2CSDE86 Application Development Frameworks

Lab-2 Task

Submitted by: Labdhi Sheth 18BCE101

Aim:

Design a view, templates, and filters for different modules. Implement crispy form an inbuilt authentication for signup and login module.

Methodology:

1. Create a django project by starting the anaconda terminal and write:
   ```
   django-admin.py startproject demo_web
   ```

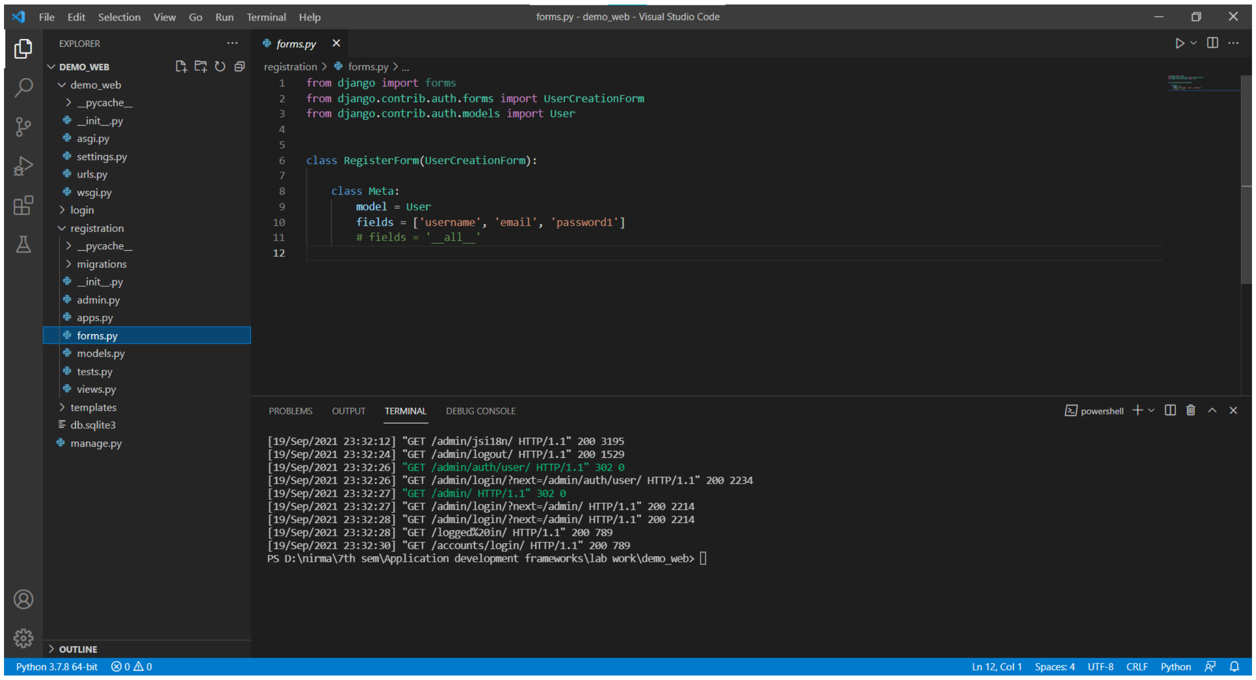
2. Open the project in vscode. Since the project will use crispy forms, install them by starting a new terminal in the IDE iself and write
   ```
   pip install django-crispy-forms
   ```

   This gives a whole new UI and additional features like the authentications to the forms.

3. Firstly we'll be making the registration app thus in the terminal write:
   ```
   python manage.py startapp registration
   ```

4. The registration app looks like this:
   a. No changes have been done to __init__.py, models.py, tests.py, apps.py, and admin.py file

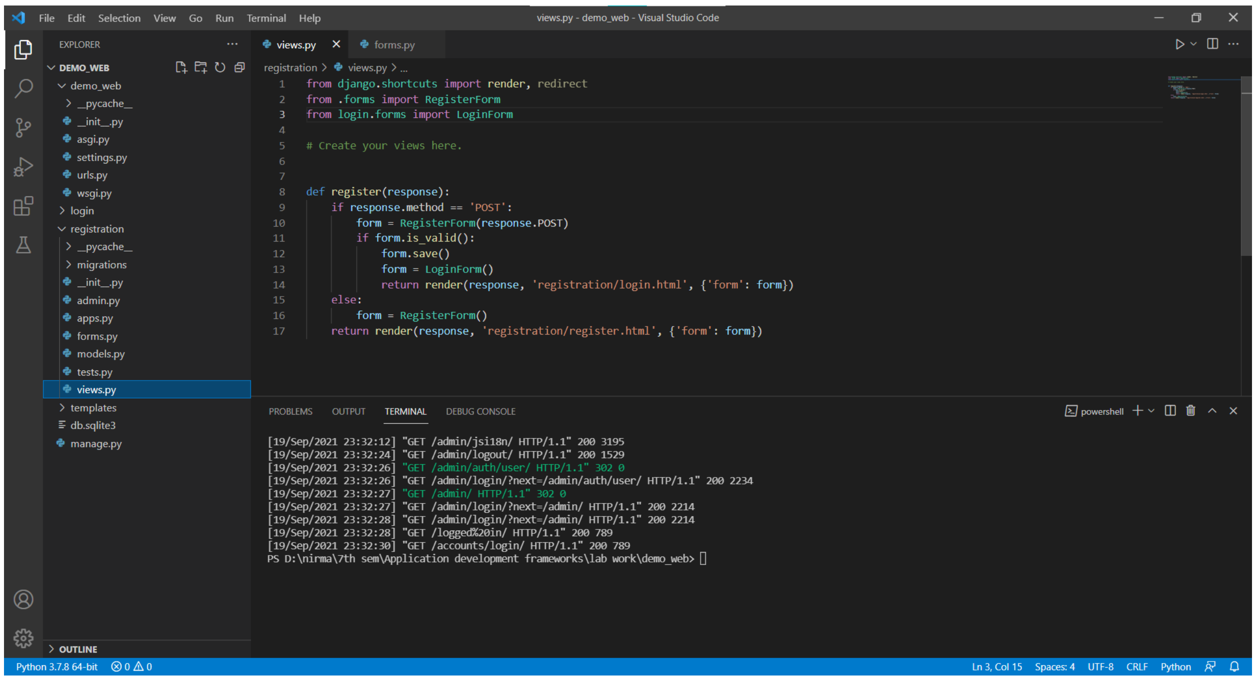
b. Create a new file names as “forms.py” under registration app.
   It creates a form for the user to register. The fields of the form are
   I. username
   Ii. email id
   Iii. password
   The class is named RegisterForm which will be further called in registration/views.py file



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


class RegisterForm(UserCreationForm):

    class Meta:
        model = User
        fields = ['username', 'email', 'password1']
        # fields = '__all__'

```

```
[19/Sep/2021 23:32:12] "GET /admin/jsi18n/ HTTP/1.1" 200 3195
[19/Sep/2021 23:32:24] "GET /admin/logout/ HTTP/1.1" 200 1529
[19/Sep/2021 23:32:26] "GET /admin/auth/user/ HTTP/1.1" 302 0
[19/Sep/2021 23:32:26] "GET /admin/login/?next=/admin/auth/user/ HTTP/1.1" 200 2234
[19/Sep/2021 23:32:27] "GET /admin/ HTTP/1.1" 302 0
[19/Sep/2021 23:32:27] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[19/Sep/2021 23:32:28] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

c. views.py:
   Herein we are calling the RegisterForm and the LoginForm, LogInForm will be discussed later in the same document. If the registration is successful then the login form is called where the user can be logged in to the system else it will redirect to the registration login again.

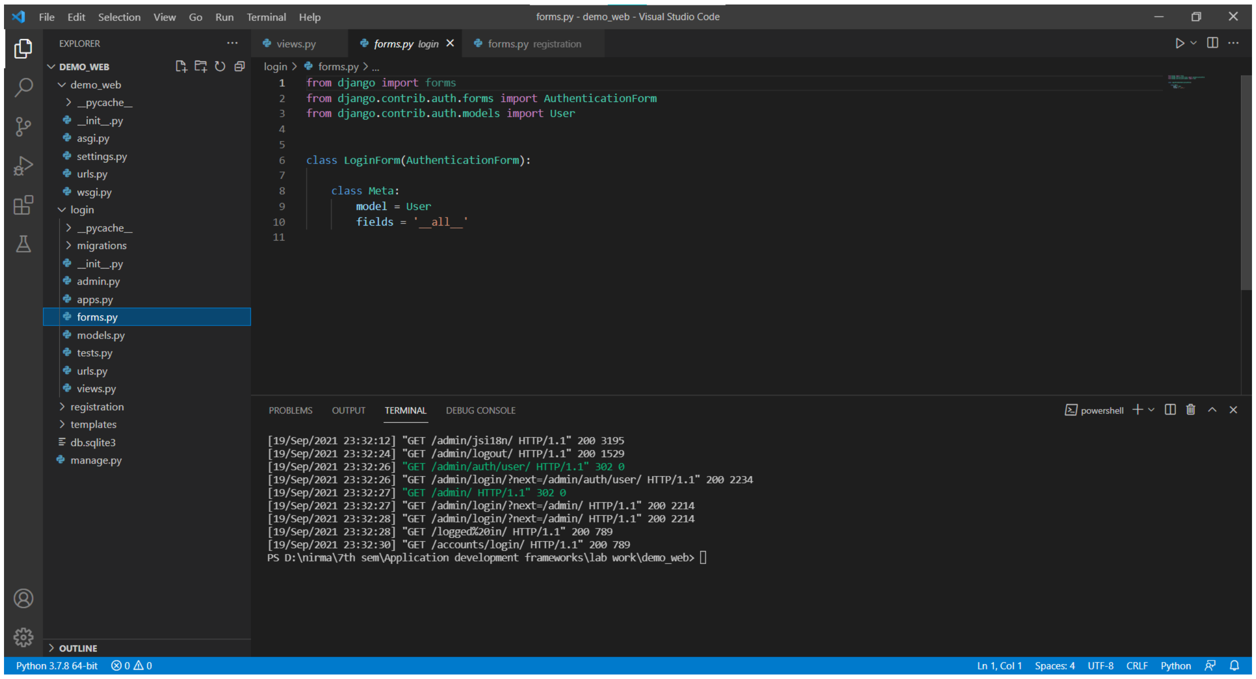


```python
from django.shortcuts import render, redirect
from .forms import RegisterForm
from login.forms import LoginForm

# Create your views here.


def register(response):
    if response.method == 'POST':
        form = RegisterForm(response.POST)
        if form.is_valid():
            form.save()
            form = LoginForm()
            return render(response, 'registration/login.html', {'form': form})
    else:
        form = RegisterForm()
    return render(response, 'registration/register.html', {'form': form})
```

```
[19/Sep/2021 23:32:12] "GET /admin/jsi18n/ HTTP/1.1" 200 3195
[19/Sep/2021 23:32:24] "GET /admin/logout/ HTTP/1.1" 200 1529
[19/Sep/2021 23:32:26] "GET /admin/auth/user/ HTTP/1.1" 302 0
[19/Sep/2021 23:32:26] "GET /admin/login/?next=/admin/auth/user/ HTTP/1.1" 200 2234
[19/Sep/2021 23:32:27] "GET /admin/ HTTP/1.1" 302 0
[19/Sep/2021 23:32:27] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[19/Sep/2021 23:32:28] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

5. After this we make login app by witing:
python manage.py startapp login

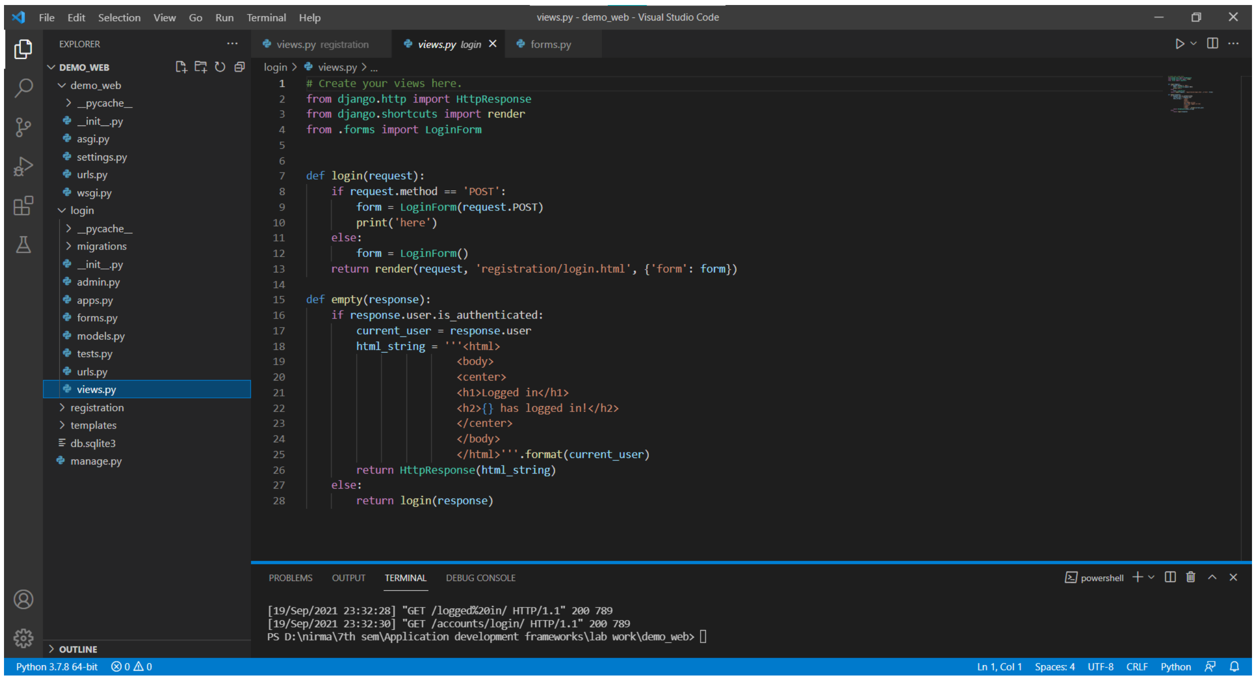
6. The login app looks like this:
   a. No changes have been done to __init__.py, models.py, tests.py, apps.py, and admin.py file
   b. Create a new file names as “forms.py” under loginapp. It creates a form for the user to register. The fields of the form are __all__



```python
from django import forms
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth.models import User


class LoginForm(AuthenticationForm):

    class Meta:
        model = User
        fields = '__all__'

```

```
[19/Sep/2021 23:32:12] "GET /admin/jsi18n/ HTTP/1.1" 200 3195
[19/Sep/2021 23:32:24] "GET /admin/logout/ HTTP/1.1" 200 1529
[19/Sep/2021 23:32:26] "GET /admin/auth/user/ HTTP/1.1" 302 0
[19/Sep/2021 23:32:26] "GET /admin/login/?next=/admin/auth/user/ HTTP/1.1" 200 2234
[19/Sep/2021 23:32:27] "GET /admin/ HTTP/1.1" 302 0
[19/Sep/2021 23:32:27] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[19/Sep/2021 23:32:28] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web> 
```

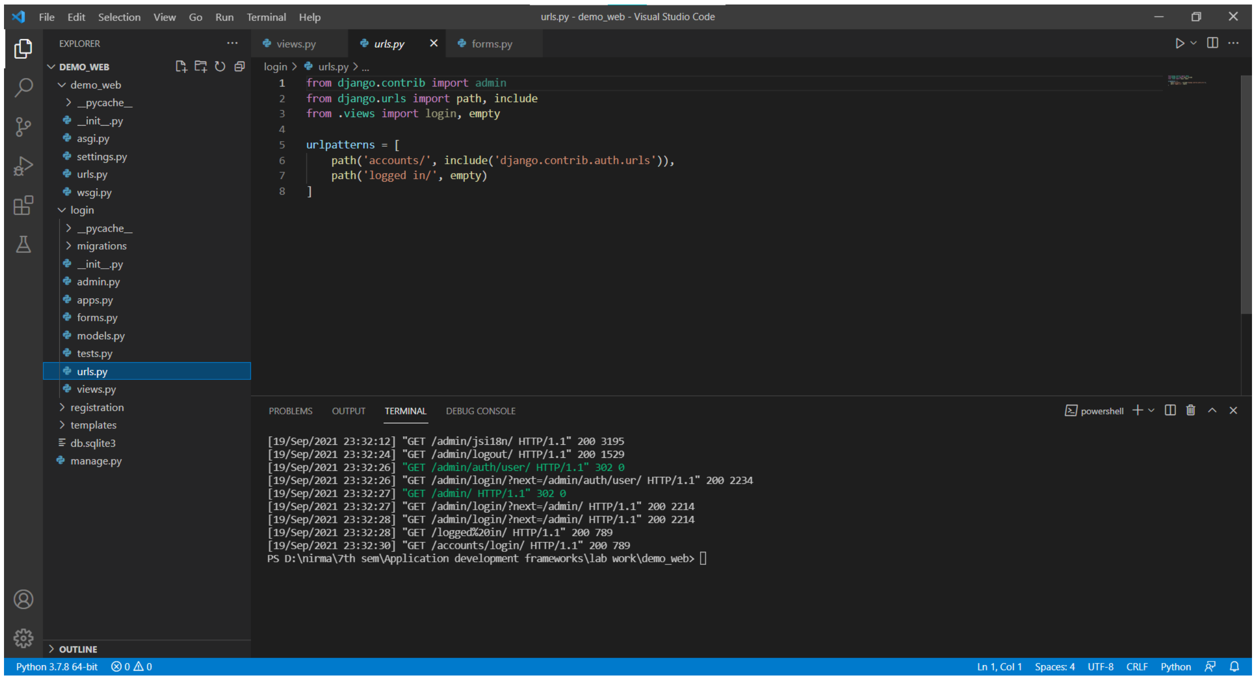
c. views.py file looks like following:
On successful login the html response in the empty function is returned to the server.



```python
# Create your views here.
from django.http import HttpResponse
from django.shortcuts import render
from .forms import LoginForm


def login(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        print('here')
    else:
        form = LoginForm()
    return render(request, 'registration/login.html', {'form': form})

def empty(response):
    if response.user.is_authenticated:
        current_user = response.user
        html_string = '''<html>
                        <body>
                        <center>
                        <h1>Logged in</h1>
                        <h2>{} has logged in!</h2>
                        </center>
                        </body>
                        </html>'''.format(current_user)
        return HttpResponse(html_string)
    else:
        return login(response)
```

```
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web> 
```

d. Create a new file called urls.py and add the following fields



```
from django.contrib import admin
from django.urls import path, include
from .views import login, empty

urlpatterns = [
    path('accounts/', include('django.contrib.auth.urls')),
    path('logged in/', empty)
]
```

```
[19/Sep/2021 23:32:12] "GET /admin/jsi18n/ HTTP/1.1" 200 3195
[19/Sep/2021 23:32:24] "GET /admin/logout/ HTTP/1.1" 200 1529
[19/Sep/2021 23:32:26] "GET /admin/auth/user/ HTTP/1.1" 302 0
[19/Sep/2021 23:32:26] "GET /admin/login/?next=/admin/auth/user/ HTTP/1.1" 200 2234
[19/Sep/2021 23:32:27] "GET /admin/ HTTP/1.1" 302 0
[19/Sep/2021 23:32:27] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[19/Sep/2021 23:32:28] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 2214
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

To use the suth app we need to add project level urls.py file. Here it has been included at accounts/ although any other url pattern is also valid.

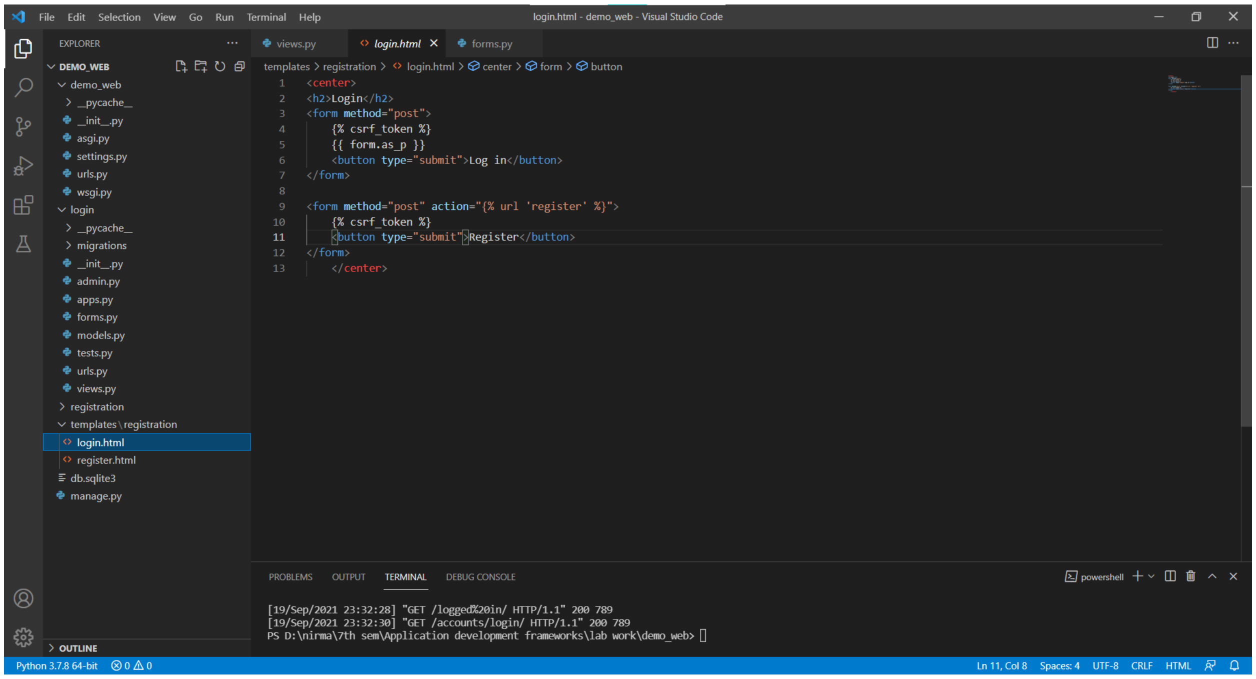
7. Create a new directory as

```
mkdir templates
cd templates
mkdir registration
cd registration
```

Thus now the path is demo_web/templates/registration
In the following directory make 2 new html files as

```
code -r login.html
code -r register.html
```

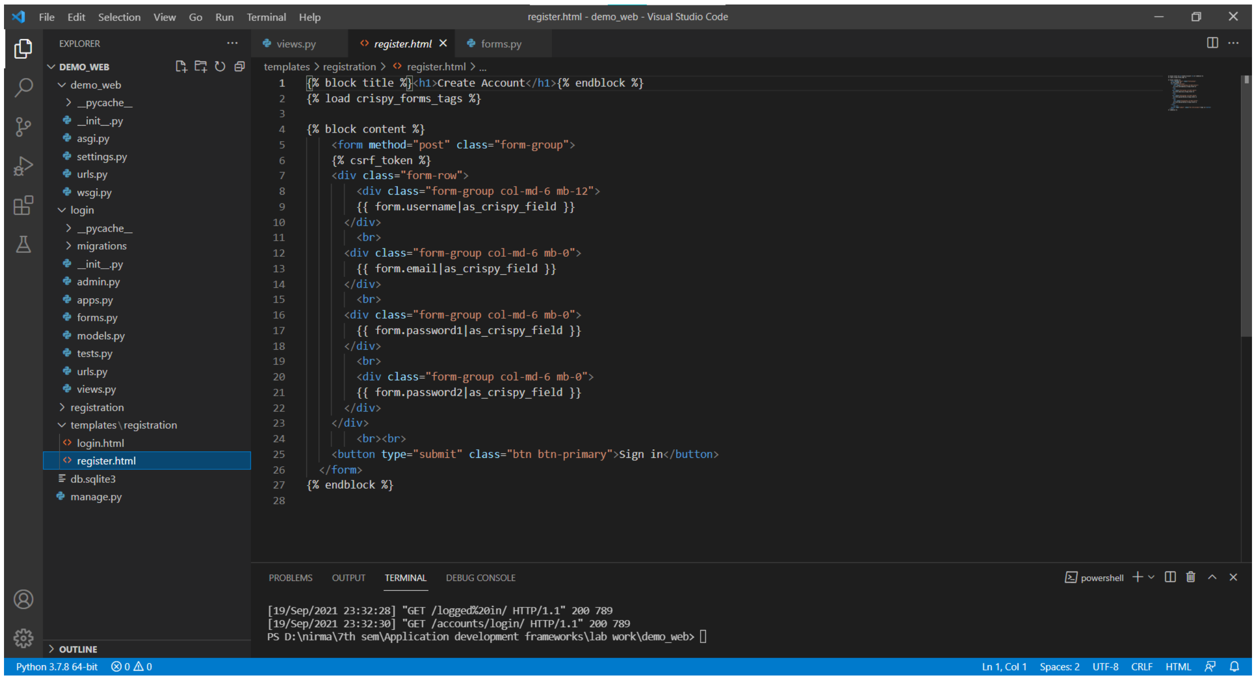
a. login.html file



```
<center>
<h2>Login</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Log in</button>
</form>

<form method="post" action="{% url 'register' %}">
    {% csrf_token %}
    <button type="submit">Register</button>
</form>
    </center>
```

```
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

b. register.html file



```
{% block title %}<h1>Create Account</h1>{% endblock %}
{% load crispy_forms_tags %}

{% block content %}
    <form method="post" class="form-group">
    {% csrf_token %}
    <div class="form-row">
        <div class="form-group col-md-6 mb-12">
        {{ form.username|as_crispy_field }}
      </div>
        <br>
      <div class="form-group col-md-6 mb-0">
        {{ form.email|as_crispy_field }}
      </div>
        <br>
      <div class="form-group col-md-6 mb-0">
        {{ form.password1|as_crispy_field }}
      </div>
        <br>
        <div class="form-group col-md-6 mb-0">
        {{ form.password2|as_crispy_field }}
      </div>
    </div>
        <br><br>
    <button type="submit" class="btn btn-primary">Sign in</button>
  </form>
{% endblock %}
```

```
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

Both of these files are used to write the design of the login and registration forms using crispy forms.

8. Django automatically installs auth app which can be seen in the INSTALLED_APPS list in settings.py. Our next step is to create a superuser by writing the following command in the terminal:

```
python manage.py createsuperuser
```

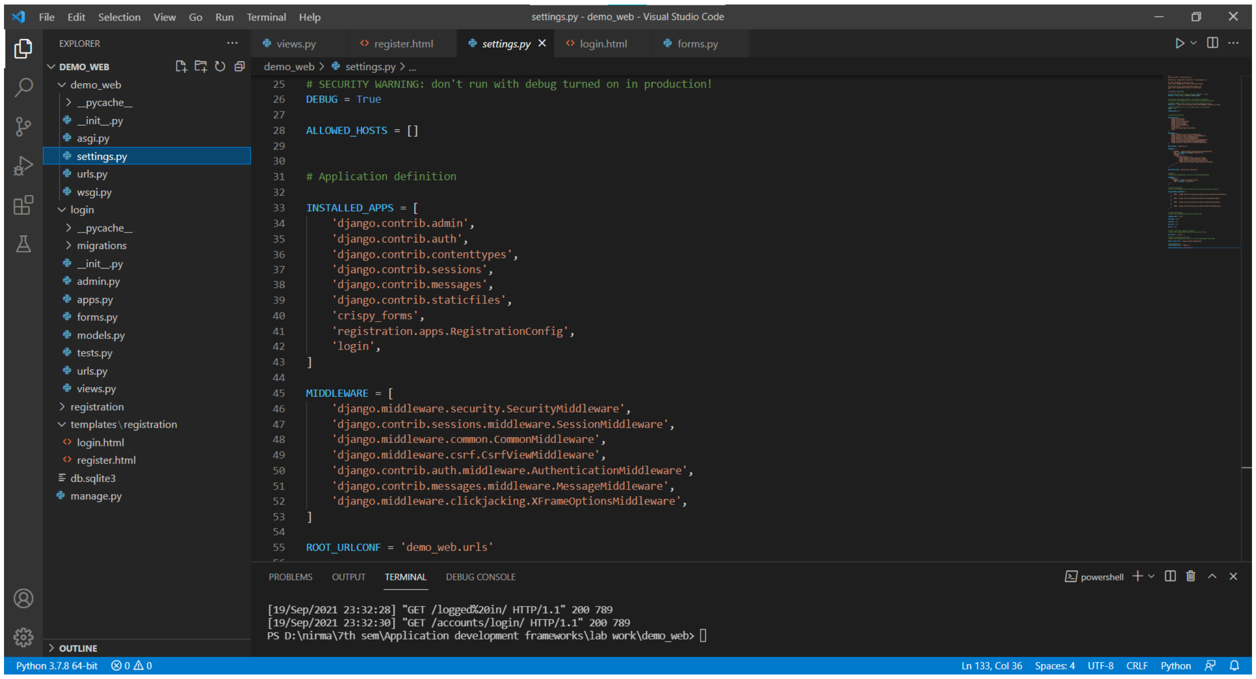
A superuser is a user which has rights to create new users and give them rights.
While creating the same, one will have to enter:

a. Super user name
b. Email address
c. Password
d. Rewrite the password for verification

The superuser login can be done by going to 127.0.0.1:8000/admin

9. The files in the demo_web looks like this:
The only changes has been done in settings.py and urls.py file.

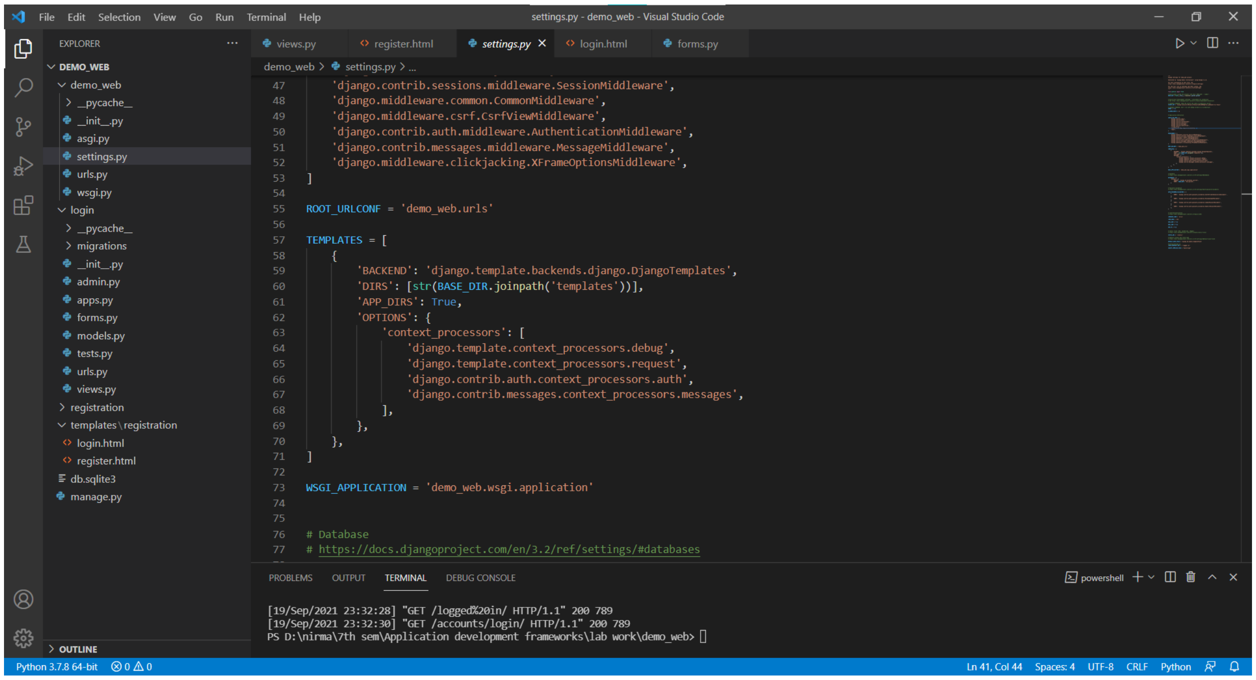
a. settings.py


settings.py - demo_web - Visual Studio Code
demo_web > settings.py > ...

```python
# 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',
    'crispy_forms',
    'registration.apps.RegistrationConfig',
    'login',
]

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 = 'demo_web.urls'
```

```
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

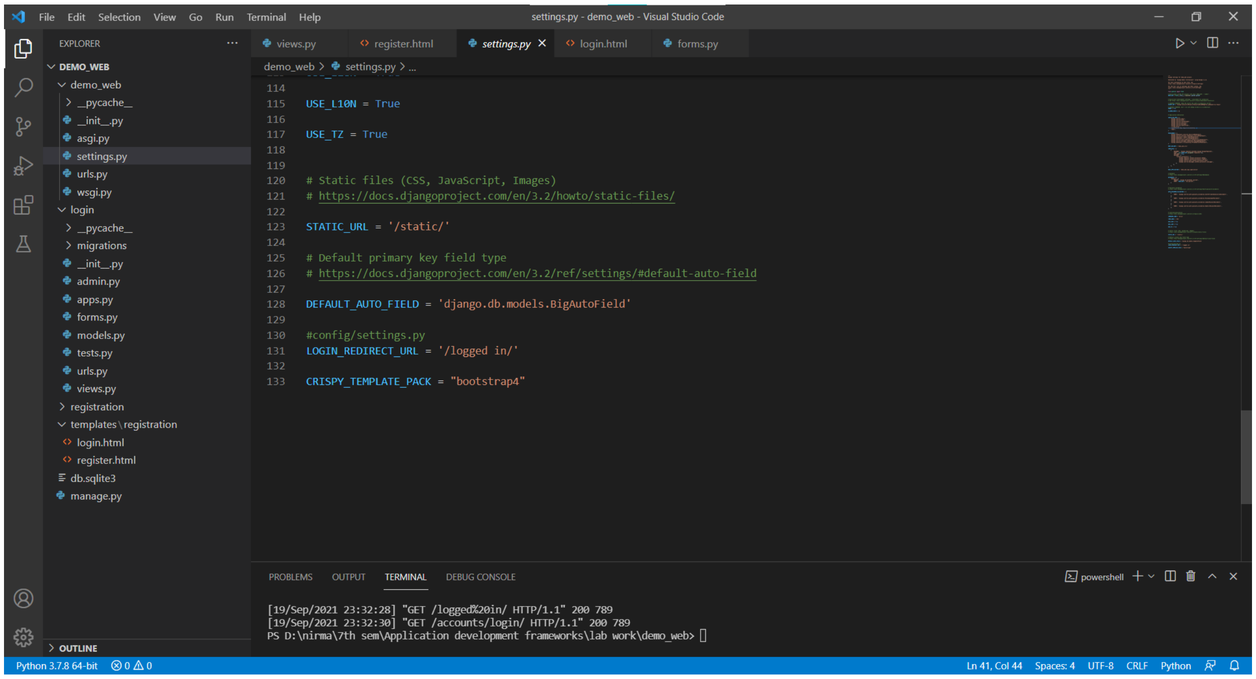
Crispy_forms, login and registration app has been added to installed apps.


settings.py - demo_web - Visual Studio Code
demo_web > settings.py > ...

```python
    '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 = 'demo_web.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [str(BASE_DIR.joinpath('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 = 'demo_web.wsgi.application'


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

```
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

The base address of templates has been added

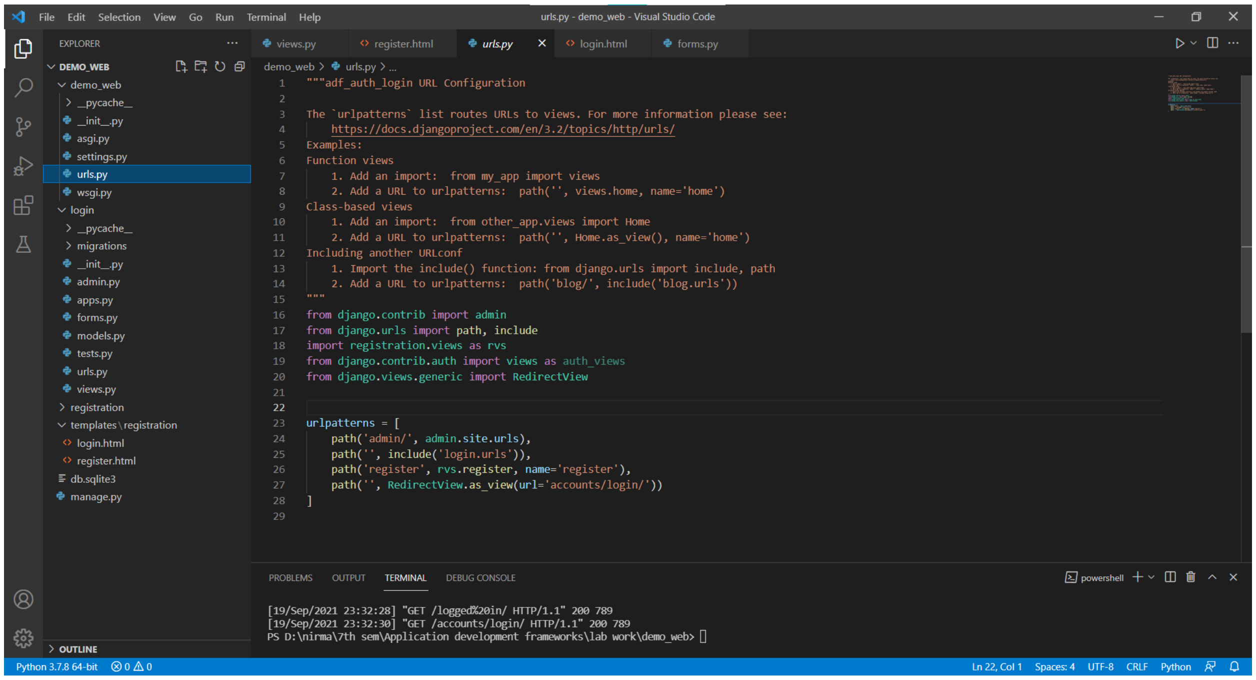


```python
USE_L10N = True

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'

#config/settings.py
LOGIN_REDIRECT_URL = '/logged in/'

CRISPY_TEMPLATE_PACK = "bootstrap4"
```

```
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

At the end of the file, the login redirect url and crispy pack has been added,

b. urls.py



```python
"""adf_auth_login 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
import registration.views as rvs
from django.contrib.auth import views as auth_views
from django.views.generic import RedirectView


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('login.urls')),
    path('register', rvs.register, name='register'),
    path('', RedirectView.as_view(url='accounts/login/'))
]
```

```
[19/Sep/2021 23:32:28] "GET /logged%20in/ HTTP/1.1" 200 789
[19/Sep/2021 23:32:30] "GET /accounts/login/ HTTP/1.1" 200 789
PS D:\nirma\7th sem\Application development frameworks\lab work\demo_web>
```

The different url patterns has been added to the main project url file. Register views are called as rvs. The default page opening is of the login urls.

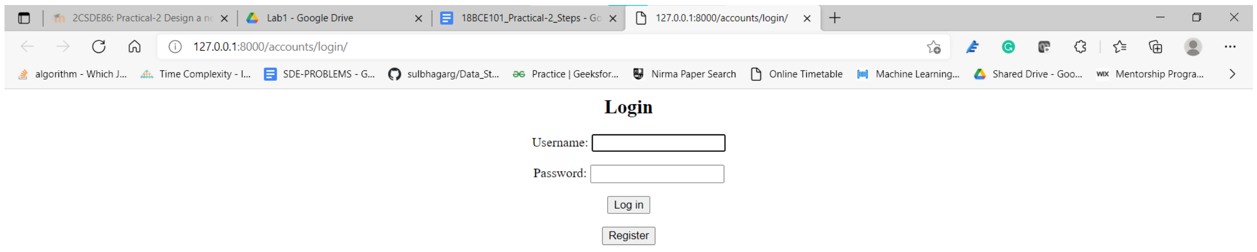
Output:

Run the following commands:

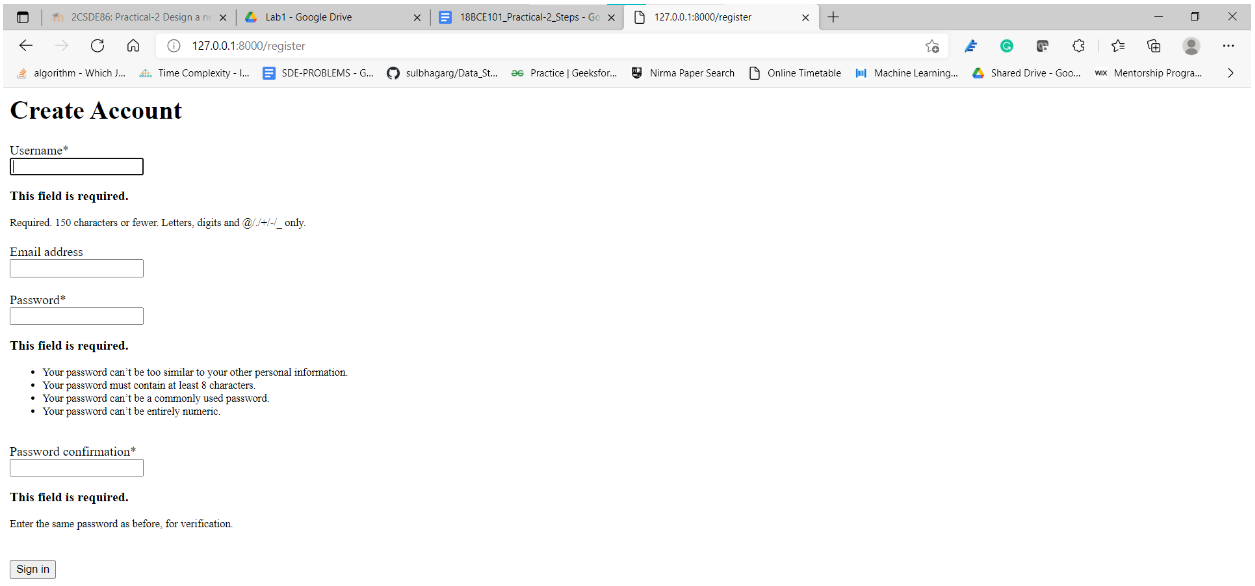
```
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
```

Click on 127.0.0.1:8000

1. Default page that opens is:



2. Since we first have to make a profile, click on "Register" which redirects us to the following page:



Create Account

Username*

This field is required.

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address

Password*

This field is required.

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

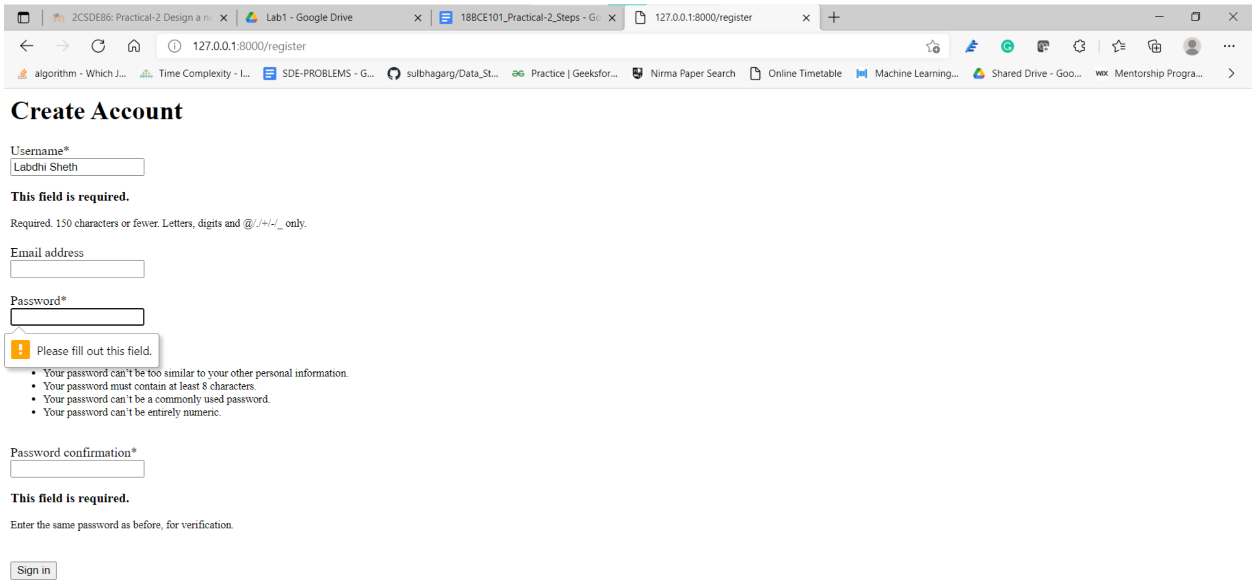
Password confirmation*

This field is required.

Enter the same password as before, for verification.

Sign in

3. Enter the details here:
   On empty cells it prompts



Create Account

Username*

Labdhi Sheth

This field is required.

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address

Password*

Please fill out this field.

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

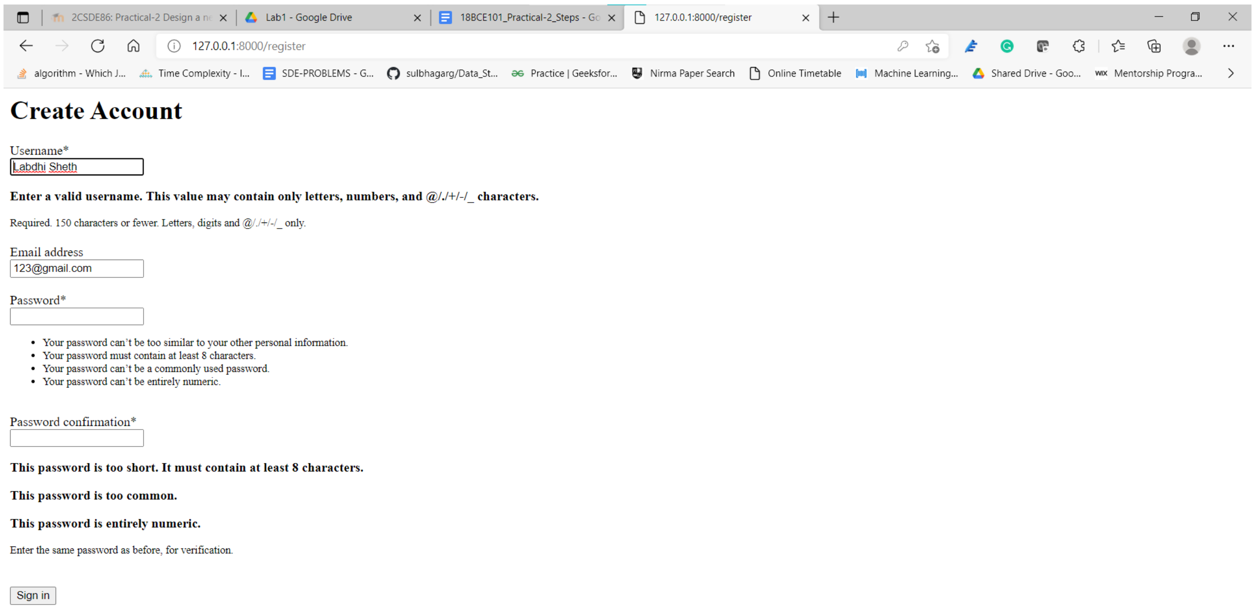
Password confirmation*

This field is required.

Enter the same password as before, for verification.

Sign in

It even validates the password as.


127.0.0.1:8000/register

Create Account

Username*
Labdhi Sheth

Enter a valid username. This value may contain only letters, numbers, and @/./+/-/_ characters.

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email address
123@gmail.com

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

This password is too short. It must contain at least 8 characters.

This password is too common.

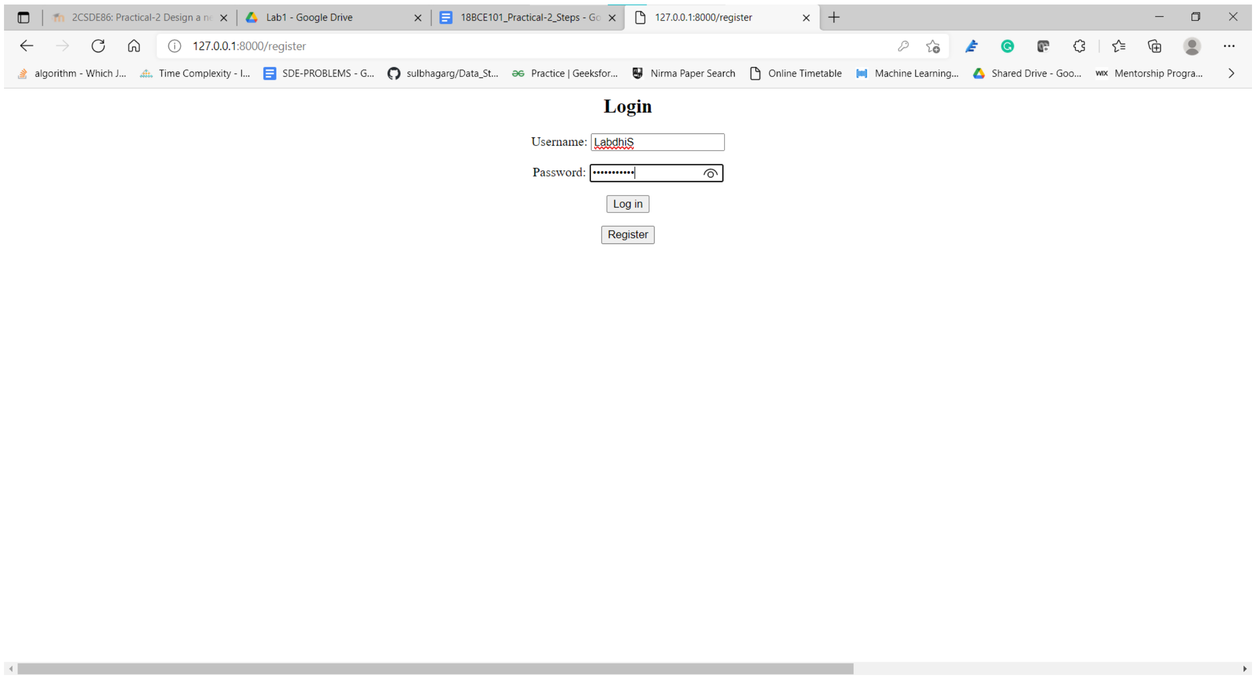
This password is entirely numeric.

Enter the same password as before, for verification.

On successful registration it redirects us back to login page

4. Login page:

   Enter the details

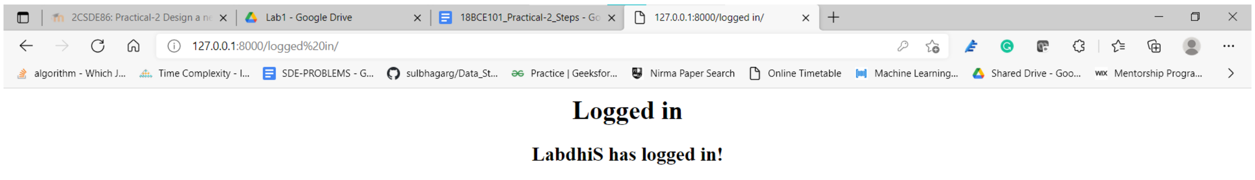

127.0.0.1:8000/register

Login

Username: LabdhiS

Password: ••••••••••

Press the login button

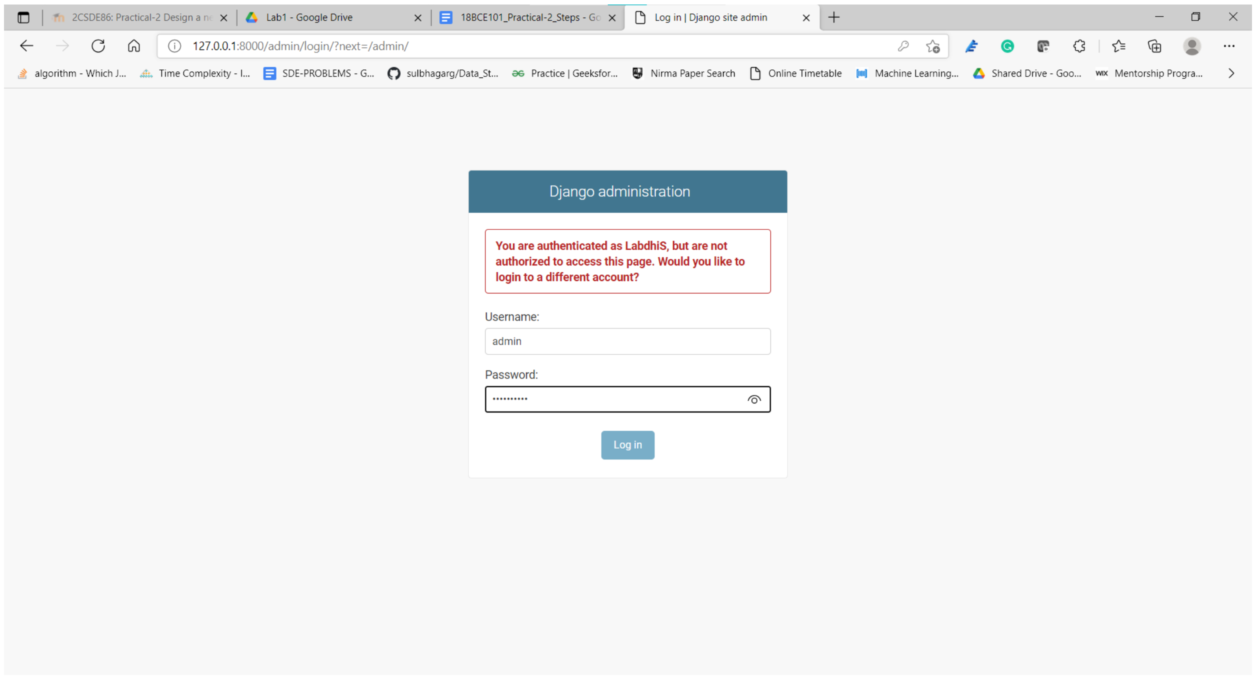


Logged in

LabdhiS has logged in!

5. Perform the super user login to see the users made. For the same move to /admin

   Enter the details that you made while making the super user

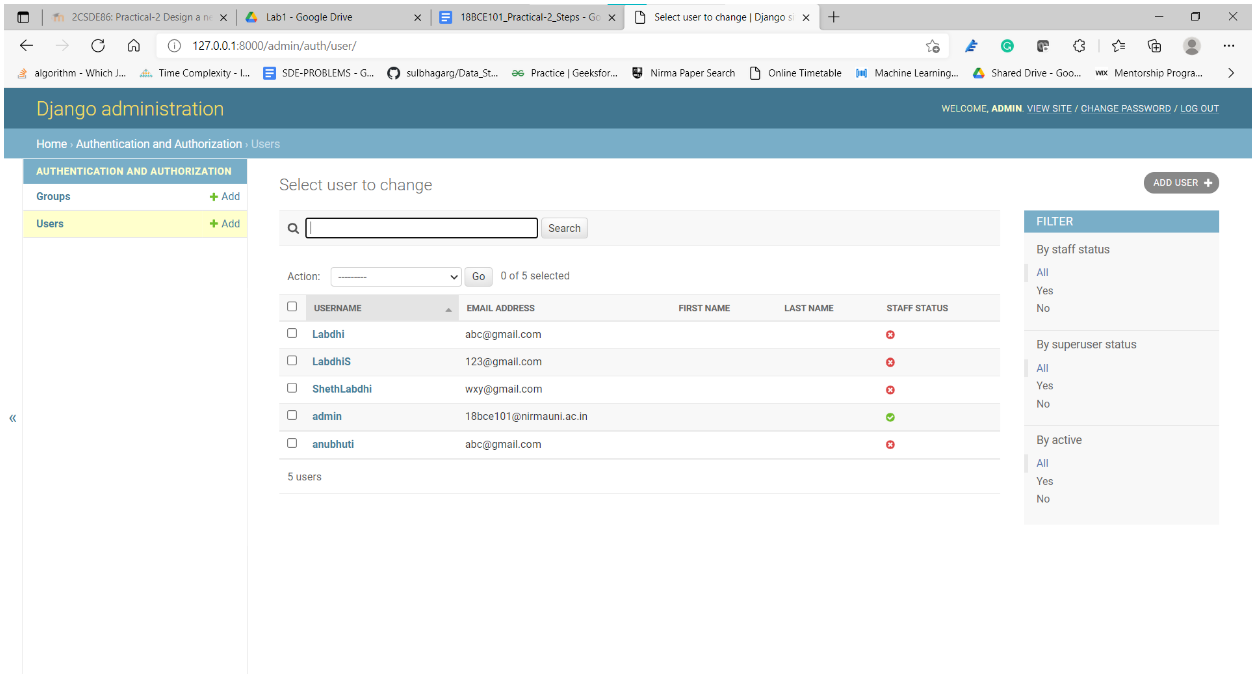


You are authenticated as LabdhiS, but are not authorized to access this page. Would you like to login to a different account?

Username: admin

Password: ••••••••••

   Expand the users section to see the other users made.



Select user to change

0 of 5 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
Labdhi	abc@gmail.com			✗
LabdhiS	123@gmail.com			✗
ShethLabdhi	wxy@gmail.com			✗
admin	18bce101@nirmauni.ac.in			✓
anubhuti	abc@gmail.com			✗

5 users

Thus all the tasks have been successfully performed.

Conclusion:

This practical taught us the crispy form and how using crispy over normal forms is useful in UI and validation. We even learned about the authentication module which is automatically built in by the Django app.