Sriram Sunderrajan

220962444

WP Lab

Week-10

27th March 2025

**PROJECT FILES**

**settings.py**

"""

Django settings for myProj project.

Generated by 'django-admin startproject' using Django 5.1.6.

For more information on this file, see

https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/5.1/ref/settings/

"""

from pathlib import Path

import os

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-o5nkkkd85o6o%&$0z6w3x5#5fcroz\*t7!2!qec8snc1xvl!4t0'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'bookApp'

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'myProj.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [

os.path.join(BASE\_DIR, "bookApp/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',

],

},

},

]

WSGI\_APPLICATION = 'myProj.wsgi.application'

# Database

# https://docs.djangoproject.com/en/5.1/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

# Password validation

# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/5.1/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/5.1/howto/static-files/

STATIC\_URL = 'static/'

# Default primary key field type

# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

**urls.py**

from django.contrib import admin

from django.urls import path, re\_path, include

urlpatterns = [

path('admin/', admin.site.urls),

re\_path('bookApp/', include('bookApp.urls'))

]

**Question – 1**

**urls.py**

from django.urls import path

from . import views

urlpatterns = [

path('', views.index, name="index"),

path('addAuthor/', views.authorIndex, name="Add Author"),

path('addPublisher/', views.publisherIndex, name="Add Publisher"),

path('addBook/', views.bookIndex, name="Add Book"),

path('retrieve/', views.retrieve, name="RetrieveRecords")

]

**models.py**

from django.db import models

class Author(models.Model):

first\_name = models.CharField(max\_length=100)

last\_name = models.CharField(max\_length=100)

email = models.EmailField()

def \_\_str\_\_(self):

return f"{self.first\_name} {self.last\_name}"

class Publisher(models.Model):

name = models.CharField(max\_length=200)

street\_address = models.CharField(max\_length=255)

city = models.CharField(max\_length=100)

state\_province = models.CharField(max\_length=100)

country = models.CharField(max\_length=100)

website = models.URLField()

def \_\_str\_\_(self):

return self.name

class Book(models.Model):

title = models.CharField(max\_length=200)

publication\_date = models.DateField()

authors = models.ManyToManyField(Author)

publisher = models.ForeignKey(Publisher, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return self.title

**forms.py**

from django import forms

from .models import Author, Publisher, Book

class AuthorForm(forms.Form):

first\_name = forms.CharField(max\_length=100)

last\_name = forms.CharField(max\_length=100)

email = forms.EmailField()

class PublisherForm(forms.Form):

name = forms.CharField(max\_length=100)

street\_address = forms.CharField(max\_length=255)

city = forms.CharField(max\_length=100)

state\_province = forms.CharField(max\_length=100)

country = forms.CharField(max\_length=100)

website = forms.URLField()

class BookForm(forms.Form):

title = forms.CharField(max\_length=100)

publication\_date = forms.DateField()

authors = forms.CharField(max\_length=300)

publisher = forms.CharField(max\_length=100)

class SearchForm(forms.Form):

name = forms.CharField(max\_length=100)

**views.py**

from django.shortcuts import render, redirect

from .forms import BookForm, AuthorForm, PublisherForm, SearchForm

from .models import Book, Author, Publisher

def index(request):

bookForm, authorForm, publisherForm = BookForm(), AuthorForm(), PublisherForm()

return render(request, "populate.html", {

"bookForm": bookForm,

"authorForm": authorForm,

"publisherForm": publisherForm

})

def bookIndex(request):

if request.method == "POST":

form = BookForm(request.POST)

if form.is\_valid():

print("Valid")

data = form.cleaned\_data

book = Book.objects.create(

title = data["title"],

publication\_date = data["publication\_date"],

publisher = Publisher.objects.get(name=data["publisher"])

)

author\_names = data["authors"].split()

for author\_name in author\_names:

author = Author.objects.filter(first\_name=author\_name)[0]

book.authors.add(author)

return redirect("index")

print("Not Valid")

def authorIndex(request):

if request.method == "POST":

form = AuthorForm(request.POST)

if form.is\_valid():

data = form.cleaned\_data

author = Author.objects.create(

first\_name = data["first\_name"],

last\_name = data["last\_name"],

email = data["email"]

)

return redirect("index")

def publisherIndex(request):

if request.method == "POST":

form = PublisherForm(request.POST)

if form.is\_valid():

data = form.cleaned\_data

publisher = Publisher.objects.create(

name = data["name"],

street\_address = data["street\_address"],

city = data["city"],

state\_province = data["state\_province"],

country = data["country"],

website = data["website"]

)

return redirect("index")

def retrieve(request):

if request.method == "POST":

form = SearchForm(request.POST)

if form.is\_valid():

name = form.cleaned\_data["name"]

book = Book.objects.filter(title=name)[0]

bookName = book.title

bookDate = book.publication\_date

authors = book.authors

print([i for i in authors.all()])

author = None

for a in authors.all():

author = a

break

if author is not None:

authorName = author.first\_name

authorEmail = author.email

else:

authorName = None

authorEmail = None

publisher = book.publisher

publisherName = publisher.name

publisherWebsite = publisher.website

context = {"authorName": authorName,

"authorEmail": authorEmail,

"publisherName": publisherName,

"publisherWebsite": publisherWebsite,

"bookName": bookName,

"bookDate": bookDate}

print(context)

return render(request, "retrieve.html", context)

else:

form = SearchForm()

return render(request, "retrieve.html", {"form": form})

**tempates: populate.html**

<!DOCTYPE html>

<html>

<head>

<title>Insert Data into Tables</title>

</head>

<body>

<h1>Insert Data</h1>

<form method="POST" action="./addAuthor/">

{% csrf\_token %}

{{ authorForm.as\_p }}

<button type="submit">Insert</button>

</form>

<form method="POST" action="./addPublisher/">

{% csrf\_token %}

{{ publisherForm.as\_p }}

<button type="submit">Insert</button>

</form>

<form method="POST" action="./addBook/">

{% csrf\_token %}

{{ bookForm.as\_p }}

<button type="submit">Insert</button>

</form>

<a href="{% url 'RetrieveRecords' %}">Go to Search</a>

</body>

</html>

**templates: retrieve.html**

<!DOCTYPE html>

<html>

<head>

<title>Retrieve</title>

</head>

<body>

<h1>Retrieve Data</h1>

<form method="POST" action="./">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Search</button>

</form>

<a href="{% url 'index' %}">Go to Populate</a>

<ul>{{ bookName }}</ul>

<ul>{{ bookDate }}</ul>

<ul>{{ authorName }}</ul>

<ul>{{ authorEmail }}</ul>

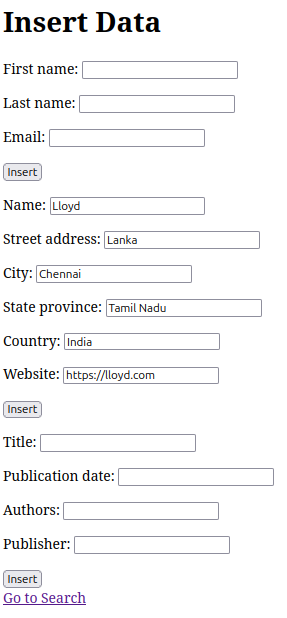
<ul>{{ publisherName }}</ul>

<ul>{{ publisherWebsite }}</ul>

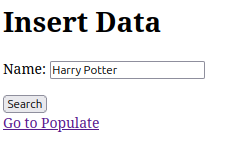
</body>

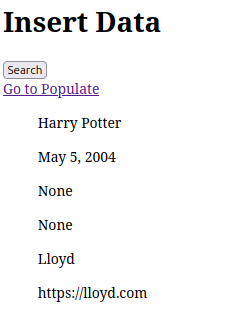
</html>

**OUTPUT**









**Question - 2**

**urls.py**

from django.urls import path

from .views import create\_product, product\_list

urlpatterns = [

path('', product\_list, name='product\_list'),

path('create/', create\_product, name='create\_product'),

]

**models.py**

from django.db import models

class Product(models.Model):

title = models.CharField(max\_length=200)

price = models.DecimalField(max\_digits=8, decimal\_places=2)

description = models.TextField()

def \_\_str\_\_(self):

return self.title

**forms.py**

from django import forms

from .models import Product

class ProductForm(forms.ModelForm):

class Meta:

model = Product

fields = ('title', 'price', 'description')

**admin.py**

from django.contrib import admin

from .models import Product

admin.site.register(Product)

**views.py**

from django.shortcuts import render, redirect

from .forms import ProductForm

from .models import Product

def create\_product(request):

if request.method == 'POST':

form = ProductForm(request.POST)

if form.is\_valid():

form.save()

return redirect('product\_list')

else:

form = ProductForm()

return render(request, 'create\_product.html', {'form': form})

def product\_list(request):

products = Product.objects.all()

return render(request, 'product\_list.html', {'products': products})

**templates: base.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Product App</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

</head>

<body>

<div class="container mt-5">

{% block content %}{% endblock %}

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

</body>

</html>

**templates: create\_product.html**

{% extends 'base.html' %}

{% block content %}

<h1 class="mb-3">Create Product</h1>

<form method="post">

{% csrf\_token %}

<div class="mb-3">

{{ form.title.label\_tag }} {{ form.title }}

</div>

<div class="mb-3">

{{ form.price.label\_tag }} {{ form.price }}

</div>

<div class="mb-3">

{{ form.description.label\_tag }} {{ form.description }}

</div>

<button type="submit" class="btn btn-primary">Create</button>

</form>

{% endblock %}

**templates: product\_list.html**

{% extends 'base.html' %}

{% block content %}

<h1 class="mb-3">Product List</h1>

<ul class="list-group">

{% for product in products %}

<li class="list-group-item">

<span>{{ product.title }} ({{ product.price }})</span>

<br>

{{ product.description }}

</li>

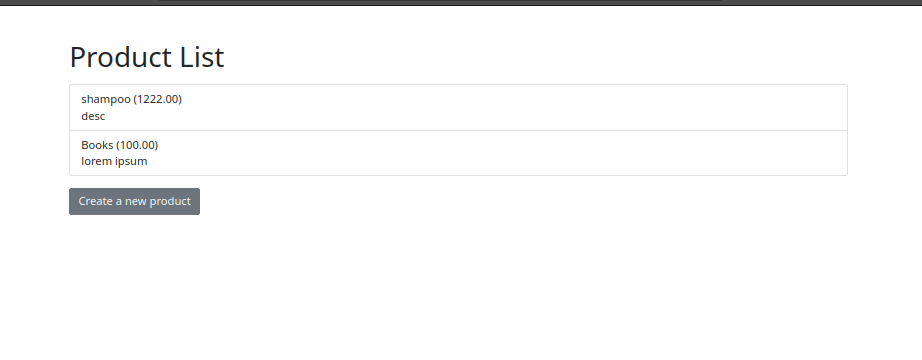
{% endfor %}

</ul>

<p><a href="{% url 'create\_product' %}" class="btn btn-secondary mt-3">Create a new product</a></p>

{% endblock %}

**OUTPUT**



**Question – 3**

**views.py**

from django.shortcuts import render, redirect

from .models import Human

from django.http import JsonResponse

def human\_list(request):

humans = Human.objects.all()

first\_names = [h.first\_name for h in humans]

return render(request, 'human\_list.html', {'first\_names': first\_names})

def get\_human\_details(request, first\_name):

try:

human = Human.objects.get(first\_name=first\_name)

return JsonResponse({

'first\_name': human.first\_name,

'last\_name': human.last\_name,

'phone': human.phone,

'address': human.address,

'city': human.city,

})

except Human.DoesNotExist:

return JsonResponse({})

def update\_human(request, first\_name):

human = Human.objects.get(first\_name=first\_name)

if request.method == 'POST':

human.last\_name = request.POST.get('last\_name')

human.phone = request.POST.get('phone')

human.address = request.POST.get('address')

human.city = request.POST.get('city')

human.save()

return redirect('human\_list')

def delete\_human(request, first\_name):

human = Human.objects.get(first\_name=first\_name)

human.delete()

return redirect('human\_list')

**forms.py**

from django import forms

from .models import Human

class HumanForm(forms.ModelForm):

class Meta:

model = Human

fields = ('first\_name', 'last\_name', 'phone', 'address', 'city')

**urls.py**

from django.urls import path

from .views import human\_list, get\_human\_details, update\_human, delete\_human

urlpatterns = [

path('', human\_list, name='human\_list'),

path('get\_details/<str:first\_name>/', get\_human\_details, name='get\_human\_details'),

path('update/<str:first\_name>/', update\_human, name='update\_human'),

path('delete/<str:first\_name>/', delete\_human, name='delete\_human'),

]

**models.py**

from django.db import models

class Human(models.Model):

first\_name = models.CharField(max\_length=100)

last\_name = models.CharField(max\_length=100)

phone = models.CharField(max\_length=20)

address = models.CharField(max\_length=200)

city = models.CharField(max\_length=50)

def \_\_str\_\_(self):

return f"{self.first\_name} {self.last\_name}"

**templates: human\_list.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Human Details</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

</head>

<body>

<div class="container mt-5">

<h1>Human Details</h1>

<select id="humanSelect" class="form-select">

{% for name in first\_names %}

<option value="{{ name }}">{{ name }}</option>

{% endfor %}

</select>

<form id="humanForm">

<div class="mb-3">

<label for="first\_name" class="form-label">First Name</label>

<input type="text" class="form-control" id="first\_name" disabled>

</div>

<div class="mb-3">

<label for="last\_name" class="form-label">Last Name</label>

<input type="text" class="form-control" id="last\_name" name="last\_name">

</div>

<div class="mb-3">

<label for="phone" class="form-label">Phone</label>

<input type="text" class="form-control" id="phone" name="phone">

</div>

<div class="mb-3">

<label for="address" class="form-label">Address</label>

<input type="text" class="form-control" id="address" name="address">

</div>

<div class="mb-3">

<label for="city" class="form-label">City</label>

<input type="text" class="form-control" id="city" name="city">

</div>

<button type="submit" class="btn btn-primary" id="updateBtn">Update</button>

<button type="button" class="btn btn-danger" id="deleteBtn">Delete</button>

</form>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

<script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>

<script>

document.addEventListener('DOMContentLoaded', function() {

const humanSelect = document.getElementById('humanSelect');

const humanForm = document.getElementById('humanForm');

humanSelect.addEventListener('change', function() {

const first\_name = this.value;

fetch('/get\_details/' + first\_name)

.then(response => response.json())

.then(data => {

document.getElementById('first\_name').value = data.first\_name;

document.getElementById('last\_name').value = data.last\_name || '';

document.getElementById('phone').value = data.phone || '';

document.getElementById('address').value = data.address || '';

document.getElementById('city').value = data.city || '';

});

});

humanForm.addEventListener('submit', function(event) {

event.preventDefault();

const first\_name = document.getElementById('first\_name').value;

const last\_name = document.getElementById('last\_name').value;

const phone = document.getElementById('phone').value;

const address = document.getElementById('address').value;

const city = document.getElementById('city').value;

$.ajax({

type: 'POST',

url: '/update/' + first\_name + '/',

data: {

'last\_name': last\_name,

'phone': phone,

'address': address,

'city': city,

'csrfmiddlewaretoken': '{{ csrf\_token }}',

},

success: function() {

alert('Details updated successfully!');

},

error: function(xhr) {

alert('Error updating details: ' + xhr.statusText);

}

});

});

document.getElementById('deleteBtn').addEventListener('click', function() {

const first\_name = document.getElementById('first\_name').value;

if (confirm('Are you sure you want to delete this record?')) {

$.ajax({

type: 'GET',

url: '/delete/' + first\_name + '/',

success: function() {

alert('Record deleted successfully!');

// Refresh dropdown

location.reload();

},

error: function(xhr) {

alert('Error deleting record: ' + xhr.statusText);

}

});

}

});

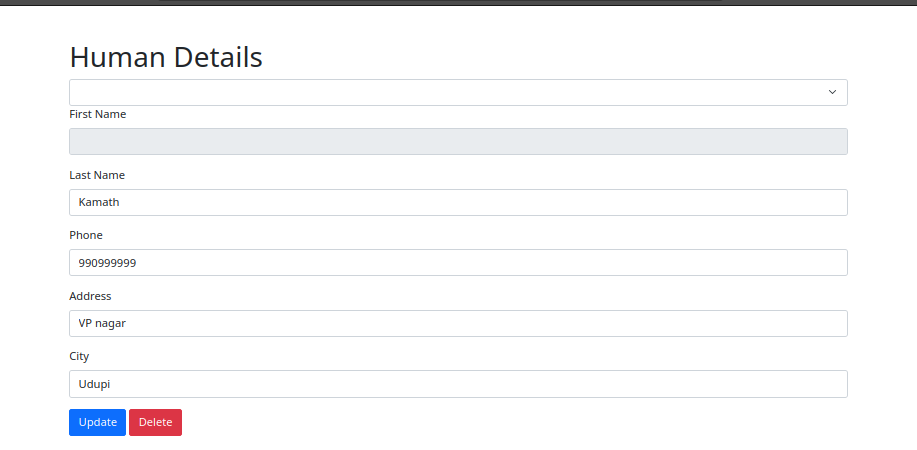
});

</script>

</body>

</html>

**OUTPUT**

****