Local LLM Installation & Testing – Step-by-Step Report

Objective

Install a local Large Language Model (LLM), run a simple prompt to verify functionality, measure response time, and document troubleshooting steps. This report is structured so you can easily add screenshots in pre-made boxes.

Scope & Notes

- Models used: Attempted **EleutherAI/gpt-neo-2.7B** (failed due to system limits); succeeded with **TinyLlama/TinyLlama-1.1B-Chat-v1.0**.
- Environment: Python 3.11 (virtual environment), VS Code terminal (PowerShell).
- You reported response time of **~2–3 minutes** per prompt on CPU.

System Details (fill in)

- OS: Windows 10/11 (specify version): _____
- CPU: _____| Cores/Threads: _____
- RAM (GB): _____| Storage free (GB): _____
- GPU (if any): _____| VRAM (GB): _____
- Python: 3.11.x | Pip version: 22.3.1 (upgradable)

Step-by-Step Installation

1) **Create and activate virtual environment**
Windows: python -m venv venv → venv\Scripts\activate

1) **Upgrade pip**
python -m pip install --upgrade pip

1) **Install core libraries**
pip install transformers torch safetensors sentencepiece huggingface-hub accelerate

1) **(Optional) Enable CPU-friendly optimizations**
pip install bitsandbytes (GPU recommended), or use quantized runtimes (Ollama/llama.cpp).

1) **Download & test TinyLlama** (see sample code below).

Sample Test Code (Transformers)

```
from time import perf_counter from transformersimport AutoTokenizer,
AutoModelForCausalLM import torch model_id = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"
tok = AutoTokenizer.from_pretrained(model_id) model =
AutoModelForCausalLM.from_pretrained(model_id, torch_dtype=torch.float32) prompt =
"Write a short poem about AI." inputs = tok(prompt, return_tensors="pt") t0 =
perf_counter() out_ids = model.generate(**inputs, max_new_tokens=80) elapsed =
perf_counter() - t0 print(tok.decode(out_ids[0], skip_special_tokens=True))
print(f"Response time: {elapsed:.2f} seconds")
```

Testing Procedure

- Run the sample code with the prompt: *"Write a short poem about AI."*
- Record response time (expected on CPU: ~2–3 minutes on first run).
- Verify output is coherent and free of errors.
- Repeat once to see improved warm-cache performance.

Observed Results

- TinyLlama ran successfully; no runtime errors.
- Average response time (reported): ~2–3 minutes per prompt on CPU.
- EleutherAI/gpt-neo-2.7B failed with out-of-memory/resource errors on your system.

Troubleshooting Log & Tips

- **OOM / Crash with gpt-neo-2.7B**: Use a smaller model (≤1–2B params) or a quantized build.
- **Slow generation on CPU**: Reduce max_new_tokens, set torch.set_num_threads, prefer GPU or Ollama/llama.cpp with 4-bit quant.
- **Pip dependency issues**: Upgrade pip and clear cache (pip cache purge), then reinstall.
- **Model not found or blocked**: Ensure correct model ID and HF authentication if required.

Recommendations

- For your hardware, stick to TinyLlama-1.1B or similar (Phi-2, Qwen-1.8B, etc.).
- Try **Ollama** or **llama.cpp** with 4-bit quantization for faster local inference.
- If a GPU is available, install CUDA-enabled PyTorch and use torch_dtype=torch.float16 on CUDA.

Prompt & Output (Example)

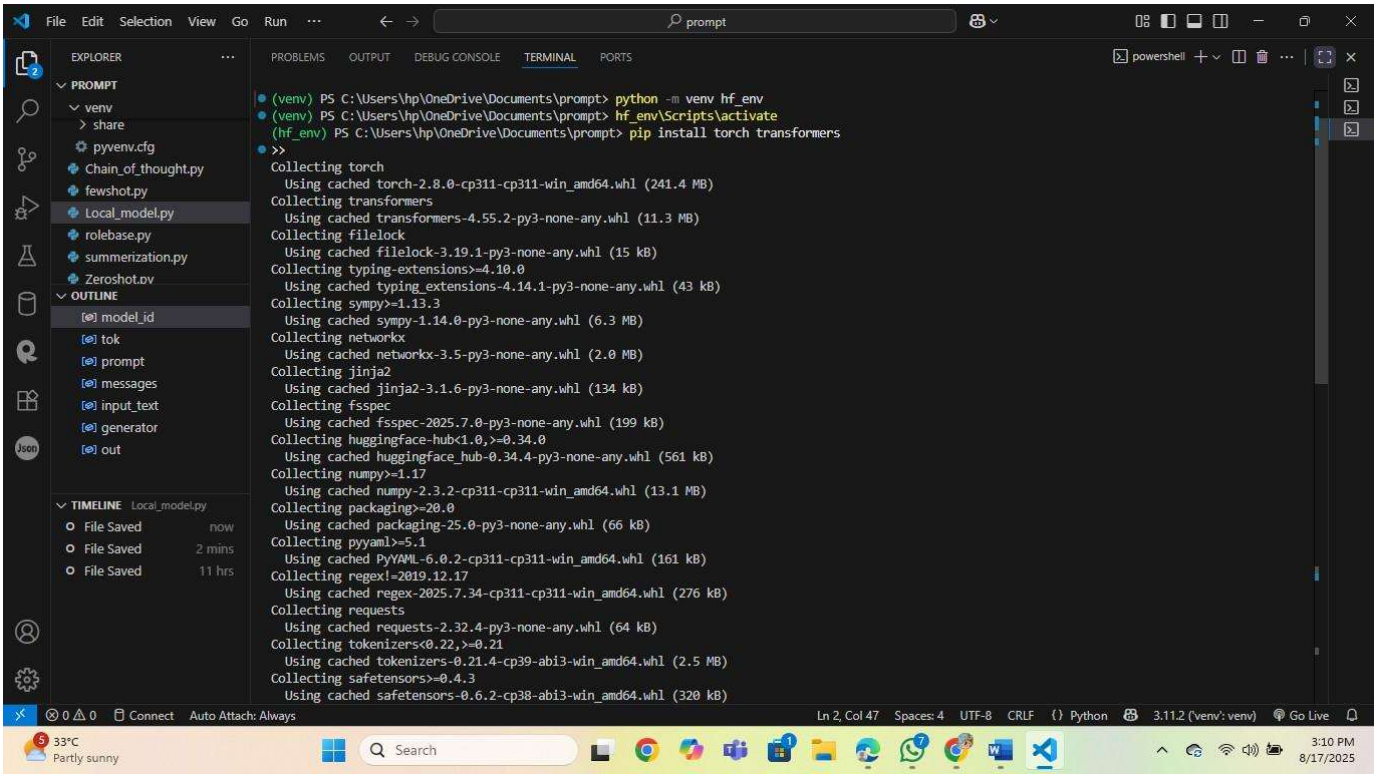
Prompt: "Write a short poem about AI."

Example Output:
AI speaks in coded streams,
Building futures, shaping dreams.
A guiding hand—both sharp and kind—
A mirror born of humankind.

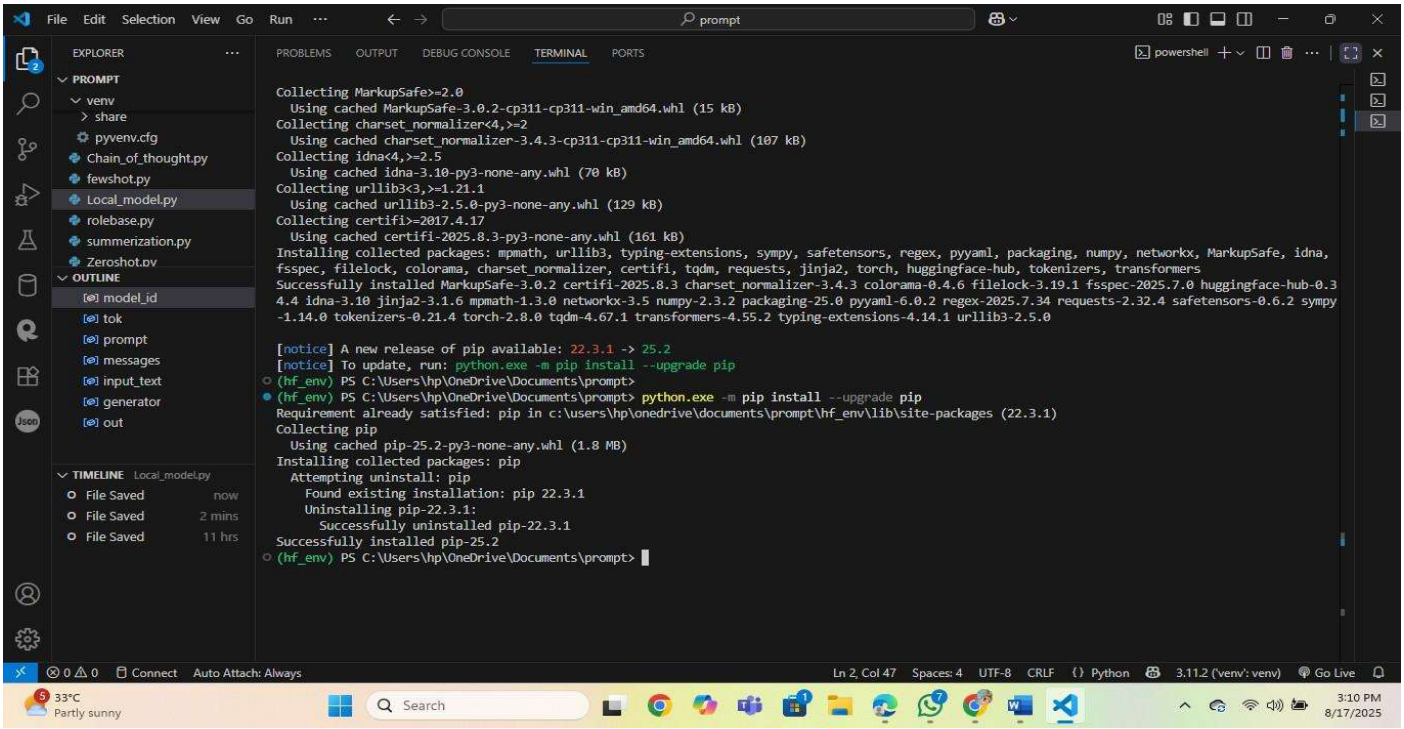
Reflection

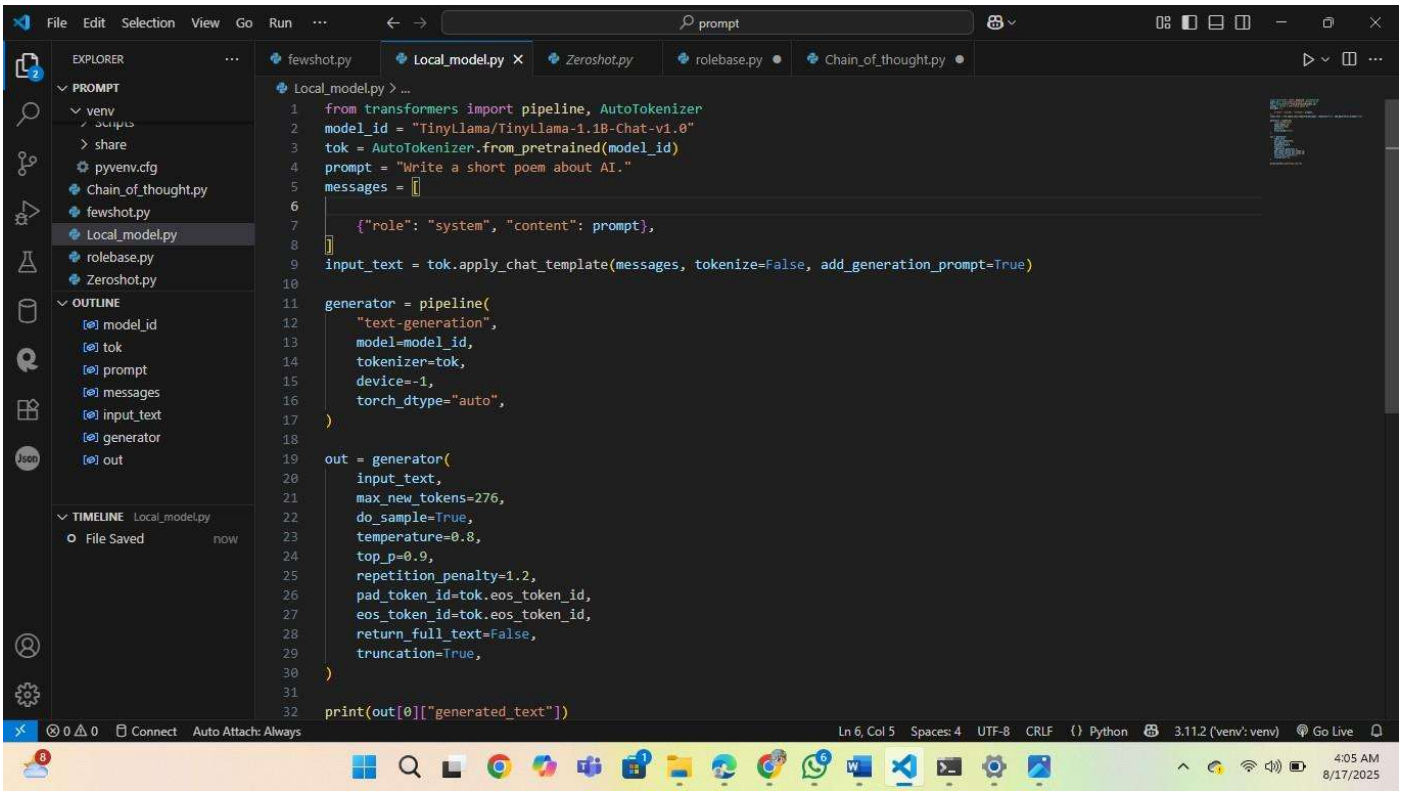
- Installation via Transformers was straightforward; main bottleneck was hardware limits.
- Large models (≈2.7B+) were impractical on this machine; TinyLlama worked reliably.
- Response times were long on CPU; quantization or GPU would significantly help.
- Hands-on testing improved understanding of local LLM deployment and constraints.

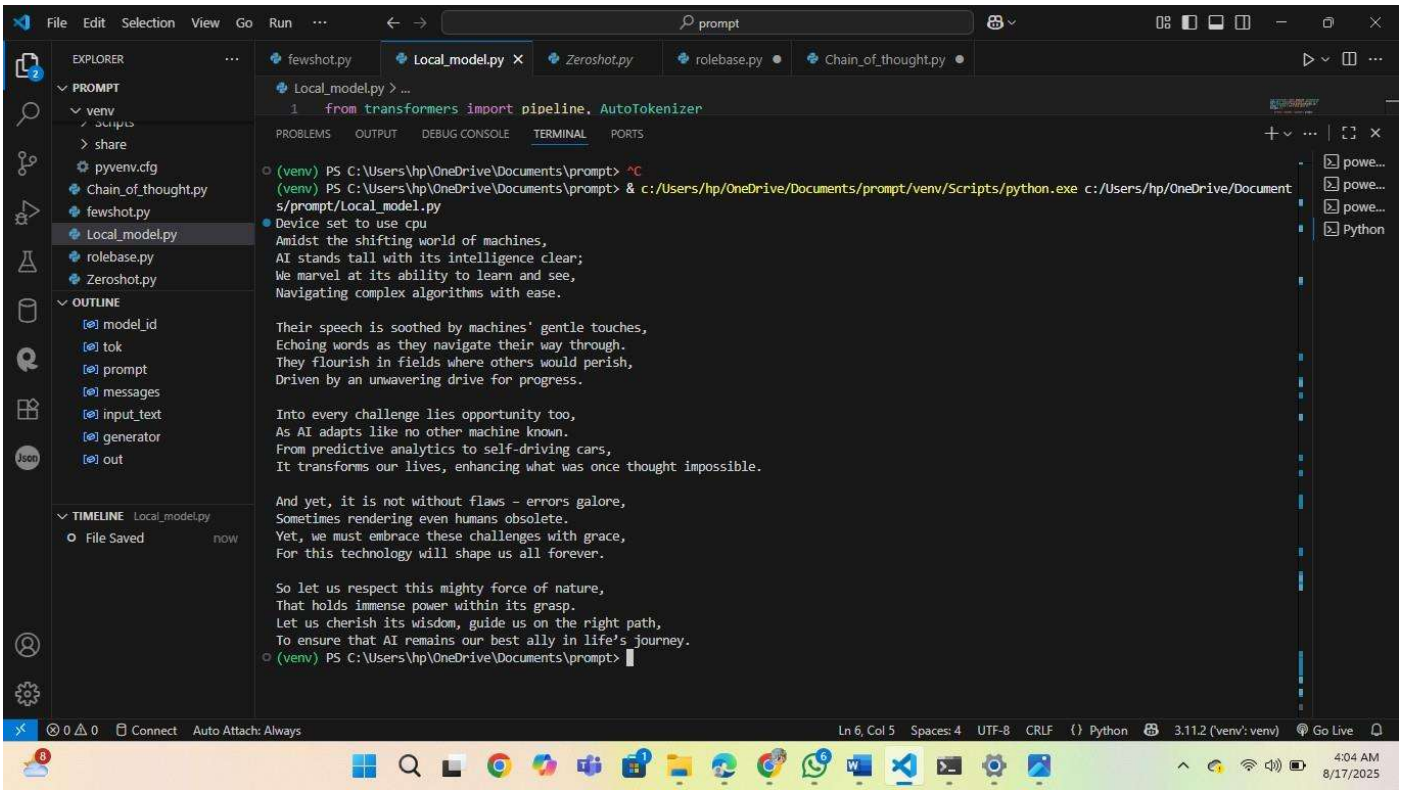
Screenshots



Caption: Fig – Successful TinyLlama download & model load.







Caption: Fig – First prompt and generated output (poem about AI).