DAY -1 Session 2

1. File Structure

- Django's File Structure:
  When working with templates and static files, the typical Django project structure looks like this:

```
myproject/
├── myapp/
│   ├── templates/
│   │   └── myapp/
│   │       └── template_file.html
│   ├── static/
│   │   └── myapp/
│   │       └── css/
│   │           └── style.css
├── myproject/
│   ├── settings.py
│   ├── urls.py
├── manage.py
```

- **Templates Directory**:
  Contains HTML files used for rendering views.
- **Static Directory**:
  Contains static files such as CSS, JavaScript, and images.

2. Requirements.txt

- **Purpose**:
  A requirements.txt file lists all the Python packages required for the project. It is used for easy installation of dependencies.
- **Creating**:
  You can generate this file by running:

```
#pip freeze > requirements.txt
```

- **Usage**:
  Install the listed packages using:

```
#pip install -r requirements.txt
```

3. Create Views

- **Definition**:
  Views in Django are Python functions or classes that take web requests and return web responses, usually by rendering a template.

In your views.py file

```
from django.shortcuts import render,redirect

from django.http import HttpResponse

from django.views.decorators.csrf import csrf_exempt

# Create your views here.

def demo(request):

   return HttpResponse('This is a demo view')
```

4. Template

- **Definition**:
  Templates are HTML files that define the structure and layout of a webpage. They can include dynamic content using Django Template Language (DTL).

5. Static Files

- **Definition**:
  Static files include assets like CSS, JavaScript, and images that don't change during the application's lifecycle.

Add this in settings.py file

```
STATIC_URL = 'static/'

STATICFILES_DIRS = [

  os.path.join(BASE_DIR,'static'),
```