Pigeonesis.

A PROJECT REPORT

Submitted in partial fulfillment for the award of the degree of

B.TECH

in

Computer Science & Engineering

By

Parekh Saloni 19BCB0009
Deshpande Arnav Sunil 19BCB0065

Under the Guidance of

Prof. Dr. Senthilkumar N.

School of Computer Science & Engineering

June 2021

DECLARATION BY THE CANDIDATES

We hereby declare that the project report entitled **“Piegonesis”** submitted by me to VIT University, Vellore in partial fulfillment of the requirement for the award of the degree of **B.Tech.(CSE)** is a record of Jth component of project work carried out by me under the guidance of **Prof. Dr. Senthilkumar N.** We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore [illegible] [illegible]

Date:04 -06-2021 Parekh Saloni, Deshpande Arnav Sunil

INDEX

1. Introduction

1.1 Background

In this time of COVID-19, everyone is away from their friends, family and loved ones. In this challenging period the thing that keeps us all connected is the various ways of communication in which Social Media plays a very big part. We have various social media websites like Twitter,Whatsapp,Instagram,Facebook and a lot more. So we took our inspiration from them and decided to make a social media website from whatall we learned in Internet Web Programming. We decided to name the website as Pigeonesis based upon our history where pigeons were used as a carrier for messages. They were often known as messengers. This website will basically allow people to be a part of their loved ones life by knowing what's going on in their lives through their posts and pictures. They can even like the picture if they find it beautiful and good.

1.2 Objective of the Project

We all know how important Relationship building is. And what's better than a social media website to form connections. Social networking can help people find and build relationships with other thought leaders and influential individuals and brands. It's all about getting to know one another and staying connected to each other and their lives. It is definitely a medium where people can share a part of their life to the people whom they consider family and friends. In this project we have accommodated the features like a person can share posts and the other persons can like the post. Also the person can set up his profile. He can also search for other people's profiles. We have also taken care that if someone posts something unethical or disturbing then that person can also be blocked. Also we can update the posts made. There are also certain restrictions like a person can like the other person's post only once. Also the admin interface that has been created takes this to a whole new level.

1.3 Organization of Report

In this report we have started with mentioning what our project is and why we have chosen this project and how it is useful. Also in the report we have shown our project schedule and how over the course of time how we have managed to complete it. Also we have attached screenshots of how our project website actually looks like. Also Code Snippets have been attached in the report. Along with all this the requirements for our project have also been mentioned in this report along with our final output. Lastly the references as well as conclusions have also been mentioned.

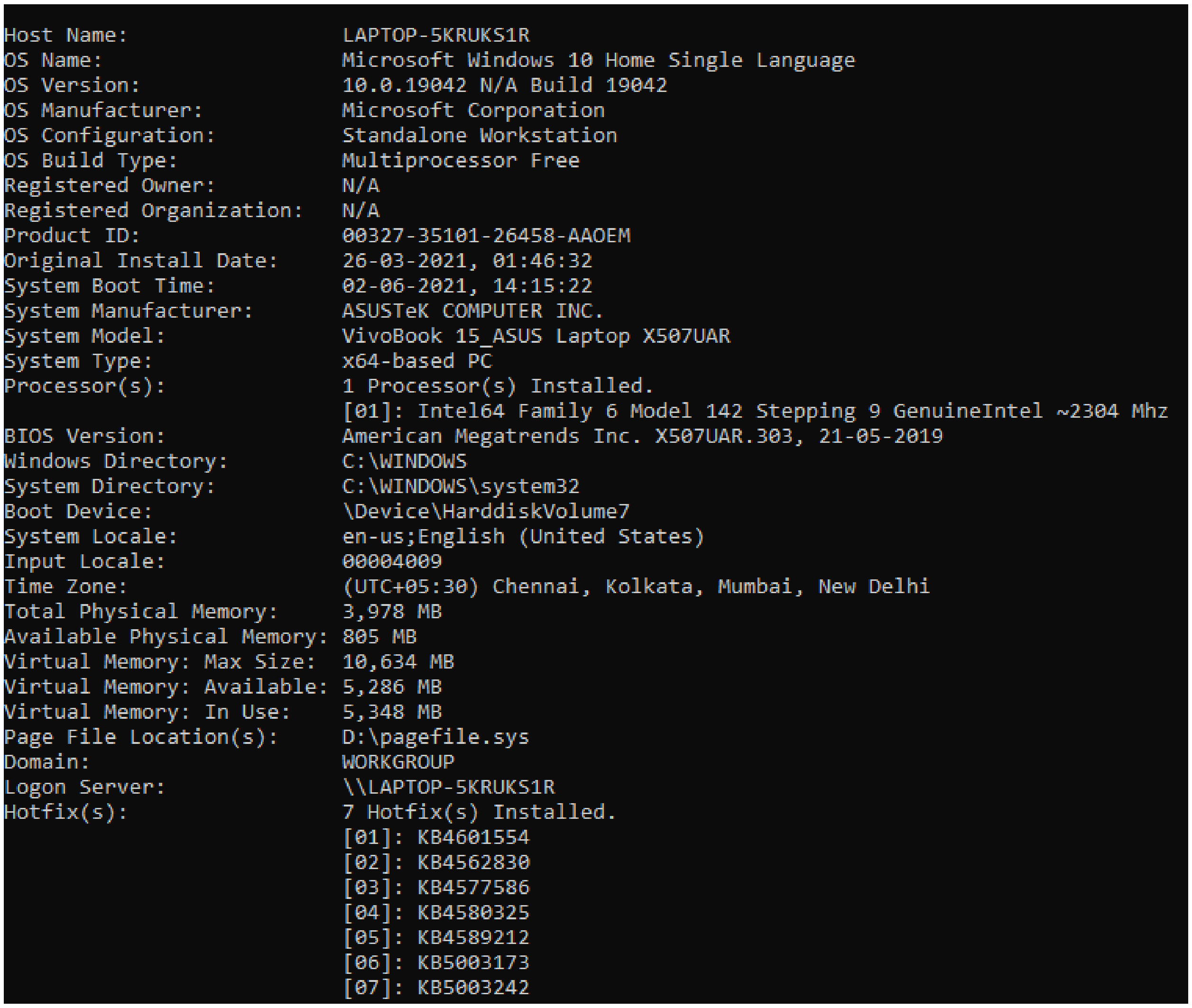
2. Overview and Planning

2.1 Proposed System Overview

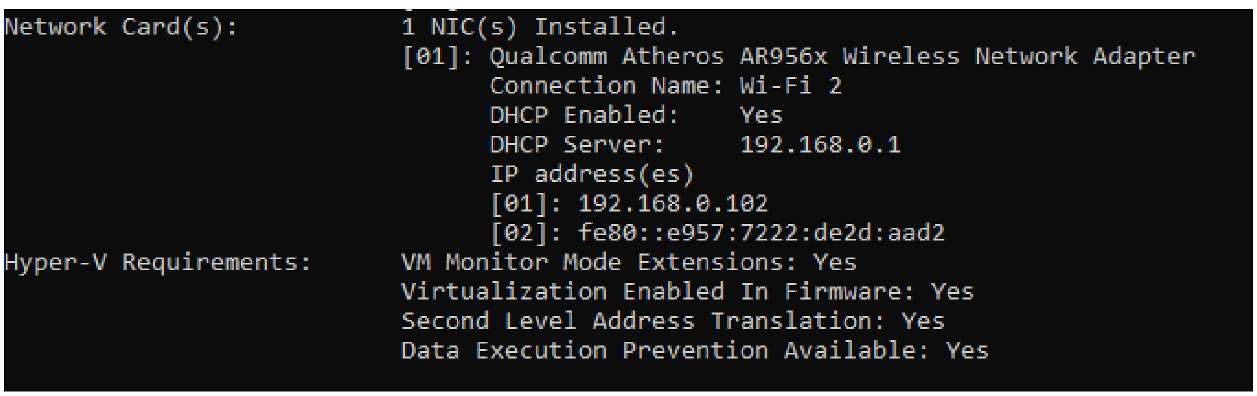
As proposed in Review - 0, we decided to get done with the solid HTML CSS codes till Review-1 along with incorporating all the specifications expected in Review-1. Then during the time between Review-1 and Review-2 we decided to convert the solid HTML CSS codes to Django framework and also validate all the forms and start with the database. And for the Review-3 we decided to host the website and link the frontend with the backend and implement the search option and show the complete implementation of our project in Review-3.

2.2 Hardware Requirements

Hardware Requirements constitute any working OS, a functioning Boot Device, and sufficient RAM (sufficient enough for the browser to run).
Here are the Hardware Specifications of the System, that we ran the Project smoothly on:



```
Host Name:                 LAPTOP-5KRUKS1R
OS Name:                   Microsoft Windows 10 Home Single Language
OS Version:                10.0.19042 N/A Build 19042
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          N/A
Registered Organization:   N/A
Product ID:                00327-35101-26458-AAOEM
Original Install Date:     26-03-2021, 01:46:32
System Boot Time:          02-06-2021, 14:15:22
System Manufacturer:       ASUSTeK COMPUTER INC.
System Model:              VivoBook 15_ASUS Laptop X507UAR
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 142 Stepping 9 GenuineIntel ~2304 Mhz
BIOS Version:              American Megatrends Inc. X507UAR.303, 21-05-2019
Windows Directory:         C:\WINDOWS
System Directory:          C:\WINDOWS\system32
Boot Device:               \Device\HarddiskVolume7
System Locale:             en-us;English (United States)
Input Locale:              00004009
Time Zone:                 (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi
Total Physical Memory:     3,978 MB
Available Physical Memory: 805 MB
Virtual Memory: Max Size:  10,634 MB
Virtual Memory: Available: 5,286 MB
Virtual Memory: In Use:    5,348 MB
Page File Location(s):     D:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\LAPTOP-5KRUKS1R
Hotfix(s):                 7 Hotfix(s) Installed.
                           [01]: KB4601554
                           [02]: KB4562830
                           [03]: KB4577586
                           [04]: KB4580325
                           [05]: KB4589212
                           [06]: KB5003173
                           [07]: KB5003242
```



```
Network Card(s):           1 NIC(s) Installed.
                           [01]: Qualcomm Atheros AR956x Wireless Network Adapter
                                 Connection Name: Wi-Fi 2
                                 DHCP Enabled:    Yes
                                 DHCP Server:     192.168.0.1
                                 IP address(es)
                                 [01]: 192.168.0.102
                                 [02]: fe80::e957:7222:de2d:aad2
Hyper-V Requirements:      VM Monitor Mode Extensions: Yes
                           Virtualization Enabled In Firmware: Yes
                           Second Level Address Translation: Yes
                           Data Execution Prevention Available: Yes
```

2.3 Software Requirements

Software Requirements for the Project are:

1.Virtual Environment - pipenv.

Can be installed by the command "pipenv install"

Can be activated by the command "pipenv shell"

2.Python 3 installed.

3.Django 3 installed.

4.Dotenv Installed.

5.To host the Project on the Local Server:

Go to the Source Folder and type -

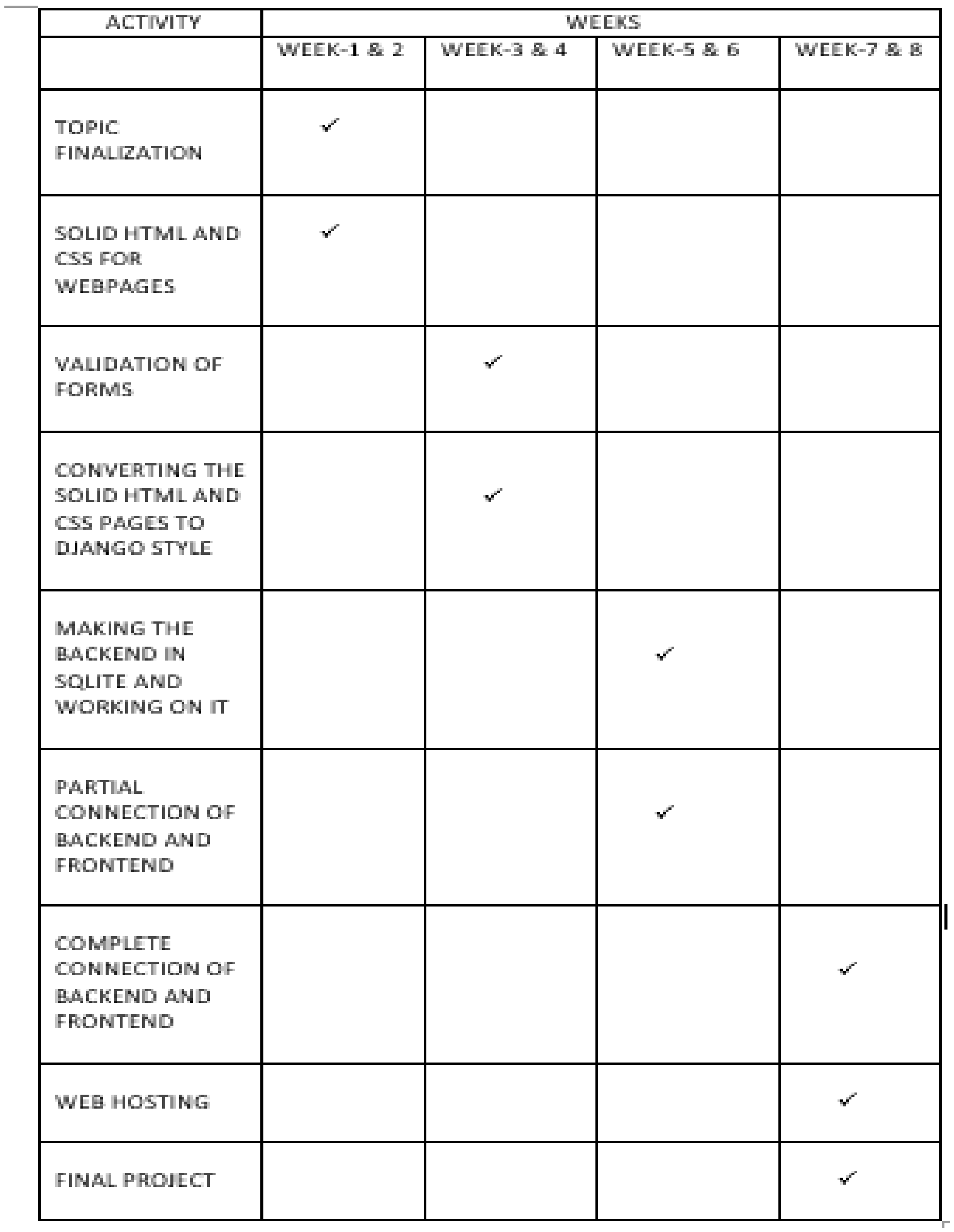
"python manage.py makemigrations"

"python manage.py migrate"

"python manage.py runserver"

6.The Site can be enjoyed at your Localhost, on your desired browser.

2.4 Project Schedule (Gantt chart)



ACTIVITY	WEEKS			
	WEEK-1 & 2	WEEK-3 & 4	WEEK-5 & 6	WEEK-7 & 8
TOPIC FINALIZATION	✓			
SOLID HTML AND CSS FOR WEBPAGES	✓			
VALIDATION OF FORMS		✓		
CONVERTING THE SOLID HTML AND CSS PAGES TO DJANGO STYLE		✓		
MAKING THE BACKEND IN SQLITE AND WORKING ON IT			✓	
PARTIAL CONNECTION OF BACKEND AND FRONTEND			✓	
COMPLETE CONNECTION OF BACKEND AND FRONTEND				✓
WEB HOSTING				✓
FINAL PROJECT				✓

3. System Implementation

3.1 Code and/or Architecture Development

The Entire Code along with dependencies, can be accessed at **our Github Link**.

A few important pages have been pasted below:

(a) Few HTML files:

search.html

```
"""

Django settings for facebook project.


Generated by 'django-admin startproject' using Django 2.2.5.


For more information on this file, see

https://docs.djangoproject.com/en/2.2/topics/settings/


For the full list of settings and their values, see

https://docs.djangoproject.com/en/2.2/ref/settings/

"""


import os

from os.path import join, dirname

from dotenv import load_dotenv
```

```
dotenv_path = join(dirname(__file__), '.env')

load_dotenv(dotenv_path)


# Build paths inside the project like this: os.path.join(BASE_DIR, ...)

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))




# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/


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

# SECRET_KEY = os.environ.get("SECRET_KEY")

SECRET_KEY = "asdfasdfasdfasdfs"

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

DEBUG = True


ALLOWED_HOSTS = ["rohanjnr.pythonanywhere.com", "localhost", "127.0.0.1"]




# Application definition


INSTALLED_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',
    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',


    'facebook.apps.accounts',

    'facebook.apps.posts',

    'facebook.apps.api',


    # third party

    'crispy_forms'

]


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 = 'facebook.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, "facebook", "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 = 'facebook.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.2/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/2.2/ref/settings/#auth-password-validato
rs



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/2.2/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/2.2/howto/static-files/

STATIC_ROOT = 'static_dep'

STATIC_URL = '/static/'

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "facebook", "static")

]



MEDIA_ROOT = os.path.join(BASE_DIR, "facebook", "media")

MEDIA_URL = "/media/"



LOGIN_URL = "login"

LOGIN_REDIRECT_URL = "profile"
```

login.html

```
{% extends "base.html" %}

{% load static %}

{% block head %}

<meta charset="UTF-8">

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

<link
href="https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,200;1,3
00&display=swap" rel="stylesheet">

<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.m
in.css">

<link
href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap"
rel="stylesheet">

<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Tangerine">
```

```
<link rel="stylesheet" href="{% static 'css/login-style.css' %}">

{% endblock %}

{% block content %}


<section>

    <div class="form-container">

        <div class="error-box">Display error here.</div>

        <div class="allPaper">

            <div class="signup-form">

                <div class="exit">

                    <div class="exit1"></div>

                    <div class="exit2"></div>

                </div>

                <form autocomplete="off" action="{% url 'register' %}"
method='POST'>

                    {% csrf_token %}

                    <div class="headding">sign up</div>

                    <div class="username">

                        <i class="fas fa-user"></i>

                        <span class="single-form">

                            {{ form.username }}

                            <label for="s-name"><span>Name</span></label>

                        </span>

                    </div>

                    <div class="email">
```

```
                        <i class="fas fa-at"></i>

                        <span class="single-form">

                            {{ form.email }}

                        <label for="s-email"><span>Email</span></label>

                        </span>

                    </div>

                    <div class="password">

                        <i class="fas fa-key"></i>

                        <span class="single-form">

                            {{ form.password1 }}

                        <label
for="s-password"><span>Password</span></label>

                        </span>

                    </div>

                    <div class="password">

                        <i class="fas fa-key"></i>

                        <span class="single-form">

                            {{ form.password2 }}

                        <label for="s-password"><span> Confirm
Password</span></label>

                        </span>

                    </div>

                    <div class="submit-btn">

                        <button>Sign Up</button>

                    </div>
```

```
                </form>

                <span class="alternate">Already have an account? <button
id="login">login</button></span>

                 <span class="alternate">Have Some Doubts? <button
id="login"> <a href="contactus.html"> Contact us </a></button></span>

            </div>

            <div class="paper">

                <div class="start">

                    <div class="start-up">

                        <div class="end"></div>

                    </div>

                    <div class="start-down">

                        <div class="end"></div>

                    </div>

                </div>

            </div>

            <div class="login-form" id="login-form">

                <div class="exit">

                    <div class="exit1"></div>

                    <div class="exit2"></div>

                </div>

                <form autocomplete="off" method='POST'>

                    {% csrf_token %}

                    <div class="headding">login</div>

                    <div class="email">
```

```html
                        <i class="fas fa-at"></i>
                        <span class="single-form">
                        <input type="text" name="username">
                        <label for="s-email"><span>Username</span></label>
                        </span>
                    </div>
                    <div class="password">
                        <i class="fas fa-key"></i>
                        <span class="single-form">
                        <!-- <input required type="password"
name="s-email"> -->
                        <input type="password" name="password">
                        <label for="s-email"><span>Password</span></label>
                        </span>
                    </div>
                    <div class="submit-btn">
                        <button>Log In</button>
                    </div>
                    <div class="extra">
                        <span class="alternate">Don't have an account?
<button id="signup">signup</button></span>
                    </div>
                </form>
            </div>
        </div>
```

```
    </div>

</section>

<script>


document.querySelector('.allPaper').classList.add('anim');


setTimeout(function(){document.querySelector('.paper').style.display="none
"},1200);

    document.querySelector('#login').addEventListener("click",(e)=>{

        e.preventDefault();

        document.querySelector(".login-form").style.transition = "opacity
0s ease 0s";


setTimeout(()=>{document.querySelector('.login-form').style.opacity =
"1";},0)

        document.querySelector(".allPaper").style.animation="cardFlipB 2s
ease forwards";



    })

    document.querySelector('#signup').addEventListener("click",(e)=>{

        e.preventDefault();

         document.querySelector(".allPaper").style.animation="cardFlip 2s
ease forwards";


setTimeout(()=>{document.querySelector('.login-form').style.opacity="0";},
700)

        })

    for(let i=0;i<2;i++)
    document.querySelectorAll(".exit")[i].addEventListener("click",()=>{

        console.log("exit")

        window.location.href="./";

    })


</script>

{% endblock %}
```

(b)Few CSS files:

about.css

```
@import
url('https://fonts.googleapis.com/css?family=ZCOOL+QingKe+HuangYou&display
=swap');

*{

    margin: 0;

    padding: 0;

}

body{

    background-color: #5CDB95;

    font-family: 'ZCOOL QingKe HuangYou', cursive;

}

main{

    background: orange;

    padding: 200px 200px;

    padding-top: 50px;
```

```css
    margin-top: -50px;

    height: 17rem;

    border-bottom: orangered 3px solid;

    text-align: center;

}

main p{

    margin: 10px;

    padding: 5px;

    font-size: 30px;

}

main img{

    width: 250px;

    height: 250px;

    border-radius: 50%;

}

section{

    text-align: center;

    margin: 10px;

    padding: 5px;

}

section h2{

    font-size: 40px;

}

.project_img img{
    width: 500px;

    height: auto;

    padding: 0;

    margin: 0;

}

.card{

    background: linear-gradient(45deg, orangered, yellow);

    margin: 40px 20%;

    padding: 0;

    display: grid;

    grid-template-columns: 1fr 1fr;

    grid-gap: 20px;

}

section h3{

    font-size: 32px;

    color: grey;

    padding: 10px;

}

.description{

    text-align: left;

}

ul{

    list-style: square;

    color: white;
```

```
}

a{

    text-decoration: none;

    color: white;

}

.view{

    font-size: 30px;

}

.contact ul{

    margin: 10px;

    display: grid;

    grid-template-columns: 1fr 1fr 1fr;

    font-size: 25px;

    list-style: none;

    color: black;

    grid-column-gap: -30px;

}
```

contactus.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500
;600;700;800;900&display=swap');

*

{

    margin:0;
    padding:0;

    box-sizing: border-box;

    font-family:'poppins',sans-serif;

    /*color:white;*/

}

.contact

{

    position: relative;

    min-height:100vh;

    padding: 50px 100px;

    display:flex;

    justify-content:center;

    align-items:center;

    flex-direction:column;

    /*background : url("bgimg1.jpg");*/

    background-color: black;

    background-size:cover;

}

.contact .content

{

    max-width:800px;

    text-align:center;

}

.contact .content h2
```

```
{

    font-size: 36px;

    font-weight: 500;

    color: #fff;

}

.contact .content p

{

    font-weight: 300;

    color: #fff;

}

.container

{

    width:100%;

    display : flex;

    justify-content:center;

    align-items:center;

    margin-top:30px;

}

.container .contactInfo

{

    width:50%;

    display:flex;

    flex-direction:column;

}
```

```
.container .contactInfo .box

{

    position:relative;

    padding:20px 0;

    display:flex;

}

.container .contactInfo .box .icon

{

    min-width:60px;

    height:60px;

    background-color:#fff;

    display:flex;

    justify-content:center;

    align-items:center;

    border-radius:50%;

    font-size:22px;

}

.container .contactInfo .box .text

{

    display:flex;

    margin-left:20px;

    font-size:16px;

    color:#fff;

    flex-direction: column;
    font-weight: 300;

}

.container .contactInfo .box .text h3

{

    font-weight: 500;

    color : #98AFC7;

}

.contactForm

{

    width: 40%;

    padding: 40px;

    background: #98AFC7;

}

.contactForm h2

{

    font-size: 30px;

    color:#333;

    font-weight:500;

}

.contactForm .inputBox

{

    position: relative;

    width: 100%;

    margin-top:10px;
```

```
}

.contactForm .inputBox input,

.contactForm .inputBox textarea

{

    width:100%;

    padding:5px 0;

    font-size: 16px;

    margin: 10px 0;

    border: none;

    border-bottom: 2px solid #333;

    outline:none;

    resize: none;

}

.contactForm .inputBox span

{

    position: absolute;

    left:0;

    padding: 5px 0;

    font-size: 16px;

    margin: 10px 0;

    pointer-events:none;

    transition: 0.5s;

    color: #666;

}
```

```
.contactForm .inputBox input:focus ~ span,

.contactForm .inputBox input:valid ~ span,

.contactForm .inputBox textarea:focus ~ span,

.contactForm .inputBox textarea:valid ~ span

{

    color: #e91e63;

    font-size: 12px;

    transform: translateY(-20px);

}

.contactForm .inputBox input[type = "submit"]

{

    width: 100px;

    background: black;

    color:#fff;

    border:none;

    cursor:pointer;

    padding:10px;

    font-size:18px;

}



@media (max-width: 991px)

{

    .contact

    {
        padding: 50px;

    }

    .container

    {

        flex-direction: column;

    }

    .container .contactInfo

    {

        margin-bottom: 40px;

    }

    .container .contactInfo,

    .contactForm

    {

        width: 100%;

    }

}
```

profile.css

```
.master{

    display: grid;

    grid-template-columns: 1fr 1fr 4fr;

    height: 100%;

    margin: 0;

    padding: 0;
```

```
}

.posts-container{

    font-size: 17px;

    text-align: left;

    max-height: 100%;

    overflow-y: auto;

    margin-right: 30px;

}

.heading{

    font-size: 4rem;

}

.posts-container hr{

    width: 70%;

}

.profile-posts{

    margin: 5px;

    padding: 20px;

    border-radius: 10px;

    border: 2px #E2DEDE solid;

    margin-right: 20px;

    -webkit-box-shadow: inset 0px 0px 50px -10px #000000;

    box-shadow: inset 0px 0px 50px -10px #000000;

}

.user-data{
    padding-left: 5px;

    font-weight: 500;

}

.post-button{

    margin: 10px;

}

.post-button button{

    border-radius: 10px;

    font-size: 20px;

    padding: 5px;

}

.active button{

    background-color: orange;

    border-color: orange;

}

.date{

    font-size: 13px;

}

a{

    text-decoration: none;

    color: #FF5D00;



}
```

(c) Backend / Database Files:

Tables are created and stored in models.py file.

manage.py

```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'facebook.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

settings.py

```
"""

Django settings for facebook project.


Generated by 'django-admin startproject' using Django 2.2.5.


For more information on this file, see

https://docs.djangoproject.com/en/2.2/topics/settings/


For the full list of settings and their values, see

https://docs.djangoproject.com/en/2.2/ref/settings/

"""


import os

from os.path import join, dirname

from dotenv import load_dotenv




dotenv_path = join(dirname(__file__), '.env')

load_dotenv(dotenv_path)


# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
```

```python
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
# SECRET_KEY = os.environ.get("SECRET_KEY")
SECRET_KEY = "asdfasdfasdfasdfs"
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ["rohanjnr.pythonanywhere.com", "localhost", "127.0.0.1"]


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'facebook.apps.accounts',

    'facebook.apps.posts',

    'facebook.apps.api',


    # third party

    'crispy_forms'

]


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 = 'facebook.urls'


TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, "facebook", "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 = 'facebook.wsgi.application'



# Database

# https://docs.djangoproject.com/en/2.2/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/2.2/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/2.2/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/2.2/howto/static-files/

STATIC_ROOT = 'static_dep'

STATIC_URL = '/static/'

STATICFILES_DIRS = [

    os.path.join(BASE_DIR, "facebook", "static")

]

MEDIA_ROOT = os.path.join(BASE_DIR, "facebook", "media")

MEDIA_URL = "/media/"

LOGIN_URL = "login"

LOGIN_REDIRECT_URL = "profile"
```

POSTS :

forms.py

```
from django import forms

from .models import Post, Comment


from crispy_forms.helper import FormHelper

from crispy_forms.layout import Layout, Field




class PostForm(forms.ModelForm):


    class Meta:

        model = Post


        fields = ["title", "description", "img"]




class CommentForm(forms.ModelForm):


    class Meta:

        model = Comment

        fields = ["comment"]


    helper = FormHelper()
```

models.py

```
from django.db import models

from django.contrib.auth.models import User

from django.db.models import Count

from django.db.models.signals import post_delete


from .signals import delete_pic


from PIL import Image




class PostManager(models.Manager):


    def get_posts(self, status):

        return
self.get_queryset().annotate(num_comments=Count("comment")).filter(archive
d=status)


class Post(models.Model):

    user = models.ForeignKey(User, on_delete=models.CASCADE)

    title = models.CharField(max_length=150)

    created = models.DateTimeField(auto_now_add=True, auto_now=False)

    updated = models.DateTimeField(auto_now_add=False, auto_now=True)

    img = models.ImageField(upload_to="post_imgs", blank=True, null=True)
    description = models.TextField()

    likes = models.ManyToManyField(User, related_name="likes")

    archived = models.BooleanField(default=False)


    objects = PostManager()


    def __str__(self):

        return f"{self.user} : {self.title}"


    class Meta:

        ordering = ['-created']


    def save(self):

        super().save()

        if self.img:

            img = Image.open(self.img.path)


            if img.height > 300 or img.width > 300:

                output_size = (600, 600)

                img.thumbnail(output_size)

                img.save(self.img.path)



class Comment(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    post = models.ForeignKey(Post, on_delete=models.CASCADE)

    commented_time = models.DateTimeField(auto_now_add=True,
auto_now=False)

    comment = models.CharField(max_length=250)




post_delete.connect(delete_pic, Post)
```

urls.py

```
from django.urls import path

from . import views

from django.contrib.auth import views as auth_views



urlpatterns = [

    path("add-post/", views.add_post_view, name="add-post"),

    path("posts/", views.display_posts_view, name="display-posts"),

    path("posts/<int:pk>", views.detail_post_view, name="detail-post"),

    path("posts/<int:pk>/add-comment", views.comments_view,
name="add-comment"),

    path("posts/<int:pk>/add-like/<str:destination>", views.like_view,
name="like"),

    path("posts/<int:pk>/delete-post", views.delete_post_view,
name="delete-post"),

    path("posts/<int:pk>/update-post", views.update_post_view,
name="update-post"),
    path("posts/<int:pk>/archive", views.archive_post_view,
name="archive"),

    path("posts/<int:pk>/un-archive", views.unarchive_post_view,
name="un-archive")

]
```

views.py

```
from django.shortcuts import render, redirect

from django.contrib.auth.decorators import login_required

from django.http import HttpResponseRedirect

from django.db.models import Q

from django.contrib import messages

from facebook.apps.accounts.models import Block



from .models import Post

from .forms import PostForm, CommentForm




@login_required

def add_post_view(request):

    if request.method == "POST":

        form = PostForm(request.POST, request.FILES)

        if form.is_valid():

            obj = form.save(commit=False)
            obj.user = request.user
            # obj.img.save()
            obj.save()
            messages.success(request, "Post has been created.")
            return redirect("/posts")
    form = PostForm()
    template_name = "posts/add_post.html"
    context = {
        "form":form
    }
    return render(request, template_name, context)


@login_required
def display_posts_view(request):
    blocked = Block.objects.filter(blocked_by =
request.user).values_list('blocked').distinct()
    blocked_by = Block.objects.filter(blocked =
request.user).values_list('blocked_by').distinct()
    if (blocked.exists()) and (blocked_by.exists()):
        allBlocked = blocked | blocked_by
    elif blocked.exists():
        allBlocked = blocked
    elif blocked_by.exists():
        allBlocked = blocked_by
    else:
        allBlocked = []

    print(allBlocked)

    posts = Post.objects.get_posts(False).exclude(user__in = allBlocked)

    liked = []

    like_no = []

    # print(posts)

    for post in posts:

        liked.append(check_like(request.user, post)[0])

        like_no.append(check_like(request.user, post)[1])


    master_list = zip(posts, liked, like_no)

    context = {

        "master":master_list

    }



    template_name = "posts/display_posts.html"

    return render(request, template_name, context)



@login_required

def detail_post_view(request, pk):

    post = Post.objects.get(pk=pk)

    comments = post.comment_set.all()

    num_comments = len(comments)

    template_name = "posts/detail_post.html"
    liked, like_no = check_like(request.user, post)


    context = {

        "post":post,

        "comments":comments,

        "liked":liked,

        "like_no":like_no,

        "num_comments":num_comments

    }


    return render(request, template_name, context)


@login_required

def comments_view(request, pk):

    print("out")

    if request.method == "POST":

        print("here")

        form = CommentForm(request.POST)

        if form.is_valid():

            obj = form.save(commit=False)

            post = Post.objects.get(pk=pk)

            obj.post = post

            obj.user = request.user

            obj.save()
            return redirect(f"/posts/{pk}")


    form = CommentForm()

    template_name = "posts/add_comment.html"

    context={

    "form":form

    }

    return render(request, template_name, context)


def check_like(user, post):

    likes = post.likes

    if user in likes.all():

        return True, len(likes.all())

    return False, len(likes.all())


def like_view(request, pk, destination):

    post = Post.objects.get(pk=pk)

    liked, like_no = check_like(request.user, post)

    if liked:

        post.likes.remove(request.user)

    else:

        post.likes.add(request.user)

    print(post.likes.all())

    post.save()
    print(request.POST)

    next_url = request.POST.get("next")

    return redirect(next_url)


@login_required

def delete_post_view(request, pk):

    obj = Post.objects.get(pk=pk)

    if request.method == "POST":

        obj.delete()

        messages.error(request, "Post has been deleted.")

        return redirect("profile")

    template_name = "posts/delete_post.html"

    context = {

        "post":obj

    }

    return render(request, template_name, context)


@login_required

def update_post_view(request, pk):

    obj = Post.objects.get(pk=pk)

    if request.method == "POST":

        post = PostForm(request.POST, request.FILES, instance=obj)

        if post.is_valid():

            post.save()
            messages.success(request, "Post has been updated.")

            return redirect("profile")


    form = PostForm(instance=obj)

    template_name = "posts/update-post.html"

    context = {

        "form":form

    }

    return render(request, template_name, context)


@login_required

def archive_post_view(request, pk):

    obj = Post.objects.get(pk=pk)

    obj.archived = True

    obj.save()

    return redirect("profile")


@login_required

def unarchive_post_view(request, pk):

    obj = Post.objects.get(pk=pk)

    obj.archived = False

    obj.save()

    return redirect("profile")
```

3.2 Test Results

UNIT TESTING

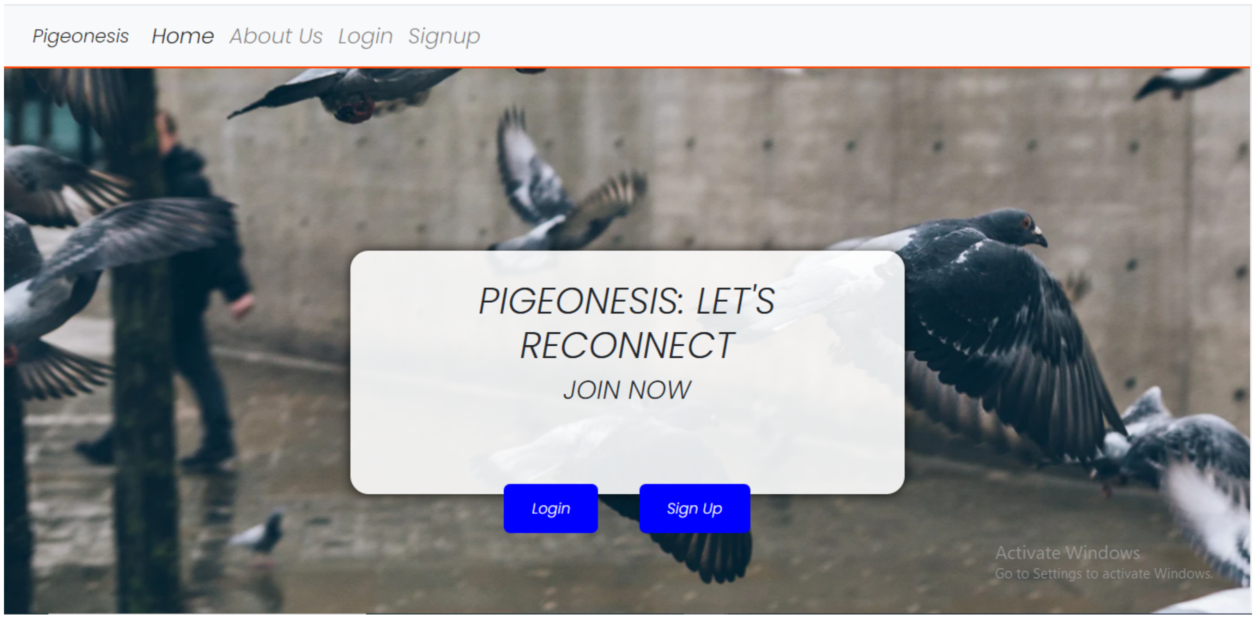
Test Case Id	Test Case Description	Status of Test Case
1	Welcome Page-Home page,Details of About Us, Login and Sign up options	Passed
2	Login with Valid Email Id and Valid Password	Passed
3	Login with Valid Email Id and Invalid Password	Passed
4	Login with Invalid Email Id and Valid Password	Passed
5	Login with Invalid Email Id and Invalid Password	Passed
6	Sign up with valid name, valid email Id and password which is not registered before	Passed
7	Sign up with Invalid name format ,Invalid email Id format and invalid password format	Passed
8	Sign up with Valid Name and Email ID format but passwords and confirm passwords do not match	Passed
9	Update profile with valid name and valid phone number and upload a profile pic	Passed
10	Update profile by giving a valid URL	Passed
11	Update profile with valid profile pic format (jpg and png format only)	Failed
12	Update profile with invalid profile pic format	Passed
13	Dashboard landing page. Should open only after	Passed

	successful login..	
14	Searching allowed with a valid username only	Passed
15	Add post allowed with valid image and caption	Passed
16	Allow Edit post	Passed
17	Allow archive post	Passed
18	Allow unarchive option	Passed
19	Allow likes to other people's posts.	Passed
20	Allow the comment option	Passed
21	Contact us Page	Passed
22	Delete account facility	Passed
23	Local cache memory clearance check after deleting account.	Failed
24	Logout option.	Passed

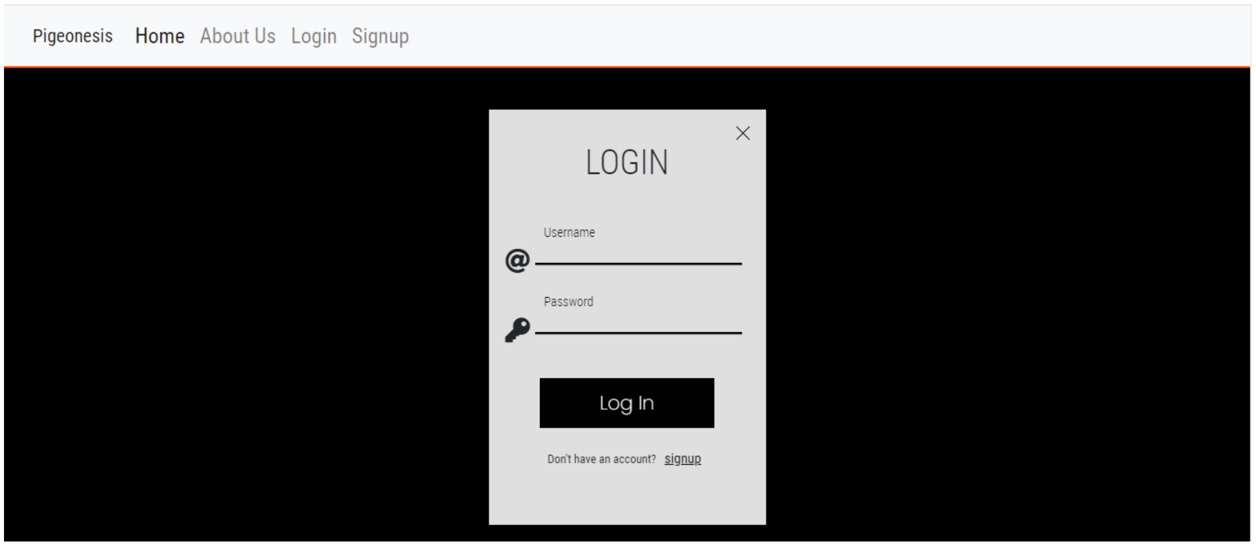
4. Results and Discussion

4.1 Output/Results

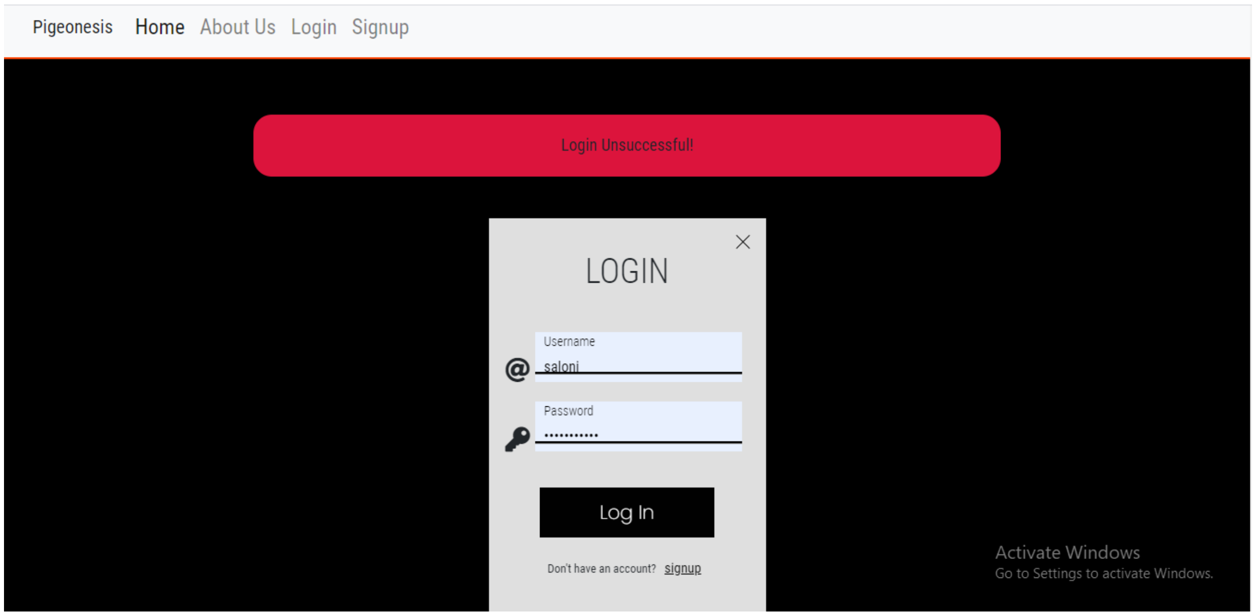
1. Landing Page



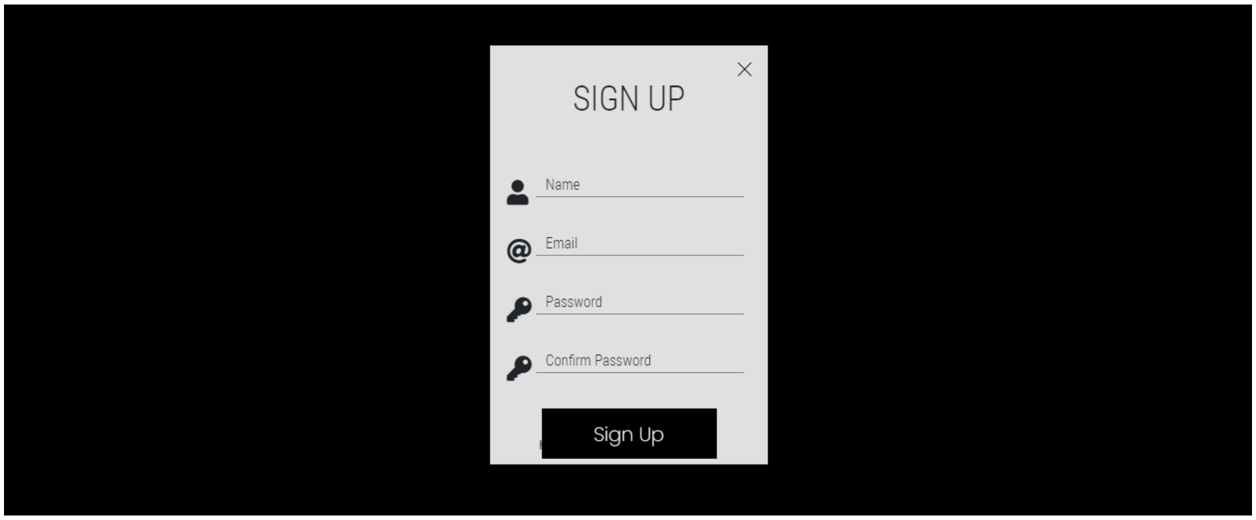
2. LOGIN PAGE



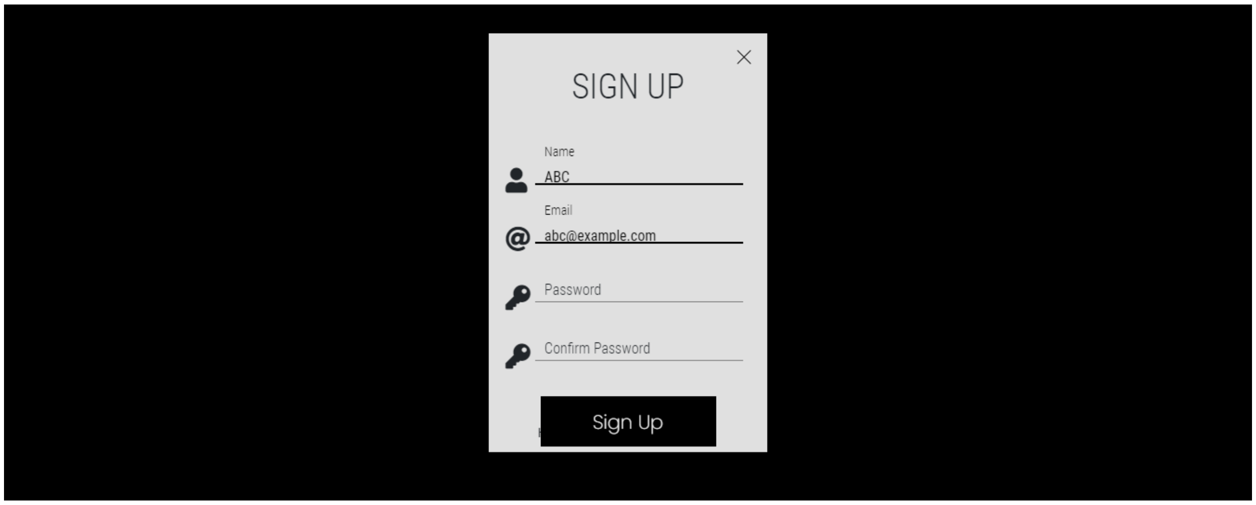
ITS VALIDATION



3. SIGN UP PAGE



ITS VALIDATION


SIGN UP

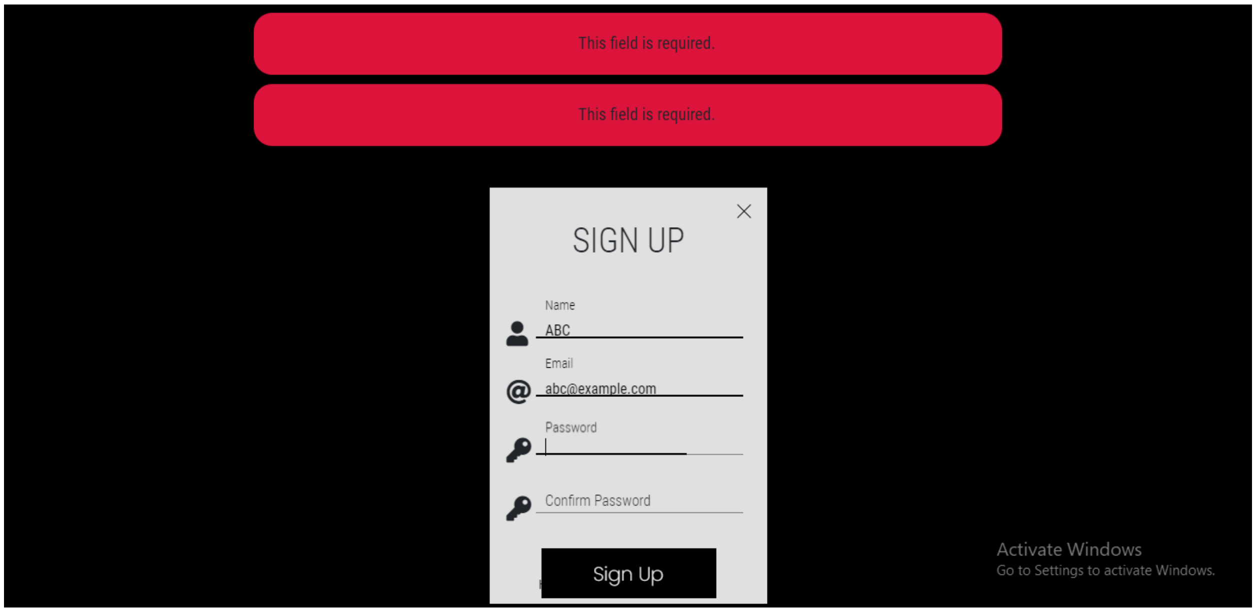
Name: ABC

Email: abc@example.com

Password

Confirm Password

Sign Up


This field is required.

This field is required.

SIGN UP

Name: ABC

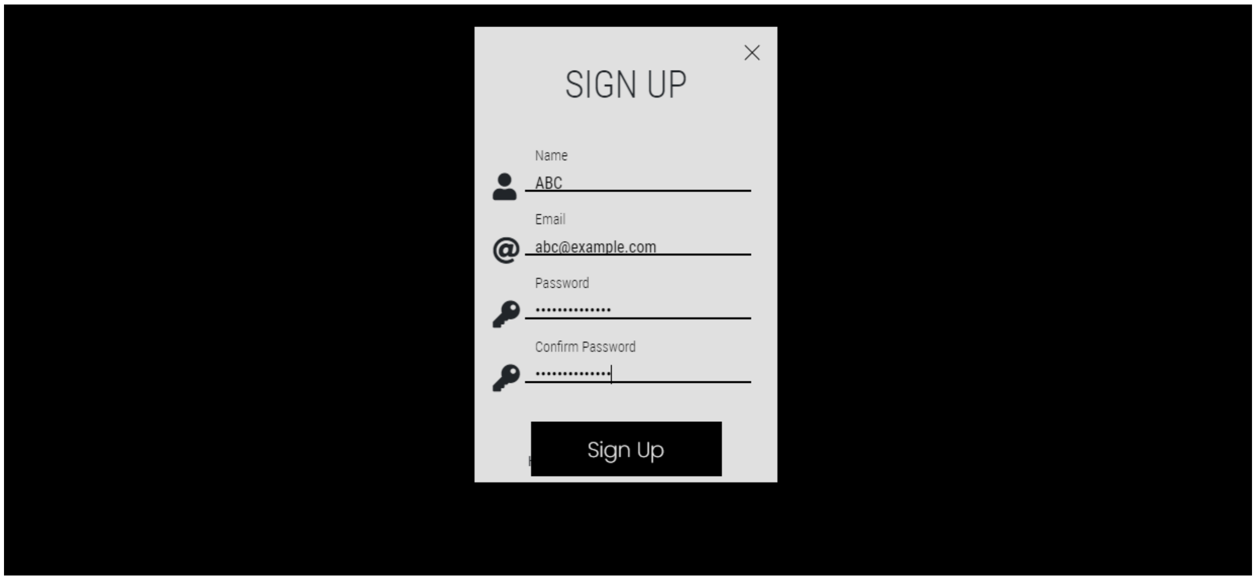
Email: abc@example.com

Password

Confirm Password

Sign Up

Activate Windows
Go to Settings to activate Windows.


SIGN UP

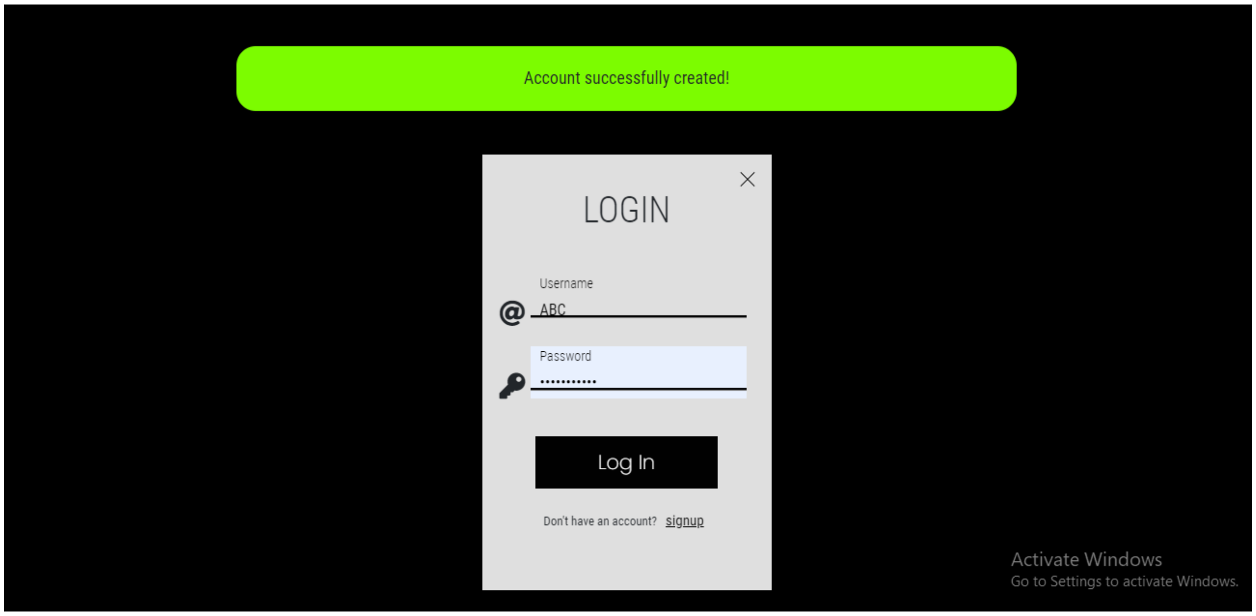
Name: ABC

Email: abc@example.com

Password: ••••••••••••••

Confirm Password: ••••••••••••••

Sign Up


Account successfully created!

LOGIN

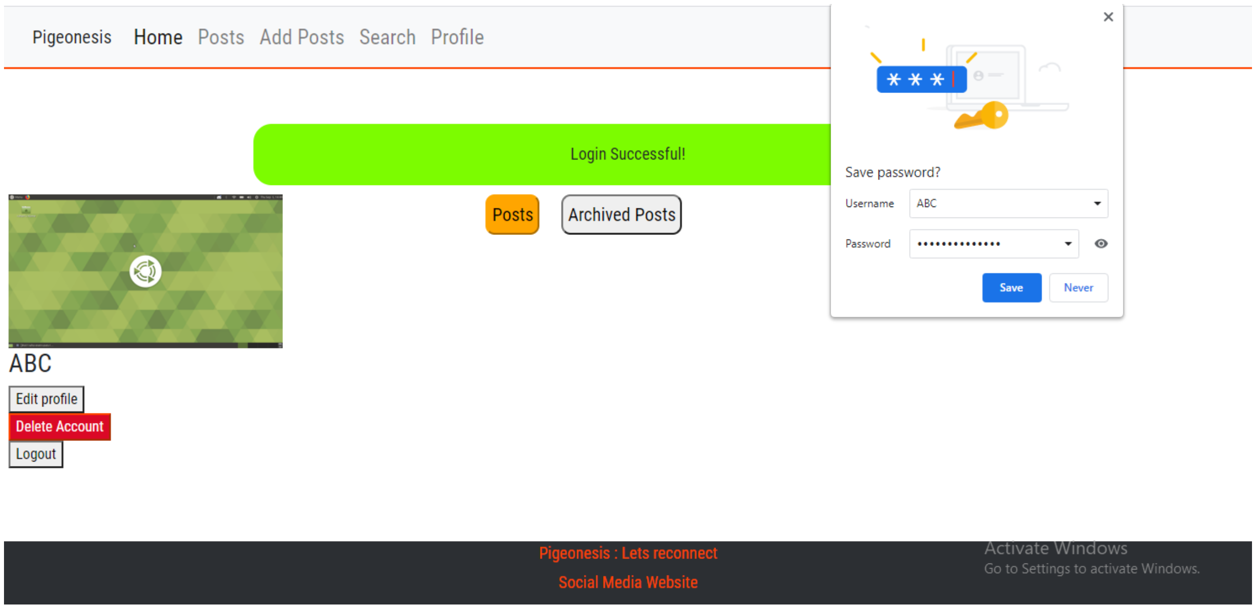
Username: ABC

Password: ••••••••••••

Log In
Don't have an account? signup
Activate Windows
Go to Settings to activate Windows.

And with successful creation,it jumps back to the login page.

Logged In Successfully -


Login Successful!

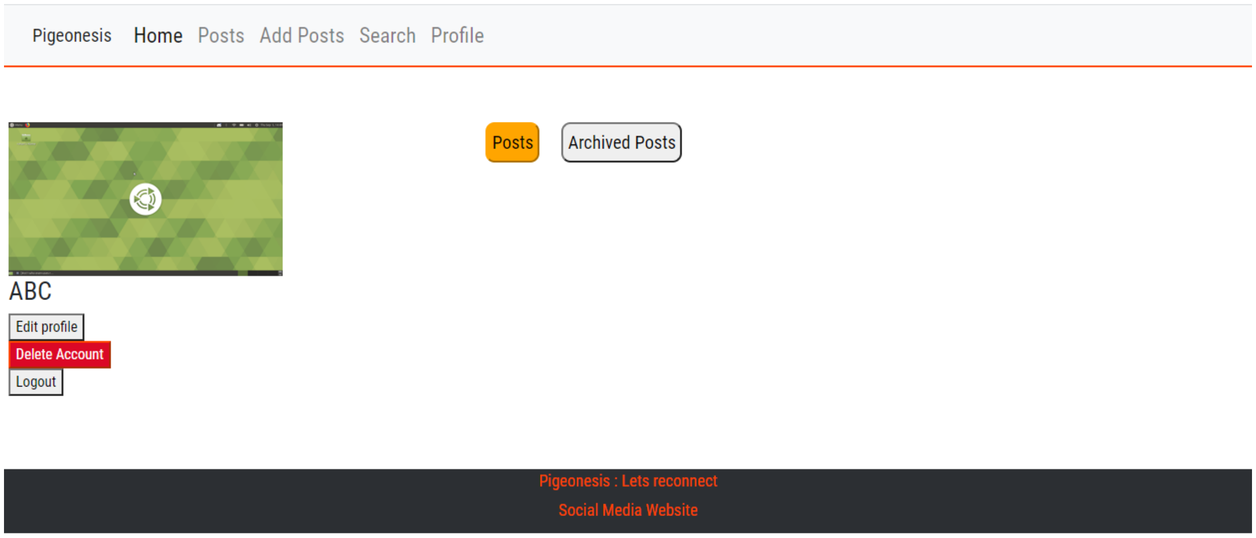
ABC

Save password?

Username: ABC

Password: ••••••••••••••

4. HOME PAGE


Pigeonesis Home Posts Add Posts Search Profile

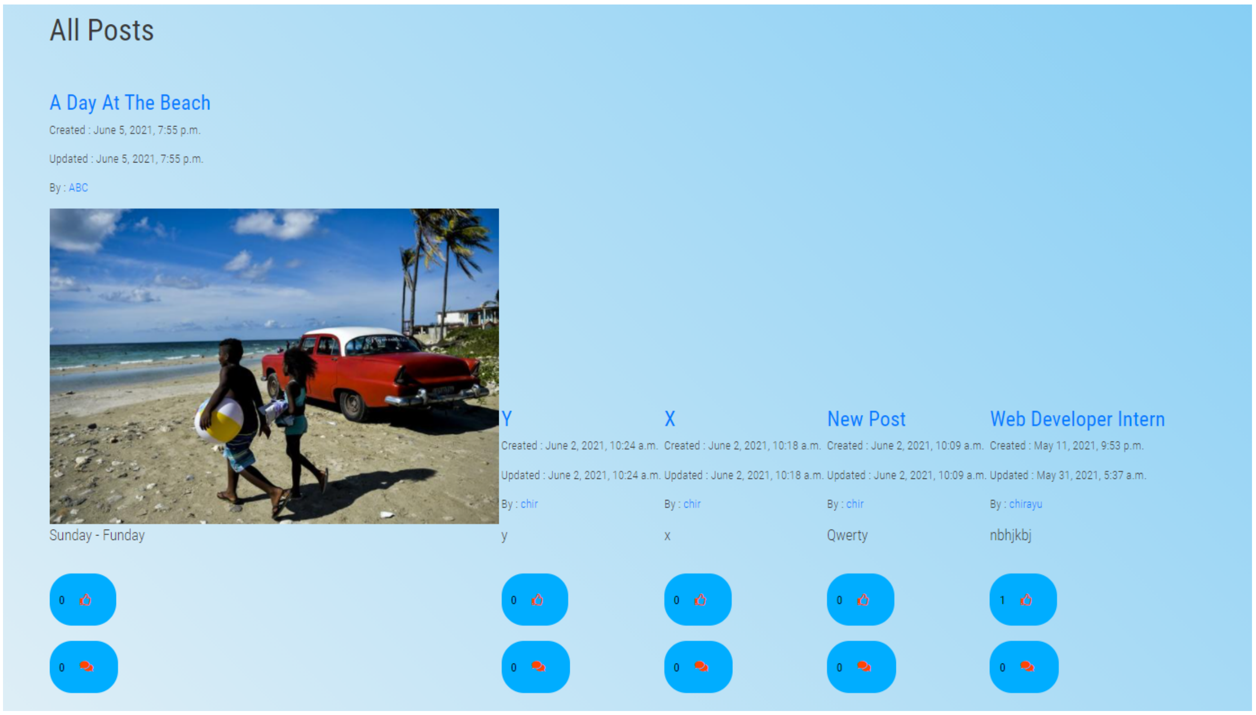
Posts Archived Posts

ABC

Edit profile
Delete Account
Logout

Pigeonesis : Lets reconnect
Social Media Website

5. POSTS PAGE


All Posts

A Day At The Beach
Created : June 5, 2021, 7:55 p.m.
Updated : June 5, 2021, 7:55 p.m.
By : ABC
Sunday - Funday
0
0

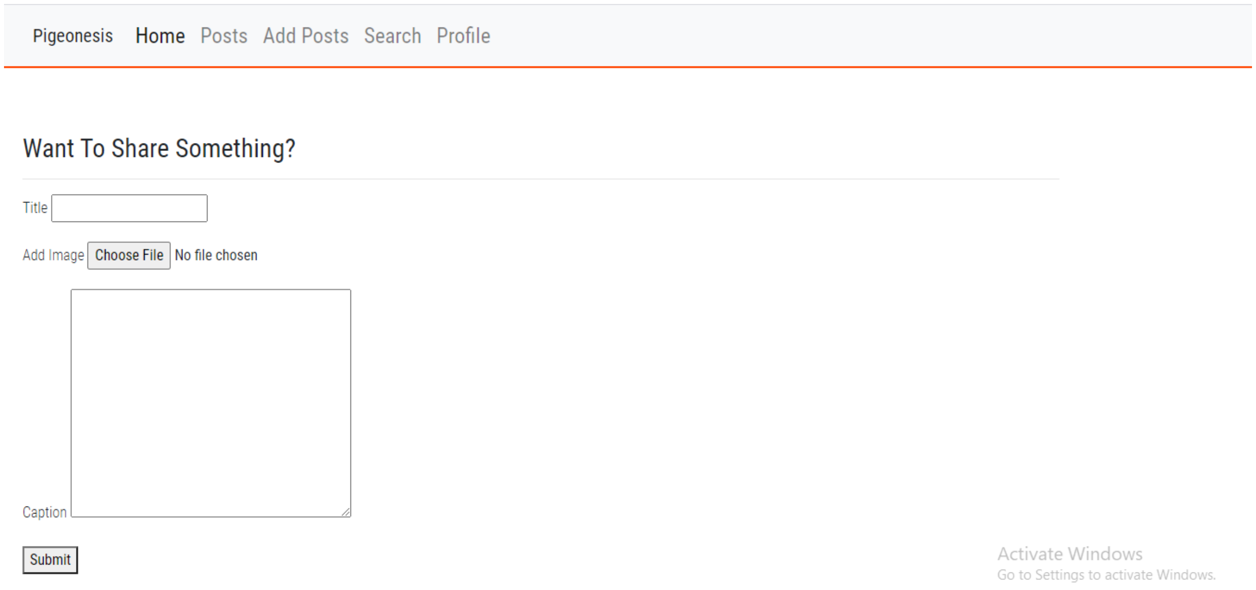
Y
Created : June 2, 2021, 10:24 a.m.
Updated : June 2, 2021, 10:24 a.m.
By : chir
y
0
0

X
Created : June 2, 2021, 10:18 a.m.
Updated : June 2, 2021, 10:18 a.m.
By : chir
x
0
0

New Post
Created : June 2, 2021, 10:09 a.m.
Updated : June 2, 2021, 10:09 a.m.
By : chir
Qwerty
0
0

Web Developer Intern
Created : May 11, 2021, 9:53 p.m.
Updated : May 31, 2021, 5:37 a.m.
By : chirayu
nbhjkbj
1
0

6. ADD POST PAGE



Want To Share Something?

Title

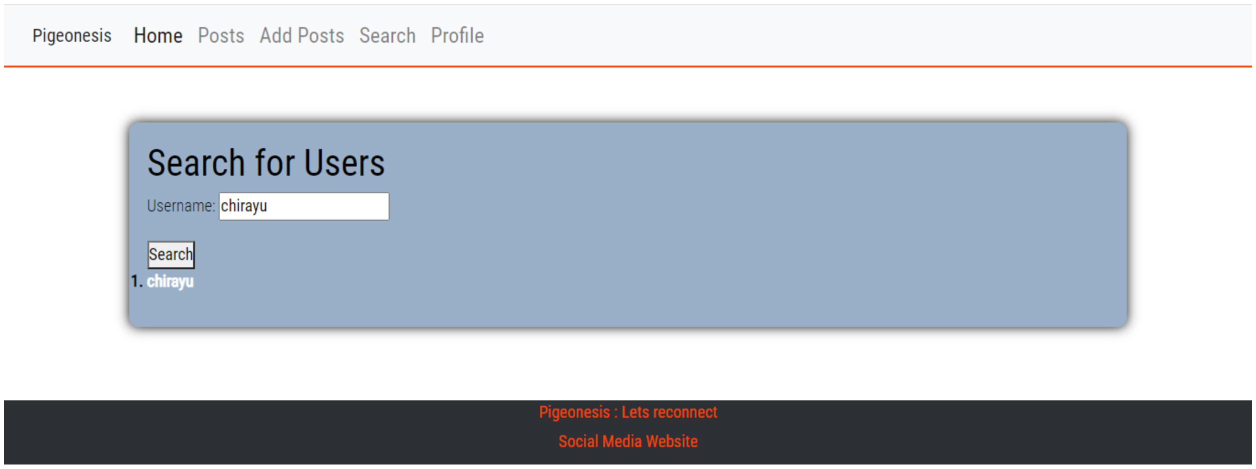
Add Image Choose File No file chosen

Caption

Submit

Activate Windows
Go to Settings to activate Windows.

7. SEARCH



Search for Users

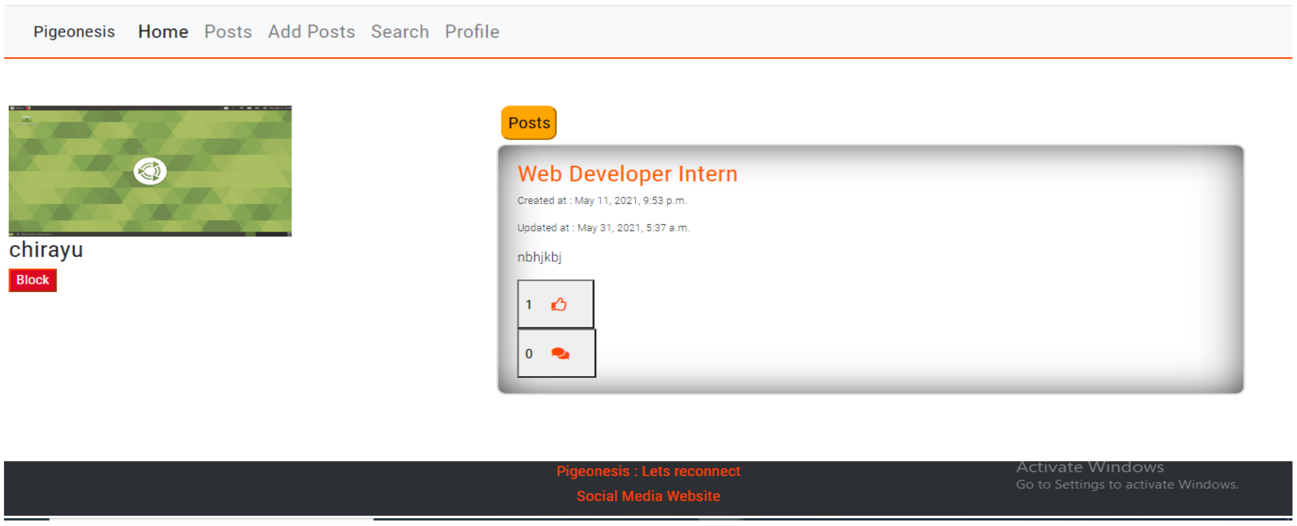
Username: chirayu

Search

1. chirayu

Pigeonesis : Lets reconnect
Social Media Website

ON CLICKING ON THE USER YOU CAN SEE THE POSTS MADE BY THEM -



chirayu

Web Developer Intern

Created at : May 11, 2021, 9:53 p.m.

Updated at : May 31, 2021, 5:37 a.m.

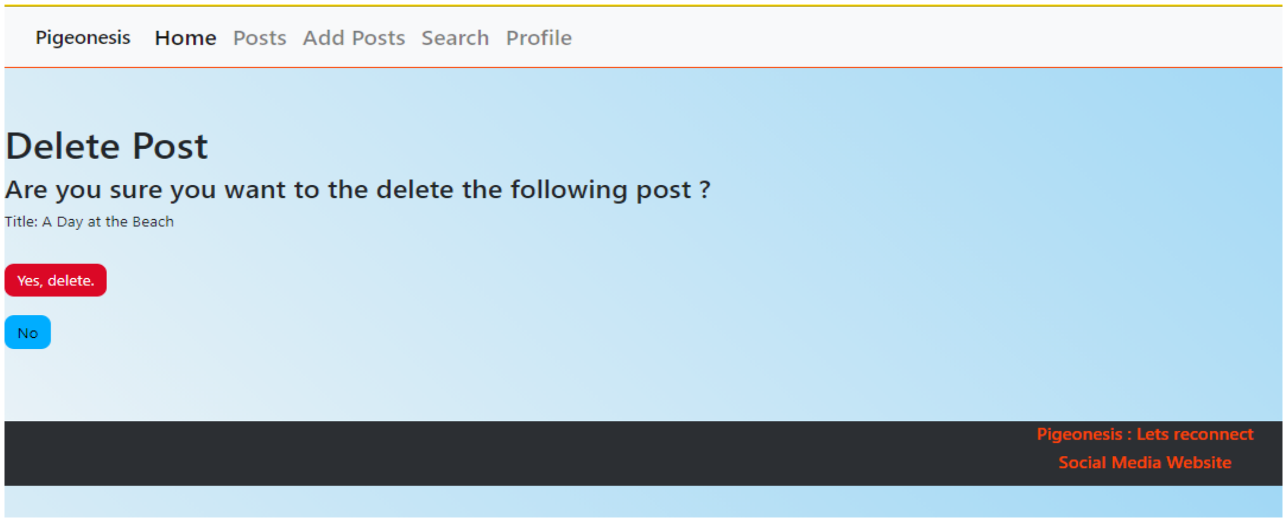
nbhjkbj

1

0

Pigeonesis : Lets reconnect
Social Media Website
Activate Windows
Go to Settings to activate Windows.

8. Delete Posts

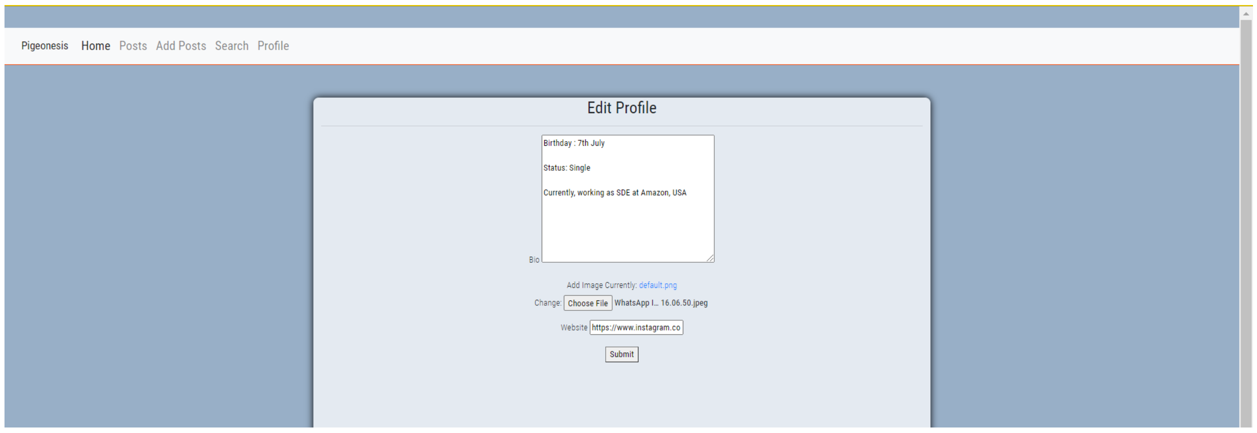


Delete Post

Are you sure you want to the delete the following post ?

Title: A Day at the Beach

9. EDIT PROFILE



Edit Profile

Bio: Birthday : 7th July

Status: Single

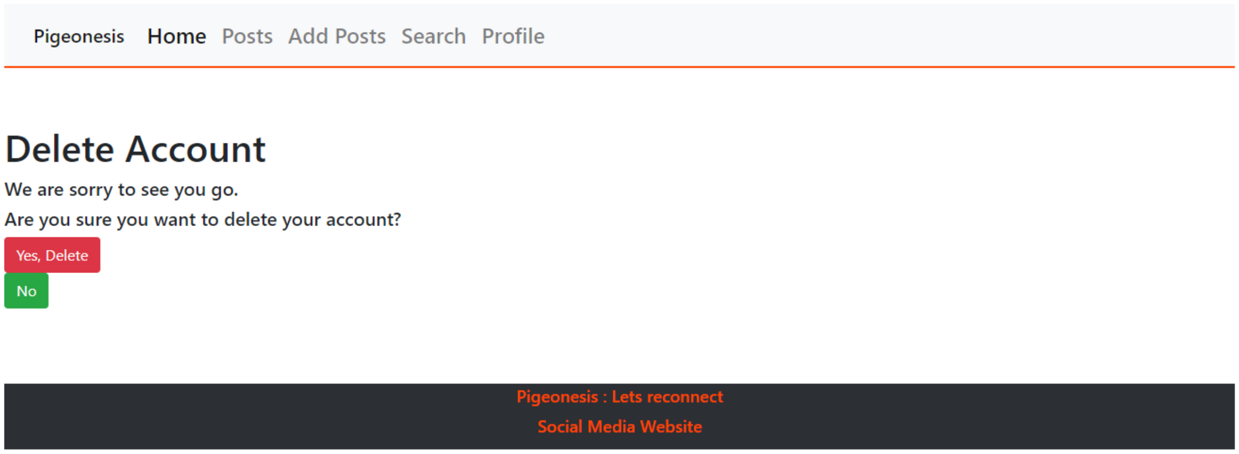
Currently, working as SDE at Amazon, USA

Add Image Currently: default.png

Change: Choose File WhatsApp I... 16.06.50.jpeg

Website https://www.instagram.co

10. DELETE ACCOUNT



Pigeonesis Home Posts Add Posts Search Profile

Delete Account

We are sorry to see you go.

Are you sure you want to delete your account?

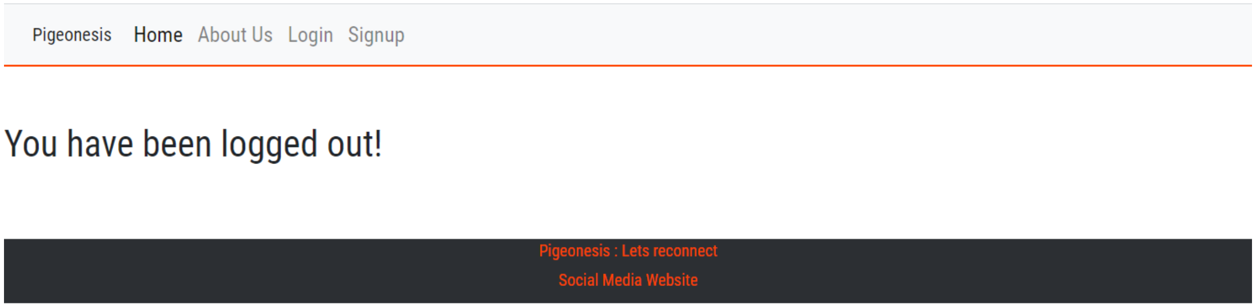
Yes, Delete

No

Pigeonesis : Lets reconnect

Social Media Website

11. LOG OUT



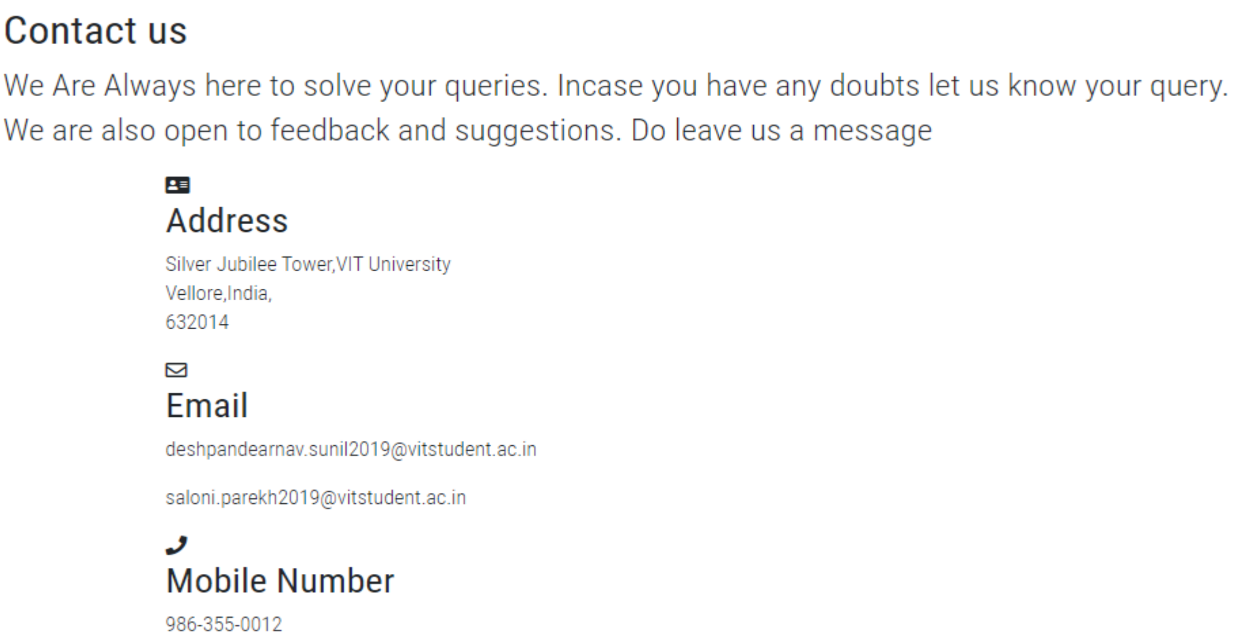
Pigeonesis Home About Us Login Signup

You have been logged out!

Pigeonesis : Lets reconnect

Social Media Website

12. ABOUT US



Contact us

We Are Always here to solve your queries. Incase you have any doubts let us know your query. We are also open to feedback and suggestions. Do leave us a message

Address

Silver Jubilee Tower,VIT University
Vellore,India,
632014

Email

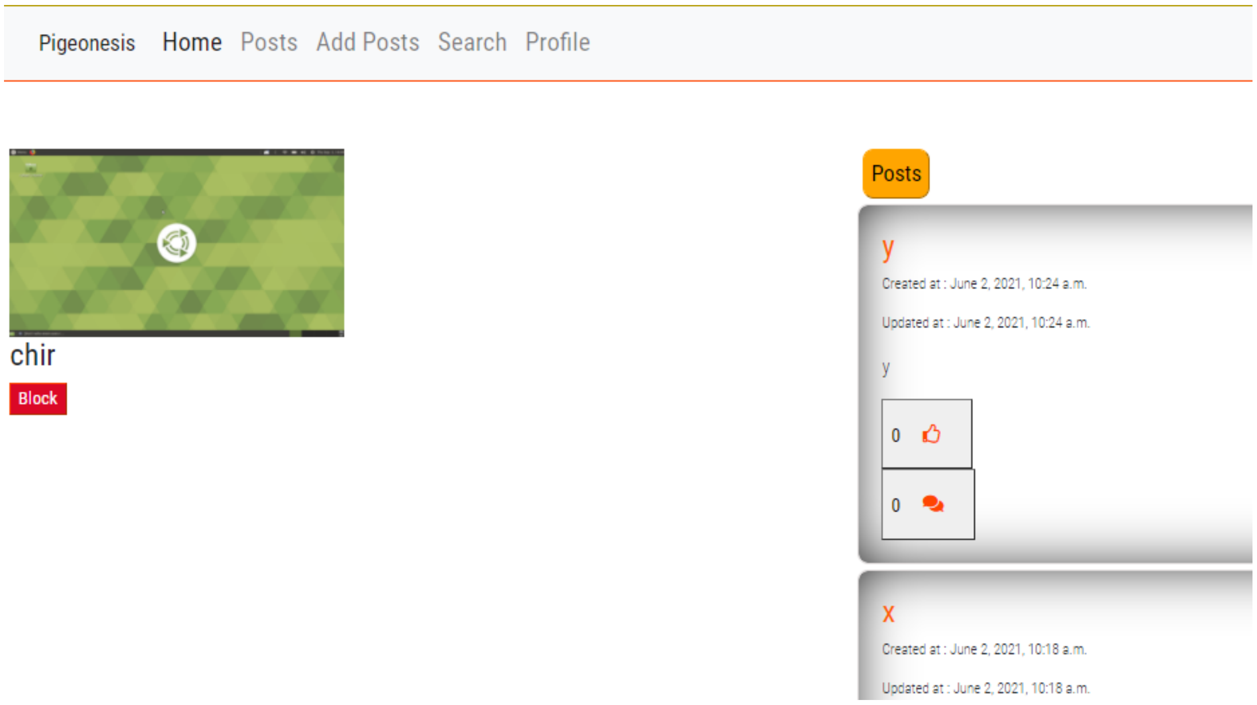
deshpandearnav.sunil2019@vitstudent.ac.in

saloni.parekh2019@vitstudent.ac.in

Mobile Number

986-355-0012

13. Block User


Pigeonesis Home Posts Add Posts Search Profile

chir

Block

Posts

y

Created at : June 2, 2021, 10:24 a.m.

Updated at : June 2, 2021, 10:24 a.m.

y

0

0

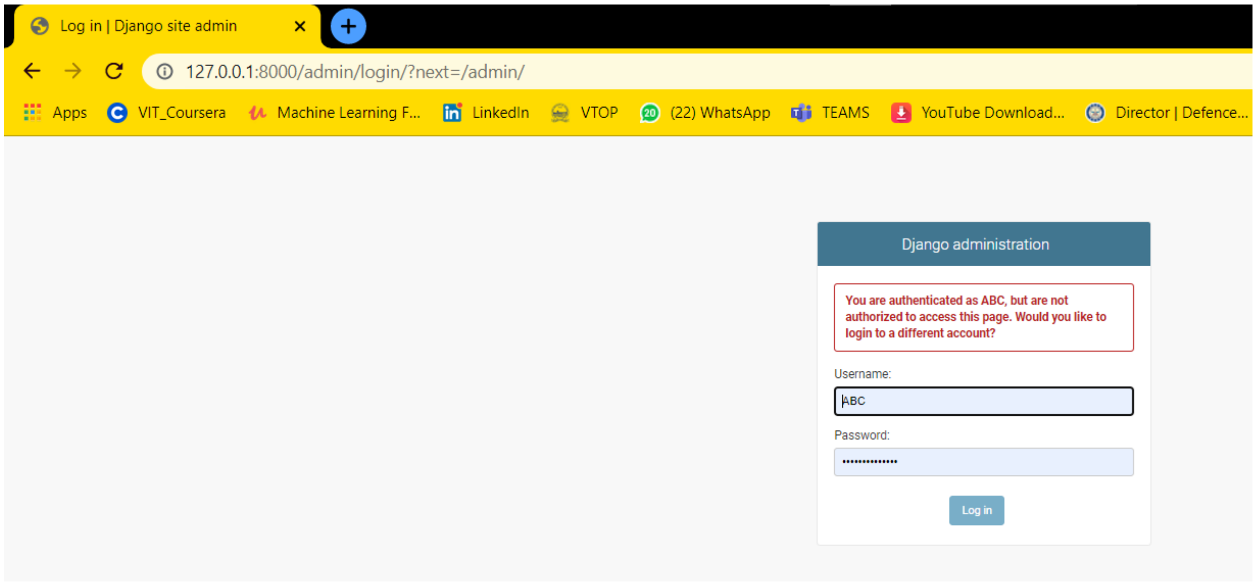
x

Created at : June 2, 2021, 10:18 a.m.

Updated at : June 2, 2021, 10:18 a.m.

If User is Blocked, You cannot view his Posts anymore.
Only an Admin can Unblock him for you.

14. ADMIN INTERFACE


Log in | Django site admin

127.0.0.1:8000/admin/login/?next=/admin/

Apps VIT_Coursera Machine Learning F... LinkedIn VTOP (22) WhatsApp TEAMS YouTube Download... Director | Defence...

Django administration

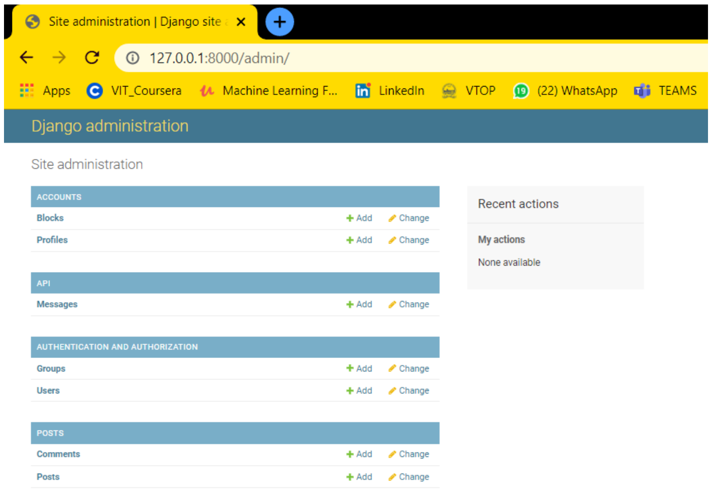
You are authenticated as ABC, but are not authorized to access this page. Would you like to login to a different account?

Username: ABC

Password: ••••••••••••••

Log in

Normal Users, don't have admin access to the site.



Django administration

Site administration

ACCOUNTS		
Blocks	+ Add	Change
Profiles	+ Add	Change

API		
Messages	+ Add	Change

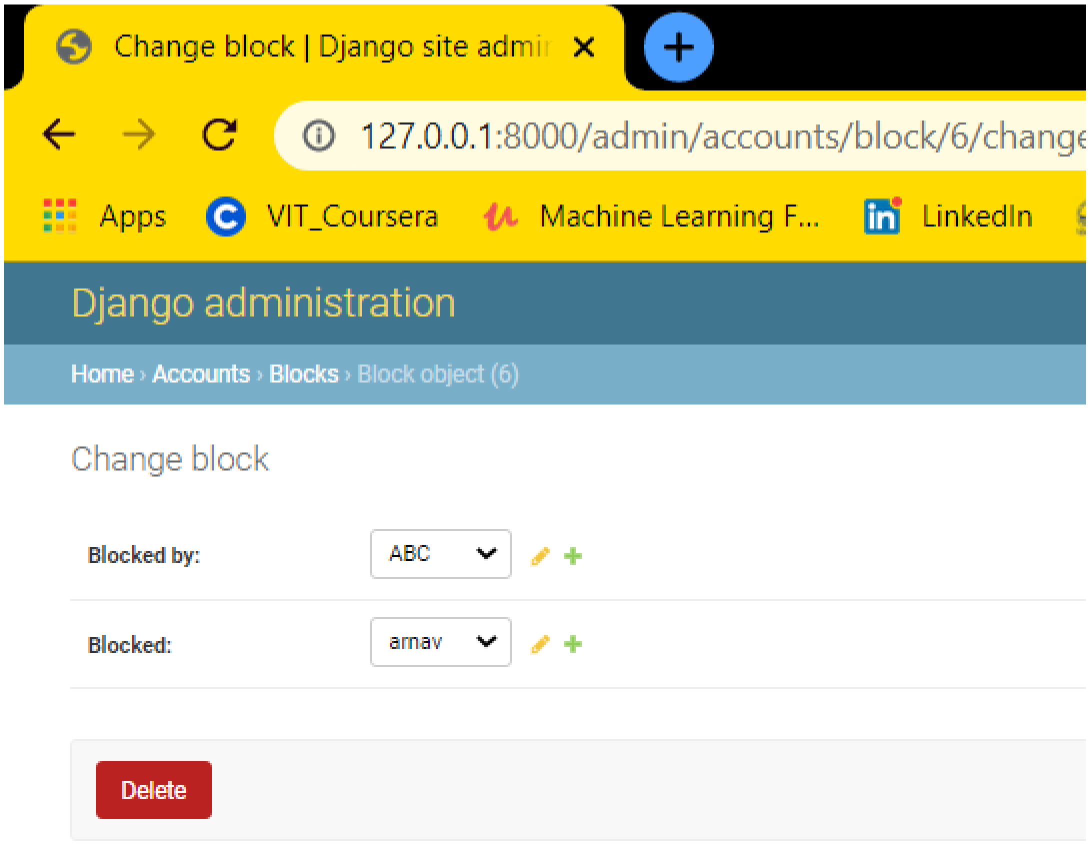
AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

POSTS		
Comments	+ Add	Change
Posts	+ Add	Change

Recent actions

My actions

None available


Change block | Django site admi
127.0.0.1:8000/admin/accounts/block/6/change
Apps
VIT_Coursera
Machine Learning F...
LinkedIn

Django administration

Home › Accounts › Blocks › Block object (6)

Change block

Blocked by:	ABC
Blocked:	arnav

Delete

Blocked Users can be changed. Posts can be taken down. Users can be added / deleted. Comments can be deleted.

4.2 Discussion

The results indicate the various specifications which we have incorporated in our social media website. We have a landing page which provides the users with the choices whether they wish to Login or Sign up based on the fact whether they do have an account or not. Based on what they have selected they are directed to the particular page where they need to enter their details correctly in order to prevent an error message. Then we have a home page. A profile page has also been created where the user is given multiple options like edit profile,add post,view post etc. On clicking the Edit profile, the user can modify the details of his/her profile with a new profile picture and URL. On clicking the Add post the person can add any post that he/she wishes to.The post can either be just a text or text along with picture or just a picture. Once the user adds the post it gets saved to his/her profile under the posts section. The user can also choose to archive some of the post that it doesn't wish to keep it open to everyone and can later unarchive it also from the unarchive column.The user can also edit the post at any time through the Edit Post option. The user can also have a look at other peoples posts and can like and comment the posts with a maximum of 1 like allowed for each post made. The user is also given the privilege to search for any person who is there on the same social media platform. The person is requested to search with the correct name. Also if they feel that someone is posting violating or inappropriate posts then they can click on the block button and if more than 3 people block the same person then that person's account will be permanently deleted from the website. By doing this we have tried to provide a safe and secure environment. Also we have implemented an admin interface which adds more charm to this project and brings in a new dimension.Also the option for logout is there on the Home screen. The user is also given the freedom to delete his/her account at any point of time through the delete account option.

5. Conclusion

Through this report we have discussed our project which was based on a clone of a social media website which we choose to name as Pigeonesis. Through pigeonesis we have shown how people can stay connected with other peoples life and also allowing others to be a part of their life. Various interesting features have been added along with a very important feature which is the blocking feature. Through this website we have tried to help as many people as possible in such difficult times where all people fear is loneliness and isolation.

In this project we tried to implement the things we got to learn during the Internet Web Programming class. We got a chance to put the learned knowledge into practice and gain a lot of practical knowledge. Also we explored a new language, Django. Overall, this project was a great additional learning bonus for us as developers, topped with the able feedback of our Course Instructor, Prof. Dr. Senthilkumar, hope it adds value to your life as well.

6. References

HTML Tutorial. [Online]. Available: https://www.w3schools.com/html/.

SQLite Documentation. [Online]. Available: https://www.sqlite.org/docs.html.

CSS Tutorial. [Online]. Available: https://www.w3schools.com/css/.

“Documentation,” *Django*. [Online]. Available: https://docs.djangoproject.com/en/3.2/.

J. Juneau, J. Baker, V. Ng, L. Soto, and F. Wierzbicki, “Web Applications With Django,” *The Definitive Guide To Jython*, pp. 281–325, 2010.

Manjurulhoque, “manjurulhoque/django-social-network,” *GitHub*. [Online]. Available: https://github.com/manjurulhoque/django-social-network.

Wagtail, “wagtail/wagtail,” *GitHub*. [Online]. Available: https://github.com/wagtail/wagtail.

Whatwg, “whatwg/html,” *GitHub*. [Online]. Available: https://github.com/whatwg/html.

pranavg000, “pranavg000/Social-Network-Django,” *GitHub*. [Online]. Available: https://github.com/pranavg000/Social-Network-Django.