I want to start on anew django project call TMS with app call tour. This application will be use by the tour travel planner . After login , he/she should be able to choose enter different location , i.e KISUMU, MOMBASA, NAIROBI. The location or destination locations should have this details :

1. Name

2: date to be visit i.e 1jan 2025 -6th january 2025

3. small description of aplace

4. Gallery. should have 4-5 gallery images , which can be scrool left to right or right to left . Display one image at a time

5. map of place

6. Country

7. Rooms with images

8. Restaurants availble around with images and name

9.Activities and service

10. Innformation

note : Gallery, Rooms, Restaurants, Activities and service and Innformation should be tabs on top as the reference of that image upbove :

all this data should be enter by tour travel planner after register and login. so we should first setup the logins and register page , then tour travel plan dashboard to enter those details. Remember this information is for one location but we can have more than 5 locations:

corrent project structure :

TMS/

├── manage.py

├── requirements.txt

├── media/ # Uploaded files (gallery, rooms, restaurants)

├── static/ # (optional if you add custom CSS/JS later)

├── templates/ # Shared templates here

│ ├── base.html

│ └── tour/

│ ├── destination\_list.html

│ └── destination\_detail.html

├── TMS/

│ ├── \_\_init\_\_.py

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

├── tour/

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── urls.py

│ ├── views.py

│ └── migrations/

styling need to be professional

my current models look like this :  
if need any changes please update accordingly :  
  
#tour/models

from django.db import models

class Destination(models.Model):

    name = models.CharField(max\_length=100)

    country = models.CharField(max\_length=100)

    start\_date = models.DateField()

    end\_date = models.DateField()

    description = models.TextField()

    map\_embed\_code = models.TextField(help\_text="Embed iframe from Google Maps")

    def \_\_str\_\_(self):

        return self.name

class DestinationImage(models.Model):

    destination = models.ForeignKey(Destination, on\_delete=models.CASCADE, related\_name='images')

    image = models.ImageField(upload\_to='destination\_gallery/')

class Room(models.Model):

    destination = models.ForeignKey(Destination, on\_delete=models.CASCADE, related\_name='rooms')

    name = models.CharField(max\_length=100)

    image = models.ImageField(upload\_to='rooms/')

class Restaurant(models.Model):

    destination = models.ForeignKey(Destination, on\_delete=models.CASCADE, related\_name='restaurants')

    name = models.CharField(max\_length=100)

    image = models.ImageField(upload\_to='restaurants/')

class Activity(models.Model):

    destination = models.ForeignKey(Destination, on\_delete=models.CASCADE, related\_name='activities')

    title = models.CharField(max\_length=100)

    description = models.TextField()

class Information(models.Model):

    destination = models.ForeignKey(Destination, on\_delete=models.CASCADE, related\_name='info')

    content = models.TextField()

views :  
  
#tour/views

from django.shortcuts import render

# Create your views here.

from django.shortcuts import render, get\_object\_or\_404

from .models import Destination

def destination\_list(request):

    destinations = Destination.objects.all()

    return render(request, 'tour/destination\_list.html', {'destinations': destinations})

def destination\_detail(request, id):

    destination = get\_object\_or\_404(Destination, id=id)

    return render(request, 'tour/destination\_detail.html', {'destination': destination})

You're doing a great job — you're close! ✅  
Let me walk through your setup, suggest fixes, and guide you to the **next step: uploading images** for Gallery, Rooms, and Restaurants.

**✅ ✅ ✅ What's Already Working:**

* **Models**: Destination, Room, Restaurant, Activity, Information — ✔️
* **Dashboard for each planner** — ✔️
* **Register/Login views and templates** — ✔️
* **Destination form with user linking** — ✔️
* **Templates using Bootstrap — good base styling** — ✔️

**⚠️ Issues + Corrections**

**🔁 1. Duplicate Bootstrap CSS/JS**

In base.html, you included both:

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

✅ **Fix:** Keep only one version:

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

Same for JS at bottom.

**📌 2. dashboard() and dashboard\_view() in views.py — You only need one**

✅ **Fix:** Remove the unused one. Keep this:

@login\_required

def dashboard\_view(request):

destinations = request.user.destinations.all()

return render(request, 'tour/dashboard.html', {'destinations': destinations})

**🔧 3. DestinationForm HTML tag left open in forms.py**

At the very bottom, this is HTML inside a Python file:

{% extends "base.html" %}

{% block content %}

...

{% endblock %}

✅ **Fix:** Remove it — it belongs in register.html, not in forms.py.

**✅ Now Let’s Proceed to:**

**📸 Image Upload for:**

* Gallery (4–5 images)
* Rooms
* Restaurants

**🧩 1. Create Forms for Media Uploads**

**📄 forms.py**

from .models import DestinationImage, Room, Restaurant

class DestinationImageForm(forms.ModelForm):

class Meta:

model = DestinationImage

fields = ['image']

class RoomForm(forms.ModelForm):

class Meta:

model = Room

fields = ['name', 'image']

class RestaurantForm(forms.ModelForm):

class Meta:

model = Restaurant

fields = ['name', 'image']

**🧩 2. Views for Uploading Media**

**📄 Add to views.py:**

from .forms import DestinationImageForm, RoomForm, RestaurantForm

@login\_required

def upload\_images(request, destination\_id):

destination = get\_object\_or\_404(Destination, id=destination\_id, user=request.user)

if request.method == 'POST':

img\_form = DestinationImageForm(request.POST, request.FILES)

if img\_form.is\_valid():

image = img\_form.save(commit=False)

image.destination = destination

image.save()

return redirect('destination\_detail', id=destination.id)

else:

img\_form = DestinationImageForm()

return render(request, 'tour/upload\_image.html', {'form': img\_form, 'destination': destination})

**📄 3. Template: upload\_image.html**

{% extends 'base.html' %}

{% block content %}

<h2>Upload Image for {{ destination.name }}</h2>

<form method="post" enctype="multipart/form-data">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="btn btn-primary">Upload</button>

</form>

{% endblock %}

**🧭 4. URL Routing**

**📄 tour/urls.py:**

path('destination/<int:destination\_id>/upload/', upload\_images, name='upload\_image'),

**💡 Optional Enhancements**

Would you like me to generate these next?

* 🧾 **Planner profile page** with edit details
* 📝 **Edit/Delete** destinations and media
* 📋 Multi-upload interface for gallery (upload 4–5 images at once)
* ✅ Drag/drop media management (with JS)
* 🔒 Make sure only planners can edit their own destinations

Let me know how you'd like to continue 👇  
→ **Image uploads only**,  
→ **Add Edit/Delete next**,  
→ or **Both together (complete planner media control panel)?**

Excellent! Let’s build **a complete media control panel** for planners:  
✅ Upload images (gallery, rooms, restaurants)  
✅ Edit & delete destinations  
✅ Secure access (planners can only manage their own content)

**✅ Part 1: Media Upload (Gallery, Room, Restaurant)**

**📄 forms.py (Add or update)**

from .models import DestinationImage, Room, Restaurant

class DestinationImageForm(forms.ModelForm):

class Meta:

model = DestinationImage

fields = ['image']

class RoomForm(forms.ModelForm):

class Meta:

model = Room

fields = ['name', 'image']

class RestaurantForm(forms.ModelForm):

class Meta:

model = Restaurant

fields = ['name', 'image']

**✅ Part 2: Upload Views (Gallery, Rooms, Restaurants)**

**📄 views.py**

@login\_required

def upload\_gallery\_image(request, destination\_id):

destination = get\_object\_or\_404(Destination, id=destination\_id, user=request.user)

if request.method == 'POST':

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

if form.is\_valid():

image = form.save(commit=False)

image.destination = destination

image.save()

return redirect('destination\_detail', id=destination.id)

else:

form = DestinationImageForm()

return render(request, 'tour/upload\_image.html', {'form': form, 'title': 'Upload Gallery Image'})

@login\_required

def upload\_room(request, destination\_id):

destination = get\_object\_or\_404(Destination, id=destination\_id, user=request.user)

if request.method == 'POST':

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

if form.is\_valid():

room = form.save(commit=False)

room.destination = destination

room.save()

return redirect('destination\_detail', id=destination.id)

else:

form = RoomForm()

return render(request, 'tour/upload\_image.html', {'form': form, 'title': 'Add Room'})

@login\_required

def upload\_restaurant(request, destination\_id):

destination = get\_object\_or\_404(Destination, id=destination\_id, user=request.user)

if request.method == 'POST':

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

if form.is\_valid():

restaurant = form.save(commit=False)

restaurant.destination = destination

restaurant.save()

return redirect('destination\_detail', id=destination.id)

else:

form = RestaurantForm()

return render(request, 'tour/upload\_image.html', {'form': form, 'title': 'Add Restaurant'})

**✅ Part 3: Edit & Delete Destination**

**📄 views.py (Add below)**

@login\_required

def edit\_destination(request, id):

destination = get\_object\_or\_404(Destination, id=id, user=request.user)

if request.method == 'POST':

form = DestinationForm(request.POST, instance=destination)

if form.is\_valid():

form.save()

return redirect('dashboard')

else:

form = DestinationForm(instance=destination)

return render(request, 'tour/edit\_destination.html', {'form': form, 'destination': destination})

@login\_required

def delete\_destination(request, id):

destination = get\_object\_or\_404(Destination, id=id, user=request.user)

if request.method == 'POST':

destination.delete()

return redirect('dashboard')

return render(request, 'tour/delete\_destination.html', {'destination': destination})

**✅ Part 4: URL Routing**

**📄 tour/urls.py**

path('destination/<int:destination\_id>/upload/gallery/', upload\_gallery\_image, name='upload\_gallery'),

path('destination/<int:destination\_id>/upload/room/', upload\_room, name='upload\_room'),

path('destination/<int:destination\_id>/upload/restaurant/', upload\_restaurant, name='upload\_restaurant'),

path('destination/<int:id>/edit/', edit\_destination, name='edit\_destination'),

path('destination/<int:id>/delete/', delete\_destination, name='delete\_destination'),

**✅ Part 5: Templates**

**📄 upload\_image.html**

{% extends "base.html" %}

{% block content %}

<h2>{{ title }}</h2>

<form method="post" enctype="multipart/form-data">{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="btn btn-primary">Upload</button>

</form>

{% endblock %}

**📄 edit\_destination.html**

{% extends "base.html" %}

{% block content %}

<h2>Edit: {{ destination.name }}</h2>

<form method="post">{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="btn btn-warning">Update</button>

</form>

{% endblock %}

**📄 delete\_destination.html**

{% extends "base.html" %}

{% block content %}

<h2>Delete "{{ destination.name }}"?</h2>

<form method="post">{% csrf\_token %}

<button class="btn btn-danger" type="submit">Confirm Delete</button>

<a class="btn btn-secondary" href="{% url 'dashboard' %}">Cancel</a>

</form>

{% endblock %}

**✅ Part 6: Dashboard Buttons**

Update your dashboard.html destination cards to add:

<a href="{% url 'edit\_destination' dest.id %}" class="btn btn-sm btn-outline-secondary">Edit</a>

<a href="{% url 'delete\_destination' dest.id %}" class="btn btn-sm btn-outline-danger">Delete</a>

<a href="{% url 'upload\_gallery' dest.id %}" class="btn btn-sm btn-outline-info">Add Gallery</a>

<a href="{% url 'upload\_room' dest.id %}" class="btn btn-sm btn-outline-info">Add Room</a>

<a href="{% url 'upload\_restaurant' dest.id %}" class="btn btn-sm btn-outline-info">Add Restaurant</a>

**✅ What’s Next?**

You now have:

* ✅ Planners uploading all media
* ✅ Edit/Delete destinations
* ✅ UI protected by login
* ✅ Separate upload forms per section

**Want to go further?**

* 🧾 Profile page: show planner details, destinations count, edit email/password
* 🧠 Dashboard stats: destinations created, total images, etc.
* ☁️ Multi-file gallery upload with JavaScript drag & drop (Dropzone.js)
* 🔒 Add permissions for admin-level users

Let me know which you'd like next!

Great — let’s fix the **Login page** first, then move on to the **Planner profile page**, and optionally add stats/dashboard widgets after that.

**✅ FIX LOGIN PAGE NOT SHOWING**

**🔎 Problem**

You’ve already included:

path('', include('django.contrib.auth.urls')),

But you’re **missing the login template** that Django expects:

* registration/login.html

**✅ 1. Create Login Template**

**📄 templates/registration/login.html**

{% extends "base.html" %}

{% block content %}

<h2>Login</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="btn btn-primary">Login</button>

</form>

<p class="mt-3">

Don't have an account?

<a href="{% url 'register' %}">Register here</a>

</p>

{% endblock %}

**✅ 2. Add Login Link in Navbar**

**In base.html:**

Replace:

<li class="nav-item"><a class="nav-link" href="#">Login</a></li>

With:

{% if user.is\_authenticated %}

<li class="nav-item"><a class="nav-link" href="{% url 'dashboard' %}">Dashboard</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>

{% endif %}

**✅ 3. Configure Login Redirects (Optional)**

In your settings.py add:

LOGIN\_REDIRECT\_URL = 'dashboard'

LOGOUT\_REDIRECT\_URL = 'login'

**✅ Now You’ll See:**

* Login at /accounts/login/
* Works with the template above
* Redirects to the planner dashboard after login

**✅ NEXT: Add Planner Profile Page?**

Would you like the profile page to show:

* Username & email
* Count of destinations created
* Button to edit email/password?

Confirm and I’ll generate it now.

Great — let's build a **Planner Profile Page** that includes:

✅ Username & email  
✅ Number of destinations created  
✅ Buttons to update email or password  
✅ Protected so only the logged-in user sees their info

**✅ 1. Create View for Profile**

**📄 views.py**

Add this:

from django.contrib.auth.models import User

from django.contrib.auth.forms import PasswordChangeForm

from django.contrib.auth import update\_session\_auth\_hash

from django.contrib import messages

@login\_required

def planner\_profile(request):

destinations = request.user.destinations.all()

destination\_count = destinations.count()

if request.method == 'POST':

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

if email:

request.user.email = email

request.user.save()

messages.success(request, "Email updated successfully.")

return render(request, 'tour/profile.html', {

'user': request.user,

'destination\_count': destination\_count

})

**✅ 2. Add Profile URL**

**📄 tour/urls.py**

path('profile/', planner\_profile, name='profile'),

**✅ 3. Create Profile Template**

**📄 templates/tour/profile.html**

{% extends "base.html" %}

{% block content %}

<h2>Planner Profile</h2>

<div class="card p-4">

<p><strong>Username:</strong> {{ user.username }}</p>

<p><strong>Email:</strong> {{ user.email }}</p>

<p><strong>Total Destinations:</strong> {{ destination\_count }}</p>

<form method="post">

{% csrf\_token %}

<div class="mb-3">

<label for="email" class="form-label">Update Email</label>

<input type="email" name="email" id="email" class="form-control" value="{{ user.email }}" required>

</div>

<button class="btn btn-primary" type="submit">Update Email</button>

</form>

<hr>

<a href="{% url 'password\_change' %}" class="btn btn-secondary">Change Password</a>

</div>

{% if messages %}

{% for message in messages %}

<div class="alert alert-success mt-3">{{ message }}</div>

{% endfor %}

{% endif %}

{% endblock %}

**✅ 4. Enable Password Change URLs**

These are already included by:

path('', include('django.contrib.auth.urls')),

Django provides:

* /accounts/password\_change/ – change password
* /accounts/password\_change/done/ – confirmation

**✅ 5. Update Navbar to Link to Profile**

In base.html navbar, update authenticated links:

{% if user.is\_authenticated %}

<li class="nav-item"><a class="nav-link" href="{% url 'dashboard' %}">Dashboard</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>

{% endif %}

**✅ DONE — You Now Have:**

* 🌐 /profile/ for planner info
* ✏️ Update email
* 🔒 Password change link
* 🧠 Destination stats

**✅ Want to Add:**

* 🧑 Upload profile picture for planner?
* 📊 Dashboard stats (like destinations this month)?
* 📨 Contact form / support tab?

Let me know what you'd like next!

Awesome! Let's extend the **Planner Profile Page** with:

1. 🧑 Profile picture upload
2. 📊 Dashboard stats (destinations this month, total rooms, images, etc.)
3. 📨 Contact/support tab (for users to reach admins or support)

**✅ PART 1: Upload Profile Picture**

**🧩 1. Extend User with Profile Model**

Create a PlannerProfile model linked to User.

**📄 models.py**

from django.db import models

from django.contrib.auth.models import User

class PlannerProfile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

photo = models.ImageField(upload\_to='profile\_pics/', default='profile\_pics/default.jpg')

def \_\_str\_\_(self):

return f"{self.user.username}'s Profile"

**🧩 2. Signal to Auto-create Profile**

**📄 models.py (bottom of file)**

from django.db.models.signals import post\_save

from django.dispatch import receiver

@receiver(post\_save, sender=User)

def create\_user\_profile(sender, instance, created, \*\*kwargs):

if created:

PlannerProfile.objects.create(user=instance)

@receiver(post\_save, sender=User)

def save\_user\_profile(sender, instance, \*\*kwargs):

instance.plannerprofile.save()

**🧩 3. Form to Update Profile Picture**

**📄 forms.py**

from .models import PlannerProfile

class ProfilePicForm(forms.ModelForm):

class Meta:

model = PlannerProfile

fields = ['photo']

**🧩 4. Update views.py for profile upload**

from .forms import ProfilePicForm

@login\_required

def planner\_profile(request):

destinations = request.user.destinations.all()

destination\_count = destinations.count()

profile = request.user.plannerprofile

if request.method == 'POST':

if 'email' in request.POST:

request.user.email = request.POST['email']

request.user.save()

messages.success(request, "Email updated.")

if 'photo' in request.FILES:

profile\_form = ProfilePicForm(request.POST, request.FILES, instance=profile)

if profile\_form.is\_valid():

profile\_form.save()

messages.success(request, "Profile picture updated.")

else:

profile\_form = ProfilePicForm(instance=profile)

return render(request, 'tour/profile.html', {

'user': request.user,

'destination\_count': destination\_count,

'profile\_form': profile\_form,

'profile': profile

})

**🧩 5. Template Update: profile.html**

Add this inside the card:

<hr>

<h5>Profile Picture</h5>

<img src="{{ profile.photo.url }}" alt="profile" width="120" class="rounded mb-2">

<form method="post" enctype="multipart/form-data">

{% csrf\_token %}

{{ profile\_form.as\_p }}

<button type="submit" class="btn btn-outline-primary btn-sm">Update Photo</button>

</form>

**✅ PART 2: Dashboard Stats**

**📄 Add to dashboard\_view in views.py**

@login\_required

def dashboard\_view(request):

destinations = request.user.destinations.all()

rooms = sum(d.rooms.count() for d in destinations)

galleries = sum(d.images.count() for d in destinations)

activities = sum(d.activities.count() for d in destinations)

return render(request, 'tour/dashboard.html', {

'destinations': destinations,

'rooms': rooms,

'galleries': galleries,

'activities': activities

})

**📄 Add to dashboard.html:**

<div class="alert alert-info d-flex justify-content-between">

<div>Total Destinations: <strong>{{ destinations|length }}</strong></div>

<div>Rooms: <strong>{{ rooms }}</strong></div>

<div>Gallery Images: <strong>{{ galleries }}</strong></div>

<div>Activities: <strong>{{ activities }}</strong></div>

</div>

**✅ PART 3: Contact Support Form (Optional)**

**Want me to add a contact form that lets planners:**

* Send a message to admin
* Or submit a support request

Should I include that too?

Let me know and I’ll generate it!

Perfect! Let’s now add a **Contact Support form** so planners can reach out to the admin/support team.

**✅ What You'll Get:**

* 📩 Contact form (Name, Email, Subject, Message)
* ✅ Confirmation message after sending
* 📨 Email sent to your admin/support inbox

**🔧 Step 1: Enable Email Sending in settings.py**

Use console backend first (for testing). Later, you can connect Gmail or SMTP.

EMAIL\_BACKEND = 'django.core.mail.backends.console.EmailBackend'

DEFAULT\_FROM\_EMAIL = 'support@tms.com'

ADMIN\_EMAIL = 'admin@example.com' # You can replace this

In production use:

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

EMAIL\_HOST\_USER = 'your\_gmail@gmail.com'

EMAIL\_HOST\_PASSWORD = 'your\_app\_password'

**📄 Step 2: Create the Contact Form**

**forms.py**

from django import forms

class ContactForm(forms.Form):

name = forms.CharField(max\_length=100)

email = forms.EmailField()

subject = forms.CharField(max\_length=150)

message = forms.CharField(widget=forms.Textarea)

**📄 Step 3: Add Contact View**

**views.py**

from django.core.mail import send\_mail

from .forms import ContactForm

from django.conf import settings

def contact\_support(request):

if request.method == 'POST':

form = ContactForm(request.POST)

if form.is\_valid():

name = form.cleaned\_data['name']

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

subject = form.cleaned\_data['subject']

message = form.cleaned\_data['message']

full\_message = f"From: {name} <{email}>\n\n{message}"

send\_mail(

subject=subject,

message=full\_message,

from\_email=settings.DEFAULT\_FROM\_EMAIL,

recipient\_list=[settings.ADMIN\_EMAIL],

)

messages.success(request, "Your message has been sent successfully!")

return redirect('contact\_support')

else:

form = ContactForm()

return render(request, 'tour/contact\_support.html', {'form': form})

**📄 Step 4: URL Route**

**urls.py**

path('support/', contact\_support, name='contact\_support'),

**📄 Step 5: Template — contact\_support.html**

{% extends "base.html" %}

{% block content %}

<h2>Contact Support</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="btn btn-primary">Send Message</button>

</form>

{% if messages %}

{% for message in messages %}

<div class="alert alert-success mt-3">{{ message }}</div>

{% endfor %}

{% endif %}

{% endblock %}

**📄 Step 6: Add Link in Navbar or Footer**

In base.html:

<li class="nav-item">

<a class="nav-link" href="{% url 'contact\_support' %}">Support</a>

</li>

**✅ ALL DONE!**

You now have:

* 🧑 Profile with photo
* 📊 Dashboard with real stats
* 📬 Support form with email sending

**🔜 Want to continue with:**

* 🌐 Multi-language (Swahili/English)?
* 🧹 Cleaner UI cards with tabs (gallery, rooms)?
* 📦 Export planner data to PDF?
* 🔒 Admin-only section to view all planner submissions?

Let me know where you want to go next!

Excellent choices! Here's how we’ll proceed:

**✅ Task 1: 🧹 Cleaner UI Cards with Tabs (Gallery, Rooms, etc.)**

We’ll redesign the destination\_detail.html page using Bootstrap **tabs**, so users can easily navigate between:

* 📸 Gallery
* 🛏️ Rooms
* 🍽️ Restaurants
* 🏄 Activities & Services
* ℹ️ Information

**📄 Update destination\_detail.html**

{% extends 'base.html' %}

{% block content %}

<h2>{{ destination.name }} ({{ destination.country }})</h2>

<p><strong>Visit Date:</strong> {{ destination.start\_date }} - {{ destination.end\_date }}</p>

<p>{{ destination.description }}</p>

<div class="ratio ratio-16x9 mb-3">{{ destination.map\_embed\_code|safe }}</div>

<!-- Tabs Navigation -->

<ul class="nav nav-tabs mb-3" id="destinationTabs" role="tablist">

<li class="nav-item" role="presentation">

<button class="nav-link active" data-bs-toggle="tab" data-bs-target="#gallery">Gallery</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" data-bs-toggle="tab" data-bs-target="#rooms">Rooms</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" data-bs-toggle="tab" data-bs-target="#restaurants">Restaurants</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" data-bs-toggle="tab" data-bs-target="#activities">Activities</button>

</li>

<li class="nav-item" role="presentation">

<button class="nav-link" data-bs-toggle="tab" data-bs-target="#info">Information</button>

</li>

</ul>

<!-- Tabs Content -->

<div class="tab-content">

<div class="tab-pane fade show active" id="gallery">

{% for img in destination.images.all %}

<img src="{{ img.image.url }}" class="img-fluid mb-2" style="max-width: 100%;">

{% empty %}

<p>No gallery images uploaded.</p>

{% endfor %}

</div>

<div class="tab-pane fade" id="rooms">

{% for room in destination.rooms.all %}

<div class="mb-2">

<strong>{{ room.name }}</strong><br>

<img src="{{ room.image.url }}" style="max-width: 100%;">

</div>

{% empty %}

<p>No rooms added yet.</p>

{% endfor %}

</div>

<div class="tab-pane fade" id="restaurants">

{% for rest in destination.restaurants.all %}

<div class="mb-2">

<strong>{{ rest.name }}</strong><br>

<img src="{{ rest.image.url }}" style="max-width: 100%;">

</div>

{% empty %}

<p>No restaurants available.</p>

{% endfor %}

</div>

<div class="tab-pane fade" id="activities">

{% for act in destination.activities.all %}

<div class="mb-3">

<strong>{{ act.title }}</strong>

<p>{{ act.description }}</p>

</div>

{% empty %}

<p>No activities added yet.</p>

{% endfor %}

</div>

<div class="tab-pane fade" id="info">

{% for i in destination.info.all %}

<p>{{ i.content }}</p>

{% empty %}

<p>No extra information provided.</p>

{% endfor %}

</div>

</div>

{% endblock %}

✅ Add some custom CSS in static/style.css if you want paddings, shadows, etc.

**✅ Task 2: 📦 Export Planner Data to PDF**

We’ll use xhtml2pdf or WeasyPrint. Here’s a setup using xhtml2pdf (easier to start):

**🧩 Install Library**

pip install xhtml2pdf

**📄 views.py (Add export view)**

from django.http import HttpResponse

from django.template.loader import get\_template

from xhtml2pdf import pisa

@login\_required

def export\_destination\_pdf(request, id):

destination = get\_object\_or\_404(Destination, id=id, user=request.user)

template = get\_template('tour/export\_pdf.html')

html = template.render({'destination': destination})

response = HttpResponse(content\_type='application/pdf')

response['Content-Disposition'] = f'attachment; filename="{destination.name}.pdf"'

pisa\_status = pisa.CreatePDF(html, dest=response)

return response

**📄 urls.py**

path('destination/<int:id>/export/', export\_destination\_pdf, name='export\_pdf'),

**📄 templates/tour/export\_pdf.html**

Keep it simple for now:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<style>

body { font-family: DejaVu Sans, sans-serif; }

h1 { color: #007bff; }

</style>

</head>

<body>

<h1>{{ destination.name }}</h1>

<p><strong>Country:</strong> {{ destination.country }}</p>

<p><strong>Dates:</strong> {{ destination.start\_date }} - {{ destination.end\_date }}</p>

<p><strong>Description:</strong> {{ destination.description }}</p>

<h3>Rooms</h3>

<ul>

{% for room in destination.rooms.all %}

<li>{{ room.name }}</li>

{% endfor %}

</ul>

<h3>Restaurants</h3>

<ul>

{% for r in destination.restaurants.all %}

<li>{{ r.name }}</li>

{% endfor %}

</ul>

<h3>Activities</h3>

<ul>

{% for a in destination.activities.all %}

<li>{{ a.title }} - {{ a.description }}</li>

{% endfor %}

</ul>

<h3>Additional Info</h3>

{% for info in destination.info.all %}

<p>{{ info.content }}</p>

{% endfor %}

</body>

</html>

**📍 Button in dashboard or detail page**

<a href="{% url 'export\_pdf' destination.id %}" class="btn btn-outline-dark">Export to PDF</a>

**✅ You Now Have:**

* Cleaner detail page with tabs for all destination sections
* Export-to-PDF button per destination (planner-specific)

**✅ Next Possible Steps:**

* 🖼 Multi-image upload (gallery drag/drop)
* 📦 Export all destinations as ZIP of PDFs
* 📍 Use Leaflet.js or Mapbox for interactive maps instead of iframe
* 🛠️ Admin approval of submitted destinations?

Let me know which feature you'd like next!