IT-LAB_ WEEK 8 SUBMISSIONS

NAME:Sagnik Chatterjee

Reg: 180905478

Roll No: 61

Sec: B

Sem :6

Project level settings :
week8/settings.py

```
"""
Django settings for week8v2 project.

Generated by 'django-admin startproject' using Django 3.2.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/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/3.2/howto/deployment/checklist
/

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

```
SECRET_KEY =
'django-insecure-j3%$lxa5-10w#zi+a=k5!z8bpj32!ttx5f%r!z0+h8#k_s8
u-n'

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

ALLOWED_HOSTS = ['127.0.0.1']


# Application definition

INSTALLED_APPS = [
    'prob4.apps.Prob4Config',
    'prob3.apps.Prob3Config',
    'prob2.apps.Prob2Config',
    'prob1.apps.Prob1Config',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

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 = 'week8v2.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(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',
            ],
        },
    },
]

WSGI_APPLICATION = 'week8v2.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME':os.path.join(BASE_DIR,'db.sqlite3'),
    }
}
```

```
# Password validation
#
https://docs.djangoproject.com/en/3.2/ref/settings/#auth-passwor
d-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarity
Validator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator
',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidato
r',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/
```

```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'

# Default primary key field type
#
https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto
-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

week8/urls.py

```
"""week8 URL Configuration

The `urlpatterns` list routes URLs to views. For more
information please see:
   https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
   1. Add an import:  from my_app import views
   2. Add a URL to urlpatterns:  path('', views.home,
name='home')
```

```
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(),
name='home')
Including another URLconf
    1. Import the include() function: from django.urls import
include, path
    2. Add a URL to urlpatterns:  path('blog/',
include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('q1/',include('prob1.urls')),
    path('q2/',include('prob2.urls')),
    path('q3/',include('prob3.urls')),
    path('q4/',include('prob4.urls')),

]
```

Q4 Create a Django Page for entry of a Product information (title, price and description) and save it into the db. Create the index page where you would view the product entries in an unordered list.

prob4/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('',views.home,name="home"),
    path('entry',views.entry,name="entry"),
    path('index',views.index,name="index")
```

```
]
```

prob4/views.py

```
from django.shortcuts import render
from .forms import ProductForm
from .models import Product
# Create your views here.
def home(request):
    return render(request,'prog4.html')

def entry(request):
    form1 = ProductForm(request.POST)
    form = ProductForm()
    if form1.is_valid():
        title = form1.cleaned_data['title']
        price = form1.cleaned_data['price']
        desc = form1.cleaned_data['desc']
        Product.objects.create(title = title,price = price,desc =
desc)
    return render(request,'prog4p1.html',{"form":form})

def index(request):
    products = Product.objects.all()
    return render(request,'prog4p2.html',{"products":products})
```

prob4/models.py

```
from django.db import models

# Create your models here.
class Product(models.Model):
    title = models.CharField(max_length=100)
    price = models.IntegerField()
    desc = models.TextField()
```

prob4/forms.py

```
from django import forms

class ProductForm(forms.Form):
   title = forms.CharField(max_length=100)
   price = forms.IntegerField()
   desc =
forms.CharField(widget=forms.Textarea(),label="description")
```

Prog4.html

```
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
   <title>Q4</title>
 </head>
 <body>
   <p>
     Create a Django Page for entry of a Product information
(title, price and
     description) and save it into the db. Create the index page
where you
     would view the product entries in an unordered list.
   </p>
   <a href="{% url 'entry' %}">Enter a new product</a><br />
   <a href="{% url 'index' %}">View Products</a>
 </body>
</html>
```

Prog4p1.html

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <form action="entry" method="POST">
       {% csrf_token %}
       <table>
           {{form.as_table}}
       </table>
       <input type="submit" value="add">
   </form>
   <a href="{% url 'home' %}">Go back to home</a>
</body>
</html>
```

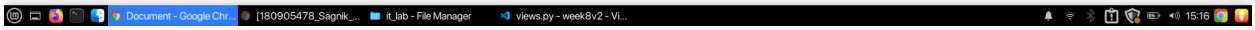
Prog4p2.html

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <h1>Products:</h1><br>
   <ul>
```

```
        {% for product in products %}
        <li>{{product.title}} <br>₹{{product.price}}
<br>{{product.desc}}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

Products:

- product1
  ₹1234
  "At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat."



Q2 Consider the following tables: WORKS(person-name,Company-name,Salary) LIVES(Person_name, Street, City) Assume Table data suitably. Design a Django webpage and include an option to insert data into WORKS table by accepting data from the user using TextBoxes. Also, include an option to retrieve the names of people who work for a particular company along with the cities they live in (particular company name must be accepted from the user).

prob2/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
   path('',views.home,name="home"),
   path('portal',views.portal,name="portal"),
   path('search',views.search,name="search")
]
```

prob2/forms.py

```
from django import forms


class Employee(forms.Form):
```

```
    name = forms.CharField(max_length=100)
    company = forms.CharField(max_length=100)
    salary = forms.IntegerField()
    street = forms.CharField(max_length=200)
    city = forms.CharField(max_length=50)

class Company(forms.Form):
    company = forms.CharField(max_length=100)
```

prob2/models.py

```
from django.db import models
from django.db.models.fields.related import ForeignKey

# Create your models here.
class Works(models.Model):
    name = models.CharField(max_length=100)
    company = models.CharField(max_length=100)
    salary = models.IntegerField()

class Lives(models.Model):
    name = models.CharField(max_length=100)
    street = models.CharField(max_length=200)
    city = models.CharField(max_length=50)
```

prob2/views.py

```
from django.shortcuts import render
from .models import Works,Lives
from .forms import Employee,Company
# Create your views here.
def home(request):
    return render(request,'prog2.html')

def portal(request):
    form = Employee()
    form1 = Employee(request.POST)
```

```
    if form1.is_valid():
        name = form1.cleaned_data['name']
        company = form1.cleaned_data['company']
        salary = form1.cleaned_data['salary']
        street = form1.cleaned_data['street']
        city = form1.cleaned_data['city']

Works.objects.create(name=name,company=company,salary=salary)
        Lives.objects.create(name=name,street=street,city=city)
    return render(request,'prog2p1.html',{"form":form})

def search(request):
    form = Company()
    form1 = Company(request.POST)
    if form1.is_valid():
        company = form1.cleaned_data["company"]
        employa = Works.objects.all().filter(company = company)
        employees = []
        for e in employa:
            employees.append(Lives.objects.get(name = e.name))
        return
render(request,"prog2p2.html",{"form":form1,"employees":employee
s})
    return render(request,"prog2p2.html",{"form":form})
```

Prog2.html

```
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
   <title>Q2 | Part 1</title>
 </head>
 <body>
```

```
    <p>
      Consider the following tables:
WORKS(person-name,Company-name,Salary)
      LIVES(Person_name, Street, City) Assume Table data
suitably. Design a
      Django webpage and include an option to insert data into
WORKS table by
      accepting data from the user using TextBoxes. Also, include
an option to
      retrieve the names of people who work for a particular
company along with
      the cities they live in (particular company name must be
accepted from the
      user).
    </p>
    <a href="{% url 'portal' %}">update employee portal</a><br />
    <a href="{% url 'search' %}">Find the employee list of a
company</a>
  </body>
</html>
```

Prog2p1.html

```
<html lang="en">
 <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>Q2 | Part 2</title>
 </head>
 <body>
    <p>
      Consider the following tables:
WORKS(person-name,Company-name,Salary)
```

```
     LIVES(Person_name, Street, City) Assume Table data
suitably. Design a
     Django webpage and include an option to insert data into
WORKS table by
     accepting data from the user using TextBoxes. Also, include
an option to
     retrieve the names of people who work for a particular
company along with
     the cities they live in (particular company name must be
accepted from the
     user).
   </p>
   <form action="portal" method="POST">
     {% csrf_token %}
     <table>
       {{form.as_table}}
     </table>
     <br />
     <input type="submit" value="insert" />
   </form>
   <a href="{% url 'home' %}">Go back to home</a>
 </body>
</html>
```

Prog2p2.html

```
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
   <title>Q2 | Part 3</title>
 </head>
 <body>
```

```
    <p>
      Consider the following tables:
WORKS(person-name,Company-name,Salary)
      LIVES(Person_name, Street, City) Assume Table data
suitably. Design a
      Django webpage and include an option to insert data into
WORKS table by
      accepting data from the user using TextBoxes. Also, include
an option to
      retrieve the names of people who work for a particular
company along with
      the cities they live in (particular company name must be
accepted from the
      user).
    </p>
    <form action="search" method="POST">
      {% csrf_token %} {{form}}
      <br />
      <input type="submit" value="search" />
    </form>
    <br />
    <table>
      <thead>
        <td>name</td>
        <td>city</td>
      </thead>
      {% for employee in employees %}
      <tr>
        <td>{{employee.name}}</td>
        <td>{{employee.city}}</td>
      </tr>
      {% endfor %}
    </table>
    <a href="{% url 'home' %}">Go back to home</a>
```

```
 </body>
</html>
```


Q2 | Part 1
127.0.0.1:8000/q2/
Consider the following tables: WORKS(person-name,Company-name,Salary) LIVES(Person_name, Street, City) Assume Table data suitably. Design a Django webpage and include an option to insert data into WORKS table by accepting data from the user using TextBoxes. Also, include an option to retrieve the names of people who work for a particular company along with the cities they live in (particular company name must be accepted from the user).
update employee portal
Find the employee list of a company
Q2 | Part 1 - Google Chr...
[180905478_Sagnik_...
it_lab - File Manager
prog4.html - week8v2 -...
15:09

Q3 There are three tables in the database an author table has a first name, a last name and an email address. A publisher table has a name, a street address, a city, a state/ province, a country, and a Web site. A book table has a title and a publication date. It also has one or more authors (a

many-to-many relationship with authors) and a single publisher (a one-to-many relationship - aka foreign key - to publishers). Design a form which populates and retrieves the information from the above database using Django.

prob3/views.py

```
from django.shortcuts import render
from .forms import
AuthorForm,PublisherForm,BookForm,AuthorSearch,PublisherSearch,B
ookSearch
from .models import Au,Publisher,Book
# Create your views here.
def home(request):
   return render(request,'prog3.html')

def publisherEntry(request):
   form = PublisherForm()
   form1 = PublisherForm(request.POST)
   if form1.is_valid():
       name = form1.cleaned_data["name"]
       street = form1.cleaned_data["street"]
       city = form1.cleaned_data["city"]
       state = form1.cleaned_data["state"]
       country = form1.cleaned_data["country"]
       site = form1.cleaned_data["site"]
       Publisher.objects.create(name = name,street = street,city
= city,state = state,country = country,site = site)
   return render(request,'prog3p1.html',{"form":form})

def authorEntry(request):
   form = AuthorForm()
   form1 = AuthorForm(request.POST)
   if form1.is_valid():
       fname = form1.cleaned_data["fname"]
```

```
        lname = form1.cleaned_data["lname"]
        email = form1.cleaned_data["email"]
        Au.objects.create(fname = fname,lname = lname,em = email)
    return render(request,'prog3p2.html',{"form":form})

def bookEntry(request):
    form = BookForm()
    form1 = BookForm(request.POST)
    if form1.is_valid():
        a = form1.cleaned_data
        title = a["title"]
        pdate = a["pdate"]
        pname = a["pname"]
        anames = a["anames"].split()
        print(anames)
        publisher = Publisher.objects.get(name = pname)
        authors = []
        book = Book(title = title,pdate = pdate,publisher =
publisher)
        book.save()
        for i in anames:
            a = Au.objects.get(fname = i)
            book.authors.add(a)
        book.save()
    return render(request,'prog3p3.html',{"form":form})

def searchBook(request):
    form = BookSearch()
    form1 = BookSearch(request.POST)
    if form1.is_valid():
        title = form1.cleaned_data["title"]
        book = Book.objects.get(title = title)
        return
render(request,'prog3p4.html',{"form":form1,"book":book})
```

```
    return render(request,'prog3p4.html',{"form":form})

def searchAuthor(request):
    form = AuthorSearch()
    form1 = AuthorSearch(request.POST)
    if form1.is_valid():
        fname = form1.cleaned_data["fname"]
        author = Au.objects.get(fname = fname)
        return
render(request,'prog3p5.html',{"form":form1,"author":author})
    return render(request,'prog3p5.html',{"form":form})

def searchPublisher(request):
    form = PublisherSearch()
    form1 = PublisherSearch(request.POST)
    if form1.is_valid():
        name = form1.cleaned_data["name"]
        publisher = Publisher.objects.get(name = name)
        return
render(request,'prog3p6.html',{"form":form1,"publisher":publishe
r})
    return render(request,'prog3p6.html',{"form":form})
```

prob3/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('',views.home,name="home"),

path('publisherEntry',views.publisherEntry,name="publisherEntry"
),
    path('authorEntry',views.authorEntry,name="authorEntry"),
    path('bookEntry',views.bookEntry,name="bookEntry"),
    path('searchBook',views.searchBook,name="searchBook"),
    path('searchAuthor',views.searchAuthor,name="searchAuthor"),
```

```
path('searchPublisher',views.searchPublisher,name="searchPublish
er"),
]
```

prob3/models.py

```
from django.db import models
from django.db.models.aggregates import Count

# Create your models here.

class Publisher(models.Model):
    name = models.CharField(max_length=100)
    street = models.CharField(max_length=200)
    city = models.CharField(max_length=50)
    state = models.CharField(max_length=50)
    country = models.CharField(max_length=50)
    site = models.URLField()
class Au(models.Model):
    fname = models.CharField(max_length=100)
    lname = models.CharField(max_length=100)
    em = models.EmailField()

class Book(models.Model):
    title = models.CharField(max_length=200)
    pdate = models.DateField()
    authors = models.ManyToManyField(Au)
    publisher =
models.ForeignKey(Publisher,on_delete=models.CASCADE)
```

prob3/forms.py

```
from django import forms

class PublisherForm(forms.Form):
```

```
    name = forms.CharField(max_length=100)
    street = forms.CharField(max_length=200)
    city = forms.CharField(max_length=50)
    state = forms.CharField(max_length=50)
    country = forms.CharField(max_length=50)
    site = forms.URLField()

class AuthorForm(forms.Form):
    fname = forms.CharField(max_length=100, label="first name")
    lname = forms.CharField(max_length=100, label="last name")
    email = forms.EmailField()

class BookForm(forms.Form):
    title = forms.CharField(max_length=200)
    pdate = forms.DateField(label="publication date")
    pname = forms.CharField(max_length=100,label="Publisher
name")
    anames = forms.CharField(max_length=400,label="Enter first
names of authors by space seperation")

class BookSearch(forms.Form):
    title = forms.CharField(max_length=200)

class AuthorSearch(forms.Form):
    fname = forms.CharField(max_length=100, label="enter the
first name")

class PublisherSearch(forms.Form):
    name = forms.CharField(max_length=100)
```

Prob3.html

```
<html lang="en">
 <head>
```

```
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
   <title>Q3 | part 1</title>
 </head>
 <body>
   <p>
     There are three tables in the database an author table has
a first name, a
     last name and an email address. A publisher table has a
name, a street
     address, a city, a state/ province, a country, and a Web
site. A book
     table has a title and a publication date. It also has one
or more authors
     (a many-to-many relationship with authors) and a single
publisher (a
     one-to-many relationship - aka foreign key - to
publishers). Design a form
     which populates and retrieves the information from the
above database
     using Django.
   </p>
   <a href="{% url 'publisherEntry' %}">Register a
publisher</a><br />
   <a href="{% url 'authorEntry' %}">Register a author</a><br />
   <a href="{% url 'bookEntry' %}">Register a book</a><br />
   <a href="{% url 'searchBook' %}">Search for a book</a><br />
   <a href="{% url 'searchAuthor' %}">Search for a author</a><br
/>
   <a href="{% url 'searchPublisher' %}">Search for a
publisher</a>
 </body>
```

```
</html>
```

Prob3p1.html

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Publisher Entry</title>
</head>
<body>
   <h1>Publisher Registration:</h1>
   <form action="publisherEntry" method="POST">
       {% csrf_token %}
       <table>
           {{form.as_table}}
       </table>
       <input type="submit" value=register>
   </form>
   <a href="{% url 'home' %}">Go back to home</a>
</body>
</html>
```

Prob3p2.html

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Author Entry</title>
</head>
<body>
```

```
    <h1>Author Registration:</h1>
    <form action="authorEntry" method="POST">
        {% csrf_token %}
        <table>
            {{form.as_table}}
        </table>
        <input type="submit" value=register>
    </form>
    <a href="{% url 'home' %}">Go back to home</a>
</body>
</html>
```

Prob3p3.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Book Entry</title>
</head>
<body>
    <h1>Book Registration:</h1>
    <form action="bookEntry" method="POST">
        {% csrf_token %}
        <table>
            {{form.as_table}}
        </table>
        <input type="submit" value=register>
    </form>
    <a href="{% url 'home' %}">Go back to home</a>
</body>
</html>
```

Prob3p4.html

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Book Search</title>
</head>
<body>
   <h1>Search for book</h1>
   <form action="searchBook" method="POST">
       {% csrf_token %}
       {{form}}
       <br><input type="submit" value = "Search">
   </form>
   <table>
       <thead>
           <td>Title</td>
           <td>Published Date</td>
           <td>Name of the Publisher</td>
           <td>Name of the authors</td>
       </thead>
       <tr>
           <td>{{book.title}}</td>
           <td>{{book.pdate}}</td>
           <td>{{book.publisher.name}}</td>
           <td>
               {% for author in book.authors.all %}
               {{author.fname}} {{author.lname}} <br>
               {% endfor %}
           </td>
       </tr>
   </table>
   <a href="{% url 'home' %}">Go back to home</a>
```

```
</body>
</html>
```

Prob3p5.html

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Author Search</title>
</head>
<body>
   <h1>Search for Author</h1>
   <form action="searchAuthor" method="POST">
       {% csrf_token %}
       {{form}}
       <br><input type="submit" value = "Search">
   </form>
   <table>
       <thead>
           <td>First Name</td>
           <td>Last Name</td>
           <td>email</td>
       </thead>
       <tr>
           <td>{{author.fname}}</td>
           <td>{{author.lname}}</td>
           <td>{{author.em}}</td>
       </tr>
   </table>
   <a href="{% url 'home' %}">Go back to home</a>
</body>
</html>
```

Prob3p6.html

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Publisher Search</title>
</head>
<body>
   <h1>Search for Publisher</h1>
   <form action="searchPublisher" method="POST">
       {% csrf_token %}
       {{form}}
       <br><input type="submit" value = "Search">
   </form>
   <table>
       <thead>
           <td>Name</td>
           <td>Street</td>
           <td>City</td>
           <td>State</td>
           <td>Country</td>
           <td>Website</td>
       </thead>
       <tr>
           <td>{{publisher.name}}</td>
           <td>{{publisher.street}}</td>
           <td>{{publisher.city}}</td>
           <td>{{publisher.state}}</td>
           <td>{{publisher.country}}</td>
           <td>{{publisher.site}}</td>
       </tr>
   </table>
   <a href="{% url 'home' %}">Go back to home</a>
```

```
</body>
</html>
```

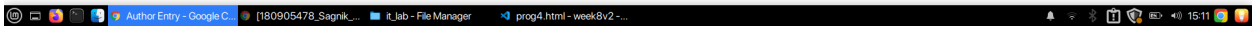

Q3 | part 1
127.0.0.1:9000/q3/
There are three tables in the database an author table has a first name, a last name and an email address. A publisher table has a name, a street address, a city, a state/ province, a country, and a Web site. A book table has a title and a publication date. It also has one or more authors (a many-to-many relationship with authors) and a single publisher (a one-to-many relationship - aka foreign key - to publishers). Design a form which populates and retrieves the information from the above database using Django.
Register a publisher
Register a author
Register a book
Search for a book
Search for a author
Search for a publisher

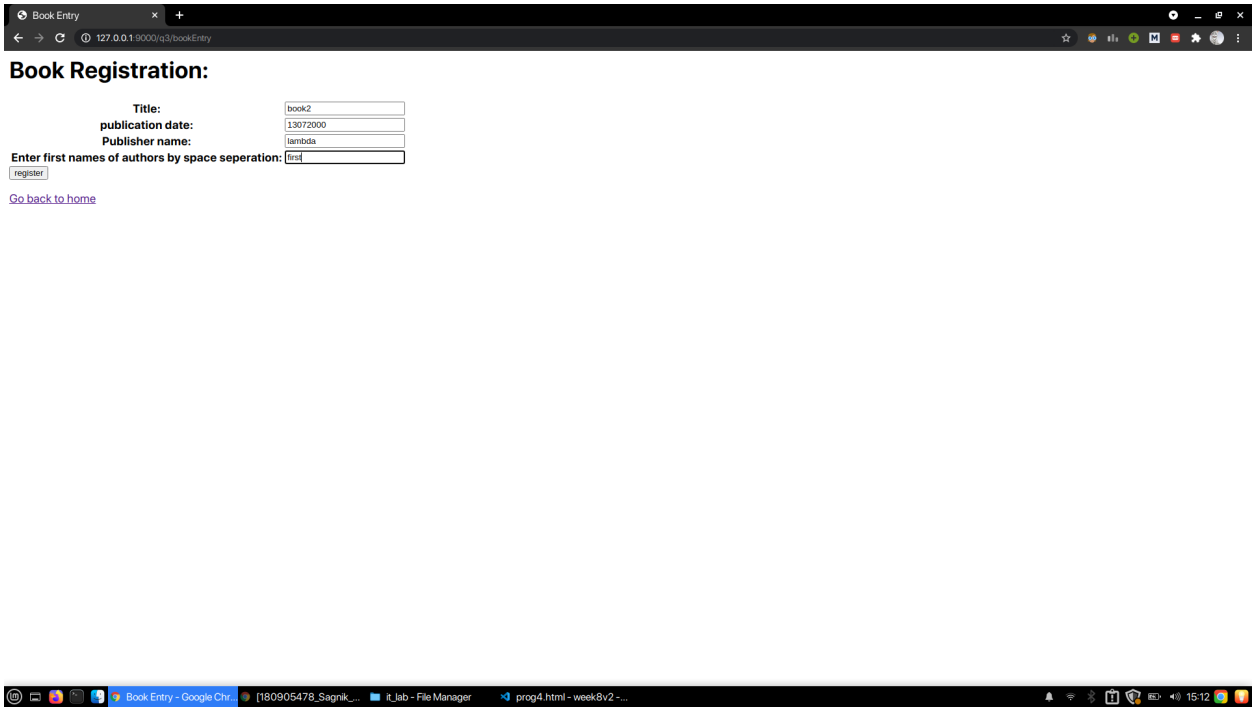
Q3 | part 1 - Google Chr...
[180905478_Sagnik_...
it_lab - File Manager
prog4.html - week8v2 -...
15:10


Publisher Entry
127.0.0.1:9000/q3/publisherEntry
Publisher Registration:
Name: author1
Street: abc
City: def
State: ghi
Country: india
Site: https://manipal.edu
register
Go back to home

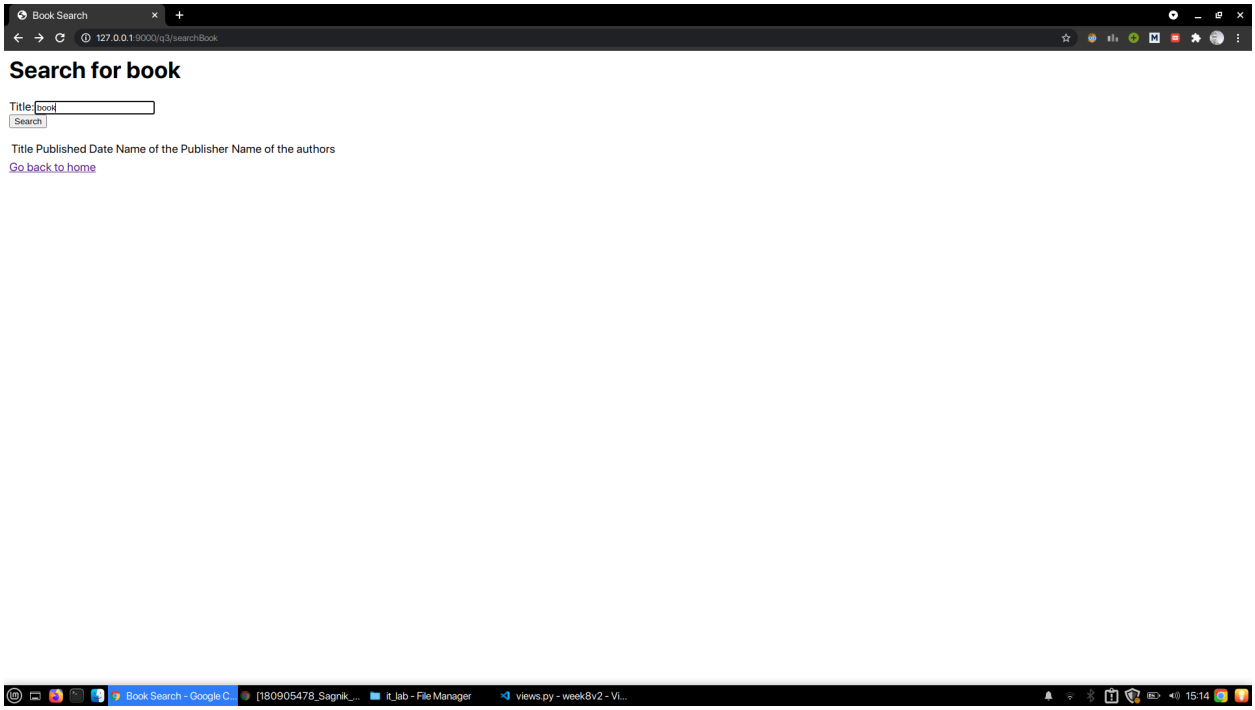
Publisher Entry - Google...
[180905478_Sagnik_...
it_lab - File Manager
prog4.html - week8v2 -...
15:11


Author Entry
127.0.0.1:9000/q3/authorEntry
Author Registration:
first name: first
last name: last
Email: author@email.com
register
Go back to home


Author Entry - Google C...
[180905478_Sagnik_...
it_lab - File Manager
prog4.html - week8v2 -...
15:11


Book Entry
127.0.0.1:9000/q3/bookEntry
Book Registration:
Title: book2
publication date: 13072000
Publisher name: lambda
Enter first names of authors by space seperation: first
register
Go back to home

Book Entry - Google Chr...
[180905478_Sagnik_...
it_lab - File Manager
prog4.html - week8v2 -...
15:12


Book Search
127.0.0.1:9000/q3/searchBook
Search for book
Title:book
Search
Title Published Date Name of the Publisher Name of the authors
Go back to home
Book Search - Google C...
[180905478_Sagnik_...
it_lab - File Manager
views.py - week8v2 - Vi...
15:14


Author Search
127.0.0.1:9000/q3/searchAuthor
Search for Author
enter the first name:lambda
Search
First Name Last Name email
Go back to home
Author Search - Google...
[180905478_Sagnik_...
it_lab - File Manager
views.py - week8v2 - Vi...
15:15

Q1 Design a web site using Django, which is a website directory – A site containing links to other websites. A web page has different categories. • A category table has a name, number of visits, and number of likes. • A page table refers to a category, has a title, URL, and many views. Design a form that populates the above database and displays it.

prob1/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('',views.home,name="home"),
    path('category',views.category,name="category"),
    path('page/',views.page,name = "page"),
    path('display/',views.display,name="display")
]
```

prob1/forms.py

```
from prob1.models import Category
from django import forms
```

```
class CategoryForm(forms.Form):
   name = forms.CharField(max_length=100)
   numberOfVisits = forms.IntegerField()
   numberOfLikes = forms.IntegerField()

class PageForm(forms.Form):
   category = forms.CharField(max_length=100)
   title = forms.CharField(max_length=100)
   url = forms.URLField()
   view = forms.IntegerField()
```

prob1/models.py

```
from django.db import models
class Category(models.Model):
   name = models.CharField(max_length=100,primary_key=True)
   numberOfVisits = models.IntegerField()
   numberOfLikes = models.IntegerField()
# Create your models here
class Page(models.Model):
   category = models.CharField(max_length=100)
   title = models.CharField(max_length=100)
   url = models.URLField(primary_key=True)
   view = models.IntegerField()
```

prob1/views.py

```
from django.shortcuts import render
from .forms import CategoryForm,PageForm
from .models import Category,Page
# Create your views here.
def home(request):
```

```
    return render(request,'prog1.html')

def category(request):
    form1 = CategoryForm()
    form = CategoryForm(request.POST)
    if form.is_valid():
        name = form.cleaned_data["name"]
        nov = form.cleaned_data["numberOfVisits"]
        nol = form.cleaned_data["numberOfLikes"]
        Category.objects.create(name = name, numberOfVisits =
nov,numberOfLikes = nol)
    return render(request,'prog1p1.html',{"form":form1})

def page(request):
    form1 = PageForm()
    form = PageForm(request.POST)
    if form.is_valid():
        category = form.cleaned_data['category']
        title = form.cleaned_data['title']
        url = form.cleaned_data['url']
        view = form.cleaned_data['view']
        Page.objects.create(category = category,title = title,url
= url,view = view)
    return render(request,'prog1p2.html',{"form":form1})

def display(request):
    pages = Page.objects.all()
    categories = Category.objects.all()
    return
render(request,'prog1p3.html',{"pages":pages,"categories":catego
ries})
```

prog1.html

```
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
   <title>Q1  | Part 1</title>
 </head>
 <body>
   <p>
     Design a web site using Django, which is a website
directory - A site
     containing links to other websites. A web page has
different categories. •
     A category table has a name, number of visits, and number
of likes. • A
     page table refers to a category, has a title, URL, and many
views. Design
     a form that populates the above database and displays it.
   </p>
   <a href="{% url 'category' %}">Enter Information to category
table</a><br />
   <a href="{% url 'page'}">Enter Information to Page
table</a><br />
   <a href="{% url 'display'}">Display Category table and page
table</a><br />
 </body>
</html>
```

Prog1p1.html

```
<html lang="en">
 <head>
   <meta charset="UTF-8" />
```

```
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
   <title>Q1 | part 2</title>
 </head>
 <body>
   <p>
     Design a web site using Django, which is a website
directory - A site
     containing links to other websites. A web page has
different categories. •
     A category table has a name, number of visits, and number
of likes. • A
     page table refers to a category, has a title, URL, and many
views. Design
     a form that populates the above database and displays it.
   </p>
   <form action="category" method="POST">
     {% csrf_token %}
     <table>
       {{form.as_table}}
     </table>
     <input type="submit" value="insert" />
   </form>
   <br />
   <a href="{% url 'home' %}">Go back to home</a>
 </body>
</html>
```

Prog1p2.html

```
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>Q1 | Part 3</title>
  </head>
  <body>
    <p>
      Design a web site using Django, which is a website
directory - A site
      containing links to other websites. A web page has
different categories. •
      A category table has a name, number of visits, and number
of likes. • A
      page table refers to a category, has a title, URL, and many
views. Design
      a form that populates the above database and displays it.
    </p>
    <form action="page" method="POST">
      {% csrf_token %}
      <table>
        {{form.as_table}}
      </table>
      <input type="submit" value="insert" />
    </form>
    <a href="{% url ('home') %}">back to home</a>
  </body>
</html>
```

Prog1p3.html

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
```

```
  <title>Q1 | Part 4</title>
 </head>
 <body>
   <p>
     Design a web site using Django, which is a website
directory – A site
     containing links to other websites. A web page has
different categories. •
     A category table has a name, number of visits, and number
of likes. • A
     page table refers to a category, has a title, URL, and many
views. Design
     a form that populates the above database and displays it.
   </p>
   <h1>Category Table:</h1>
   <br />
   <table>
     <thead>
       <td>Name</td>
       <td>Number of Visits</td>
       <td>Number of likes</td>
     </thead>
     {% for category in categories %}
     <tr>
       <td>{{category.name}}</td>
       <td>{{category.numberOfVisits}}</td>
       <td>{{category.numberOfLikes}}</td>
     </tr>
     {% endfor %}
   </table>
   <br />
   <h1>Page table</h1>
   <table>
     <thead>
```

```
      <td>Category</td>
      <td>Title</td>
      <td>URL</td>
      <td>View</td>
    </thead>
    {% for page in pages %}
    <tr>
      <td>{{page.category}}</td>
      <td>{{page.title}}</td>
      <td>{{page.url}}</td>
      <td>{{page.view}}</td>
    </tr>
    {% endfor %}
  </table>
  <br />
  <a href="{% url 'home' %}">back to home</a>
 </body>
</html>
```


Q1 | Part 1
127.0.0.1:9000/q1/
Design a web site using Django, which is a website directory – A site containing links to other websites. A web page has different categories. • A category table has a name, number of visits, and number of likes. • A page table refers to a category, has a title, URL, and many views. Design a form that populates the above database and displays it.
Enter Information to category table
Enter Information to Page table
Display Category table and page table
Q1 | Part 1 - Google Chr...
[180905478_Sagnik_...
it_lab - File Manager
prog4.html - week8v2 -...
15:07


Q1 | part 2
127.0.0.1:9000/q1/category
Design a web site using Django, which is a website directory – A site containing links to other websites. A web page has different categories. • A category table has a name, number of visits, and number of likes. • A page table refers to a category, has a title, URL, and many views. Design a form that populates the above database and displays it.
Name: chatterjee
Numberofvisits: 123123
Numberoflikes: 1233233
insert
Go back to home
Q1 | part 2 - Google Chr...
[180905478_Sagnik_...
it_lab - File Manager
prog4.html - week8v2 -...
15:08