```
import torch
import numpy as np
import random
from transformers import AutoTokenizer, AutoModelForCausalLM
import warnings

# Ignore UserWarning category from the transformers library
warnings.filterwarnings("ignore", category=UserWarning, module="transformers")

# Set the random seed for reproducibility
random.seed(42)
np.random.seed(42)
torch.manual_seed(42)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(42)

torch.cuda.empty_cache()

model_name = "Qwen/Qwen2.5-0.5B-Instruct"

tokenizer = AutoTokenizer.from_pretrained(model_name)

# Ensure device_map is manually set when only one GPU is available
if torch.cuda.device_count() == 8:
    device = "cuda:0"
    conversation = AutoModelForCausalLM.from_pretrained(
        model_name,
        torch_dtype=torch.float16
    ).to(device)  # Explicitly move model to the single GPU
else:
    conversation = AutoModelForCausalLM.from_pretrained(
        model_name,
        device_map="auto",
        torch_dtype=torch.float16
    )

def chatbot_response(user_input_en):
    torch.cuda.empty_cache()
    print("\nQuestion:", user_input_en)

    # Determine the device of the model
    device = next(conversation.parameters()).device

    # Move inputs to the correct device
    inputs = tokenizer(user_input_en, return_tensors="pt").to(device)

    # Generate a response
    generate_ids = conversation.generate(
        inputs.input_ids,
        max_length=2000,  # Allow up to 2000 tokens
        temperature=1.2,  # Increase temperature for variability
        top_p=0.9,        # Use nucleus sampling for diversity
        do_sample=True,   # Enable sampling for less deterministic output
        eos_token_id=tokenizer.eos_token_id
    )

    # Decode the response
    response_en = tokenizer.batch_decode(
        generate_ids,
        skip_special_tokens=True,
```

```
        clean_up_tokenization_spaces=False
    )[0].replace(user_input_en, "").strip()

    print("Response:", response_en)
    return response_en

if __name__ == "__main__":
    print("Welcome to the Console Chatbot! Type 'exit' to quit.")
    while True:
        user_input = input("\nYou: ")
        if user_input.lower() in ["exit", "quit"]:
            print("Exiting... Goodbye!")
            break
        chatbot_response(user_input)
```