CivicHacks 2026 – Open Source AI Live Demo

Step-by-Step Guide

Workshop: *Why Open Source AI Changes Everything – And How to Use It This Weekend*

What This Demo Does

This demo builds a **complete civic AI application** live on stage, in three progressive steps, proving that open source AI is free, powerful, and accessible to anyone:

Step	What Happens	Time on Stage	Audience Sees
Step 1	Run a local AI model	~60 seconds	"AI runs on a laptop for free"
Step 2	Connect it to real civic data (RAG)	~90 seconds	"It can analyze our city's data"
Step 3	Wrap it in a web app	~60 seconds	"That's a real product – built in minutes"

The demo uses **real Boston and Massachusetts civic datasets** covering all four hackathon tracks (EcoHack, CityHack, EduHack, JusticeHack). The audience votes on which track to demo, creating investment and ownership.

Prerequisites (Do This Before the Session)

1. Install Ollama

Ollama lets you run large language models locally with a single command.

macOS:

```shell
# Download from https://ollama.com or use Homebrew
```

```
brew install ollama
```

Linux:

```Shell
curl -fsSL https://ollama.com/install.sh | sh
```

Windows: Download the installer from ollama.com

2. Pull the Model

```Shell
# This downloads ~4.7 GB — do this on reliable wifi BEFORE the event
ollama pull llama3.1
```

Verify it works:

```Shell
ollama run llama3.1 "Say hello in 10 words or less"
```

3. Install Python Dependencies

```Shell
# Requires Python 3.10+
cd civichacks-demo
pip install -r requirements.txt
```

This installs:

- llama-index — RAG framework for connecting AI to data
- llama-index-llms-ollama — Ollama integration
- llama-index-embeddings-huggingface — Local embeddings (no API key needed)
- gradio — Web UI framework

4. Pre-warm Everything (Critical for Live Demo!)

Run each step once before you go on stage. This ensures:

- Models are loaded in memory (first run is slower)
- Embedding model is downloaded and cached
- No surprise downloads during the live demo

```
Shell
# Pre-warm Step 1
python scripts/demo_step1_ollama.py

# Pre-warm Step 2 (do each track)
python scripts/demo_step2_rag.py eco
python scripts/demo_step2_rag.py city
python scripts/demo_step2_rag.py edu
python scripts/demo_step3_app.py justice

# Pre-warm Step 3 (start it, verify it loads, then Ctrl+C)
python scripts/demo_step3_app.py
```

5. Hardware Requirements

Component	**Minimum**	**Recommended**
RAM	8 GB	16 GB
GPU VRAM	Not required (CPU works)	8+ GB (much faster)
Storage	10 GB free	20 GB free
CPU	Any modern 4-core	Apple Silicon or recent Intel/AMD

Apple Silicon Macs (M1/M2/M3/M4) are ideal — the unified memory architecture handles Llama 3.1 8B beautifully at 15-25 tokens/second.

No GPU? It still works on CPU, just slower (~3-5 tokens/second). For the live demo, this is actually fine — the audience can see it generating in real time.

Demo Flow – What to Do on Stage

STEP 1: "The 60-Second AI" (During Opening Segment, ~Minute 3)

Setup: Terminal open, font size large enough for the back row.

Script:

> "I just told you that GPT-4-class models are free and open. Let me prove it. This laptop has no special hardware. I'm going to ask a local AI model – Llama 3.1, running right here, no cloud, no API key – a question about civic tech."

Run:

```Shell
python scripts/demo_step1_ollama.py
```

What happens: The script sends a civic-themed prompt to the local Ollama instance and streams the response token by token. The audience watches the AI generate in real time. At the end, it prints the time elapsed and "$0.00" cost.

Talking point while it generates:

> "Watch it go – this is the same architecture behind GPT-4. It's generating on this laptop's [CPU/GPU]. No internet required. No data leaving this machine. And it cost me nothing to run."

After it finishes:

> "That took [X] seconds. Cost: zero. And that's just the raw model. Over the next 40 minutes, we're going to turn that into a real civic tech application."

STEP 2: "Connecting AI to Real Civic Data" (During Stack Evaluation, ~Minute 20)

Setup: The audience has already voted on a track. Have the track key ready.

Script:

> "OK, so you've got a local AI. But a model by itself doesn't know anything about Boston. The magic happens when you connect it to real data. This is called RAG – Retrieval Augmented Generation – and it takes about 15 lines of Python."

Optionally show the code file briefly (the key lines are well-commented):

```shell
Shell
# Show the relevant code if you want – it's intentionally readable
cat scripts/demo_step2_rag.py | head -80
```

Run (using the audience's voted track):

```shell
Shell
# If they voted CityHack:
python scripts/demo_step2_rag.py city

# For all queries (if time allows):
python scripts/demo_step2_rag.py city --all
```

What happens: The script loads the track-specific civic dataset, builds a vector index, and queries it. The AI's response is grounded in actual data – it cites specific statistics, neighborhoods, and findings from the documents.

Talking point while the index builds:

> "It's reading the city's data and building a search index. This is the same technique behind every enterprise AI chatbot – except we're doing it locally, for free, with open source tools."

After the answer:

> "Look at that – it pulled specific numbers from the city's own data. [Reference a specific stat from the answer]. That's not the model making things up. That's RAG – the model retrieves relevant chunks of real data before generating its answer. This is how you build civic tech that's actually trustworthy."

STEP 3: "From Script to Web App" (During Templates & Resources, ~Minute 32)

Setup: This is the reveal moment. You've gone from terminal → data pipeline → now a real web application.

Script:

> "We've got a local AI that can answer questions about civic data. But hackathon judges aren't going to lean over your shoulder and watch a terminal. You need a demo. Watch how fast we can go from script to web app."

Run:

```Shell
python scripts/demo_step3_app.py
```

What happens: Gradio launches a web server and opens a polished chat interface in the browser. It has track selection, example questions, a chat interface, and a footer showing the full stack. The audience sees a real, shareable web application.

Walk through the UI:

1. Point out the track selector — switch between datasets
2. Click an example question — watch it query and respond
3. Highlight the footer: "Ollama + LlamaIndex + Gradio · Cost: $0.00"

Talking point:

> "This is a production-quality web interface built with Gradio. The entire UI is about 40 lines of Python. No React, no JavaScript, no frontend experience needed. And if you want to share it with judges or deploy it publicly, Hugging Face Spaces will host it for free."

The kicker:

> "Let's count what we used: Ollama — free. Llama 3.1 — free, open source. LlamaIndex — free, open source. Gradio — free, open source. Hosting — free. Total cost to build a civic AI application: zero dollars. Total time from scratch: under an hour. That's what open source AI makes possible."

Troubleshooting

"Ollama isn't responding"

```
Shell
# Check if Ollama is running
ollama list
# If not, start it
ollama serve
```

"The model is slow"

- CPU inference for 8B models: expect 3-8 tokens/second (still fine for demos)
- If it's painfully slow, pre-generate the Step 1 response and show it as a "replay" while noting you ran it live earlier during sound check
- Make sure no other heavy applications are competing for memory

"Embedding model download is slow"

The HuggingFace embedding model (all-MiniLM-L6-v2) downloads on first use (~80 MB). **Run Step 2 at least once before the session** to cache it.

"Gradio won't open in the browser"

```
Shell
# Try specifying the browser
BROWSER=chrome python scripts/demo_step3_app.py
# Or open manually: http://localhost:7860
```

Backup Plan: Pre-Recorded Demo

If wifi or hardware fails, have a screen recording of the full demo flow ready. Record it at the venue during setup so the environment looks authentic.

Adapting This Demo for Your Own Use

Swap in Your Own Data

Replace the files in data/ with any text documents relevant to your use case. The RAG pipeline handles:

- .txt files (used here)
- .pdf files (add llama-index-readers-file to requirements)
- .csv files
- .docx files
- Web pages (use SimpleWebPageReader)

Change the Model

```shell
# Smaller/faster model for limited hardware
ollama pull phi3:mini        # 3.8B, runs on almost anything
ollama pull llama3.2:3b      # 3B, very fast

# Larger/better model if you have the RAM
ollama pull llama3.1:70b     # Needs ~40GB RAM, but incredible quality

# Best for reasoning tasks
ollama pull deepseek-r1:7b   # Strong reasoning, MIT license
```

Then update the model name in the scripts:

```python
Settings.llm = Ollama(model="phi3:mini")  # Change model here
```

Deploy to Hugging Face Spaces (Free)

1. Create an account at huggingface.co
2. Create a new Space (select "Gradio" as the SDK)
3. Push your code
4. Note: For HF Spaces, you'll need to swap Ollama for the HF Inference API or a hosted model endpoint (Ollama runs locally, not on HF's servers)

Deploy to Streamlit Cloud (Free Alternative)

Convert demo_step3_app.py to use Streamlit instead of Gradio — both are excellent choices. Streamlit Cloud offers free hosting with GitHub integration.

Project Structure

```None
civichacks-demo/
├── README.md            ← You are here
├── requirements.txt     ← Python dependencies
├── data/                ← Civic datasets (one per track)
│   ├── ecohack_boston_environment.txt
│   ├── cityhack_boston_311.txt
│   ├── eduhack_boston_schools.txt
│   └── justicehack_ma_justice.txt
└── scripts/             ← Demo scripts (run in order)
    ├── demo_step1_ollama.py   ← Step 1: Basic local AI
    ├── demo_step2_rag.py      ← Step 2: RAG with civic data
    └── demo_step3_app.py      ← Step 3: Full web application
```

Resources Mentioned in the Workshop

Resource	**URL**	**What It Is**
Ollama	ollama.com	Run LLMs locally
AI Templates	aitemplates.io	Production-ready AI app templates (Red Hat)
Hugging Face	huggingface.co	Models, datasets, free deployment
Artificial Analysis	artificialanalysis.ai	Independent model benchmarks
LlamaIndex	docs.llamaindex.ai	RAG framework
LangChain	langchain.com	LLM application framework

Resource	URL	What It Is
Gradio	gradio.app	ML web UI framework
CrewAI	crewai.com	Multi-agent framework
awesome-langchain	github.com/kyrolabs/awesome-langchain	Curated tool list
awesome-llm-agents	github.com/kaushikb11/awesome-llm-agents	Agent framework list

---

License

This demo code is released under the **Apache 2.0 License**. Use it, fork it, teach with it, build on it.

The civic datasets included are **synthetic/illustrative** — they are realistic but fabricated for demonstration purposes. For real civic data, visit:

- **data.boston.gov** — City of Boston open data portal
- **mass.gov/open-data** — Massachusetts state data
- **data.gov** — Federal open data

Built for CivicHacks 2026 at Boston University. Go build something that matters.