HTTP Response in Django:

An HTTP response is the data returned by a server in response to a client's request (e.g., a user's browser requesting a webpage). In Django, responses are handled using the HttpResponse class or other response classes like JsonResponse for JSON data.

Common HTTP Responses in Django:

1. **HttpResponse: Sends back a basic response with custom content. Http Response is used to display Static messages.**

```
from django.http import HttpResponse

def my_view(request):

    return HttpResponse("Hello, world!")
```

2. JsonResponse: Returns JSON data, typically used for APIs.

```
from django.http import JsonResponse
def my_json_view(request):
    data = {"key": "value"}
    return JsonResponse(data)
```

Time in Django:

Django provides tools for working with time and dates, especially useful for tasks like timestamps, scheduling, or filtering data by time.

1. **Current Date and Time: You can get the current time using Python's datetime library.**

```
from datetime import datetime

def time_view(request):

    current_time = datetime.now()
```

```
        return HttpResponse(f"The current time is {current_time}")
```

2. Django Timezone: Django has built-in timezone handling for global projects.

```
from django.utils import timezone

def timezone_view(request):
    now = timezone.now()
    return HttpResponse(f"The current timezone-aware time is {now}")
```

Conditions in Django:

Conditions in Django views allow you to control what responses are sent based on logic, user requests, or database values. Commonly, conditions are used to manage permissions, validate data, or dynamically render content.

1. Basic Condition in a View:

```
from django.http import HttpResponseForbidden

def my_view(request):
    if not request.user.is_authenticated:
        return HttpResponseForbidden("You are not allowed to view this page.")
    return HttpResponse("Welcome!")
```

2. Conditional Rendering in Templates: In templates, you can use {% if %} statements to conditionally show content.

```
{% if user.is_authenticated %}
    <p>Welcome, {{ user.username }}!</p>
{% else %}
    <p>Please log in.</p>
```

{% endif %}

These components (HTTP responses, time handling, and conditional logic) allow you to create dynamic, user-specific, and time-sensitive functionality in Django.

We can define multiple methods in views.py.
In this project we have:

1. Multiple functions
2. Multiple applications
3. Project level urls
4. Application level urls

Installation: pip install pytz // python time zone
In settings.py:
Timezone = 'Asia/Kolkata' instead of 'UTC' for our local time

Project creation: django-admin startproject class2
Then cd class2
Next app creation: django-admin startapp myapp - first app
And next second app creation: django-admin startapp myapp2 - second app

In myapp - views.py file:

```
from django.shortcuts import render
from django.http import HttpResponse
from django.utils import timezone
# import datetime
import pytz

# Create your views here.
# function based view
# def display(request):
#     s = "<h1>Hello, Welcome to Django Class! </h1>"
```

```
#     return HttpResponse(s)

# def display2(request):
#     return HttpResponse("Hello, Django World!")

# def noon_message(request):
#     time = datetime.datetime.now()
#     formatted_time = time.strftime("%d-%m-%Y %H:%M:%S")
#     return HttpResponse("<h1>Hello, Goodafternoon! now the time is " +
formatted_time+ "</h1>")

def greeting(request):
    current_time = timezone.now()

    current_time_utc = timezone.now()
    ist_tz = pytz.timezone("Asia/Kolkata")
    current_time_utc = current_time_utc.astimezone(ist_tz)
    hour = current_time_utc.hour

    if 6 <= hour <= 12:
        greeting_message = "Good Morning"
    elif 12 <= hour <= 16:
        greeting_message = "Good Afternoon"
    elif 16 <= hour <= 21:
        greeting_message = "Good Evening"
    else:
        greeting_message = "Good Night"

    formatted_date = current_time_utc.strftime("%d-%m-%Y %H:%M:%S")
    return HttpResponse(f"{greeting_message} Today the date and time is
{formatted_date}")
```

In second app myapp2 - views.py file:

```python
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def home(request):
    return HttpResponse("<p> Home Page")

def contacts(request):
    return HttpResponse("<p> Contact Page")

def services(request):
    return HttpResponse("<p> Services Page")
```

Project level - class2 - settings.py file:

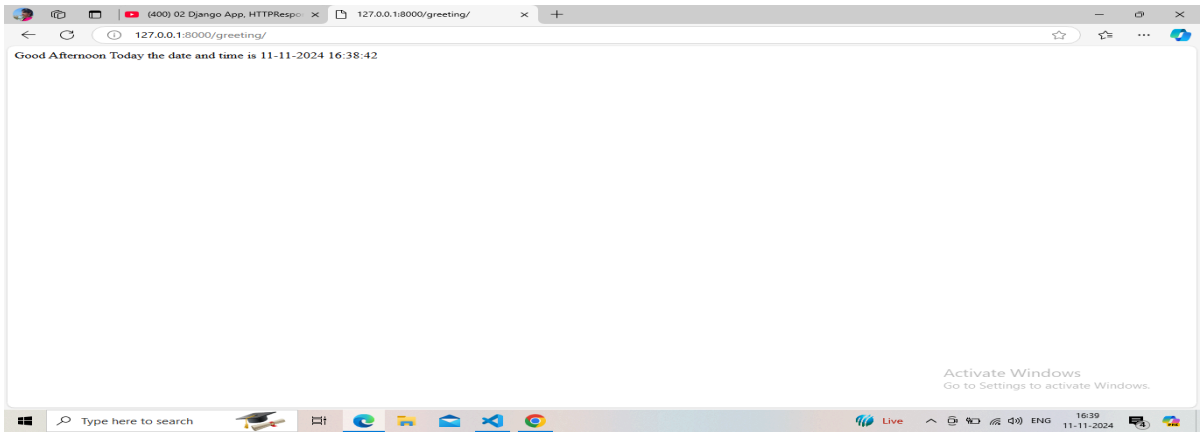
```python
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    'myapp',
    'myapp2',
]
```

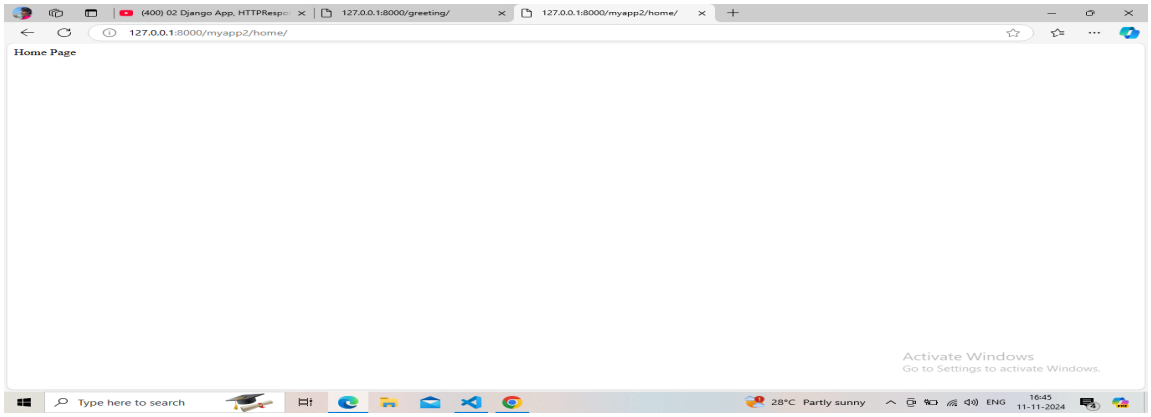
```python
TIME_ZONE = "Asia/Kolkata"
```

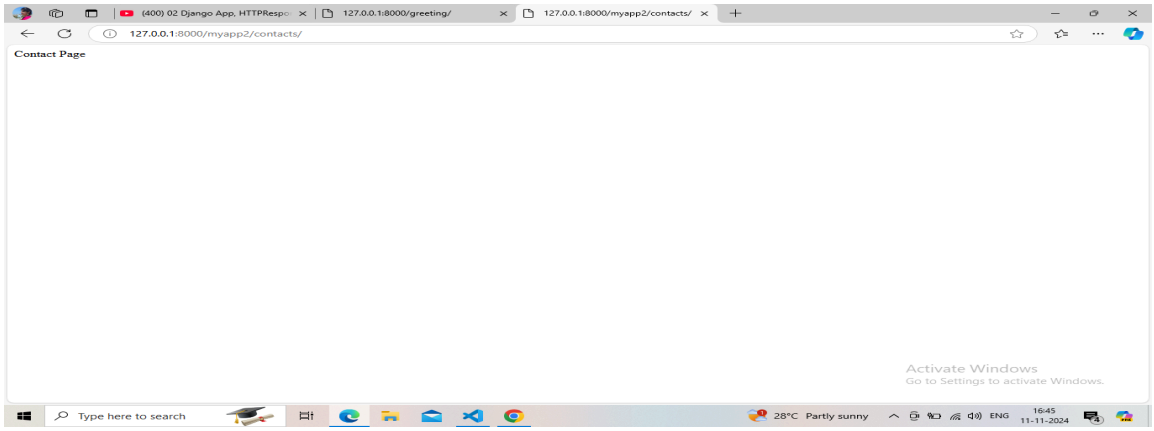
Project level - class2 - urls.py file:

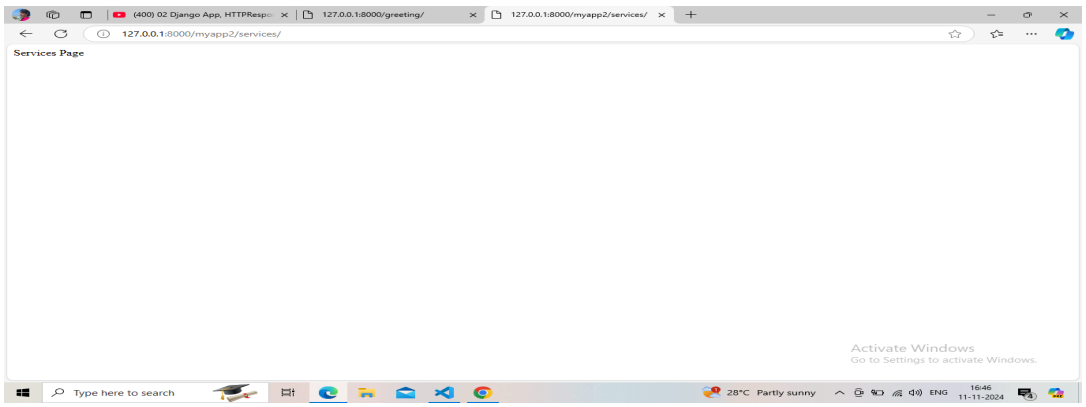
```python
from django.contrib import admin
from django.urls import path, include
from myapp import views

urlpatterns = [
    path("admin/", admin.site.urls),
    # path('greetings/', views.display),
    # path('greetings2/', views.display2),
    # path('greetings3/', views.noon_message),
    # path('myapp2/', include('myapp2.urls')),
    path('greeting/', views.greeting)
]
```

output:






(400) 02 Django App, HTTPRespo
127.0.0.1:8000/greeting/
127.0.0.1:8000/myapp2/contacts/
127.0.0.1:8000/myapp2/contacts/
Contact Page
Activate Windows
Go to Settings to activate Windows.
Type here to search
28°C Partly sunny
ENG
16:45
11-11-2024


(400) 02 Django App, HTTPRespo
127.0.0.1:8000/greeting/
127.0.0.1:8000/myapp2/services/
127.0.0.1:8000/myapp2/services/
Services Page
Activate Windows
Go to Settings to activate Windows.
Type here to search
28°C Partly sunny
ENG
16:46
11-11-2024