Ameya AI – Hands-On Workshop Facilitator Master Guide

This document is the single source of truth for facilitation. It is designed for reference during the session and can be shared with participants AFTER completion.

Principles:
- No slides
- No theory
- Hands-on only
- Local-first, enterprise-shaped AI

Open-Source Models for Local AI (New Section)

This section explains where to find open-source models, how to choose the right model for a local machine, and how Ollama Local differs from Ollama Cloud. This content is critical context for Session 2.

Where to Find Open-Source Models

Ollama:
- Simplest way to run modern LLMs locally
- Handles model lifecycle, quantization, and serving
- Exposes a local HTTP API (localhost:11434)
- Best choice for workshops and enterprise PoCs

Hugging Face:
- Largest open-source model hub
- Useful for advanced customization and fine-tuning
- Requires more setup and ML expertise

How to Choose a Model for Local Execution

Choose models based on constraints, not benchmarks:

- Hardware: CPU-only vs GPU
- Latency: interactive vs batch
- Context length: resumes vs large documents
- Accuracy vs speed trade-offs
- Compliance: data must stay local

Recommended defaults:
- llama3.2 for CPU-only machines
- mistral / qwen variants if more compute is available

Ollama Local vs Ollama Cloud

Ollama Local:
- Fully local inference
- Zero data egress
- Predictable cost
- Enterprise and regulated workloads

Ollama Cloud:
- Managed inference
- Higher throughput
- Data leaves local boundary
- Best for non-sensitive demos

Workshop Rule: Ollama Local ONLY.

Running Models Locally

```
ollama pull llama3.2
ollama run llama3.2
```

1. Workshop Overview

Session 1: AI-Native Development with Agentic IDEs (60 mins)
Session 2: Local LLMs and Enterprise AI (60 mins)

IDE: Cursor ONLY
LLM: Local LLM via Ollama

2. One-Page Facilitator Checklist

- Cursor IDE ready
- Python 3.10+
- Ollama running
- Repo cloned and tested

3. README – Exact Commands

```
git clone <repo-url>
cd ai-native-skill-platform
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt

uvicorn app.main:app --reload
```

4. Facilitator-Only Prompt Sheet

Prompt 1 – Service Scaffolding

You are an AI engineer.

Create a FastAPI service that uses a LOCAL LLM via Ollama (http://localhost:11434).

Do NOT use any cloud LLM APIs or API keys.

Endpoints:

- GET /health

- POST /upload_resume

- POST /analyze_skills

Only code. No explanations.

Prompt 2 – Local LLM + Contract Enforcement

Extend the existing FastAPI service.

Constraints:

- Use Python requests or httpx to call Ollama (/api/generate or /api/chat)

- Model name must come from env var OLLAMA_MODEL (default: llama3.2)

- /upload_resume must accept text or file and store it under ./data/resumes

- /analyze_skills must call Ollama and return structured JSON

Return JSON format:

```
{
  "skills": [],
  "years_experience": "optional",
  "role_suggestions": []
}
```

Add basic error handling if Ollama is not running.

Only code. No explanations.

Prompt 3 – Skill AI Intelligence Layer

Extend the service with intelligence features.

Add:

- skill_gap_analysis(target_role)

- weekly_learning_task_generator()

Behavior:

- Compare extracted skills against the target role

- Identify missing skills

- Generate 5–7 practical weekly learning tasks per missing skill

Return a single structured JSON response combining:

- extracted_skills

- missing_skills

- weekly_tasks

Fix any issues automatically.

Only code. No explanations.

5. Day-2 Enterprise Extension Roadmap

- Auth & RBAC
- Persistent vectors
- Observability
- Docker & deployment
- Multi-agent orchestration