

# Product Requirements Document: Universal AI Orchestration Layer (UAOL) - Backend & Architecture

**Product Name:** UAOL (Universal AI Orchestration Layer) **Version:** 1.0 - Backend Focus **Date:** December 3, 2025 **Author:** Manus AI **Goal:** To define the highly scalable, MCP-First microservices architecture, data model, and core backend services required to support a billion users and a vast ecosystem of third-party AI tools.

## 1. Core Architectural Strategy: MCP-First Microservices

The UAOL backend will be built on a **Model Context Protocol (MCP)-First** strategy, ensuring every interaction with an external tool or model is standardized. The architecture will be a decoupled, event-driven microservices design to achieve the required scale, resilience, and independent deployment cycles.

### 1.1. Key Architectural Principles

Principle	Description	Technology Focus
MCP-First	All tool integrations (internal and external) must conform to the MCP specification for seamless context and data exchange.	Standardized API Gateway, Tool Proxy Service.
Decoupling	Services must be independently deployable and scalable, communicating primarily via asynchronous message queues.	Serverless Functions (Lambda/Cloud Functions), Managed Message Queue (SQS/Kafka).
Global Scale	The data layer must support global distribution, high read/write throughput, and low latency.	Distributed SQL (CockroachDB/PlanetScale), Global Caching (Redis Cluster).
Resilience	Critical processes (e.g., workflow execution) must be fault-tolerant with automatic retries and Dead-Letter Queues (DLQs).	Job Orchestration Service with SQS/Kafka and DLQ implementation.

## 2. Backend Services (Microservices)

The core functionality will be split into the following services:

Service Name	Primary Function	Key Responsibilities
Auth & User Service	Handles user authentication, authorization, and profile management.	Google OAuth, API Key generation, User/Subscription/Credit balance management.
Tool Registry Service	Manages the metadata for all registered MCP tools (internal and third-party).	Tool registration, discovery index, versioning, and pricing (Job Credit cost).
Job Orchestration Service	The brain of the platform. Manages the lifecycle of all workflows and long-running tasks.	Workflow serialization, job queuing, status tracking, error handling (DLQ), and credit deduction/refund.
Tool Proxy Service	The single point of contact for all external MCP tool calls.	Rate limiting, request validation, secure secret injection, and proxying requests to third-party MCP servers.
Billing & Credit Service	Manages the Job Credit system, subscription tiers, and usage-based billing.	Real-time credit deduction, subscription management (Stripe integration), and usage reporting.
Data & Storage Service	Manages file uploads, long-term storage of analysis results, and chat history.	AWS S3 integration, pre-signed URL generation, and data retention policies.

## 3. Data Model (High-Level)

The core data model revolves around Users, Tools, and Jobs.

### 3.1. users Table

Field	Type	Description
-------	------	-------------

<code>user_id</code>	UUID	Primary key.
<code>email</code>	TEXT	User's email address.
<code>current_credits</code>	BIGINT	Real-time balance of Job Credits.
<code>subscription_tier</code>	ENUM	Free, Pro, Enterprise.
<code>api_key</code>	TEXT	User's private API key for external access.

### 3.2. `mcp_tools` Table (Tool Registry)

Field	Type	Description
<code>tool_id</code>	UUID	Primary key.
<code>name</code>	TEXT	Display name of the tool (e.g., "Alpha Vantage Stock API").
<code>gateway_url</code>	TEXT	The base URL for the third-party MCP server.
<code>credit_cost_per_call</code>	INT	The number of Job Credits consumed per execution.
<code>developer_id</code>	UUID	Foreign key to the developer's <code>user_id</code> .
<code>status</code>	ENUM	Pending, Approved, Disabled.

### 3.3. `processing_jobs` Table (Job Orchestration)

Field	Type	Description
<code>job_id</code>	UUID	Primary key.
<code>user_id</code>	UUID	Foreign key to the user who initiated the job.
<code>workflow_definition</code>	JSONB	The serialized definition of the VWB workflow.

status	ENUM	Queued, Running, Success, Failed, Retrying.
start_time	TIMESTAMP	Time the job was queued.
end_time	TIMESTAMP	Time the job completed.
final_output	JSONB	The result of the workflow execution.

## 4. Scalability and Performance Requirements

Requirement ID	Description	Metric/Target
AR-401	API Latency	P95 API response time for non-job-related calls (e.g., Tool Registry lookup) must be under <b>200ms</b> .
AR-402	Job Throughput	The Job Orchestration Service must be able to process <b>10,000 jobs per second</b> during peak load.
AR-403	Database Connection Pool	The database must support a minimum of <b>10,000 concurrent connections</b> to handle massive user activity.
AR-404	Caching Strategy	All Tool Registry lookups and Stock/Prediction Market data must be served from the cache with a <b>95% hit rate</b> .
AR-405	Uptime	The entire platform must maintain a minimum of <b>99.99% uptime</b> (Four Nines).

## 5. Security Requirements

- Data Encryption:** All data at rest (database, S3 storage) and in transit (API calls) must be encrypted (AES-256 and TLS 1.3, respectively).

- **Secret Management:** All API keys for LLMs and third-party services must be stored in a dedicated, secure vault (e.g., AWS Secrets Manager, HashiCorp Vault) and injected at runtime by the Tool Proxy Service.
- **Rate Limiting:** Implement aggressive, per-user and per-tool rate limiting at the API Gateway to prevent abuse and ensure fair usage.
- **Input Validation:** Strict validation and sanitization of all user-provided input, especially in the Workflow Builder and chat interface, to prevent injection attacks.