Question 1:

By default are django signals executed synchronously or asynchronously? Please support your answer with a code snippet that conclusively proves your stance. The code does not need to be elegant and production ready, we just need to understand your logic.

ANS:In synchronous fashion Django does all the signals by default. So if a signal is sent, it’s appropriate to assume that the signal’s receivers will be invocated straight away on the calling thread.

Eg

HANDLER

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

from django.dispatch import receiver

from django.contrib.auth.models import User

//This is the signal handler (receiver)

@receiver(post\_save, sender=User)

def notify\_admin(sender, instance, \*\*kwargs):

print("Signal received: A new user has been saved.")

VIEW:

from django.contrib.auth.models import User

from django.http import HttpResponse

def create\_user\_view(request):

print("Creating a new user...")

user = User.objects.create(username="simpleuser")

print("User creation complete.")

return HttpResponse("User created")

Question 2:

Do django signals run in the same thread as the caller? Please support your answer with a code snippet that conclusively proves your stance. The code does not need to be elegant and production ready, we just need to understand your logic.

Yes, **Django signals run in the same thread as the caller by default**. Since signals are executed synchronously, they share the same thread as the function that triggers the signal.

EG:

//SIGNAL HANDLER

import threading

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

from django.dispatch import receiver

from django.contrib.auth.models import User

//Signal handler to print the current thread name

@receiver(post\_save, sender=User)

def signal\_handler(sender, instance, \*\*kwargs):

print(f"Signal running in: {threading.current\_thread().name}")

:VIEW

import threading

from django.contrib.auth.models import User

from django.http import HttpResponse

def create\_user\_view(request):

print(f"View running in: {threading.current\_thread().name}")

User.objects.create(username="simpleuser")

return HttpResponse("User created")

Question 3:

By default do django signals run in the same database transaction as the caller? Please support your answer with a code snippet that conclusively proves your stance. The code does not need to be elegant and production ready, we just need to understand your logic.

ANS:

Signals in Django are executed by default in the same database transaction with the caller. This means that all the signal handlers are executed in the context of one database transaction and if the transaction is rolled back, then all the changes that the signal handler has executed will also be rolled back.

SINGNAL HANDLER:

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

from django.dispatch import receiver

from django.contrib.auth.models import User

//Signal handler that creates another user

@receiver(post\_save, sender=User)

def signal\_handler(sender, instance, \*\*kwargs):

print("Signal handler: creating 'signaluser'")

User.objects.create(username="signaluser")

VIEW:

from django.db import transaction

from django.contrib.auth.models import User

from django.http import HttpResponse

def create\_user\_view(request):

try:

with transaction.atomic():

print("Creating 'mainuser'...")

User.objects.create(username="mainuser")

raise Exception("Rolling back transaction") # Force a rollback

except Exception as e:

print(f"Exception caught: {e}")

return HttpResponse("Transaction rolled back")