: DATABASE থেকে ডেটা দেয় বা আপডেট করে।
4. TEMPLATE: VIEW ডেটা TEMPLATE এ পাঠায় → TEMPLATE HTML বানায় → ইউজারকে দেখায়।

◆ MVT EXAMPLE

• MODEL:

```
class User(models.Model):  
    name = models.CharField(max_length=100)  
    email = models.EmailField()
```

• VIEW:

```
def user_list(request):  
    users = User.objects.all()  
    return render(request, "user_list.html", {"users": users})
```

• TEMPLATE (USER_LIST.HTML):

```
<h1>All Users</h1>  
{% for u in users %}  
    <p>{{ u.name }} - {{ u.email }}</p>
```

```
{% endfor %}
```

- ◆ MVT এর সুবিধা
- ✓ TEMPLATE আর BUSINESS LOGIC আলাদা থাকে → কোড পরিষ্কার থাকে।
- ✓ VIEW (DJANGO) REQUEST-RESPONSE হ্যান্ডেল করতে সহজ করে।
- ✓ MAINTAIN করা, আপডেট করা সহজ।

- ◆ MVT ব্যবহার করে এমন FRAMEWORK
- DJANGO
- WEB2PY

👉 এক লাইনে মনে রাখো:

MVT হলো DJANGO এর MVC VERSION যেখানে –

- MODEL → DATA
- VIEW → LOGIC + REQUEST HANDLING (CONTROLLER এর কাজ করে)
- TEMPLATE → UI/HTML RENDERING

13. PROJECT START:

I. FIRST OF ALL, MAKE `views.py` IN INNER PROJECT. THEN, CONNECTED TO `urls.py`.

views.py:

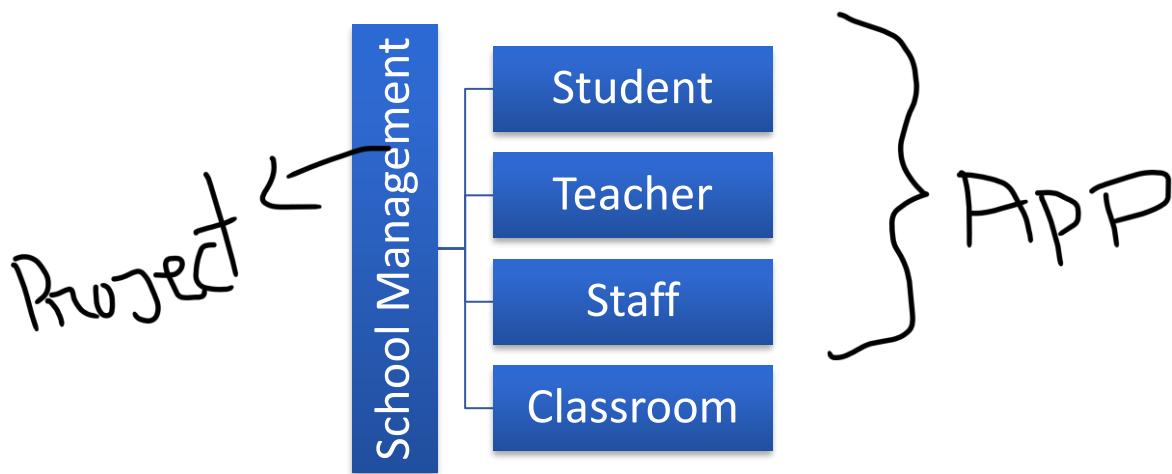
```
from django.http import HttpResponse

def home(request):
    return HttpResponse("<h1>Hello, Django! </h1>")
```

Urls.py:

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

II. Creating App



a) Project name = School Management

```
Django-admin startapp student[choose your any app name]
```

b) school_management/settings.py ফাইলে গিয়ে INSTALLED_APPS এ যোগ করো:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # custom apps
    'student',
    'teacher',
]
```

c) Student/teacher app er views.py kaj

```
from django.http import HttpResponse

def student_home(request):
    return HttpResponse("Welcome to Student App")
```

d) student/urls.py

(এই ফাইলটা তোমাকে তৈরি করতে হবে student app এর ভিতরে)

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

```
urlpatterns = [
    path('home/', views.student_home)
]
```

e) project urls.py এ connect করা:

school management/urls.py:

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

urlpatterns = [
    path('admin/', admin.site.urls),

    # apps urls
    path('student/', include('student.urls')),
    path('teacher/', include('teacher.urls')),
]
```

Ekhane Include diye student app er urls sathe connect koraisi..

f) Database migrate করা

```
python manage.py makemigrations
python manage.py migrate
//Tarpor runserver
python manage.py runserver
```

Templates

Django Template Easy Setup (Step by Step)

1. Templates ফোল্ডার বানাও

তোমার প্রোজেক্ট ফোল্ডারে একটা templates নামে ফোল্ডার বানাও।

```
mysite/
└── mysite/
    └── core/
        └── templates/
            └── home.html
```

2. settings.py তে Template path দাও

```
TEMPLATES = [
{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [BASE_DIR / "templates"],    # এই লাইন যোগ করো
```

```
'APP_DIRS': True,  
...  
},  
]  
]
```

3. *views.py* → *render*

core/views.py এ লিখো:

```
from django.shortcuts import render  
  
def home(request):  
    return render(request, "home.html")
```

4. *urls.py* → *route*

```
mysite/urls.py এ লিখো:  
from django.contrib import admin  
from django.urls import path  
from core.views import home  
  
urlpatterns = [  
    path("admin/", admin.site.urls),  
    path("", home, name="home"),  
]
```

5. *Template* বানাও

templates/home.html ফাইলের ভিতরে লিখো:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>My First Django Template</title>  
</head>  
<body>  
    <h1 style="color:red;">This is our first Django project</h1>  
    <p>Welcome to Django Templates!</p>  
</body>  
</html>
```

6. Run server

python manage.py runserver

Browser এ যাও: <http://127.0.0.1:8000/> → তোমার template দেখা যাবে ✓

(DTL = Django Template Language):

Html modde Python code kora

IF-ELIF-ELSE

Student/views.py

```
def profile(request):
    user_data = {
        'name': 'John Doe',
        'age': 17,
        'city': 'New York',
        'country': 'USA',
    }
    return render(request, "student/profile.html", context = user_data)
```

templates/student/profile.html

```
{% if age < 20 %}
    <p>Less than 20</p>
{% elif age == 20 %}
    <p>Equal to 20</p>
{% else %}
    <p>Greater than 20</p>
{% endif %}
```

For Loop:

Student/views.py

```
def profile(request):
    marks=[
        {
            'subject':'Math',
            'marks':90
        },
        {
            'subject':'English',
            'marks':85
        },
        {
            'subject':'Science',
            'marks':95
        },
        {
            'subject':'History',
            'marks':80
        },
        {
            'subject':'Geography',
            'marks':78
        },
        {
            'subject':'Computer Science',
            'marks':67
        }
    ]
    return render(request, "student/profile.html", {'marks': marks})
```

templates/stdent/index.html:

```
<table border="1">
    <tr>
```

```

<th>Subject Name</th>
<th>Marks</th>
<th>Grade</th>
</tr>

{% for mark in marks %}
<tr>
    <td>Subject: {{ mark.subject }}</td>
    <td>Mark: {{ mark.marks }}</td>
    <td>

        {% if mark.marks > 89 %}
        Grade: A+
        {% elif mark.marks > 79 %}
        Grade: A
        {% elif mark.marks > 69 %}
        Grade: B
        {% elif mark.marks > 59 %}
        Grade: C
        {% elif mark.marks > 49 %}
        Grade: D
        {% else %}
        Grade: F
    {% endif %}

    </td>
</tr>

{% endfor %}

</table>

```

Filter:

- *Upper*
 - *Lower*
 - *Length*
 - *Add*
 - *Divisibleby*
 - *Slice*
 - *Capitalize*
 - *Word Count*
 - *Truncate Words*
 - *Make list*
 - *Join*

Static File:

ঠিক আছে ভাই 😊 আমি একদম *easy step by step* দিয়ে দিচ্ছি যাতে তুমি *Django static files (CSS, JS, Images)* ভালোভাবে নোট করতে পারো।



Django Static Files – Easy Notes

1 Static Files কি?

- *Static files* হলো CSS, JS, Images, Fonts যা আপনার template এ ব্যবহার হয়।
- *Dynamic content* নয়, শুধু display এর জন্য।

2 settings.py এ Static configure করা

```
# settings.py
```

```
# base এর নিচে লিখো
STATIC_URL = '/static/'

# project এর root এ static folder রাখলে
STATICFILES_DIRS = [
    BASE_DIR / "static",
]
```

3 Project folder structure

```
myproject/
└── myapp/
    ├── templates/
    │   └── index.html
    └── views.py
├── static/
    ├── css/
    │   └── style.css
    ├── js/
    │   └── script.js
    └── images/
        └── logo.png
└── manage.py
└── settings.py
```

4 Template এ Static load করা

```
{% load static %}

<!-- CSS --&gt;
&lt;link rel="stylesheet" href="{% static 'css/style.css' %}"&gt;

<!-- JS --&gt;
&lt;script src="{% static 'js/script.js' %}"&gt;&lt;/script&gt;

<!-- Image --&gt;
&lt;img src="{% static 'images/logo.png' %}" alt="Logo"&gt;</pre>
```

5 Example CSS (static/css/style.css)

```
body {
    font-family: Arial, sans-serif;
```

```
background-color: #f0f0f0;
}

h1 {
    color: blue;
}
```

6 Example template (templates/index.html)

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Static Example</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    <h1>Hello Static Files!</h1>
    
    <script src="{% static 'js/script.js' %}"></script>
</body>
</html>
```

✓ Rule

1. *settings.py* এ *STATIC_URL* & *STATICFILES_DIRS* ঠিক করতে হবে
2. *Template* এ *{% load static %}* লিখতে হবে
3. সব *static files* *project/static/folder* এ রাখা

Apps er modde Static

 Django App এর ভিতরে Static Files ব্যবহার

1 App Structure

ধরো তোমার app এর নাম *student*

```
myproject/
  └── student/           ← তোমার app
      ├── static/          ← এখানে static রাখবে
          └── student/       ← app এর নামে সাব-ফোল্ডার রাখা ভালো
              ├── css/
              │   └── style.css
              └── images/
                  └── logo.png
      ├── templates/
          └── student/
              └── profile.html
      └── views.py
  └── manage.py
  myproject/
      └── settings.py
```

2 settings.py

তুমি যদি app এর ভিতরে static/ রাখো তাহলে আলাদা করে STATICFILES_DIRS লাগবে না।
শুধু এইটা রাখো (by default Django তে থাকে):

```
STATIC_URL = '/static/'
```

👉 Django স্ট্যাটিক ফাইলের প্রতিটি app এর ভিতরের static/ folder খুঁজে নিবে।

3 Template এ static ব্যবহার

student/templates/student/profile.html

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Student Profile</title>
    <!-- CSS connect -->
    <link rel="stylesheet" href="{% static 'student/css/style.css' %}">
</head>
<body>
    <h1>Hello {{ user.name }}</h1>

    <!-- Image connect -->
    
</body>
</html>
```

4 Example CSS (student/static/student/css/style.css)

```
body {
    background-color: #f0f0f0;
    font-family: Arial, sans-serif;
}

h1 {
    color: green;
}
```

5 Run server

```
python manage.py runserver
```

👉 এখন template এ গেলে static CSS & image load হবে।

✓ Rule মনে রাখো:

- App এর static সবসময় app_name/static/app_name/ ফোল্ডারে রাখো।

- *Template এ ব্যবহার করার সময়* → `{% static 'app_name/filepath' %}`

DTL URLs:

Django Template Language (DTL) – URL Notes

◆ 1. Direct URL (Hardcoded ✗)

```
<a href="/student/profile/">Profile</a>
```

- *Problem* → URL change হলে সব template এ manually change করতে হবে।
- ✗ Best practice না।

◆ 2. Named URL (Best Practice ✓)

```
# urls.py
path('profile/', views.profile, name="profile"),
```

```
<a href="{% url 'profile' %}">Profile</a>
```

- URL change করলেও name same থাকলে template safe থাকবে।
- Always name ব্যবহার করো।

◆ 3. Dynamic URL (parameter সহ)

```
# urls.py
path('student/<int:id>', views.student_detail, name="student_detail"),
```

```
<a href="{% url 'student_detail' 10 %}">Student 10</a>
```

👉 Output: /student/10/

◆ 4. Variable দিয়ে Dynamic URL

```
{% for student in students %}
    <a href="{% url 'student_detail' student.id %}">{{ student.name }}</a>
{% endfor %}
```

👉 Loop এর ভেতর থেকে dynamic link বানানো যায়।

◆ 5. Summary

- a) Direct link → ✗ avoid করো
- b) Named link → `{% url 'profile' %}` ✓
- c) Dynamic link → `{% url 'student_detail' 5 %}`
- d) Variable দিয়ে dynamic → `{% url 'student_detail' student.id %}`
- e) সবসময় urls.py তে name ব্যবহার করো

