**Programme Code :**

**Settings.py:**

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'user\_app',

    'admin\_app',

    'agency\_app',

    'destination',

    'accomodation',

    'booking',

    'feedback',

    'community',

    'trip\_planning',

    'notification',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

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

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

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

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

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'travel\_voyage.urls'

TEMPLATES = [

    {

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

        'DIRS': [],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'travel\_voyage.wsgi.application'

# Database

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

DATABASES = {

    'default': {

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

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

    }

}

**Views.py:**

def index(request):

    user=request.user

    destinations = Destination.objects.all()[:3]

    activities = Activity.objects.prefetch\_related('activity\_feedbacks').all()[:3]  # Corrected related name

    accommodations = Accommodation.objects.prefetch\_related('accommodation\_feedbacks', 'images').all()[:3]  # Corrected related name

    bookings=[]

    unread\_count=0

    if user.is\_authenticated:

        bookings=Booking.objects.filter(user=request.user)

        unread\_count = Notification.objects.filter(

            Q(user=request.user) |  # Notifications for the current user

            # Notifications related to bookings made by the current user

            Q(user\_\_in=bookings.values\_list('accommodation\_\_travel\_id', flat=True))  # Notifications sent by the travel agency for the user's bookings

        ).filter(is\_read=False).count()

    context = {

        'destinations': destinations,

        'activities': activities,

        'accommodations': accommodations,

        'unread\_count': unread\_count,

    }

    return render(request, 'index.html', context)

def about(request):

    return render(request,'about.html')

def doLogin(request):

    form = LoginForm()

    if request.method == "POST":

        username = request.POST.get("username")

        password = request.POST.get("password")

        if not username or not password:

            messages.error(request, 'Username and password are required.', extra\_tags='log')

            return render(request, 'login.html', {'form': form})

        user = authenticate(request, username=username, password=password)

        if user is None:

            if not Register.objects.filter(username=username).exists():

                messages.error(request, 'Invalid username.', extra\_tags='reg')

            else:

                messages.error(request, 'Invalid password.', extra\_tags='reg')

            return render(request, 'login.html', {'form': form})

        # If the user is authenticated, check if they are a superuser

        if user.is\_superuser:

            # Ensure the Register entry is updated to reflect admin rights

            register\_entry, created = Register.objects.get\_or\_create(username=user.username, defaults={'usertype': 1})

            if not created and register\_entry.usertype != 1:  # Update existing record

                register\_entry.usertype = 1

                register\_entry.is\_approved = True

                register\_entry.save()

        # Perform login

        login(request, user)

        # Fetch user data from the Register table

        data = Register.objects.get(username=user.username)

        request.session['ut'] = data.usertype

        request.session['uid'] = data.id

        # Display login success message

        messages.success(request, f'Login Successful! Welcome {data.username}.', extra\_tags='log')

        return redirect('/')

    else:

        form = LoginForm()

    return render(request, 'login.html', {'form': form})

def logout(request):

    auth.logout(request)

    return redirect('/')

def user\_register(request):

    if request.method == 'POST':

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

        if form.is\_valid():

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

            if Register.objects.filter(email=email).exists():

                login\_form = LoginForm()

                return render(request, 'login.html', {'form': login\_form, 'z': True})

            else:

                try:

                    user = form.save(commit=False)

                    user.password = make\_password(form.cleaned\_data['password'])

                    user.usertype = 0

                    user.is\_approved = True

                    user.is\_active=True

                    user.status="approved"

                    user.save()

                    messages.success(request, f'Your registration has been successful! You can login now.', extra\_tags='log')

                    return redirect('/login')

                except Exception as e:

                    form.add\_error(None, f'An error occurred while saving the form: {e}')

        return render(request, 'register.html', {'form': form})

    else:

        form = UserRegisterForm()

        title='User'

    return render(request, 'register.html', {'form': form,'title':title})

def traveller\_register(request):

    if request.method == 'POST':

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

        if form.is\_valid():

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

            if Register.objects.filter(email=email).exists():

                login\_form = LoginForm()

                return render(request, 'login.html', {'form': login\_form, 'z': True})

            else:

                try:

                    user = form.save(commit=False)

                    user.password = make\_password(form.cleaned\_data['password'])

                    user.usertype = 2

                    user.is\_approved = False

                    user.is\_active=False

                    user.save()

                    messages.success(request, 'Your registration has been successful! You can login only after admin approval.', extra\_tags='log\_dr')

                    return redirect('/login')

                except Exception as e:

                    form.add\_error(None, f'An error occurred while saving the form: {e}')

        return render(request, 'register.html', {'form': form,'title':'Driver'})

    else:

        form = TravelAgencyRegisterForm()

        title='Travel Agency'

    return render(request, 'register.html', {'form': form,'title':title})

**urls.py:**

from django.contrib import admin

from django.urls import path

from . import views

from django.contrib.auth import views as auth\_views

from .views import  \*

urlpatterns = [

    path('',views.index),

    path('login',views.doLogin,name='login'),

    path('about/',views.about,name='about'),

    path('user\_register', views.user\_register, name='user\_register'),

    path('traveller\_register',views.traveller\_register,name='traveller\_register'),

    path('logout/',views.logout),

]

**Models.py:**

from django.db import models

from django.contrib.auth.models import AbstractUser

from django.utils import timezone

class Register(AbstractUser):

    usertype = models.IntegerField(default=0)

    phone = models.CharField(max\_length=10,null=True, blank=True)

    place = models.CharField(max\_length=300,null=True, blank=True)

    profile\_image = models.FileField(upload\_to='profile\_images/', null=True, blank=True)

    address = models.CharField(max\_length=300, null=True, blank=True)

    location = models.CharField(max\_length=200, null=True, blank=True)

    latitude = models.FloatField(null=True, blank=True)

    longitude = models.FloatField(null=True, blank=True)

    experience = models.TextField(null=True, blank=True)

    languages\_spoken = models.CharField(max\_length=200, null=True, blank=True)

    available = models.BooleanField(default=True)

    guide\_license = models.CharField(max\_length=100, null=True, blank=True)

    travel\_id = models.CharField(max\_length=100, null=True, blank=True)

    certifications = models.FileField(upload\_to='certifications/', null=True, blank=True)

    is\_approved = models.BooleanField(default=False)

    status = models.CharField(max\_length=50, default='pending')

    availability = models.CharField(max\_length=20, default='available')  # Add this field

**forms.py:**

from django.contrib.auth.forms import PasswordChangeForm

from django import forms

from django.core.validators import RegexValidator

from django.core.exceptions import ValidationError

from .models import Register

class UserRegisterForm(forms.ModelForm):

    confirm\_password = forms.CharField(

        widget=forms.PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Confirm Password'})

    )

    class Meta:

        model = Register

        fields = ['username', 'email', 'phone', 'place', 'profile\_image', 'password']

        widgets = {

            'username': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Username'}),

            'email': forms.EmailInput(attrs={'class': 'form-control', 'placeholder': 'Email'}),

            'password': forms.PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Password'}),

            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Full Name'}),

            'place': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Place'}),

            'phone': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Phone'}),

            'profile\_image': forms.ClearableFileInput(attrs={'class': 'form-control-file'}),

        }

        help\_texts = {'username': None}

    def clean\_phone(self):

        phone = self.cleaned\_data.get('phone')

        if not phone:

            raise forms.ValidationError("Phone number is required.")

        # Ensure phone is treated as a string

        phone = str(phone).strip()

        # Validate the phone number with regex

        phone\_validator = RegexValidator(

            regex=r'^[6-9]\d{9}$',

            message="Phone number must start with 6, 7, 8, or 9 and must be exactly 10 digits long."

        )

        try:

            phone\_validator(phone)

        except ValidationError as e:

            raise forms.ValidationError(e.message)

        # Ensure the phone is exactly 10 digits

        if len(phone) != 10:

            raise forms.ValidationError("Phone number must be exactly 10 digits long.")

        return phone

    def clean(self):

        cleaned\_data = super().clean()

        password = cleaned\_data.get("password")

        confirm\_password = cleaned\_data.get("confirm\_password")

        if password and confirm\_password and password != confirm\_password:

            raise forms.ValidationError("Passwords do not match.")

        return cleaned\_data

class TravelAgencyRegisterForm(forms.ModelForm):

    confirm\_password = forms.CharField(

        widget=forms.PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Confirm Password'})

    )

    class Meta:

        model = Register

        fields = ['username', 'email',  'phone', 'place', 'profile\_image', 'experience', 'password','certifications']

        widgets = {

            'username': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Username'}),

            'email': forms.EmailInput(attrs={'class': 'form-control', 'placeholder': 'Email'}),

            'password': forms.PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Password'}),

            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Agency Name'}),

            'place': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Location'}),

            'phone': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Phone'}),

            'profile\_image': forms.ClearableFileInput(attrs={'class': 'form-control-file'}),

            'experience': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Experience in Travel Industry'}),

        }

        help\_texts = {'username': None}

    def clean(self):

        cleaned\_data = super().clean()

        password = cleaned\_data.get("password")

        confirm\_password = cleaned\_data.get("confirm\_password")

        if password and confirm\_password and password != confirm\_password:

            raise forms.ValidationError("Passwords do not match.")

        return cleaned\_data

class GuideRegisterForm(forms.ModelForm):

    confirm\_password = forms.CharField(

        widget=forms.PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Confirm Password'})

    )

    class Meta:

        model = Register

        fields = [

            'username', 'email',  'phone', 'place', 'profile\_image', 'experience',

            'languages\_spoken', 'guide\_license', 'certifications', 'password'

        ]

        widgets = {

            'username': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Username'}),

            'email': forms.EmailInput(attrs={'class': 'form-control', 'placeholder': 'Email'}),

            'password': forms.PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Password'}),

            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Full Name'}),

            'place': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Location'}),

            'phone': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Phone'}),

            'profile\_image': forms.ClearableFileInput(attrs={'class': 'form-control-file'}),

            'experience': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Experience'}),

            'languages\_spoken': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Languages Spoken'}),

            'guide\_license': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'License Number'}),

            'certifications': forms.ClearableFileInput(attrs={'class': 'form-control-file'}),

        }

        help\_texts = {'username': None}

    def clean(self):

        cleaned\_data = super().clean()

        password = cleaned\_data.get("password")

        confirm\_password = cleaned\_data.get("confirm\_password")

        if password and confirm\_password and password != confirm\_password:

            raise forms.ValidationError("Passwords do not match.")

        return cleaned\_data

**templates:**

**register.html**

{% include 'header.html' %}

{% load static %}

<body style="background-image: url('{% static 'images/bg\_1.jpg' %}'); background-size: cover; background-position: center;">

<section class="ftco-section contact-section ftco-no-pt mt-5">

    <div class="overlay"></div>

    <div class="container mt-5" style="position: relative; z-index: 2;">

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

            <div class="col-md-5 mt-5 bg-transparent"> <!-- Adjusted column width -->

                <div class="card shadow-lg border-0 rounded" style="background:

rgba(30, 30, 30, 0.7);

  backdrop-filter: blur(15px);">

                    <div class="card-body p-4">

                        <h2 class="text-center text-white mb-4" style="font-weight: bold;">Register As {{ title }}</h2>

                        <form action="" method="post" class="request-form contact-form">

                            {% csrf\_token %}

                            <div class="form-group">

                              {{ form.as\_p }}

                            </div>

                            <div class="form-group mt-4">

                                <input type="submit" value="Register" class="btn btn-primary btn-block py-3 px-4">

                            </div>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

</section>

<style>

    label{

        color: #fff;

    }

</style>

</body>

{% include 'footer.html' %}

**Login.html:**

{% include 'header.html' %}

{% load static %}

<body style="background-image: url('{% static 'images/bg\_1.jpg' %}'); background-size: cover; background-position: center;">

    <section class="ftco-section contact-section ftco-no-pt mt-5">

        <div class="overlay"></div>

        <div class="container mt-5" style="position: relative; z-index: 2;">

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

                <div class="col-md-5 mt-5 bg-transparent"> <!-- Adjusted column width -->

                    <div class="card shadow-lg border-0 rounded" style="background:

rgba(30, 30, 30, 0.7);

  backdrop-filter: blur(15px);">

                        <div class="card-body p-4">

                            <h2 class="text-center text-white mb-4" style="font-weight: bold;">Login</h2>

                            <form action="" method="post" class="request-form contact-form">

                                {% csrf\_token %}

                                <div class="form-group">

                                  {{ form.as\_p }}

                                </div>

                                <div class="form-group mt-4">

                                    <input type="submit" value="Login" class="btn btn-primary btn-block py-3 px-4">

                                </div>

                            </form>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </section>

    <style>

        label{

            color: #fff;

        }

    </style>

    </body>

    {% include 'footer.html' %}