Abhikarta-LLM

Enterprise AI Orchestration Platform

Version 1.4.6

11+
Providers

100+
Models

$0
Local Models

The AI Revolution in Enterprise

From Traditional AI to Generative AI

Traditional AI excelled at classification and prediction. Generative AI (GenAI) creates new content - text, code, images, analysis. Large Language Models (LLMs) like GPT-4, Claude, and Llama have transformed what machines can accomplish.

Enterprise Impact

McKinsey estimates GenAI could add $2.6-4.4 trillion annually across industries

82% of organizations are exploring AI agents for automation

Knowledge work productivity gains of 20-40% are achievable

The Challenge

Only 44% have security policies for AI. Most lack governance, oversight, and control.

What is Agentic AI?

Definition

Agentic AI refers to AI systems that can autonomously plan, reason, use tools, and take actions to accomplish goals. Unlike simple chatbots, agents iterate until objectives are achieved.

Chatbot

Single response to query
No tool access
No memory across turns

AI Agent

Plans and iterates to goal
Uses tools (DB, API, code)
Maintains context and memory

Agent Components: LLM + Tools + Memory + Reasoning + Goals

Why AI Orchestration Matters

The Need for Orchestration

Real enterprise tasks require multiple AI agents working together - research teams, approval chains, multi-step workflows. Orchestration coordinates these agents for reliable, governed outcomes.

Without Orchestration

Shadow AI - employees use ChatGPT without oversight
No audit trail for compliance or debugging
Costs spiral without rate limiting or quotas
No human oversight on critical decisions

With Orchestration

Centralized governance and visibility
Complete audit logging for compliance
Cost control with usage limits
Human-in-the-loop at every level

Why Abhikarta-LLM for Enterprise?

Purpose-Built for Enterprise AI

Abhikarta-LLM addresses the governance gap in enterprise AI adoption. While other frameworks focus on capabilities, Abhikarta prioritizes security, oversight, and organizational alignment.

Multi-Provider

11+ LLM providers
No vendor lock-in

Visual Design

No-code agent builder
DAG workflow editor

AI Organizations

Patent-pending
Hierarchy + HITL

Enterprise RBAC

Model-level perms
Full audit trails

Result: Safe, governed AI that aligns with how enterprises actually operate

Table of Contents

1. Market Challenges
2. Platform Architecture
3. Multi-Provider LLM Support
4. Agent Framework
5. Workflow DAG System
6. Agent Swarms
7. AI Organizations (Patent)
8. RBAC and Security
9. Notifications Integration
10. Use Cases with Examples
11. Competitive Analysis
12. Appendix

Section

Market Challenges

Problems solved by Abhikarta-LLM

Challenge: Provider Lock-In

The Problem

Organizations commit to single LLM provider (OpenAI, Anthropic, etc.)

Provider-specific code patterns create high switching costs

Pricing changes impact budgets with no alternatives

Service disruptions halt production workloads

Our Solution

Unified abstraction layer across 11+ providers

Same code works with any provider - switch via config

Automatic failover between providers on errors

Best-of-breed model selection per use case

Challenge: AI Governance Gaps

The Problem

Shadow AI proliferates - employees use ChatGPT without oversight

Sensitive data leaked to external AI services

No audit trail for AI-generated content

Regulatory compliance gaps (GDPR, HIPAA, SOX)

Our Solution

Centralized platform with RBAC at model level

Complete audit logging of all LLM interactions

Rate limiting per user, team, organization

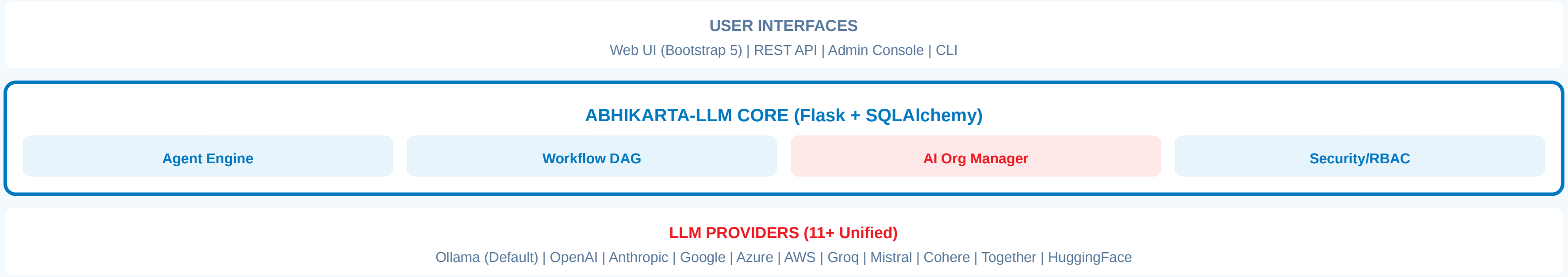
Local Ollama models for sensitive workloads

Section

Platform Architecture

System design and components

Architecture Overview



USER INTERFACES

Web UI (Bootstrap 5) | REST API | Admin Console | CLI

ABHIKARTA-LLM CORE (Flask + SQLAlchemy)

- **Agent Engine**
- **Workflow DAG**
- **AI Org Manager**
- **Security/RBAC**

LLM PROVIDERS (11+ Unified)

Ollama (Default) | OpenAI | Anthropic | Google | Azure | AWS | Groq | Mistral | Cohere | Together | HuggingFace

Core Components

Agent Framework

Modular agents: Persona + Tools + Memory + Knowledge Base

6 reasoning patterns: ReAct, CoT, ToT, Reflexion, Hierarchical, Goal

MCP tool integration for external services

Workflow Engine

DAG orchestration with topological execution

12+ node types including Python code nodes

Parallel execution with HITL approval gates

AI Organization Manager (Patent Pending)

Digital twin of corporate hierarchy

Task delegation down, response aggregation up

Human mirrors with configurable autonomy

Security Layer

RBAC with role to permission to model mapping

API key management with scoped access

Rate limiting and complete audit trails

Section

Multi-Provider LLM Support

Unified access to 11+ providers

Supported LLM Providers

Ollama (DEFAULT)
Free, Local, Private

OpenAI
GPT-4o, o1, o3

Anthropic
Claude 4.5

Google
Gemini 2.0

UI Configuration (Admin - Providers)

Add/edit providers with API keys securely stored in database
Configure base URLs for self-hosted endpoints (Azure, vLLM)
Set rate limits (RPM/TPM) per provider with test connectivity
Enable/disable providers and models with one click

Unified API Benefits

Key API Endpoints

POST /api/v1/complete - Unified completion across any provider

POST /api/v1/chat - Multi-turn conversations with history

POST /api/v1/embed - Generate embeddings for RAG

POST /api/v1/agents/id/execute - Run agent with tools

POST /api/v1/workflows/id/execute - Run workflow DAG

Benefits

Single API for any provider - switch via provider parameter

Consistent error handling with automatic retry logic

Streaming support for all providers

Standardized usage metrics and cost tracking

OpenAPI/Swagger documentation at /api/docs

Section

Agent Framework

Building intelligent AI agents

What is an AI Agent?

Definition

An AI Agent is an autonomous entity combining an LLM with tools, memory, and reasoning patterns to accomplish tasks. Unlike chatbots, agents plan, use tools, iterate until goals are achieved.

Agent Components

Persona: System prompt defining role, expertise, constraints
LLM: The language model from any provider
Tools: Functions the agent can call (DB, API, File, MCP)
Memory: Conversation history, working memory, long-term KB
Reasoning: Pattern for thinking (ReAct, CoT, ToT, Reflexion)
HITL: Human checkpoints for oversight

Agent Reasoning Patterns

ReAct (Reason + Act)

Think then Act then Observe then Repeat

Best for: multi-step tasks with tools

Chain-of-Thought (CoT)

Step-by-step reasoning before answer

Best for: math, logic, analysis

Tree-of-Thoughts (ToT)

Explore multiple paths, backtrack

Best for: creative, open-ended

Reflexion

Self-critique and improve iteratively

Best for: quality refinement

Hierarchical

Manager delegates to worker agents

Best for: complex decomposition

Goal-Based

Define goal, plan, execute, replan

Best for: autonomous objectives

Agent Tool Integration

Built-in Tool Types

Database: Query SQLite, PostgreSQL, MySQL

API: Call REST/GraphQL with auth headers

File: Read/write files, parse PDF, Excel, CSV

Search: Web search, vector RAG retrieval

Python: Execute Python in sandbox

MCP (Model Context Protocol) Integration

Connect external MCP servers as agent tools

Pre-built: Filesystem, GitHub, Slack, Postgres, Puppeteer

Auto-discovery of MCP tool schemas

UI Tool Management (Admin - Tools)

Browse/enable tools per agent | Configure parameters | Test before save

Creating Agents via UI

Visual Agent Designer (Agents - New)

1. Basic Info: Name, description, category, tags
2. Provider/Model: Select from dropdown (Ollama default)
3. Persona: Rich text editor for system prompt
4. Tools: Drag-drop from tool library, configure params
5. Knowledge Base: Upload docs for RAG retrieval
6. Reasoning: Select pattern (ReAct, CoT, etc.)
7. HITL: Configure approval checkpoints
8. Test: Interactive chat to validate before save

Agent Definition: JSON Format

```json
{"name": "Research Assistant",
 "provider": "ollama", "model": "llama3.3:70b",
 "persona": "You are a research assistant...",
 "reasoning_pattern": "react",
 "tools": [
   {"type": "web_search", "config": {"max_results": 5}},
   {"type": "file_read"},
   {"type": "mcp", "server": "github"}
 ],
 "knowledge_base": {"vector_store": "chroma"},
 "hitl": {"approval_required": ["web_search"]},
 "max_iterations": 10, "temperature": 0.7}
```

Python Usage

```python
from abhikarta import Agent
agent = Agent.from_json("agent.json")
result = agent.run("Analyze Q3 earnings")
```

Section

Workflow DAG System

Visual pipeline orchestration

Workflow DAG Overview

What is a Workflow DAG?

A Directed Acyclic Graph (DAG) represents multi-step AI pipelines where nodes are processing steps and edges define data flow. Workflows enable parallel execution, conditional logic, and human checkpoints.

Key Capabilities

12+ Node Types: LLM, Agent, Tool, Python, Condition, Human...

Parallel Execution: Nodes without dependencies run concurrently

Conditional Branching: Route based on output values

Error Handling: Retry, fallback, or fail-fast per node

HITL Nodes: Pause for human approval at any step

Creating Workflows via UI

Visual Workflow Designer (Workflows - New)

1. Canvas: Drag-drop nodes from palette onto canvas
2. Connect: Draw edges between node outputs and inputs
3. Configure: Click node to edit parameters in sidebar
4. Python Code: Add Python nodes with syntax highlighting
5. Variables: Define workflow inputs, pass between nodes
6. Validate: Check for cycles, missing connections
7. Test Run: Execute with sample inputs before deploy

Python Integration

Inline Python with workflow context | Import existing files | pip install per workflow

Workflow Definition: JSON Format

```
{"name": "Document Analysis Pipeline",
 "nodes": [
   {"id": "extract", "type": "tool", "tool": "file_read"},
   {"id": "summarize", "type": "llm", "provider": "ollama",
    "prompt": "Summarize: {{extract.output}}"},
   {"id": "approve", "type": "human", "message": "Review summary"},
   {"id": "notify", "type": "tool", "tool": "slack_send"}
 ],
 "edges": [
   {"from": "extract", "to": "summarize"},
   {"from": "summarize", "to": "approve"},
   {"from": "approve", "to": "notify"}
 ]}
```

Python Usage

```
result = Workflow.from_json("pipeline.json").run({"file": "report.pdf"})
```

Section

Agent Swarms

Dynamic multi-agent coordination

What is an Agent Swarm?

Definition

An Agent Swarm is a collection of autonomous agents that collaborate dynamically. Unlike fixed workflows, swarms use event-driven coordination where agents respond to tasks based on capabilities.

Key Characteristics

Event-Driven: Agents react to events, not predefined sequences
Self-Organizing: Agents claim tasks based on capabilities
Scalable: Add/remove agents without restructuring
Fault Tolerant: Other agents compensate for failures

Messaging: Kafka, RabbitMQ, ActiveMQ, built-in pub/sub

Section

AI Organizations

Patent-pending hierarchical governance

What is an AI Organization?

Definition (Patent Pending)

An AI Organization is a digital twin of a corporate hierarchy where each position is occupied by an AI agent. Tasks flow down (delegation) and responses flow up (aggregation), mirroring how real organizations work.

Problems Solved

Accountability: Clear ownership for AI decisions
Delegation: Complex tasks decomposed naturally
Human Oversight: Every AI has a human mirror
Compliance: Matches regulatory org structures

AI Organization: JSON Definition

```
{"name": "Research Division",
 "positions": [
   {"id": "director", "title": "Research Director",
    "agent": "strategic_agent",
    "human_mirror": "john.smith@company.com",
    "autonomy": "supervised"},
   {"id": "lead1", "title": "Team Lead",
    "agent": "ai_specialist",
    "reports_to": "director",
    "autonomy": "semi_autonomous"},
   {"id": "analyst1", "title": "Analyst",
    "agent": "research_agent",
    "reports_to": "lead1"}
 ]}
```

Benefits: Mirrors real org | Clear accountability | Scalable AI teams | Human oversight built-in

Section

RBAC and Security

Enterprise-grade access control

Role-Based Access Control

RBAC Model

Users: Individual accounts with authentication

Roles: Admin, Developer, Analyst, Viewer (+ custom)

Permissions: Granular actions per resource type

Model Access: Control which models each role can use

Usage Limits: RPM/TPM quotas per role

Security Features

API Keys: Scoped keys with expiration

Audit Logs: Every action recorded with user/timestamp

Rate Limiting: Prevent abuse and cost overruns

Data Isolation: Multi-tenant data separation

Notifications Integration

Supported Channels

Slack: Bot integration with interactive buttons
Microsoft Teams: Webhook and bot support
Email: SMTP with templates
Webhooks: POST to any endpoint

AI Org Integration

Human Mirror Alerts: Notify when AI needs approval
Escalation: Auto-escalate if no response in timeout
Interactive Buttons: Approve/Reject from notification
Summary Reports: Daily/weekly AI activity digests

Section

Use Cases

Real-world applications with examples

Use Case: Customer Service Bot

Key Benefits

70% auto-resolution

24/7 availability

RAG over FAQ

CRM integration

HITL for complaints

```
{"name": "Support Bot",
 "provider": "ollama",
 "tools": ["rag", "api"],
 "hitl": {"enabled": true}}
```

Use Case: Document Analysis

Key Benefits

90% faster review

DAG pipeline

PDF parsing

Human approval gate

```
{"name": "Contract Review",
 "provider": "ollama",
 "tools": ["rag", "api"],
 "hitl": {"enabled": true}}
```

Use Case: Research and Reports

Key Benefits

Hours to minutes

AI Org with swarm

Web+DB search

Director aggregates

```
{"name": "Research Team",
 "provider": "ollama",
 "tools": ["rag", "api"],
 "hitl": {"enabled": true}}
```

Use Case: Developer Productivity

Key Benefits

Ollama for privacy

GitHub MCP

Auto docs and tests

```
{"name": "Code Reviewer",
 "provider": "ollama",
 "tools": ["rag", "api"],
 "hitl": {"enabled": true}}
```

Use Case: Risk and Compliance

Key Benefits

24/7 monitoring

Transaction analysis

Pattern detection

Audit ready

```
{"name": "Compliance Swarm",
 "provider": "ollama",
 "tools": ["rag", "api"],
 "hitl": {"enabled": true}}
```

Section

Competitive Analysis

Market positioning

Competitive Comparison

Feature	Abhikarta	LangChain	AutoGen	CrewAI
Multi-Provider	**11+ Native**	Via Plugins	Limited	Limited
Visual Workflow	**Built-in DAG**	LangGraph	None	None
AI Org Hierarchy	**Patent Pending**	None	Basic	Basic
Enterprise RBAC	**Full + Model**	None	None	None
HITL Controls	**Comprehensive**	Interrupt	Basic	Basic

Section

Appendix

Acknowledgements and licensing

Open Source Acknowledgements

LLM and AI

LangChain, LlamaIndex, Ollama, OpenAI SDK, Anthropic SDK, Sentence Transformers, ChromaDB, FAISS, Transformers

Web Framework

Flask, SQLAlchemy, Pydantic, Bootstrap 5, Jinja2, Gunicorn

Infrastructure

Docker, PostgreSQL, Redis, Kafka, RabbitMQ

Utilities

Click, Rich, PyYAML, Requests, aiohttp, NumPy, Pandas, PyPDF

Licensing and Intellectual Property

Proprietary License

Abhikarta-LLM is proprietary software. All rights reserved. Unauthorized copying, modification, distribution, or use is strictly prohibited.

Patent Pending

AI Organization Management technology and hierarchical AI governance framework are patent pending innovations.

Copyright

Copyright 2025-2030 Ashutosh Sinha. All Rights Reserved.

Contact: ajsinha@gmail.com

Thank You

Abhikarta-LLM

Enterprise AI Orchestration Platform v1.4.6

11+ Providers

100+ Models