Django Step by Step Process

Step 1: Create Project

1. Go to VS Code Terminal
2. Hit command django-admin startproject Book_Store

Step 2: Create App

1. Go inside the Book_Store folder
2. Open VS Code in that folder (CMD Command – code .)
3. Go to VS Code Terminal
4. Hit command python manage.py startapp store

Step 3: Add the app to settings.py file

1. Inside the Book_Store folder open settings.py file
2. Add ‘store’ inside the INSTALLED_APPS

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

Step 4: Connect Database

1. Open VS Code Terminal
2. Hit command python manage.py makemigrations
3. Hit command python manage.py migrate

Step 5: Create urls.py in store folder

1. Go inside the store folder
2. Create urls.py file
3. Import Modules

Modules:

- from django.urls import path
- from store import views

Step 6: Connect Book_Store’s urls.py with store’s urls.py

1. Open Book_Store urls.py
2. Import Module from django.urls import include
3. Add a new path to urlpatterns connecting it with store urls.py file

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include("store.urls")),
]
```

Step 7: Add urlpatterns and path to store urls.py file

1. Open store urls.py
2. Create urlpatterns list and add path of home page

```
urlpatterns = [
    path('', views.home, name="home"),
]
```

Step 8: Create first view

1. Open store views.py
2. Create a method home

```
def home(request):
	return render(request, "index.html")
```

Step 9: Create Templates folder and connect to settings.py

1. Create templates folder in the base directory(the folder which has manage.py file)
2. Open settings.py file inside Book_Store folder
3. Add BASE_DIR / "templates“ to DIRS key of TEMPLATES dictionary

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / "templates"],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Step 10: Add index.html file inside templates folder

1. Open templates folder and create index.html file
2. Write Hello Django in h1 tag

Step 11: Run Server

1. Open VS Code Terminal
2. Run Command python manage.py runserver
3. Check port http://127.0.0.1:8000/

Step 12: Add and connect static folder

1. Create static folder in the base directory(the folder which has manage.py file)
2. Open settings.py file inside Book_Store folder
3. Add under STATIC_URL

```
STATICFILES_DIRS = [
    BASE_DIR / "static“
]
```

Step 13: Create a model inside store models.py

1. Open store models.py
2. Create a class Book

```
class Book(models.Model):
    bid=models.AutoField(primary_key=True)
    name=models.CharField(max_length=255)
    author=models.CharField(max_length=300)
    publisher=models.CharField(max_length=400)
    date=models.DateField(verbose_name='Publish Date')
    summary=models.TextField()
    price=models.DecimalField(max_digits=10,decimal_places=0)


    def __str__(self):
        return f"Book ID:{self.bid} Name:{self.name}"
```

Step 14: Register to admin.py

1. Open admin.py in store folder
2. Import models (from store.models import *)
3. Register Book model

```
admin.site.register(Book)
```

Step 15: Create an admin user

1. Open VS Code Terminal
2. Run command python manage.py createsuperuser
3. Give details and create user

Step 16: Update database and run server

1. Open VS Code Terminal
2. Hit command python manage.py makemigrations
3. Hit command python manage.py migrate
4. Hit command python manage.py runserver

Step 17: Login to Admin Panel

1. Open chrome http://127.0.0.1:8000/admin and login
2. Add few books data

Step 18: Send the Data from the views

1. Open store views.py and add from store.models import * at the top
2. Add to views bk=Book.objects.all()
3. Add data to a dictionary data={‘book’: Book}
4. Send the data

```
def home(request):
        return render(request, "index.html“, data)
```

Step 19: Show the Data on home page

1. Open index.html and start a Django template for loop

```
{% for b in book %}
<h1>{{ b.name }}</h1>
{% endfor %}
```

Step 20: User Sign Up Step 1

1. Open index.html and add <a href="/signup">Sign Up</a>
2. Open store urls.py file and add a new path

```
path('signup', views.signup, name="signup"),
```

Step 21: User Sign Up Step 2

1. Create forms.py file inside store directory
2. Add modules

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
```

Step 22: User Sign Up Step 3

Add SignUpForm class

```
class SignUpForm(UserCreationForm):
    username=forms.CharField(label="Username", widget=forms.TextInput(
        attrs={
            'placeholder': 'jonny_english_mi7',
            'class': 'form-control'
        }))
    first_name = forms.CharField(label="First Name", widget=forms.TextInput(
        attrs={
            'placeholder': 'Jonny',
            'class': 'form-control'
        }))
    last_name = forms.CharField(label="Last Name", widget=forms.TextInput(
        attrs={
            'placeholder': 'English',
            'class': 'form-control'
        }))
    email = forms.CharField(label="Email", widget=forms.TextInput(
        attrs={
            'placeholder': 'jonny@mi7.com',
            'class': 'form-control'
        }))
    password1 = forms.CharField(label="Password", widget=forms.PasswordInput(
        attrs={
            'placeholder': 'Password',
            'class': 'form-control'
        }))
    password2 = forms.CharField(label="Re-Enter Password", widget=forms.PasswordInput(
        attrs={
            'placeholder': 'Confirm Password',
            'class': 'form-control w-100'
        }))
    class Meta:
        model=User
        fields=['username','first_name','last_name', 'email']
```

Step 23: User Sign Up Step 4

Open views.py and create a method signup

```
def signup(request):
    if request.method == "POST":
        form = SignUpForm(request.POST)
        if form.is_valid():
            try:
                form.save()
            except Exception as e:
                print(e)
    else:
        form = SignUpForm()
    data = {"form": form}
    return render(request, “forms.html", data)
```

Step 24: User Sign Up Step 5

Create forms.html file in templates and add a form

```
<form method="post">
{% csrf_token %}
{{ form }}
</form>
```

Step 25: User Sign In Step 1

1. Open index.html and add <a href="/signin">Sign In</a>
2. Open store urls.py file and add a new path

```
path('signin', views.signin, name="signin"),
```

Step 26: User Sign In Step 2

1. Open forms.py file of store directory
2. Add modules

```
from django.contrib.auth.forms import AuthenticationForm
```

Step 27: User Sign In Step 3

Add SignInForm class

```
class SignInForm(AuthenticationForm):
    username = forms.CharField(label="Username", widget=forms.TextInput(
        attrs={
            'class': 'form-control border-primary',
            'placeholder': 'Enter your username'
        }))
    password = forms.CharField(label="Password",
widget=forms.PasswordInput(
        attrs={
            'class': 'form-control border-primary',
            'placeholder': 'Enter your password'
        }))
```

Step 28: User Sign In Step 4

1. Add from django.contrib.auth import login,logout,authenticate
2. Open views.py and create a method signin

```
def signin(request):
    if request.method == "POST":
        form = SignInForm(request=request, data=request.POST)
        if form.is_valid():
            uname = form.cleaned_data['username']
            upass = form.cleaned_data['password']
            user = authenticate(username=uname, password=upass)
            if user is not None:
                login(request, user)
                return redirect('/profile')
    else:
        form = SignInForm()
    data = {'form': form}
    return render(request, ‘forms.html', data)
```

Step 29: Create Profile Step 1

1. Open index.html and add <a href="/profile">Visit Profile</a>
2. Open store urls.py file and add a new path

```
path(‘profile', views.profile, name=“profile"),
```

Step 30: Create Profile Step 2

1. Open views.py of store
2. Add modules

```
from django.contrib.auth.decorators import login_required
```

Step 31: Create Profile Step 3

Create a method in views.py profile

```
@login_required(login_url='/signin')
def profile(request):
    return render(request, “profile.html”)
```

Step 32: Create Profile Step 4

Create profile.html file in templates folder

```
<h1>Hello {{ request.user.first_name }}</h1>
<a href="/signout">Logout</a>
```

Step 33: Add logout path to urls

1. Open store urls.py
2. Add a path

```
path(‘signout', views.signout, name=“signout"),
```

Step 34: Create signout method

Create a signout method

```
def signout(request):
    logout(request)
    return redirect("/")
```

Step 35: Add profile page

1. Create a button on index.html page <a href=“/profile”>Profile</a>
2. Add path to store urls.py path(‘profile', views.profile, name=“profile"),
3. Add to views.py from django.contrib.auth.decorators import login_required
4. Add profile method

```
@login_required(login_url='/signin’)
def profile(request):
    data=Book.objects.all()
    return render(request, “profile.html”,{‘books’:data})
```

Step 36: Add profile.html

```
{% for b in books %}
<div>
<h3>{{ b.name }}</h3>
<a href="/delete/{{ b.bid }}">Delete</a>
<hr>
</div>
{% endfor %}
```

Step 37: Add Delete function

1. Add path to store urls.py path(‘delete/<int:id>', views.delete, name=“delete"),
2. Add delete function

```
def delete(request, id):
    b=Book.objects.all()
    b.delete()
    return redirect(‘/profile’)
```