NEHRU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

An ISO 9001:2015 & ISO 14001:2015 Certified Institution, Affiliated to Anna University, Chennai
Approved by AICTE, New Delhi, Recognized by UGC with 2(f) & 12(B)
Re-accredited by NAAC "A+", NBA Accredited (UG Courses): AERO | CSE
Nehru Gardens, Thirumalayampalayam, Coimbatore – 641 105.

DEPARTMENT OF INFORMATION TECHNOLOGY

RECORD NOTEBOOK

NAME :_____

REG.NO. :_____

BRANCH : _____

SUBJECT CODE & TITLE:_____

NEHRU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

An ISO 9001:2015 & ISO 14001:2015 Certified Institution, Affiliated to Anna University, Chennai
Approved by AICTE, New Delhi, Recognized by UGC with 2(f) & 12(B)
Re-accredited by NAAC "A+", NBA Accredited (UG Courses): AERO | CSE
Nehru Gardens, Thirumalayampalayam, Coimbatore – 641 105

Certified that this is a bonafide record of work done in _____

by Mr./Ms._____

Reg.No._____of this Institution for the V semester course in Information Technology branch during the academic year 2024-2025(ODD).

Staff In charge :

Date : **Head of the Department**

Submitted for the university practical examination held on_____at **Nehru Institute of Engineering and Technology,** Coimbatore – 105.

University Register Number	

....

Internal Examiner **External Examiner**

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	MARKS	STAFF SIGN
1		Develop a portfolio website for yourself which gives details about yourself for a potential recruiter.		
2		Create a web application to manage the TO-DO list of users, where users can login and manage their to-do items.		
3		Create a simple micro blogging application (like twitter) that allows people to post their content which can be viewed by people who follow them.		
4		Create a food delivery website where users can order food from a particular restaurant listed in the website.		
5		Develop a classifieds web application to buy and sell used products		
6		Develop a leave management system for an organization where users can apply different types of leaves such as casual leave and medical leave. They also can view the available number of days.		
7		Develop a simple dashboard for project management where the statuses of various tasks are available. New tasks can be added and the status of existing tasks can be changed among Pending, InProgress or Completed		
8		Develop an online survey application where a collection of questions is available and users are asked to answer any random 5 questions.		
CONTENT BEYOND SYLLABUS				
1		Develop a program to create and build a password strength check.		

NEHRU INSTITUTE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
An ISO 9001:2015 & ISO 14001:2015 Certified Institution, Affiliated to Anna University, Chennai
Approved by AICTE, New Delhi, Recognized by UGC with 2(f) & 12(B)
Re-accredited by NAAC "A+", NBA Accredited (UG Courses): AERO | CSE
Nehru Gardens, Thirumalayampalayam, Coimbatore – 641 105.

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION AND MISSION OF THE INSTITUTION

VISION

Our Vision is to mould the youngsters to acquire sound knowledge in technical and scientific fields to face the future challenges by continuous upgradation of all resources and processes for the benefit of humanity as envisaged by our great leader Pandit Jawaharlal Nehru.

MISSION

- To build a strong centre of learning and research in engineering and technology.
- To facilitate the youth to learn and imbibe discipline, culture and spirituality.
- To produce quality engineers, dedicated scientists and leaders.
- To encourage entrepreneurship.
- To face the challenging needs of the global industries.

VISION AND MISSION OF THE DEPARTMENT

VISION

To produce highly competent and innovative Computing and Business Systems Professionals with Managerial Skills, Social Values to serve the Nation and to meet the industry challenges.

MISSION

M1: To impart technical knowledge through innovative students-centric teaching learning processes and research.

M2: To groom students technologically superior and ethically stronger and responsible throughout the professional career to compete globally.

M3: To produce competent engineers with professional ethics, spirit of innovation and managerial skills to cater the needs of the industries and society.

DEPARTMENT OF INFORMATION TECHNOLOGY

PROGRAM EDUCATIONAL OBJECTIVES(PEOs)

PEO1: Acquire and Apply knowledge in Computer Science, Mathematics, Science and inter-disciplinary engineering principles in order to excel in computer professional career.

PEO2: Analyze real life problems adapting to new Computing Technologies for professional excellence and ethical attitude in order to provide economically feasible engineering solutions.

PEO3: Carry out complex engineering problems with best practices exhibiting communication skills, team work and interpersonal skills to enable continued computer professional development through life-long learning.

PROGRAM SPECIFIC OUTCOMES(PSOs)

PSO1: Professional Skills: Acquaint in-depth knowledge on the basic and advanced computer science domains like Data Sciences, Cryptography, Cloud and Distributed Computing, Neural Networks and Artificial Intelligence.

PSO2: Analyze and recommend the appropriate IT infrastructure required for the implementation of a project.

PSO3: Entrepreneurship and Successful Career: Apply the standard practices to have successful career path in the field of information and communication technology and entrepreneurship.

PROGRAM OUTCOMES(POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct Investigations of Complex Problems:** Use research-based knowledge andresearchmethodsincludingdesignofexperiments,analysisandinterpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern Tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-Long Learning:** Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

NEHRU INSTITUTE OF ENGINEERING AND TECHNOLOGY
Autonomous
An ISO 9001: 2015 & 14001: 2015 Certified Institution, Affiliated to Anna University, Chennai
Approved by AICTE, New Delhi, Recognized by UGC with Section 2(f) and 12(B)
Re-accredited by NAAC "A+", NBA Accredited (UG Courses: AERO & CSE)
"Nehru Garden",Thirumalayampalayam, Coimbatore - 641 105.

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL

ANNA UNIVERSITY REGULATION 2021

Laboratory Code : IT3511

Laboratory Name : FULL STACK WEB DEVELOPMENT LABORATORY

Semester / Year : V / III

Degree / Branch : B.Tech / IT

PREPARED BY:

Mr.M.Sathish Kumar,

Assistant Professor,

Department of IT

VISION AND MISSION OF THE DEPARTMENT

VISION

To produce highly competent and innovative Computing and Business Systems Professionals with Managerial Skills, Social Values to serve the Nation and to meet the industry challenges.

MISSION

M1: To impart technical knowledge through innovative students-centric teaching learning processes and research.

M2: To groom students technologically superior and ethically stronger and responsible throughout the professional career to compete globally.

M3: To produce competent engineers with professional ethics, spirit of innovation and managerial skills to cater the needs of the industries and society.

PEOS & PSOS

PROGRAM EDUCATIONAL OBJECTIVES

The Graduates of the Information Technology Programme will be able to

PEOs	PROGRAM EDUCATIONAL OBJECTIVES
PEO1	Acquire and Apply knowledge in Computer Science, Mathematics, Science and inter-disciplinary engineering principles in order to excel in computer professional career.
PEO2	Analyze real life problems adapting to new Computing Technologies for professional excellence and ethical attitude in order to provide economically feasible engineering solutions.
PEO3	Carry out complex engineering problems with best practices exhibiting communication skills, team work and interpersonal skills to enable continued computer professional development through life-long learning.

PROGRAM SPECIFIC OUTCOMES

The Students of the Information Technology Programme will be able to

PSO	PROGRAM SPECIFIC OUTCOMES
PSO1	**Professional Skills:** Acquaint in-depth knowledge on the basic and advanced computer science domains like Data Sciences, Cryptography, Cloud and Distributed Computing, Neural Networks and Artificial Intelligence.
PSO2	Analyze and recommend the appropriate IT infrastructure required for the implementation of a project.
PSO3	**Entrepreneurship and Successful Career:** Apply the standard practices to have successful career path in the field of information and communication technology and entrepreneurship.

PROGRAM OUTCOMES

PO	PROGRAM OUTCOMES (POS)-12 GRADUATE ATTRIBUTES
PO1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and

	leader in a team, to manage projects and in multidisciplinary environments.
PO12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Please DO!

• Be on time, Clean up yourself; enter the lab with proper dress code and lab coat

• Sign the login register and occupy only the allotted system

• Do use the computer equipments properly and inform the instructor, if there is a problem

• Touch the keyboard lightly and maintain silence

• Do some example exercises beyond the syllabus

• Read and understand how to carry out an experiment thoroughly & complete the lab workbook before coming to the laboratory

• Do a proper shutdown after using computers and arrange the chairs properly before leaving the lab

• Respect the equipment. Don't damage, remove, or disconnect any labels, parts, cables, or equipment.

Please DON'T!

• Don't bring food or drinks inside the room

• Do not remove anything from the computer laboratory without permission

• Don't play games on the computers

• Don't make undue noise in the laboratories

• Don't disturb other users

• Don't use the on/off switch to reboot

• Do not read or modify other users' files.

• Do not install or download any software or modify or delete any system files on any lab computers.

• Leave the bags outside and do not use pen drives

• Do not touch, connect or disconnect any plug or cable without your instructor/laboratory technician's permission

• Do not misbehave in the computer laboratory

• If you leave the lab, do not leave your personal belongings unattended.

Syllabus	
V Semester	
Course Code: IT3511	**Course Name: Full Stack Web Development Laboratory**
Objective(s)	• To develop full stack applications with clear understanding of user interface, business logic and data storage. • To design and develop user interface screens for a given scenario • To develop the functionalities as web components as per the requirements • To implement the database according to the functional requirements • To integrate the user interface with the functionalities and data storage.
Outcome(s)	CO1: Design full stack applications with clear understanding of user interface, business logic and data storage. CO2: Design and develop user interface screens CO3: Implement the functional requirements using appropriate tool CO4: Design and develop database based on the requirements CO5: Integrate all the necessary components of the application

LIST OF EXPERIMENTS	
1	Develop a portfolio website for yourself which gives details about yourself for a potential recruiter.
2	Create a web application to manage the TO-DO list of users, where users can login and manage their to-doitems.
3	Create a simple micro blogging application (like twitter) that allows people to post their content which can be viewed by people who follow them.
4	Create a food delivery website where users can order food from a particular restaurant listed in the website.
5	Develop a classifieds web application to buy and sell used products.
6	Develop a leave management system for an organization where users can apply different types of leaves such as casual leave and medical leave. They also can view the available number of days.
7	Develop a simple dashboard for project management where the statuses of various tasks are available. New tasks can be added and the status of existing tasks can be changed among Pending, InProgress or Completed.
8	Develop an online survey application where a collection of questions is available and users are asked to answer any random 5 questions.
	Total hours **60**

CONTENT BEYOND SYLLABUS:	
1	Develop a program to create and build a password strength check.

CO’s-PO’s & PSO’s MAPPING

CO’s	PO’s												PSO’s		
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
1	3	3	3	1	3	1	1	1	2	1	1	1	2	2	1
2	3	3	3	2	3	1	1	1	2	1	1	1	2	2	1
3	3	3	3	3	3	1	1	1	2	1	1	1	2	2	1
4	3	3	3	3	3	2	1	1	1	1	2	1	1	2	1
5	3	3	3	3	2	1	1	1	1	1	1	1	2	2	1
AVg.	3	3	3	2	3	1	1	1	1	1	1	1	2	2	1

1-low, 2-medium, 3-high,‘-“- no correlation

Ex.No : 1

Develop a portfolio website for yourself which gives details about yourself for a potential recruiter.

Aim: To develop a full stack personal portfolio web application that uses python django framework

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin start project project name
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using django in settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
    }
}
```

- Now application structure will look like below image project structure

```
appname/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

- The models.py file contains code for database design using django object relational mapper(ORM)
- After models creation use python manage.py makemigrations and python manage.py migrate for table creations
- In views.py the business logics will take place template folder contain all html file and all static files placed inside static folder

Front End code:

1.Home.html:

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

{% load static %}

{% block content %}

    <div class="container">

        <div class="row home ">

            <div class="col-lg-12 text-center my-auto">

                <small class="s-color">Welcome to my portfolio website!</small>

                <div class="animated">

                    Hey, I'm<br>

                    <span class="color-primary" id="animated-name"></span>

                </div>

                <div class="row justify-content-center">

                    <div class="col-lg-6 text-center">

                        <p class="s-color mt-5">{{info.mini_about}}</p>

                    </div>

                </div>

                <div class="social-icons m-0 mt-3" style="justify-content: center;">

                    <a href="{{info.github}}" target="_blank">

                        <i class="fab fa-github"></i>

                    </a>

                    <a href="{{info.linkedin}}" target="_blank">

                        <i class="fab fa-linkedin-in"></i>

                    </a>

                    <a href="{{info.instagram}}" target="_blank">

                        <i class="fab fa-instagram"></i>

                    </a>
```

```
                <a href="{{info.facebook}}" target="_blank">
                    <i class="fab fa-facebook-f"></i></i>
                </a>
                <a href="{{info.twitter}}" target="_blank">
                    <i class="fab fa-twitter"></i>
                </a>
            </div>
            {% include 'nav.html' %}
        </div>
    </div>
     <a href="/#education" class="scroll_down">
        <span class="scroll_mouse">
            <span class="scroll_wheel"></span>
        </span>
     </a>
     <!-- Experiences and Education -->
     {% include 'experiences_and_education.html' %}
     <!-- Languages and Tools -->
     {% include 'languages_and_tools.html' %}
    <!-- Projects -->
     {% include 'projects.html' %}
     <!-- About and Contact -->
     {% include 'about_and_contact.html'%}
   </div>
{% endblock content %}
```

Backend (model.py):

```
from django.db import models
import re
from ckeditor.fields import RichTextField
class Information(models.Model):
    name_complete = models.CharField(max_length=50, blank=True, null=True)
    avatar = models.ImageField(upload_to="avatar/", blank=True, null=True)
    mini_about = models.TextField(blank=True, null=True)
    about = models.TextField(blank=True, null=True)
    born_date = models.DateField(blank=True, null=True)
    address = models.CharField(max_length=100, blank=True, null=True)
    phone = models.CharField(max_length=20, blank=True, null=True)
    email = models.EmailField(max_length=255, blank=True, null=True)
    cv = models.FileField(upload_to='cv', blank=True, null=True)
    # Social Network
    github = models.URLField(blank=True, null=True)
    linkedin = models.URLField(blank=True, null=True)
    facebook = models.URLField(blank=True, null=True)
    twitter = models.URLField(blank=True, null=True)
    instagram = models.URLField(blank=True, null=True)

    def__str__(self):
        return self.name_complete
class Competence(models.Model):
    title = models.CharField(max_length=50, blank=False, null=False)
    description = models.TextField(blank=False, null=False)
    image = models.FileField(upload_to='competence/', blank=False, null=False)

    def__str_(self):
```

```
        return self.title
class Education(models.Model):
    title = models.CharField(max_length=50, blank=False, null=False)
    description = models.TextField(blank=False, null=False)
    the_year = models.CharField(max_length=50, blank=False, null=False)

    def__str_(self):
        return self.title
class Experience(models.Model):
    title = models.CharField(max_length=50, blank=False, null=False)
    description = models.TextField(blank=False, null=False)
    the_year = models.CharField(max_length=50, blank=False, null=False)
    def__str_(self):
        return self.title
class Project(models.Model):
    title = models.CharField(max_length=200, blank=False, null=False)
    slug = models.SlugField(max_length=200, blank=True, null=True)
    description = RichTextField(blank=False, null=False)
    image = models.ImageField(upload_to="projects/", blank=False, null=False)
    tools = models.CharField(max_length=200, blank=False, null=False)
    demo = models.URLField()
    github = models.URLField()
    show_in_slider = models.BooleanField(default=False)

    def__str_(self):
        return self.title
   def  get_project_absolute_url(self):
        return "/projects/{}".format(self.slug)
    def save(self, *args, **kwargs):
```

```
        self.slug = self.slug_generate()
        super(Project, self).save(*args, **kwargs)
    def slug_generate(self):
        slug = self.title.strip()
        slug = re.sub(" ", "_", slug)
        return slug.lower()
class Message(models.Model):
    name = models.CharField(max_length=100, null=False, blank=False)
    email = models.EmailField(max_length=255, null=False, blank=False)
    message = models.TextField(null=False, blank=False)
    send_time = models.DateTimeField(auto_now_add=True)
    is_read = models.BooleanField(default=False)
    def__str_(self):
        return self.name
```

Business Logic(views.py)

```
from django.shortcuts import render, get_object_or_404, HttpResponse
from django.http import JsonResponse
from django.core import serializers
import json
from django.db.models import Q
from decouple import config
from django.core.mail import send_mail
from django.conf import settings
from info.forms import MessageForm
from info.models import (
    Competence,
    Education,
    Experience,
```

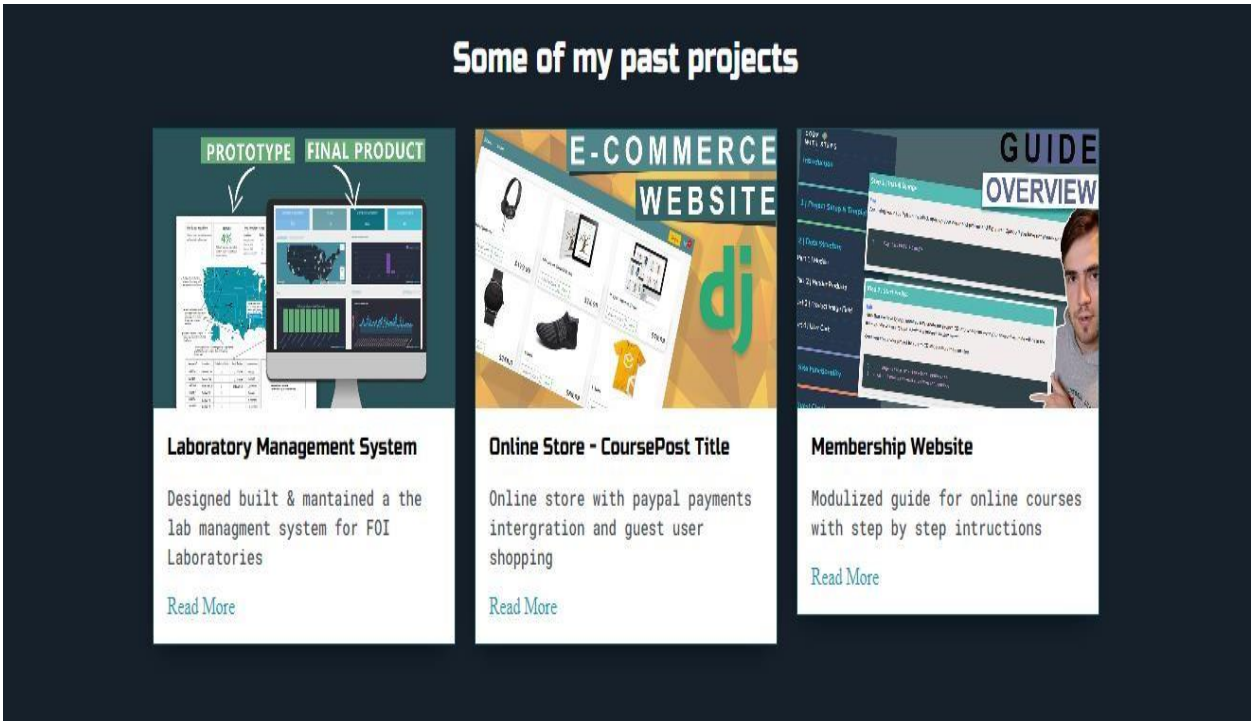
```
    Project,

    Information,

    Message

)

def email_send(data):

    old_message = Message.objects.last()

    if old_message.name == data['name'] and old_message.email == data['email'] and
old_message.message == data['message']:

        return False

    subject = 'Portfolio : Mail from {}'.format(data['name'])

    message = '{}\nSender Email: {}'.format(data['message'], data['email'])

    email_from = settings.EMAIL_HOST_USER

    recipient_list = [settings.EMAIL_HOST_USER, ]

    send_mail(subject, message, email_from, recipient_list)

    return True

def homePage(request):

    template_name = 'homePage.html'

    context = {}

    if request.method == 'POST':

        if request.POST.get('rechaptcha', None):

            form = MessageForm(request.POST)

            if form.is_valid():

                form.save(commit=False)
                data = {
                    'name': request.POST['name'],
                    'email': request.POST['email'],
                    'message': request.POST['message']
                }

                if email_send(data):
```

```
            form.save()
        return JsonResponse({'success': True})
    else:
        return JsonResponse({'success': False, 'errors': form.errors})
    return JsonResponse({'success': False, 'errors': "Oops, you have to check the recaptcha !"})
if request.method == 'GET':
    form = MessageForm()
    competences = Competence.objects.all().order_by('id')
    education = Education.objects.all().order_by('-id')
    experiences = Experience.objects.all().order_by('-id')
    projects = Project.objects.filter(show_in_slider=True).order_by('-id')
    info = Information.objects.first()
    context = {
        'info': info,
        'competences': competences,
        'education': education,
        'experiences': experiences,
        'projects': projects,
        'form': form,
        'recaptcha_key': config("recaptcha_site_key", default="")
    }
return render(request, template_name, context)
```

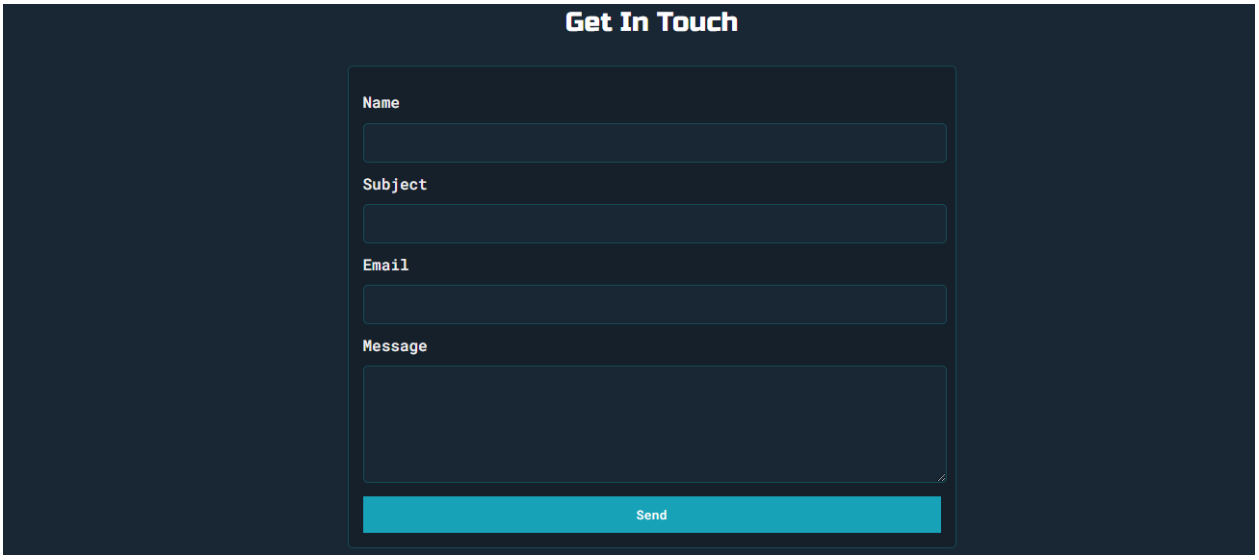
Output:

Homepage:

About Projects:



Contact:



RESULT:

Thus portfolio website which gives details about yourself for a potential recruiter is successfully executed.

Ex.No:2

Create a web application to manage the TO-DO list of users,where users can login and manage their to-do items

Aim: To create full stack TODO application with users authentication and Updation

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin startproject projectname
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using djongo in settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
    }
}
```

- The models.py file contains code for database design using django object relational mapper(ORM)
- After models creation use python manage.py makemigrations and python manage.py migrate for table creations
- In views.py the business logics will take place template folder contain all html file and all static files placed inside static folder

<u>Front End Code:</u>

Index.html:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1" />
   <title>TODO APP</title>
   <!-- Bootstrap CSS -->
   <link      href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
rel="stylesheet">
   <link type="text/css" rel="stylesheet" href="{% static '/css/style.css' %}">
   <link rel="stylesheet" href="{% static '/css/login.css' %}">
   <link rel="stylesheet" href="{% static '/css/register.css' %}">
</head>
```

```
<body>
   <nav class="navbar">
            <p class="navbar-brand">TODO APP</p>

      {% if user.is_authenticated %}
         <p id="welcome-user">Welcome, {{ request.user.username }}</p>

         <ul>
            <li class="navbar-item">
               <a class="link" href="{% url 'logout' %}">Logout</a>
            </li>
         </ul>
      {% else %}
         <ul>
            {% if request.path == '/register/' %}
               <li>
                  <a class="link" href="{% url 'login' %}">Login</a>
               </li>
            {% elif request.path == '/login/' %}
               <li>
                  <a class="link" href="{% url 'register' %}">Register</a>
               </li>
            {% endif %}
         </ul>
      {% endif %}
         </nav>
   {% block content %}
   {% endblock %}
   <!-- Below jquery javascript is required for modal functionalities -->
   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
   <!—
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script
> -->
   <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</body>
</html>
```

Login.html

```
{% extends 'todo/index.html' %}
{% load crispy_forms_tags %}

{% block content %}
<div class="content-section">
```

```
    <form action="" method="POST">
        {% csrf_token %}
        <fieldset class="form-group">
            <legend>Login</legend>
            <hr>
            {{ form | crispy }}
        </fieldset>

        <div class="form-submit">
            <button class="login-submit-btn" type="submit">Login</button>
        </div>
        <hr>
        <div id="create-account">
            <medium class="text-muted">
                Don't Have an Account?
                <a href="{% url 'register' %}" class="register-link">Sign Up</a>
            </medium>
        </div>
    </form>
</div>
{% endblock %}
```

register.html

```
{% extends 'todo/index.html' %}
{% load crispy_forms_tags %}
{% block content %}

 <div class="register-content">
        <form action="" method="POST">
            {% csrf_token %}
            <fieldset class="register-form-group">
                <legend>Register</legend>
                <hr>
                {{ form | crispy }}
            </fieldset>
            <div class="form-submit">
                <button class="btn btn-success" type="submit">Sign Up</button>
            </div>
            <hr>
             <div class="login-account">
                <medium class="text-muted">
                    Already Have an Account?
                    <a href="{% url 'login' %}" class="login-link">Login</a>
                </medium>
            </div>
```

```
            </form>
        </div>
    {% endblock %}
```

Backend(models.py):

```
from django.db import models

from django.conf import settings

class TodoItem(models.Model):

    """

    Todo Item Model

    """

    name = models.CharField(max_length=100)

    created_on = models.DateTimeField(auto_now_add=True)

    updated_on = models.DateTimeField(auto_now=True)

    is_completed = models.BooleanField(default=False)

    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE,

        related_name="todo_item")

  class Meta:

        """

        Meta Information

        """

        app_label = "todo"

        db_table = "todo_item"

        verbose_name = "todo_item"

        verbose_name_plural = "todo_items"

   def__str_(self):

        return self.name
```

views.py:

```
login_required

def home(request):
```

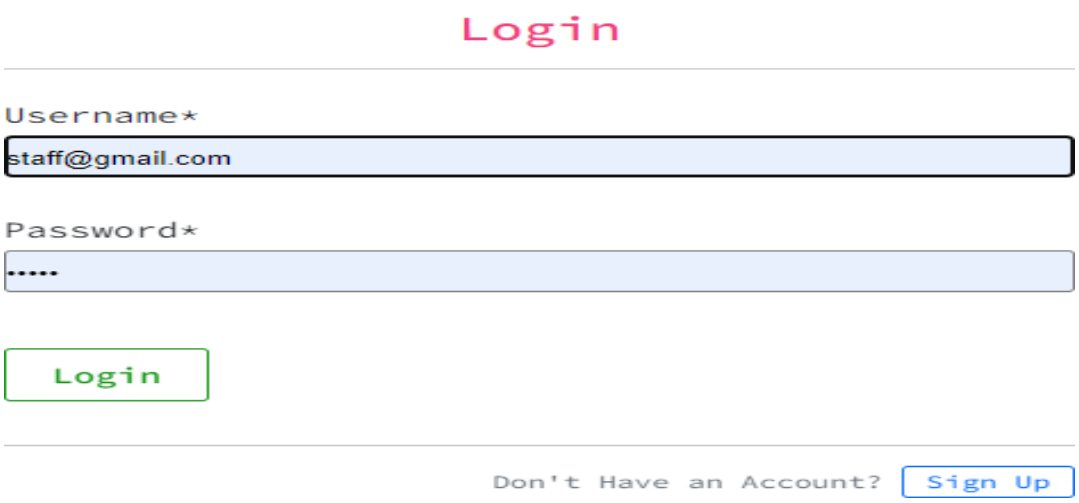
```
    """
    Create todo item and view other todo items as well.
    """
    if request.method == 'POST':
        todo_name = request.POST.get("new-todo")
        todo = TodoItem.objects.create(name=todo_name, user=request.user)
        return redirect("home")
   # todo items
    todos = TodoItem.objects.filter(user=request.user, is_completed=False).order_by("-id")
    # pagination 4 items per page
    paginator = Paginator(todos, 4)
    page_number = request.GET.get("page")
    page_obj = paginator.get_page(page_number)
    context = {"todos": todos, "page_obj": page_obj}

    # NOTE: Need to change the html file to crud.html for displaying the todo's
    return render(request, "todo/crud.html", context)
def register(request):
    """
    User Registration form
    Args:
        request (POST): New user registered
    """
    form = UserRegistrationForm()
    if request.method == "POST":
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            form.save()
```

```
        return redirect("login")
    else:
        form = UserRegistrationForm()
    context = {"form": form}
    return render(request, "todo/register.html", context)
```

Output:

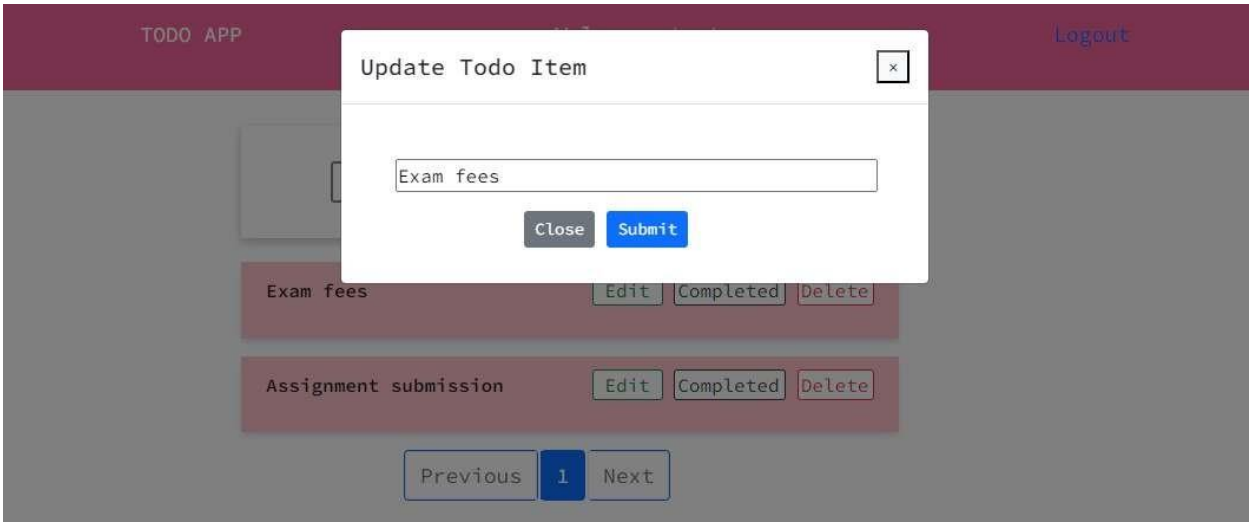
Register:

Login:



Add page:

Update:



RESULT:

Thus the web application to manage the TO-DO list of users, where users can login and manage their to-do items successfully executed.

Ex.No:3

Create a simple micro blogging application (like twitter) that allows people to post their content which can be viewed by people who follow them.

Aim: To create a simple micro blogging application with user post and follower comments on post

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin startproject projectname
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using djongo in settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
    }
}
```

- The models.py file contains code for database design using django object relational mapper(ORM)
- After models creation use python manage.py makemigrations and python manage.py migrate for table creations
- In views.py the business logics will take place template folder contain all html file and all static files placed inside static folder

<u>Frontend:</u>

Bloghome.html

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

{% load static %}

{% block content %}

    <!-- Hero Section-->

  <section style="background: url({% static 'img/inn.jpg'%}); background-size: cover; background-position: center center" class="hero">

    <div class="container">

      <div class="row">

        <div class="col-lg-7">

          <h1>Blog App</h1><a href="#" class="hero-link">Discover More</a>
```

```
        </div>
      </div><a href=".intro" class="continue link-scroll"><i class="fa fa-long-arrow-down"></i> Scroll Down</a>
    </div>
  </section>
  <!-- Intro Section-->
  <section class="intro">
    <div class="container">
      <div class="row">
        <div class="col-lg-8">
          <h2 class="h3">Some great intro here</h2>
          <p class="text-big">Place a nice <strong>introduction</strong> here <strong>to catch reader's attention</strong>. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderi.</p>
        </div>
      </div>
    </div>
  </section>
  <section class="featured-posts no-padding-top">
    <div class="container">
      <!-- Post-->
      {% for obj in object_list %}
      {% if forloop.counter < 5 %}
      <!-- Do your something here -->
        <div class="row d-flex align-items-stretch">
          {% if not forloop.first and not forloop.last %}
          <div class="image col-lg-5"><img src="{{obj.thumbnail.url}}" alt="..."></div>
          {% endif %}
        <div class="text col-lg-7">
```

```
<div class="text-inner d-flex align-items-center">
<div class="content">
<header class="post-header">
<div class="category">
{% for cat in obj.categories.all %}
<a href="#">{{cat}}</a>
{% endfor %}
</div>
<a href="{{ obj.get_absolute_url }}">
<h2 class="h4">{{obj.title}}</h2></a>
</header>
<p>{{obj.overview}}</p>
<footer class="post-footer d-flex align-items-center"><a href="#" class="author d-flex
align-items-center flex-wrap">
<div class="avatar"><img src="https://flyclipart.com/thumb2/avatar-my-profile-
profile-user-user-profile-icon-196366.png" alt="..." class="img-fluid"></div>

<div class="title"><span>{{obj.author}}</span></div></a>
<div class="date"><i class="icon-clock"></i> {{obj.timestamp | timesince}}
ago</div>
<div class="comments">
<i class="icon comment"></i>{{obj.comment_count}}</div>
</footer>
</div>
</div>
</div>
{% if forloop.first or forloop.last %}
<div class="image col-lg-5"><img src="{{obj.thumbnail.url}}" alt="..."></div>
{% endif %}
</div>
```

```
        {% endif %}

        {% endfor %}

        </div>

    </section>

    <!-- Divider Section-->

    <section style="background: url({% static 'img/divider-bg.jpg'%}); background-size: cover; background-position: center bottom" class="divider">

      <div class="container">

        <div class="row">

          <div class="col-md-7">

            <h2>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua</h2><a href="#" class="hero-link">View More</a>

          </div>
        </div>

      </div>

    </section>

    <!-- Latest Posts -->

    <section class="latest-posts">

      <div class="container">

        <header>

          <h2>Newly Updated Places</h2>

          <p class="text-big">.</p>

        </header>

        <div class="row">

          {% for obj in latest%}

          <div class="post col-md-4">

            <div class="post-thumbnail"><a href="{{ obj.get_absolute_url }}"><img src="{{ obj.thumbnail.url}}" alt="..." class="img-fluid" style="height: 300px;"></a></div>

            <div class="post-details">

              <div class="post-meta d-flex justify-content-between">
```

```
            <div class="date">{{obj.timestamp}}</div>
            <div class="category">
              {% for cat in obj.categories.all %}
              <a href="#">{{cat}}</a>
              {% endfor %}
            </div>
          </div><a href="{{ obj.get_absolute_url }}">
            <h3 class="h4">{{obj.title}}</h3></a>
          <p class="text-muted">{{obj.content}}</p>
        </div>
      </div>
     {% endfor %}
    </div>
   </div>
 </section>
  <!-- Gallery Section-->
  <section class="gallery no-padding">
   <div class="row">
    <div class="mix col-lg-3 col-md-3 col-sm-6">
      <div class="item"><a href="img/gallery-1.jpg" data-fancybox="gallery" class="image"><img src="{% static 'img/gallery-1.jpg'%}" alt="..." class="img-fluid">
          <div class="overlay d-flex align-items-center justify-content-center"><i class="icon-search"></i></div></a></div>
    </div>
    <div class="mix col-lg-3 col-md-3 col-sm-6">
      <div class="item"><a href="img/gallery-2.jpg" data-fancybox="gallery" class="image"><img src="{% static 'img/gallery-2.jpg'%}" alt="..." class="img-fluid">
          <div class="overlay d-flex align-items-center justify-content-center"><i class="icon-search"></i></div></a></div>
    </div>
```

```
        <div class="mix col-lg-3 col-md-3 col-sm-6">

          <div class="item"><a href="img/gallery-3.jpg" data-fancybox="gallery"
class="image"><img src="{% static 'img/gallery-3.jpg'%}" alt="..." class="img-fluid">

              <div class="overlay d-flex align-items-center justify-content-center"><i class="icon-
search"></i></div></a></div>
        </div>

        <div class="mix col-lg-3 col-md-3 col-sm-6">

          <div class="item"><a href="img/gallery-4.jpg" data-fancybox="gallery"
class="image"><img src="{% static 'img/gallery-4.jpg'%}" alt="..." class="img-fluid">

              <div class="overlay d-flex align-items-center justify-content-center"><i class="icon-
search"></i></div></a></div>

        </div>

      </div>

    </section>

      {% endblock content %}
```

Database(models.py):

```
class Category(models.Model):

    title = models.CharField(max_length=30)

    def__str_(self):

        return self.title

class Comment(models.Model):

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

    timestamp = models.DateTimeField(auto_now_add=True)

    content =tinymce_models.HTMLField('Content')

    post = models.ForeignKey(

        'Post', related_name='comments', on_delete=models.CASCADE)

    def__str_(self):

        return self.user.username

   class account(models.Model):

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

    email_id=models.CharField("email id",max_length=100)
```

```
    password=models.CharField("user password",max_length=100)
    def__str_(self):
        return self.name;
class Post(models.Model):
    title = models.CharField(max_length=100)
    overview = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)
    content =models.TextField(max_length=10000)
    # comment_count = models.IntegerField(default = 0)
    # view_count = models.IntegerField(default = 0)
    author = models.ForeignKey(Author, on_delete=models.CASCADE)
    thumbnail = models.ImageField()
    categories = models.ManyToManyField(Category)
    features = models.BooleanField()
    previous_post = models.ForeignKey(
        'self', related_name = 'previous', on_delete= models.SET_NULL, blank=True, null= True
    )
    next_post = models.ForeignKey(
        'self', related_name = 'next', on_delete= models.SET_NULL, blank=True, null= True
    )
    location=models.CharField(max_length=10000,null=True)
    phone=models.CharField(max_length=1000)
    timing=models.CharField(max_length=100)
    address=models.CharField(max_length=1000)
```

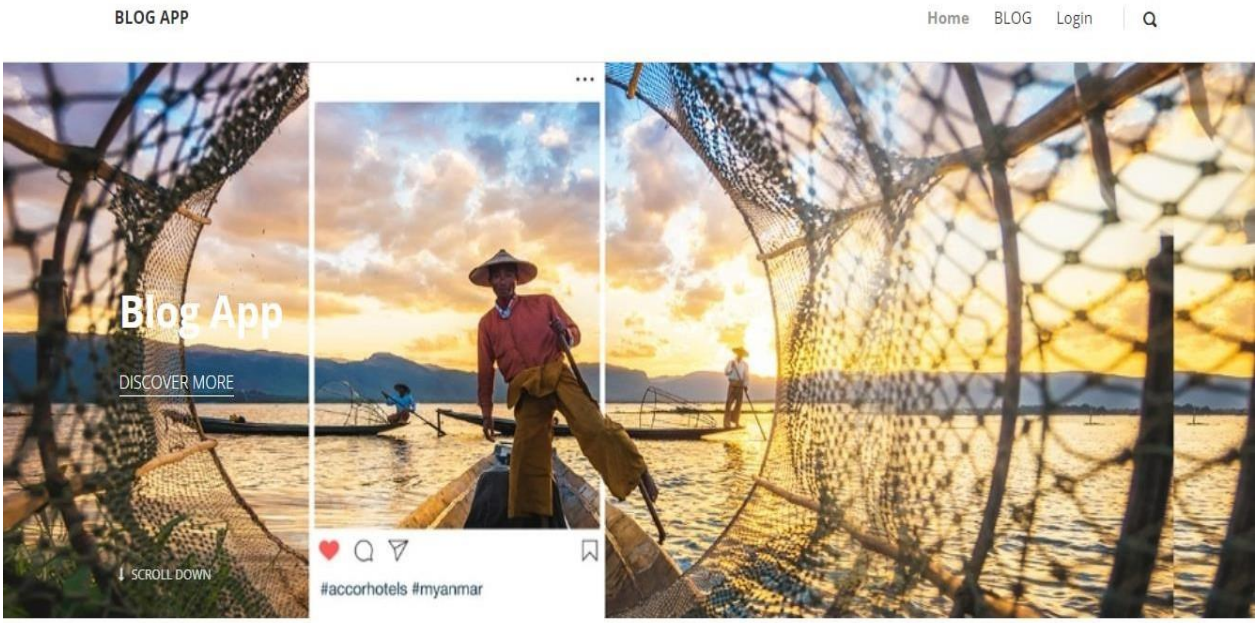
views.py:

```
def index(request):
    featured = Post.objects.all()
    latest = Post.objects.order_by('-timestamp')[0:3]
      if request.method == "POST":
        email = request.POST["email"]
        new_signup = Signup()
        new_signup.email = email
        new_signup.save()
    context = {
        'object_list' : featured,
        'latest': latest
    }
    return render(request, 'index.html',context)
def blog(request):
    category_count = get_category_count()
    most_recent = Post.objects.order_by('-timestamp')[:3]
    post_list = Post.objects.all()
    paginator = Paginator(post_list, 4)
    page_request_var = 'page'
    page = request.GET.get(page_request_var)
    try:
        paginated_queryset = paginator.page(page)
    except PageNotAnInteger:
        paginated_queryset = paginator.page(1)
    except EmptyPage:
        paginated_queryset = paginator.page(paginator.num_pages)
```

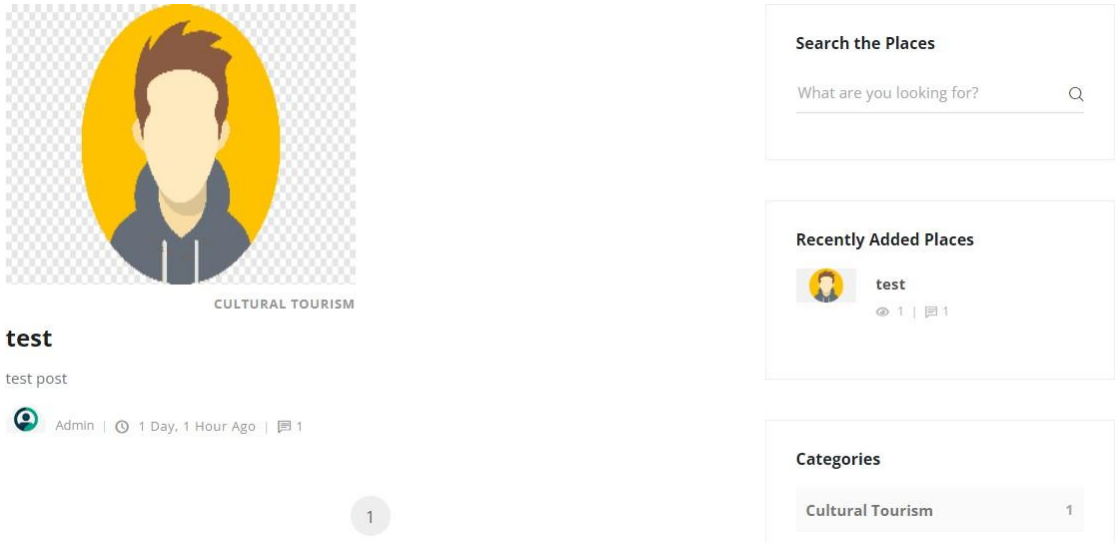
```
    context = {
    'queryset' : paginated_queryset,
    'most_recent': most_recent,
    'page_request_var': page_request_var,
    'category_count': category_count
}

return render(request, 'blog.html',context)
```

Output:

Home:



Display blog content:



Post comments:

RESULT:

Thus simple micro blogging application that allows people to post their content which can be viewed by people who follow them successfully executed.

Ex. No: 4

Create a food delivery website where users can order food from aparticular restaurant listed in the website.

Aim: To Create Full functional Food ordering application with users can select food and make order online.

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin startproject projectname
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using djongo in settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
    }
}
```

- The models.py file contains code for database design using django object relational mapper(ORM)
- After models creation use python manage.py makemigrations and python manage.py migrate for table creations
- In views.py the business logics will take place template folder contain all html file and all static files placed inside static folder

<u>Frontend code:</u>

Home.html:

```
<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

        {% load static %}

    <meta charset="utf-8">

    <link rel="stylesheet" type="text/css" href="{% static 'css/imperial_style.css' %}" />
```

```
    <link rel="stylesheet" type="text/css" href="{% static 'css/master.css' %}" />

    <link href="https://fonts.googleapis.com/css?family=Caveat:700&display=swap"
rel="stylesheet">

    <link
href="https://fonts.googleapis.com/css?family=Poppins:300,300i,400,400i,500,500i,600,600i,70
0,700i" rel="stylesheet">

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

    <link href="https://fonts.googleapis.com/css?family=Caveat:400,700&subset=cyrillic"
rel="stylesheet">

    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">

    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.3.1/css/all.css">

<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>

    <title>foodApp | Home</title>

        <link rel="shortcut icon" href="{% static 'images/favico.png' %}" type="image/x-icon">
```

```
  </head>
   <nav class="navbar navbar-expand-lg navbar-light" style="position:
fixed;width:100%;background-image: linear-gradient(180deg,#FFF,#FFF);box-shadow: 0px 0px
7px 0px #404040;">

     <a class="navbar-brand" href="/"><h2 style="color:black;font-family:sans-serif;font-
weight:bolder;"><b>Food Ordering App</b></h2></a>

     <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">

       <span class="navbar-toggler-icon"></span>

     </button>

     <div class="collapse navbar-collapse" id="navbarSupportedContent">

       <ul class="navbar-nav" style="margin-left:auto!important;margin-right:30px;">

         <li class="nav-item active">

           <a class="nav-link" id="nav-link" href="home">Home <span class="sr-
only">(current)</span></a>

         </li>

         {% if request.session.CustId is None and request.session.AdminId is not None %}

         <li class="nav-item active">

           <a class="nav-link" id="nav-link" href="addfoods">Add Food <span class="sr-
only">(current)</span></a>

         </li>

         {% endif %}

{% if request.session.CustId is not None and request.session.AdminId is None %}

             <a class="dropdown-item" href="getcustomer">Edit Profile</a>

           <div class="dropdown-divider"></div>

             <a class="dropdown-item" href="orders"><i class="fas fa-clipboard-list" style="font-
size:20px;"></i>   Orders</a>
```

```
            <a class="dropdown-item" href="orders"><i class="fas fa-history" style="font-
size:20px;"></i>  Order History</a>

          <div class="dropdown-divider"></div>

                              {% endif %}

            <a class="dropdown-item" style="background:red;width:1;border-radius:4px;"
href="logout"><i class="fas fa-sign-out-alt" style="font-
size:20px;color:white;"></i>  <font color="white">Logout</font></a>

          </div>

         </li>

         {% endif %}

       </ul>

     </div>

   </nav>

</body>

</html>
```

Database(models.py):

```
lass Food(models.Model)

          FoodId    = models.AutoField(primary_key=True)

          FoodName = models.CharField(max_length=30)

          FoodCat = models.CharField(max_length=30)

          FoodPrice = models.FloatField(max_length=15)

          FoodImage = models.ImageField(upload_to='media',default='')

          class Meta:

                    db_table = "FP_Food"

class Cust(models.Model):

          CustId    = models.AutoField(primary_key=True)

          CustFName  = models.CharField(max_length=30)
```

```
    CustLName  = models.CharField(max_length=30)
    CustCont = models.CharField(max_length=10)
    CustEmail = models.CharField(max_length=50)
    CustPass  = models.CharField(max_length=60)
    Address  = models.CharField(max_length=150,default='')
    class Meta:
        db_table = "FP_Cust"
```

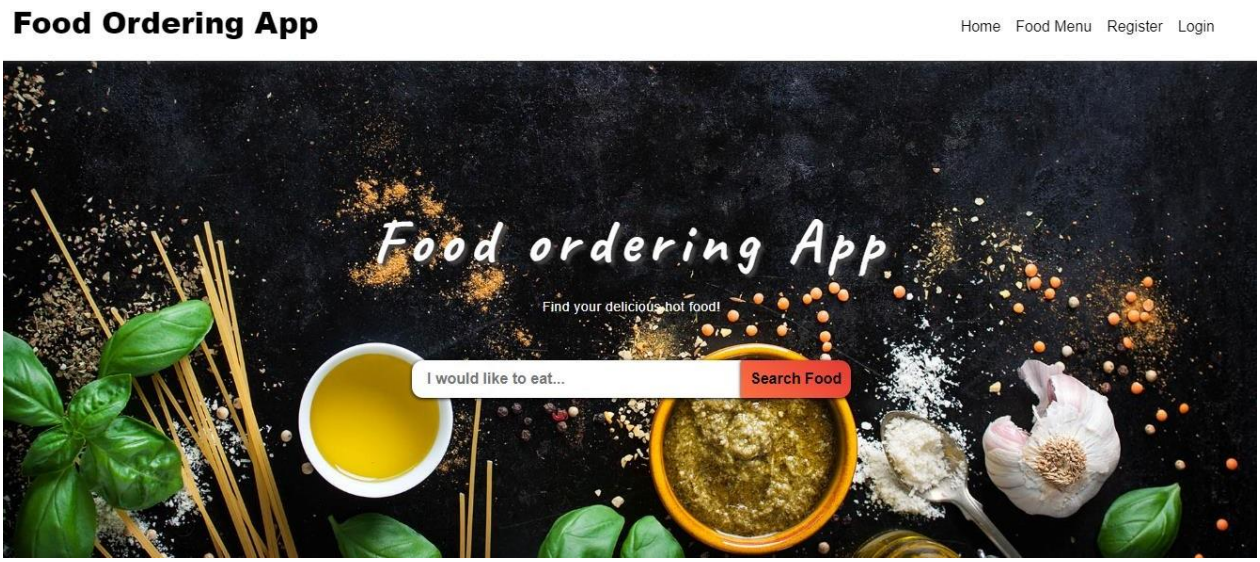
views.py:

```
def addfood(request):
    if request.method=="POST":
        form = FoodForm(request.POST,request.FILES)
        if form.is_valid():
            try:
                form.save()
                return redirect("/allfood")
            except:
                return render(request,"error.html")
            else :
        form = FoodForm()
    return render(request,'addfood.html',{'form':form})
    def showfood(request):
    foods = Food.objects.all()
    return  render(request,'foodlist.html',{'foodlist':foods})
    def deletefood(request,FoodId):
    foods = Food.objects.get(FoodId=FoodId)
```

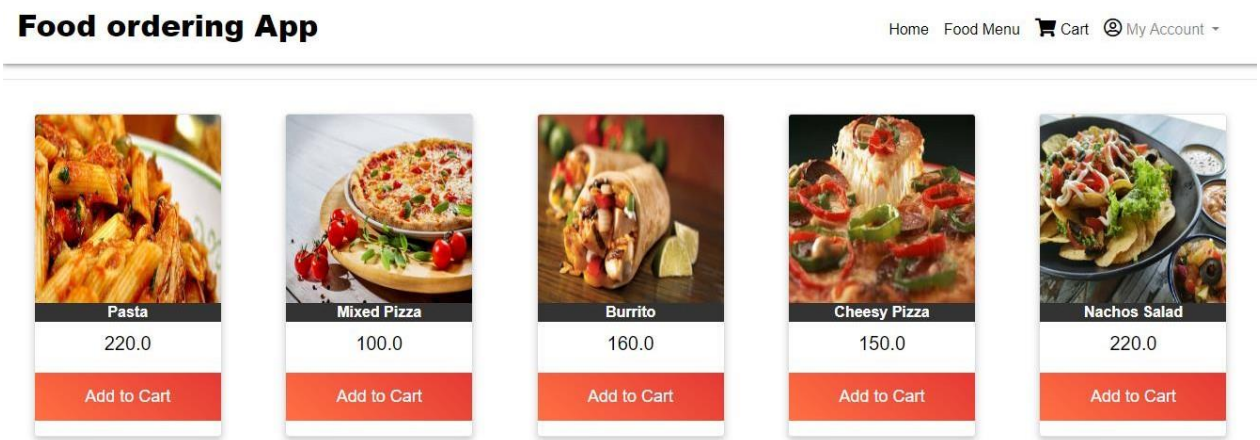
```
foods.delete()
return redirect("/allfood")
def getfood(request,FoodId):
foods = Food.objects.get(FoodId=FoodId)
return render(request,'updatefood.html',{'f':foods})
def updatefood(request,FoodId):
foods = Food.objects.get(FoodId=FoodId)
form = FoodForm(request.POST,request.FILES,instance=foods)
if form.is_valid():
        form.save()
        return redirect("/allfood")
return render(request,'updatefood.html',{'f':foods})
def addcust(request):
if request.method=="POST":
        form = CustForm(request.POST)
        if form.is_valid():
                try:
                        form.save()
                        return redirect("/login")
                except:
                        return render(request,"error.html")
                else :
        form = CustForm()
return render(request,'addcust.html',{'form':form})
```

Output:

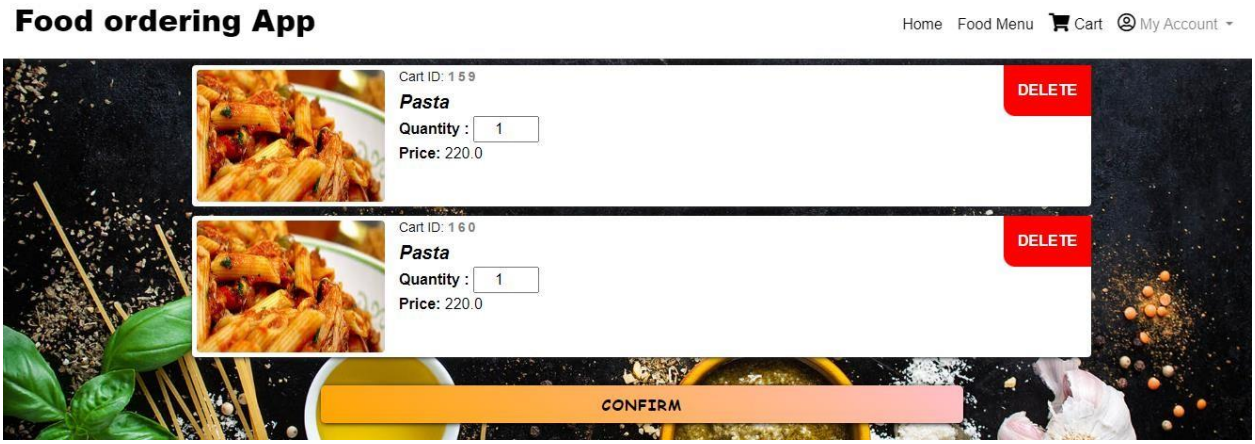
Home Page:



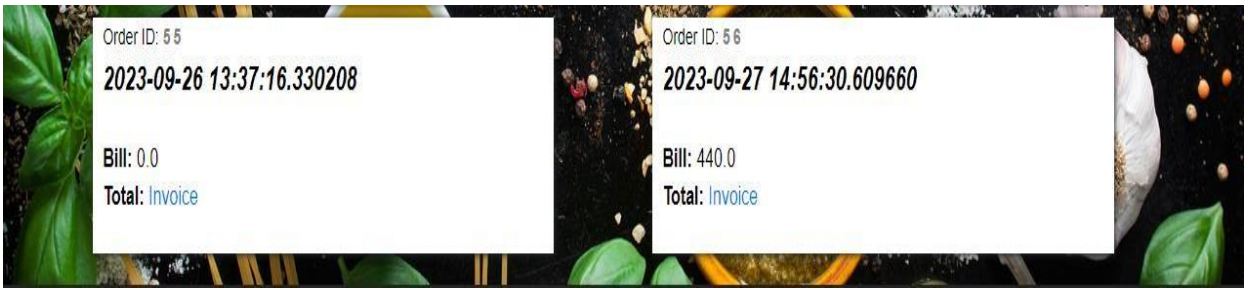
Menu Page:



Cart:



Invoice:



RESULT:

Thus food delivery website where users can order food from a particular restaurant listed in the website is successfully executed.

Ex.No: 5

Develop a classifieds web application to buy and sell used products.

Aim: To create a full stack we application where user can sell their products and can purchase the products online.

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin startproject projectname
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using djongo in settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
    }
}
```

- The models.py file contains code for database design using django object relational mapper(ORM)
- After models creation use python manage.py makemigrations and python manage.py migrate for table creations
- In views.py the business logics will take place template folder contain all html file and all static files placed inside static folder

<u>Frontend code:</u>

Base.html:

```
{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Test</title>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
```

```
crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="{% static 'store/style.css' %}">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Hello</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="{% url 'home' %}">Home <span class="sr-only">(current)</span></a>
            </li>
            {% if user.is_authenticated %}
                <li class="nav-item"><a class="nav-link" href="{% url 'product-create' %}">New product</a></li>
                <li class="nav-item"><a class="nav-link" href="{% url 'profile' %}">Profile</a></li>
                <li class="nav-item"><a class="nav-link" href="{% url 'logout' %}">Logout</a></li>
            {% else %}
                <li class="nav-item"><a class="nav-link" href="{% url 'login' %}">Login</a></li>
                <li class="nav-item"><a class="nav-link" href="{% url 'register'%}">Register</a></li>
            {% endif %}
            </ul>
        </div>
    </nav>
```

```
    <div class="container">
        <!--Displaying any flash message -->
        {% if messages %}
            {% for message in messages %}
                {{ message }}
            {% endfor%}
        {% endif %}
         <!--Main info block -->
        {% block content%} {% endblock %}
    </div>
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxM
fooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8
ifwB6" crossorigin="anonymous"></script>
</body>
```

Home.html

```
{% extends "store/base.html" %}
{% block content %}
    <h1 class="store-heading">Listed items </h1>
    <hr class="style1">
    {% for product in products %}
        <h2><a href="{% url 'product-detail' product.id %}">{{ product.name }} </a></h2>
        <h4 class="item-title"> Seller : {{ product.seller}}</h4>
        <h4 class="item-title"> Contact : {{ product.seller.email}}</h4>
        <h4 class="item-title"> Cost : {{ product.cost}}</h4>
```

```
        {% if product.image1 %}
            <img class="img-thumbnail" src={{ product.image1.url}} height=100/>
        {% endif %}
        {% if product.image2 %}
            <img class="img-thumbnail" src={{ product.image2.url}} height=100/>
        {% endif %}
        <hr>
    {% endfor %}
{% endblock content %}
```

Database(models.py):

```
from django.db import models
from django.contrib.auth.models import User
from django.urls import reverse
def user_directory_path(instance, filename):
    return 'uploads/user_{0}/{1}'.format(instance.user.id, filename)
class Product(models.Model):
    """Product model with seller as foreign key on user"""
    name =  models.CharField(max_length=30)
    description = models.TextField()
    age = models.FloatField()
    cost = models.PositiveIntegerField()
    address = models.CharField(max_length=100)
    seller = models.ForeignKey(User, on_delete=models.CASCADE)
    image1 = models.ImageField(upload_to='uploads/', null=True, blank=True)
    image2 = models.ImageField(upload_to='uploads/', null=True, blank=True)
    def_str_(self):
        return self.name
  #function to return url string for displaying product after its creation
```

```
    def get_absolute_url(self):
        return reverse('product-detail', kwargs={'pk':self.pk})
```

Views.py:

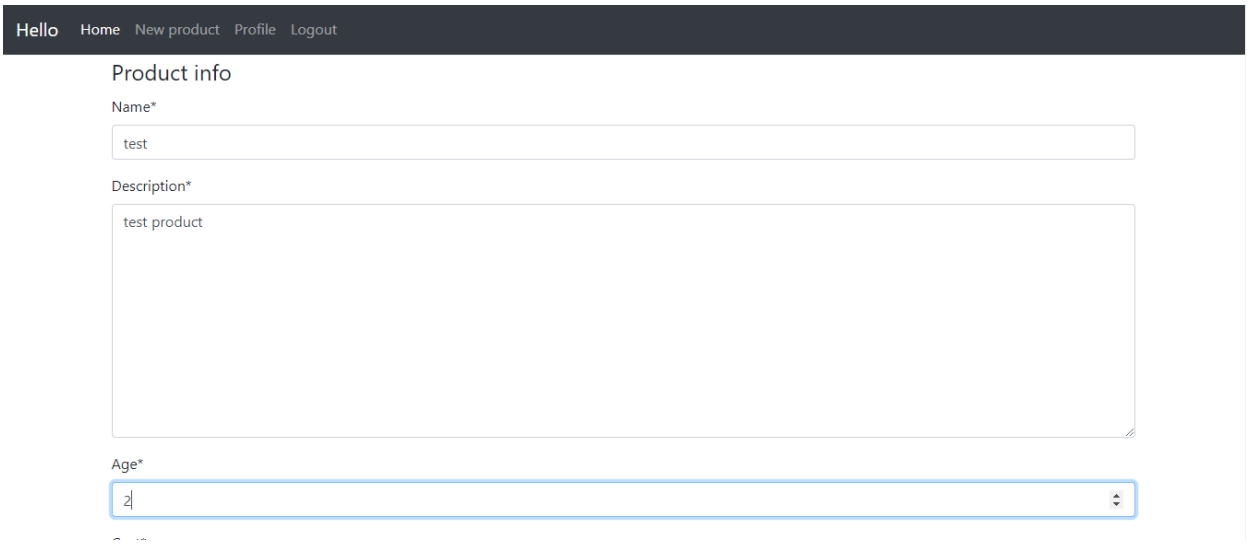
```
ef home(request):
    context = {
        'products' : Product.objects.all()
    }
    return render(request, 'store/home.html', context)
class ProductListView(ListView):
    """Class based view for displaying Product"""
    model = Product
    template_name = 'store/home.html'
    context_object_name = 'products'
   class ProductDetailView(DetailView):
    model = Product
class ProductCreateView(LoginRequiredMixin, CreateView):
    """Class based view for creating a new Product with login required mixin"""
    model = Product
    fields = ['name', 'description', 'age', 'cost', 'image1', 'image2']
        def form_valid(self, form):
        form.instance.seller = self.request.user
        return super().form_valid(form)
class ProductUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    """Class based view for updating a Product with login required mixin"""
    model = Product
    fields = ['name', 'description', 'age', 'cost', 'image1', 'image2']
        def form_valid(self, form):
        form.instance.seller = self.request.user
```

```
        return super().form_valid(form)
        # function for testing if user tying to udpate the product is seller itself
    def test_func(self):
        product = self.get_object()
        if self.request.user == product.seller:
            return True
        return False
class ProductDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = Product
    success_url = '/'  #setting success url to home
    # function for testing if user tying to delete the product is seller itself
    def test_func(self):
        product = self.get_object()
        if self.request.user == product.seller:
            return True
        return False
  @login_required
def BuyProduct(request, pk):
    """ View and logic for buying a product"""
    product = get_object_or_404(Product, pk=pk)
    user = request.user
    buyerInstance = User.objects.get(username=request.user.username)
    sellerInstance = User.objects.get(username=product.seller.username)
    #if seller tries to buy him own item
    if user == product.seller:
        return redirect('home')
    #wallet should contain more money than cost
    if user.profile.wallet >= product.cost:
```

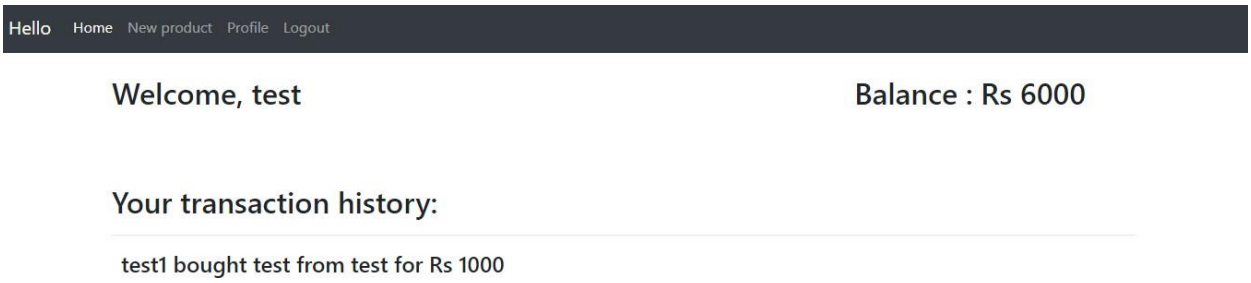
```
        user.profile.wallet -= product.cost

        product.seller.profile.wallet += product.cost

        new_trans = Transaction.objects.create(item=product.name, buyer=buyerInstance,
seller=sellerInstance, value=product.cost)

        user.save()

        product.seller.save()

        product.delete()

        messages.success(request, f'Item successfully bought!')

    else:

        messages.error(request, f'Item cannot be bought not enough money!')
    return redirect('home')
```

Output:

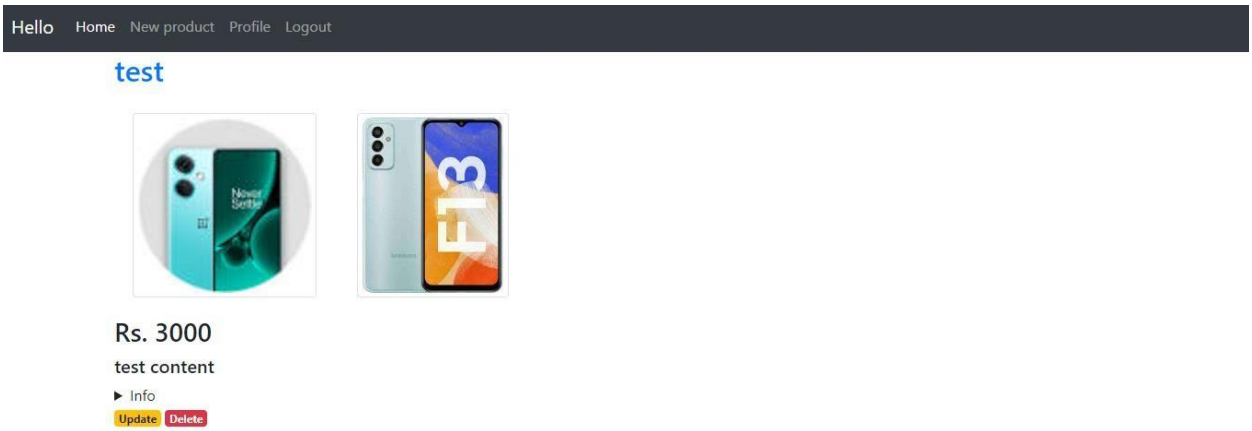
Post selling product:



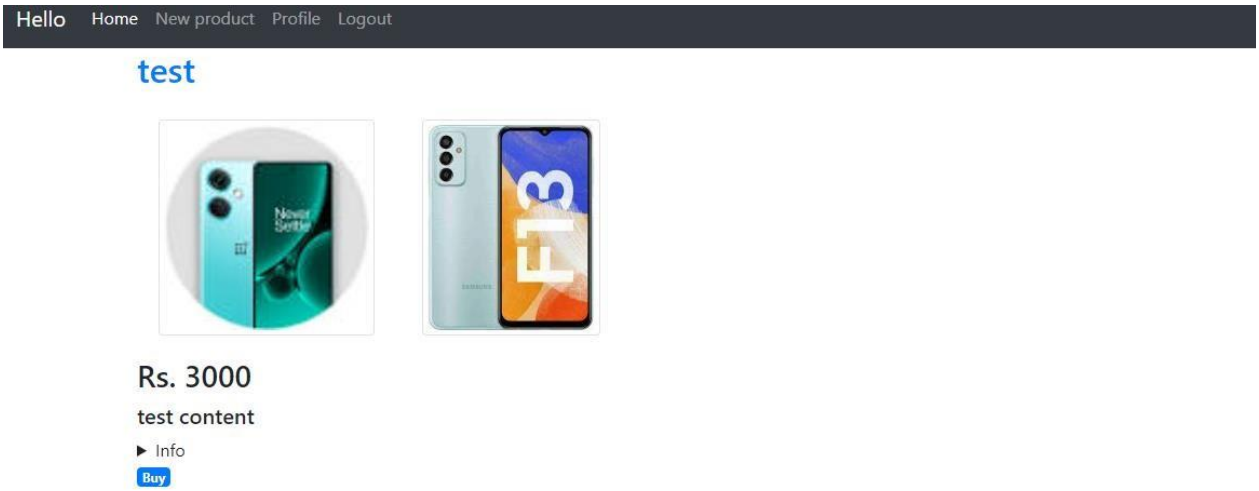
Wallet:



Listed products:



Buy Page:



RESULT:

Thus classifieds web application to buy and sell used products is successfully executed.

Ex.No: 6

Develop a leave management system for an organization where users can apply different types of leaves such as casual leave and medical leave. They also can view the available number of days.

Aim: To develop a employee leave management and approval system with tracking management using python django framework.

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin startproject projectname
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using djongo in settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
    }
}
```

- The models.py file contains code for database design using django object relational mapper(ORM)
- After models creation use python manage.py makemigrations and python manage.py migrate for table creations
- In views.py the business logics will take place template folder contain all html file and all static files placed inside static folder

<u>Frontend code:</u>

Index.html:

```
{% load static %}

<!doctype html>

<html lang="en">

<head>

    <meta charset="utf-8" />

    <link rel="icon" type="image/png" href="assets/img/favicon.ico">

    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

    <title>Leave Management</title>
```

```
        <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0' name='viewport' />
  <meta name="viewport" content="width=device-width" />
  <!-- Bootstrap core CSS     -->
  <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet" />
  <link href="{% static 'css/pe-icon-7-stroke.css' %}" rel="stylesheet" />
  <!-- Google Fonts -->
  <link href="https://fonts.googleapis.com/css?family=Anton|Russo+One" rel="stylesheet">
  <link href="https://fonts.googleapis.com/css?family=Luckiest+Guy" rel="stylesheet">
  <link href="https://fonts.googleapis.com/css?family=Alice" rel="stylesheet">
</head>
<body>
    <style type="text/css">
       body{
         position: relative;
         background: #243177;
       }
       a{
              font-variant: petite-caps;
              font-weight: 100;
              font-size: 16px;
       }
      navbar-brand{
         font-variant: petite-caps;
         font-family: 'Luckiest Guy', cursive;
         font-weight: 600;
         font-size: 3.1rem;
         color: #e4a530;
         text-shadow:1px 2px 1px rgba(0,0,0,0.2);
```

```
}
h2{
  font-family: 'Alice', serif;
  color: gold;
  font-size: 4.9rem;
}
p{
  font-variant: small-caps;
  color: #cececе;
}
p span{
  color: #a5a5a5;
  font-weight: bold;
}
span.icon{
  font-size: 15px;
}
.content{
  position: relative;
  height: 100vh;
}
.sub-title{
vertical-align: -webkit-baseline-middle;
font-size: 12px !important;
font-family: 'Luckiest Guy', cursive;
}
.back-link{
  color: #fff;
```

```
    font-size: 2.6rem;
    font-weight: 900;
    transition: all 400ms ease-in-out;
  }
  .back-link:hover{
    color: #fff;
    font-size: 2.8rem;
    transition: all 400ms ease-in-out;
  }
  .container-centered-items{
    position: absolute;
    top: 50%;
    left: 0;
    right: 0;
}
</style>
<!--CONTENTS-->
<section class="content">
    <!-- Navigation -->
        <nav class="navbar navbar-default">
            <div class="container-fluid">
              <!-- Brand and toggle get grouped for better mobile display -->
              <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                  <span class="sr-only"></span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
```

```
                    </button>

                    {% if request.user.is_authenticated %}

                    <a class="navbar-brand" href="{% url 'dashboard:dashboard' %}">Leave
Management<span class="sub-title">HRM</span></a>

                    {% else %}

                    <a class="navbar-brand" href="/">Leave Management</a>

                    {% endif %}

                  </div>

                  <!-- Collect the nav links, forms, and other content for toggling -->

                  <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

                    <ul class="nav navbar-nav">
                      <li class=""><a href="" target="_blank">Leave Management<span class="sr-
only">(current)</span></a></li>

                    </ul>

                    <ul class="nav navbar-nav navbar-right">

                      <li><a href="{% url 'accounts:login' %}">Login</a></li>

                    </ul>

                  </div><!-- /.navbar-collapse -->

                </div><!-- /.container-fluid -->

              </nav>

            <!-- /Naivagtion -->

            <div class="container-fluid container-centered-items">

              <section class="row">

                <section class="">

                  <section class="col col-lg-12 text-center">

                    <div class="center-me-please"

                      </p>

                    </div>

                  </section>
```

```
                </section>
            </section>
          </div>
    </section> <!-- /content -->
   <div>
      {% if request.user.is_authenticated %}
                <p style="position: absolute;bottom: 0;"><a
href="{{request.META.HTTP_REFERER|escape}}" class="back-link" title="back to previous
page"><i class="pe-7s-back-2"></i></a></p>
      {% endif %}
   </div>
</body>
   <!--   Core JS Files  -->
  <script src="{% static 'js/jquery.3.2.1.min.js' %}" type="text/javascript"></script>
          <script src="{% static 'js/bootstrap.min.js' %}" type="text/javascript"></script>
<script type="text/javascript">
          </script>
</html>
```

Database(Models.py):

```
import datetime
from employee.utility import code_format
from django.db import models
from employee.managers import EmployeeManager
from phonenumber_field.modelfields import PhoneNumberField
from django.utils.translation import ugettext as _
from django.contrib.auth.models import User
from leave.models import Leave
# Create your models here.
class Role(models.Model):
```

```
    name = models.CharField(max_length=125)

    description = models.CharField(max_length=125,null=True,blank=True)
    created = models.DateTimeField(verbose_name=_('Created'),auto_now_add=True)

    updated = models.DateTimeField(verbose_name=_('Updated'),auto_now=True)

    class Meta:

        verbose_name = _('Role')

        verbose_name_plural = _('Roles')

        ordering = ['name','created']

    def__str_(self):

        return self.name

class Department(models.Model):

    name = models.CharField(max_length=125)

    description = models.CharField(max_length=125,null=True,blank=True)

    created = models.DateTimeField(verbose_name=_('Created'),auto_now_add=True)

    updated = models.DateTimeField(verbose_name=_('Updated'),auto_now=True)

    class Meta:

        verbose_name = _('Department')

        verbose_name_plural = _('Departments')

        ordering = ['name','created']

     def__str__(self):

        return self.name

class Employee(models.Model):

    MALE = 'male'

    FEMALE = 'female'

    OTHER = 'other'

    NOT_KNOWN = 'Not Known'
```

```
GENDER = (
(MALE,'Male'),
(FEMALE,'Female'),
(OTHER,'Other'),
(NOT_KNOWN,'Not Known'),
)
MR = 'Mr'
MRS = 'Mrs'
MSS = 'Mss'
DR = 'Dr'
SIR = 'Sir'
MADAM = 'Madam'
TITLE = (
(MR,'Mr'),
(MRS,'Mrs'),
(MSS,'Mss'),
(DR,'Dr'),
(SIR,'Sir'),
(MADAM,'Madam'),
)
FULL_TIME = 'Full-Time'
PART_TIME = 'Part-Time'
CONTRACT = 'Contract'
INTERN = 'Intern'
```

```
    EMPLOYEETYPE = (

    (FULL_TIME,'Full-Time'),

    (PART_TIME,'Part-Time'),

    (CONTRACT,'Contract'),

    (INTERN,'Intern'),

    )

   # PERSONAL DATA

    user = models.ForeignKey(User,on_delete=models.CASCADE,default=1)

       image = models.FileField(_('Profile
Image'),upload_to='profiles',default='default.png',blank=True,null=True,help_text='upload
image size less than 2.0MB')#work on path username-date/image

    firstname = models.CharField(_('Firstname'),max_length=125,null=False,blank=False)

    lastname = models.CharField(_('Lastname'),max_length=125,null=False,blank=False)

    othername = models.CharField(_('Othername
(optional)'),max_length=125,null=True,blank=True)

    birthday = models.DateField(_('Birthday'),blank=False,null=False)

       department =  models.ForeignKey(Department,verbose_name
=_('Department'),on_delete=models.SET_NULL,null=True,default=None)

    role =  models.ForeignKey(Role,verbose_name
=_('Role'),on_delete=models.SET_NULL,null=True,default=None)

    startdate = models.DateField(_('Employement Date'),help_text='date of
employement',blank=False,null=True)

    employeetype = models.CharField(_('Employee
Type'),max_length=15,default=FULL_TIME,choices=EMPLOYEETYPE,blank=False,null=Tru
e)

    employeeid = models.CharField(_('Employee ID
Number'),max_length=10,null=True,blank=True)

    dateissued = models.DateField(_('Date Issued'),help_text='date staff id was
issued',blank=False,null=True)
```

```
# app related

 is_blocked = models.BooleanField(_('Is Blocked'),help_text='button to toggle employee block and unblock',default=False)

 is_deleted = models.BooleanField(_('Is Deleted'),help_text='button to toggle employee deleted and undelete',default=False)

 created = models.DateTimeField(verbose_name=_('Created'),auto_now_add=True,null=True)

 updated = models.DateTimeField(verbose_name=_('Updated'),auto_now=True,null=True)

 #PLUG MANAGERS

 objects = EmployeeManager()

class Meta:

    verbose_name = _('Employee')

    verbose_name_plural = _('Employees')

    ordering = ['-created']

 def__str__(self):

    return self.get_full_name

@property

 def get_full_name(self):

    fullname = ''

    firstname = self.firstname

    lastname = self.lastname

    othername = self.othername

    if (firstname and lastname) or othername is None:

        fullname = firstname +' '+ lastname

        return fullname

    elif othername:

        fullname = firstname + ' '+ lastname +' '+othername
```

```
            return fullname
        return
    @property
    def get_age(self):
        current_year = datetime.date.today().year
        dateofbirth_year = self.birthday.year
        if dateofbirth_year:
            return current_year - dateofbirth_year
        return
    @property
    def can_apply_leave(self):
        pass
    def save(self,*args,**kwargs):
        '''
        overriding the save method - for every instance that calls the save method
        perform this action on its employee_id
        added : March, 03 2019 - 11:08 PM
        '''
        get_id = self.employeeid #grab employee_id number from submitted form field
        data = code_format(get_id)
        self.employeeid = data #pass the new code to the employee_id as its orifinal or actual code
        super().save(*args,**kwargs) # call the parent save method
        # print(self.employeeid)
```

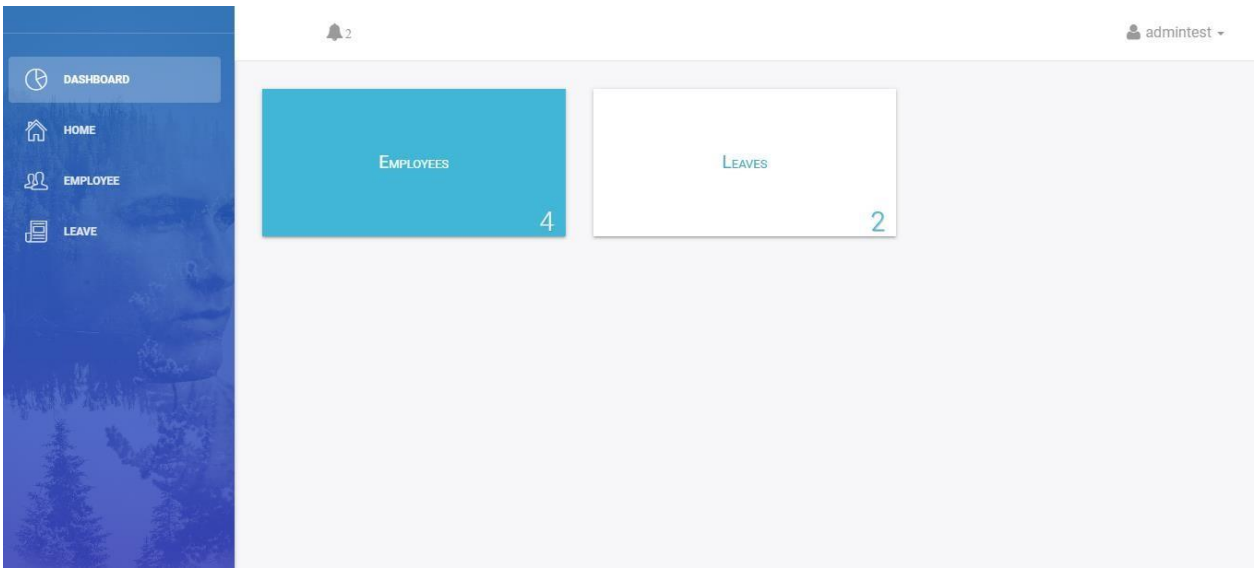
Views.py

```
ef dashboard_employees(request):
            if not (request.user.is_authenticated and request.user.is_superuser and
request.user.is_staff):
                        return redirect('/')
            dataset = dict()
```

```
    departments = Department.objects.all()

    employees = Employee.objects.all()

    #pagination

    query = request.GET.get('search')

    if query:

        employees = employees.filter(

            Q(firstname_icontains = query) |

            Q(lastname_icontains = query)

        )

    paginator = Paginator(employees, 10) #show 10 employee lists per page

    page = request.GET.get('page')

    employees_paginated = paginator.get_page(page)

    blocked_employees = Employee.objects.all_blocked_employees()

    return render(request,'dashboard/employee_app.html',dataset)

def dashboard_employees_create(request):

    if not (request.user.is_authenticated and request.user.is_superuser and
request.user.is_staff):

        return redirect('/')

    if request.method == 'POST':

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

        if form.is_valid():

            instance = form.save(commit = False)

            user = request.POST.get('user')

            assigned_user = User.objects.get(id = user)

            instance.user = assigned_user

            instance.title = request.POST.get('title')
            instance.image = request.FILES.get('image')
            instance.firstname = request.POST.get('firstname')
            instance.lastname = request.POST.get('lastname')
            instance.othername = request.POST.get('othername')
            instance.birthday = request.POST.get('birthday')
```

Output:

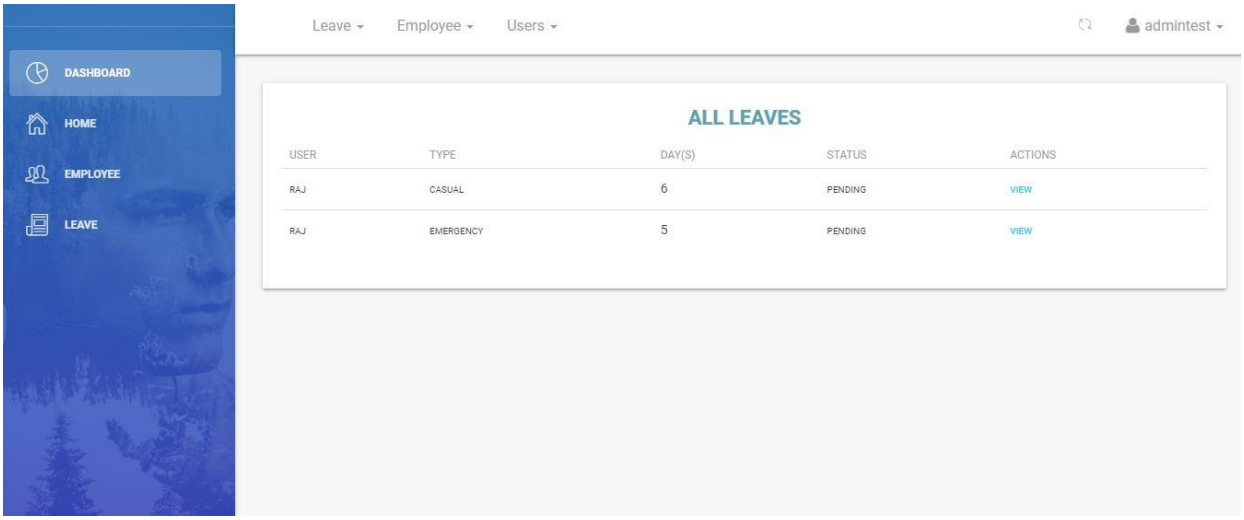
HR Page:



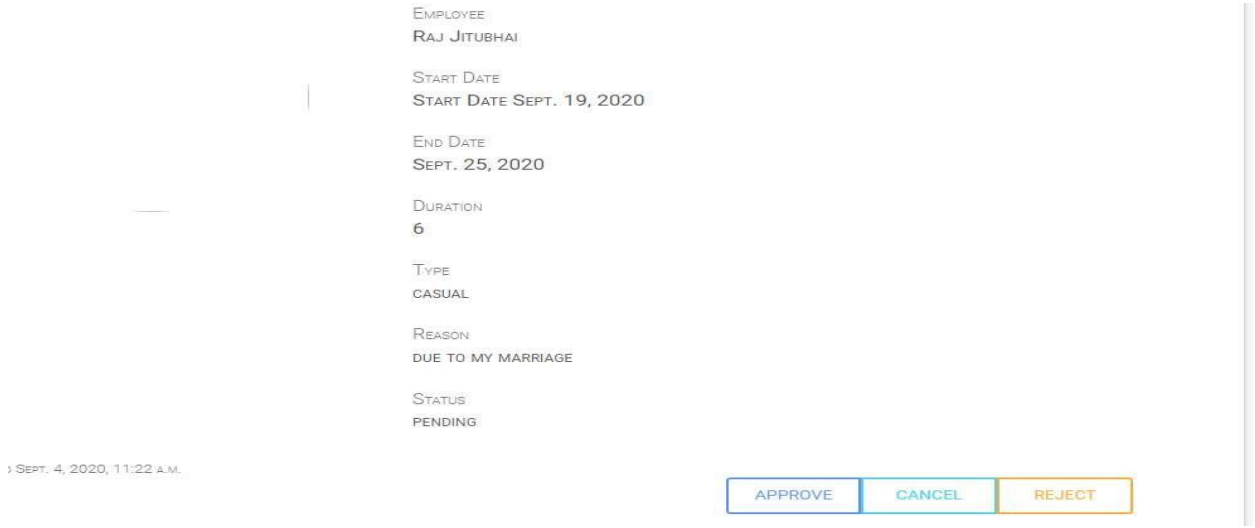
Add employee:

Apply for leave:

Pending leaves:



Approve leaves:



RESULT:

Thus a leave management system for an organization where users can apply different types of leaves such as casual leave and medical leave and they also can view the available number of days products is successfully executed.

Ex.No: 7

Develop a simple dashboard for project management where the statuses of various tasks are available. New tasks can be added and the status of existing tasks can be changed

among Pending, InProgress or Completed.

Aim: To develop a simple dashboard for project management with status of existing tasks can be changed among Pending, InProgress or Completed.

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin startproject projectname
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using djongo in settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
    }
}
```

- The models.py file contains code for database design using django object relational mapper(ORM)
- After models creation use python manage.py makemigrations and python manage.py migrate for table creations
- In views.py the business logics will take place template folder contain all html file and all static files placed inside static folder

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Project Management Dashboard</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
      background-color: #f4f4f4;
    }
    table {
      width: 100%;
      border-collapse: collapse;
      margin-bottom: 20px;
    }
    th, td {
      border: 1px solid #ddd;
      padding: 8px;
```

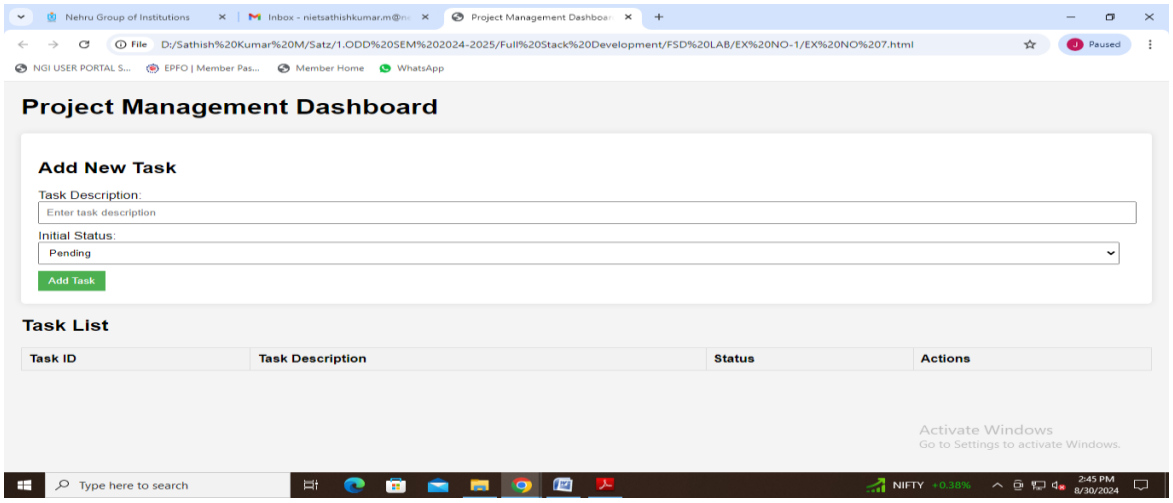
```
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
        button {
            padding: 8px 12px;
            border: none;
            color: #fff;
            cursor: pointer;
        }
        .btn-add {
            background-color: #4CAF50;
        }
        .btn-save {
            background-color: #2196F3;
        }
        .btn-delete {
            background-color: #f44336;
        }
        .form-group {
            margin-bottom: 10px;
        }
        select, input[type="text"] {
            padding: 8px;
            width: calc(100% - 18px);
        }
        .form-container {
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
            margin-bottom: 20px;
        }
    </style>
</head>
<body>

    <h1>Project Management Dashboard</h1>

    <div class="form-container">
        <h2>Add New Task</h2>
        <div class="form-group">
            <label for="new-task">Task Description:</label>
            <input type="text" id="new-task" placeholder="Enter task description">
        </div>
        <div class="form-group">
```

```
            <label for="new-status">Initial Status:</label>
            <select id="new-status">
                <option value="Pending">Pending</option>
                <option value="InProgress">In Progress</option>
                <option value="Completed">Completed</option>
            </select>
        </div>
        <button class="btn-add" onclick="addTask()">Add Task</button>
    </div>

    <h2>Task List</h2>
    <table id="task-table">
        <thead>
            <tr>
                <th>Task ID</th>
                <th>Task Description</th>
                <th>Status</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            <!-- Tasks will be dynamically added here -->
        </tbody>
    </table>

    <script>
        let taskIdCounter = 1;

        function addTask() {
            const taskDescription = document.getElementById('new-task').value;
            const taskStatus = document.getElementById('new-status').value;

            if (taskDescription.trim() === '') {
                alert('Task description cannot be empty.');
                return;
            }
            const table = document.getElementById('task-
table').getElementsByTagName('tbody')[0];
            const row = table.insertRow();
            row.setAttribute('data-id', taskIdCounter);

            row.innerHTML = `
                <td>${taskIdCounter}</td>
                <td>${taskDescription}</td>
                <td>
                    <select onchange="updateStatus(this)">
                        <option value="Pending"${taskStatus === 'Pending' ? ' selected' :
```

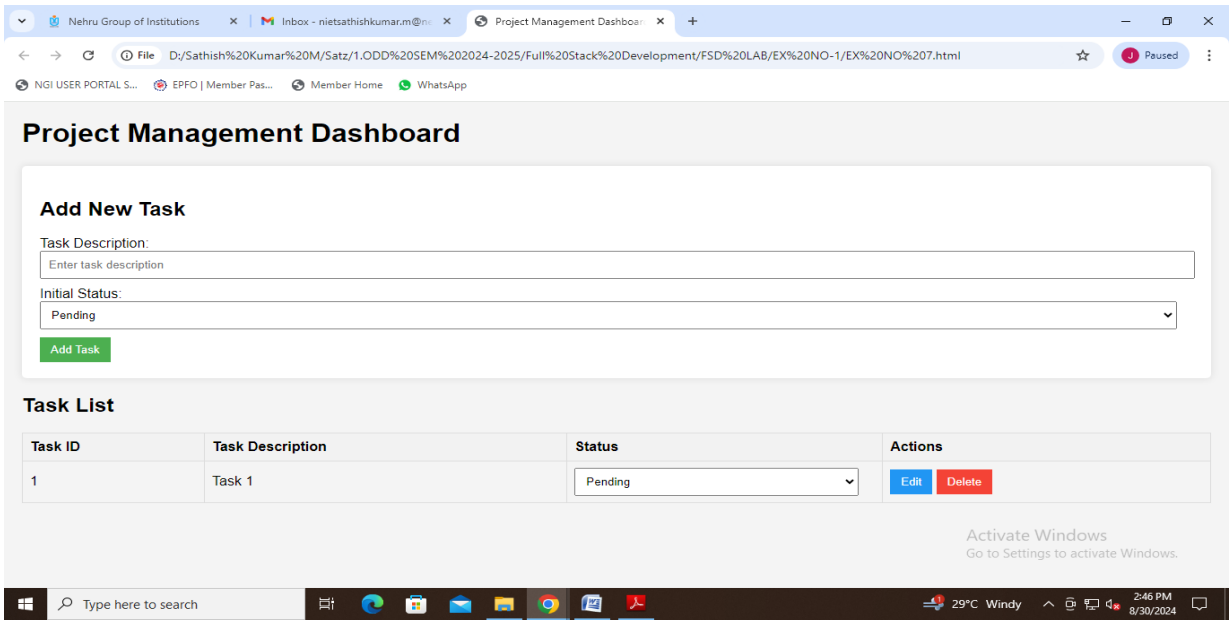
```
"}>Pending</option>
                <option value="InProgress"${taskStatus === 'InProgress' ? ' selected' : "}>In
Progress</option>
                <option value="Completed"${taskStatus === 'Completed' ? ' selected' :
"}>Completed</option>
            </select>
        </td>
        <td>
            <button class="btn-save" onclick="editTask(this)">Edit</button>
            <button class="btn-delete" onclick="deleteTask(this)">Delete</button>
        </td>
    `;
    taskIdCounter++;
    document.getElementById('new-task').value = '';
    document.getElementById('new-status').value = 'Pending';
}

function updateStatus(selectElement) {
    const status = selectElement.value;
    selectElement.closest('tr').querySelector('td:nth-child(3)').textContent = status;
}

function editTask(button) {
    const row = button.closest('tr');
    const taskDescription = row.cells[1].textContent;
    const status = row.querySelector('select').value;

    document.getElementById('new-task').value = taskDescription;
    document.getElementById('new-status').value = status;

    row.remove();
}

function deleteTask(button) {
    if (confirm('Are you sure you want to delete this task?')) {
        button.closest('tr').remove();
    }
}
</script>

</body>
</html>
```

Output:

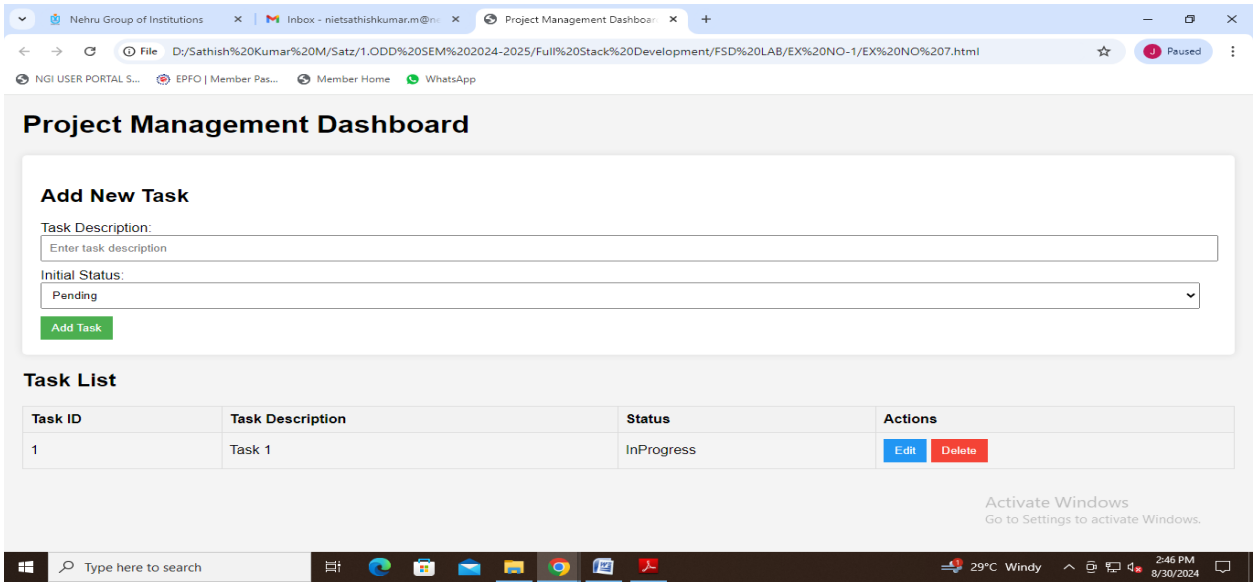
Project Management Dashboard



Task Added



Task Status



RESULT:

Thus a simple dashboard for project management where the statuses of various tasks are available, New tasks can be added and the status of existing tasks can be changed among Pending, InProgress or Completed is successfully executed.

Ex.No: 8

Develop an online survey application where a collection of questions is available and users are asked to answer any random 5 questions.

Aim: To develop an online survey application in which users are asked to answer any random 5 questions.

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin startproject projectname
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using djongo in settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'your-db-name',
    }
}
```

- The models.py file contains code for database design using django object relational mapper(ORM)
- After models creation use python manage.py makemigrations and python manage.py migrate for table creations
- In views.py the business logics will take place template folder contain all html file and all static files placed inside static folder

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Survey Application</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .survey-container {
            max-width: 600px;
            margin: 0 auto;
        }
        .question {
            margin-bottom: 15px;
```

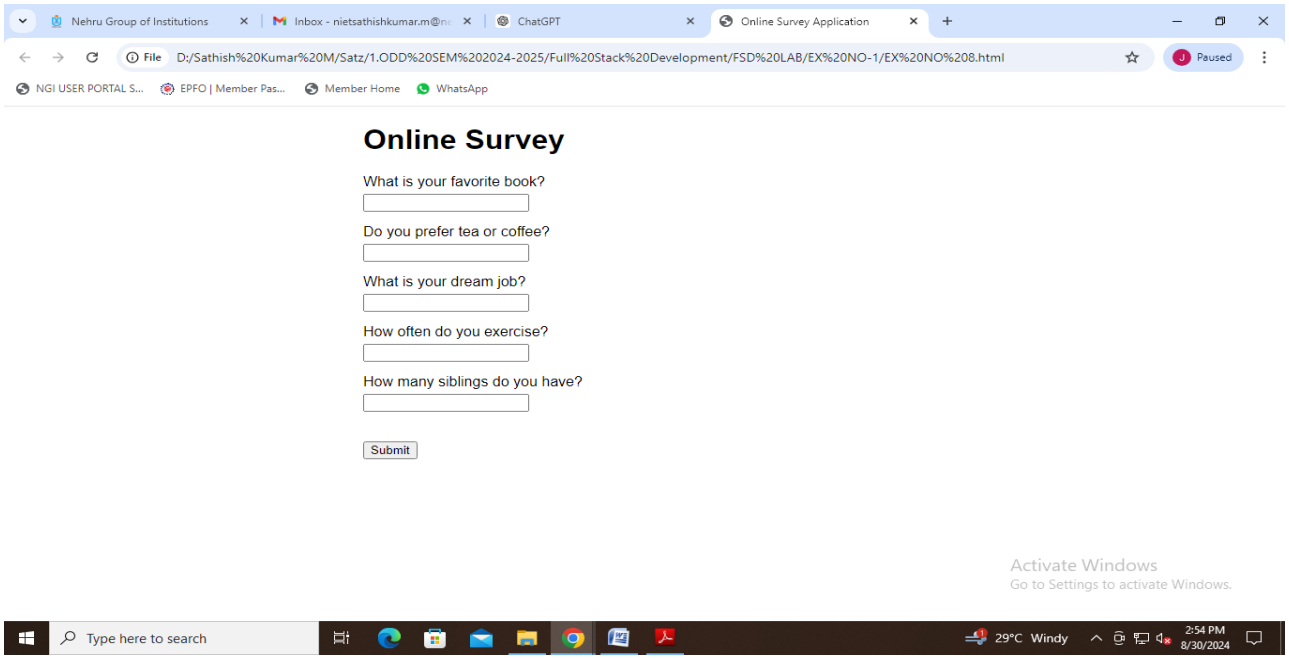
```
        }
        .question label {
            display: block;
            margin-bottom: 5px;
        }
        .question input {
            margin-right: 10px;
        }
        .submit-button {
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <div class="survey-container">
        <h1>Online Survey</h1>
        <form id="survey-form">
            <!-- Questions will be injected here -->
        </form>
        <button class="submit-button" type="button"
onclick="submitSurvey()">Submit</button>
    </div>

    <script>
        // Array of all possible questions
        const allQuestions = [
            "What is your favorite color?",
            "How many siblings do you have?",
            "What is your preferred mode of transportation?",
            "What type of music do you enjoy?",
            "What is your favorite book?",
            "Do you prefer tea or coffee?",
            "How often do you exercise?",
            "What is your dream job?",
            "What is your favorite cuisine?",
            "Where do you like to travel?"
        ];

        // Function to shuffle an array
        function shuffleArray(array) {
            for (let i = array.length - 1; i > 0; i--) {
                const j = Math.floor(Math.random() * (i + 1));
                [array[i], array[j]] = [array[j], array[i]];
            }
            return array;
        }
```

```
    // Function to get random questions
    function getRandomQuestions() {
        const shuffledQuestions = shuffleArray([...allQuestions]);
        return shuffledQuestions.slice(0, 5);
    }

    // Function to display questions on the page
    function displayQuestions() {
        const questions = getRandomQuestions();
        const form = document.getElementById('survey-form');
        form.innerHTML = '';

        questions.forEach((question, index) => {
            const questionDiv = document.createElement('div');
            questionDiv.className = 'question';

            const questionLabel = document.createElement('label');
            questionLabel.textContent = question;
            questionLabel.setAttribute('for', `question${index}`);

            const questionInput = document.createElement('input');
            questionInput.type = 'text';
            questionInput.id = `question${index}`;
            questionInput.name = `question${index}`;
            questionDiv.appendChild(questionLabel);
            questionDiv.appendChild(questionInput);

            form.appendChild(questionDiv);
        });
    }

    // Function to handle form submission
    function submitSurvey() {
        const form = document.getElementById('survey-form');
        const formData = new FormData(form);
        const responses = {};

        for (const [name, value] of formData.entries()) {
            responses[name] = value;
        }

        console.log('Survey Responses:', responses);
        alert('Thank you for completing the survey!');
    }
    // Initial setup
    window.onload = displayQuestions;
</script>
```

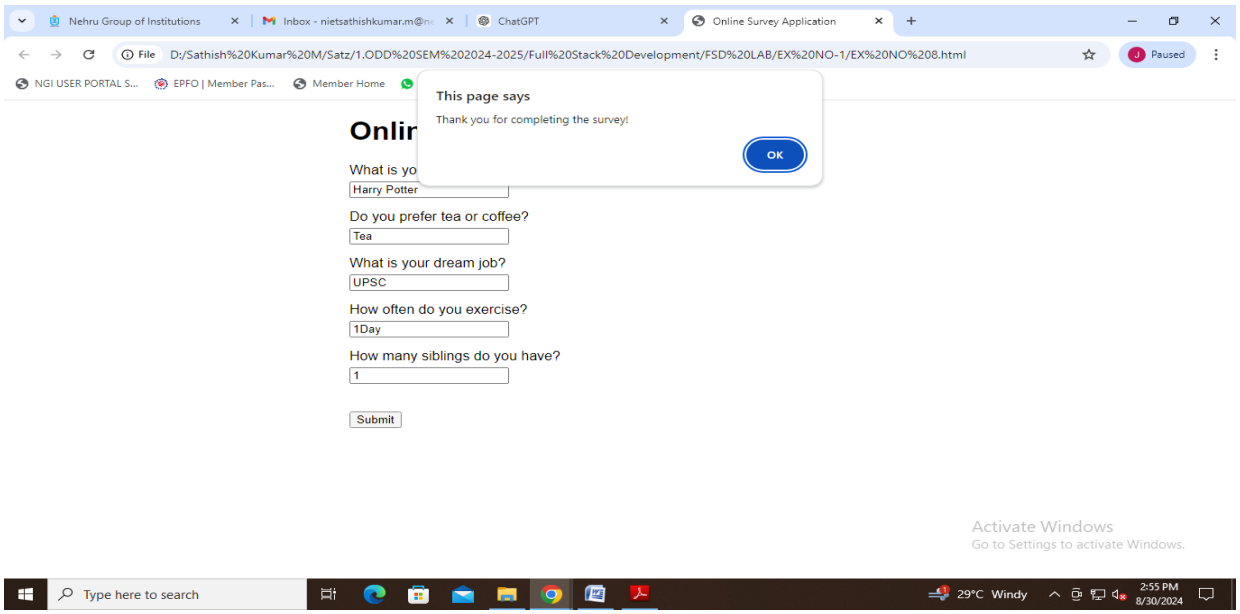
```
</body>
</html>
```

Output:

Online Survey



Survey Submission Form:



RESULT:

Thus an online survey application where a collection of questions is available and users are asked to answer any random 5 questions is successfully executed.

Topic Beyond Syllabus:

Develop a program to create and build a password strength check.

Aim: To develop a program to create and build a password strength checker.

Procedure:

- Open command prompt as administrator mode
- Then create project using django-admin startproject projectname
- Then go to project directory then create app using python manage.py start app app_name command
- Now configure database to connect with mongodb using djongo in settings.py
- The models.py file contains code for database design using django object relational mapper(ORM)

Program:

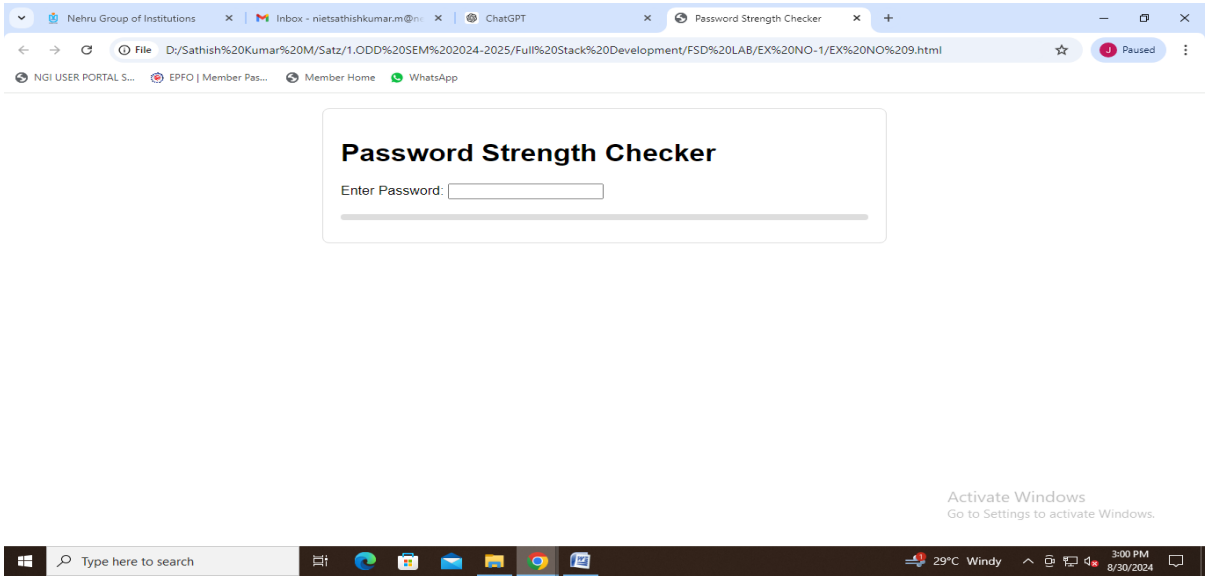
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Password Strength Checker</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .container {
            max-width: 600px;
            margin: 0 auto;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
        }
        .password-input {
            margin-bottom: 20px;
        }
        .strength-meter {
            margin-top: 10px;
            height: 8px;
            width: 100%;
            background-color: #ddd;
            border-radius: 4px;
        }
        .strength-meter span {
            display: block;
```

```
            height: 100%;
            border-radius: 4px;
        }
        .weak {
            background-color: red;
            width: 20%;
        }
        .medium {
            background-color: orange;
            width: 60%;
        }
        .strong {
            background-color: green;
            width: 100%;
        }
        .strength-message {
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Password Strength Checker</h1>
        <div class="password-input">
            <label for="password">Enter Password:</label>
            <input type="password" id="password" onkeyup="checkPasswordStrength()">
        </div>
        <div class="strength-meter">
            <span id="strength-bar"></span>
        </div>
        <div class="strength-message" id="strength-message"></div>
    </div>
    <script>
        function checkPasswordStrength() {
            const password = document.getElementById('password').value;
            const strengthBar = document.getElementById('strength-bar');
            const strengthMessage = document.getElementById('strength-message');
            let strength = 0;
            // Criteria checks
            const hasUpperCase = /[A-Z]/.test(password);
            const hasLowerCase = /[a-z]/.test(password);
            const hasNumber = /[0-9]/.test(password);
            const hasSpecialChar = /[!@#$%^&*(),.?":{}|<>]/.test(password);
            const lengthCriteria = password.length >= 8;
            if (lengthCriteria) strength += 1;
            if (hasUpperCase) strength += 1;
            if (hasLowerCase) strength += 1;
            if (hasNumber) strength += 1;
```

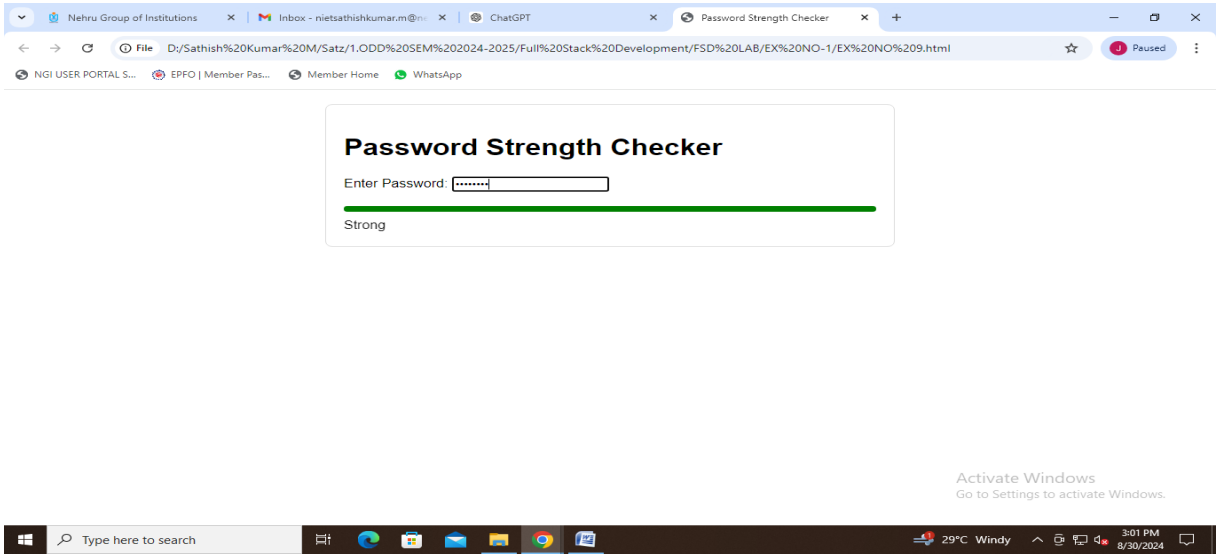
```
        if (hasSpecialChar) strength += 1;
        // Set strength bar and message
        switch (strength) {
          case 0:
          case 1:
            strengthBar.className = 'weak';
            strengthMessage.textContent = 'Very Weak';
            break;
          case 2:
          case 3:
            strengthBar.className = 'medium';
            strengthMessage.textContent = 'Moderate';
            break;
          case 4:
          case 5:
            strengthBar.className = 'strong';
            strengthMessage.textContent = 'Strong';
            break;
          default:
            strengthBar.className = '';
            strengthMessage.textContent = '';
            break;
        }
      }
    </script>
</body>
</html>
```

Output:

Password Strength Checker Front Page



Password Strength Checker



RESULT:

Thus a program to create and build a password strength check is successfully executed.