Show\_staff

def show\_staff(request):  
 context = {}  
  
 try:  
 # Retrieve all staff records from the database  
 context['staff\_list'] = staff\_model.objects.all()  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 messages.error(request, f"An error occurred while fetching staff records: {str(ex)}")  
  
 # Render the template with the context data  
 return render(request, "showstaff.html", context)  
  
show templeinfo  
def show\_templeinfo(request):  
 context = {}  
  
 try:  
 # Retrieve all temple information records from the database  
 context['info\_list'] = templeinfo\_model.objects.all()  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 messages.error(request, f"An error occurred while fetching temple information: {str(ex)}")  
  
 # Render the template with the context data  
 return render(request, "vietemp.html", context)  
def show\_priest(request):  
 context = {}  
  
 try:  
 # Fetch all priest records from the database  
 context['priest\_list'] = priest\_model.objects.all()  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 # Display an error message using Django's messaging framework  
 messages.error(request, f"An error occurred while fetching priest information: {str(ex)}")  
 # Optionally, you could log the error or handle it differently  
 # Set an empty list in the context to avoid breaking the template  
 context['priest\_list'] = []  
  
 # Render the addprie.html template with the current context, including the list of priests  
 return render(request, "viewprie.html", context)  
  
show schedule  
def show\_schedule(request):  
 context = {}  
  
 try:  
 # Retrieve all schedule records from the database  
 context['schedule\_list'] = poojaschedule\_model.objects.all()  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 messages.error(request, f"An error occurred while fetching the schedule records: {str(ex)}")  
  
 # Render the template with the context data  
 return render(request, "viewchedule.html", context)

show special  
  
def show\_special(request):  
 context = {}  
  
 try:  
 # Retrieve all special day records from the database  
 context['special\_list'] = specialday\_model.objects.all()  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 messages.error(request, f"An error occurred while fetching special day records: {str(ex)}")  
  
 # Render the template with the context data  
 return render(request, "viewspecial.html", context)

show income  
def show\_income(request):  
 context = {}  
  
 try:  
 # Retrieve all income records from the database  
 context['income\_list'] = income\_model.objects.all()  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 messages.error(request, f"An error occurred while fetching income records: {str(ex)}")  
  
 # Render the template with the context data  
 return render(request, "viewincome.html", context)

show careers  
def show\_careers(request):  
 context = {}  
  
 try:  
 # Retrieve all career records from the database  
 context['careers\_list'] = careers\_model.objects.all()  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 messages.error(request, f"An error occurred while fetching career records: {str(ex)}")  
  
 # Render the template with the context data  
 return render(request, "viewcareers.html", context)

show darshan  
  
def show\_darshan(request):  
 context = {}  
  
 try:  
 # Retrieve all darshan records from the database  
 context['darshan\_list'] = darshan\_model.objects.all()  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 messages.error(request, f"An error occurred while fetching darshan records: {str(ex)}")  
  
 # Render the template with the context data  
 return render(request, "viewdarshan.html", context)  
  
search temple  
def search\_temple(request):  
 context = {}  
  
 try:  
 # Initialize the form with POST data if available  
 frm = location\_form(request.POST or None)  
  
 # Add the form to the context for rendering  
 context['f'] = frm  
  
 except Exception as ex:  
 # Handle any unexpected exceptions  
 messages.error(request, f"An error occurred while processing the search form: {str(ex)}")  
  
 # Render the search temple template with the context data  
 return render(request, "serachtemple.html", context)