OneUI — Comprehensive Design Document (2025/09/26)

Date: Sep 26, 2025

> This design doc is self-contained and reflects the current OneUI system (v4) with recent additions: MCP Host/Client, Deep Research agent + Eval framework, IDP updates (Runway, GitHub Control Room, SRE Cockpit), and core data/tenancy architecture. It favors deterministic-first patterns with LLMs as orchestrators.

---

1. Goals & Non-Goals

Goals

- Unify enterprise workflows behind a single AI-assisted interface that **streams structured outputs** (ChatBlocks) and is **auditable, governed, and cost-controlled**.
- Provide a **deterministic-first** backbone (typed tools, rules, SQL) with **LLM orchestration** for reasoning/explanation.
- Support **multi-tenant isolation**, strong **observability**, **web-first answers with citations**, and **hybrid retrieval** (BM25 + embeddings).
- Offer **agentic workflows** (Deep Research, Marketing, Data Analytics) and **MCP integration** (host/client) for external tool control.

Non-Goals

- Not a general purpose low-code platform; OneUI exposes a curated, typed tool surface.
- Not a replacement for full observability stacks; SRE Cockpit aggregates signals via adapters.

---

2. Product Surfaces

- **Chat/Streaming**: primary conversational UI returning ChatBlocks (markdown/table/chart/job/form).
- **Agents**: MCP Host/Client, Deep Research + Eval UI.
- **Apps**: Gmail, Drive (New from Prompt), Sheets/SQL/BigQuery, HubSpot/Proxycurl, QuickBooks→Metabase.
- **IDP**: GitHub Control Room, SRE Operations Cockpit, Runway (Self-Service Blueprints).

---

3. High-Level Architecture

```
+-----------------+     +---------------------+     +--------------------+
| Frontend (Next) |<--->| FastAPI Backend     |<--->| Providers/LLMs     |
| ChatBlocks      |     | ROUTES / TOOLS      |     | (OpenAI/Anth/Gemini|
| Agents, IDP     |     | Policy Engine       |     | Qwen/GLM/OSS)      |
+--------+--------+     | Multi-LLM Router    |     +---------+----------+
         |              | Deterministic Tools |               |
         |              | MCP Gateway         |               |
         v              +----------+----------+               |
  Streaming (NDJSON)               |                          |
                                   v                          v
                         +---------------+          +-------------+
                         | Memory & RAG  |<-------->| External Web|
                         | Postgres+Obj  |          | & APIs      |
                         | pgvector/BM25 |          +-------------+
                         +-------+-------+
                                 |
                                 v
                         +---------------+
                         | Kafka Topics  |
                         | memory/jobs   |
                         +-------+-------+
                                 |
                                v
                         +---------------+
                         | Workers/Jobs  |
                         | (async tasks) |
                         +---------------+
```

---

4. Core Components

4.1 Frontend (Next.js)

- Streams NDJSON into ChatBlocks (markdown/table/chart/job/form).
- **Agents** pages: MCP Host/Client, Agent Eval UI.
- **IDP**: Runway (IaC generator), GitHub Control Room, SRE Cockpit dashboards.
- Auth: Firebase/SSO patterns; passes `X-User-Id`/session headers to backend.

4.2 Backend (FastAPI)

- `ROUTES`: per-surface endpoints (e.g., `/query/*`, `/agents/*`, `/idp/*`).
- `TOOLS`: typed deterministic tools (Sheets→SQL, SQL query, HubSpot sync, Proxycurl search, QBO ingest, n8n triggers, etc.).
- **Policy Engine**: pre-prompt (redaction, model/tool selection), tool-time policies (allow/deny; approvals), post-gen policies (PII checks, formatting, receipts).
- **Multi-LLM Router**: provider selection w/ A/B, failover, budget, and small-model preference.
- **MCP Gateway**: REST today, SSE planned; maintains server registry, capabilities, and tool/resource invocations.
- **Deep Research Agent**: multi-step plan + tool calls; emits evaluations and traces.

4.3 Memory & Retrieval

- **Postgres** tables + object storage for durable memory and artifacts.
- `memory_cards` (≤120 tokens), `messages`, `vector_index`, `table_registry`, `jobs`, `usage`.

- **pgvector** + BM25 hybrid retrieval; web-first answers with citations; RAG fallback uses vault + web sources.
- Versioned writes with replay; per-tenant and per-user isolation.

4.4 Kafka & Jobs

- Topics: `oneui.memory.v1`, `oneui.jobs.v1`.
- Workers consume jobs (ETL, enrichment, syncs); idempotent with saga/outbox patterns.
- Retry/backoff and dead-letter queues.

4.5 Observability & Cost

- **Langfuse** for traces (prompt/tool spans) + cost; **Prometheus/Grafana** for metrics; **Elastic** for logs.
- KPIs: TTFC, p95 latency, citation precision, hallucination rate, cost/run, blueprint validity, MCP connect success.

4.6 Security & Tenancy

- Namespaced per tenant (DB schema, storage prefix, optional K8s namespace).
- AuthN: OAuth2/SSO/SAML patterns; AuthZ via roles and policy gates.
- PII tagging/redaction; egress allowlists; audit logs; budgets/quotas.
- Secrets via cloud KMS/Secret Manager.

---

5. Data Model (selected tables)

```
-- Message log (streamed chat & tool messages)
CREATE TABLE messages (
  id BIGSERIAL PRIMARY KEY,
  tenant_id TEXT NOT NULL,
  session_id TEXT NOT NULL,
  role TEXT CHECK (role IN ('user','agent','tool')),
  model TEXT,
  content JSONB,        -- ChatBlocks with types/metadata
  created_at TIMESTAMPTZ DEFAULT now(),
  trace_id TEXT,
  cost_cents NUMERIC(10,4) DEFAULT 0
);

-- Short factual memories
CREATE TABLE memory_cards (
  id BIGSERIAL PRIMARY KEY,
  tenant_id TEXT NOT NULL,
  subject TEXT,              -- e.g., user or project
  content TEXT,              -- ≤120 tokens guideline
  tags TEXT[],
  created_at TIMESTAMPTZ DEFAULT now(),
  version INTEGER DEFAULT 1
);

-- Vector index for RAG
CREATE TABLE vector_index (
  id BIGSERIAL PRIMARY KEY,
  tenant_id TEXT NOT NULL,
  doc_id TEXT,
  chunk TEXT,
  embedding VECTOR(1536),
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT now()
);

-- Registry of user-created tables (Sheets→SQL, Proxycurl results, etc.)
CREATE TABLE table_registry (
  id BIGSERIAL PRIMARY KEY,
  tenant_id TEXT NOT NULL,
  table_name TEXT NOT NULL,
  owner_user_id TEXT,
  schema JSONB,
  created_at TIMESTAMPTZ DEFAULT now()
);

-- Jobs & usage
CREATE TABLE jobs (
  id BIGSERIAL PRIMARY KEY,
  tenant_id TEXT NOT NULL,
  kind TEXT,                -- etl, enrich, sync, agent_eval
  payload JSONB,
  status TEXT,              -- queued, running, success, failed
  created_at TIMESTAMPTZ DEFAULT now(),
  updated_at TIMESTAMPTZ DEFAULT now(),
  trace_id TEXT
);

CREATE TABLE usage (
  id BIGSERIAL PRIMARY KEY,
  tenant_id TEXT NOT NULL,
  user_id TEXT,
  provider TEXT,            -- openai, anthropic, etc.
  tokens_prompt BIGINT,
  tokens_output BIGINT,
  cost_cents NUMERIC(10,4),
  created_at TIMESTAMPTZ DEFAULT now()
);
```

---

6. API Surfaces (selected)

6.1 Querying & Agents

- `POST /query/*` — deterministic tools (Sheets→SQL, SQL query, HubSpot sync, Proxycurl search, QBO ingest, etc.).
- `POST /query/agent` — streamed NDJSON agent responses (Marketing/Data Analytics).

- `POST /oneui_agent/research` — Deep Research pipeline (OpenAI Agents SDK).

6.2 MCP Gateway (REST today)

- `GET /mcp/servers` — list servers; supports tags/filters.
- `GET /mcp/servers/:id/status` — health/capabilities.
- `POST /mcp/servers/:id/connect|disconnect` — manage connections.
- `GET /mcp/capabilities` — aggregated tool/resource list.
- `POST /mcp/tools/:name/call` — invoke tool; returns streamed or buffered result.
- `GET /mcp/resources?uris=...` — fetch resources.

6.3 Runway (IaC Blueprints)

- `POST /api/runway/gemini/generate` — returns strict JSON bundle {files[], readme, instructions, checksums, correlation_id}.
- **Planned**: `POST /api/runway/validate` (fmt/tflint/OPA), `POST /api/runway/open_pr`.

6.4 IDP & Apps

- GitHub read paths (refs/tree/file), PR assist (mock), Issues/PR triage (planned).
- SRE Cockpit adapters: CloudWatch/Cloud Logging/OTel (planned).
- Gmail/Drive/Sheets endpoints; QBO ingest; HubSpot sync; Proxycurl person search/enrich.

---

7. Multi-LLM Router & Policy Engine

Router

- Inputs: task type, latency/cost budget, compliance flags, observed provider health.
- Strategy: small-model-first; A/B buckets; failover on provider errors; sticky per session when required.
- Outputs: provider/model selection, temperature/max_tokens, tool-use hints.

Policy Engine

- **Pre-prompt**: redact PII, attach policy banner, select provider/tool, set budgets.
- **Tool-time**: allow/deny actions; require approvals for high-risk operations; log intents.
- **Post-gen**: validate output (schema/regex), PII re-scan, produce receipt (trace_id, costs, policies applied).

---

8. Agent Architecture

- **Deep Research**: planner → search → read → synthesize → cite; retries/backoff; evaluation metrics captured per run; emits ChatBlocks + artifacts.
- **Marketing/Data Analytics agents**: orchestrate Proxycurl→HubSpot, Sheets/SQL, and chart ChatBlocks; budgeted tool steps with receipts.
- **Evaluation Harness**: golden sets, rubric scoring (Decision/Rationale/Citations), red-team prompts; reports persisted.

---

9. MCP Integration

- **Host mode**: UI manages registry; backend proxies calls; per-server bearer/OAuth tokens; session headers (`X-Session-Id`, `X-User-Id`).
- **Client mode**: OneUI connects to cloud MCP servers (Unity/Blender/Filesystem/GitHub); **SSE** planned for streaming.
- **Policy**: tool allowlist per tenant; approvals for destructive ops; full Langfuse traces.

---

10. Deployment & Tenancy

- **Baseline**: Cloud Run + Cloud SQL; object storage buckets; Secret Manager/KMS for secrets.
- **K8s path**: GH Actions→GHCR→Argo CD GitOps; per-tenant namespaces; auto-sleeping worker pools.

- **Provisioning**: signup → create schema/storage prefix (and namespace if K8s) → apply budgets/policies → issue API keys.

11. Observability, SLOs, and Cost

- **Traces**: Langfuse with stable `trace_id`, spans for prompts/tools/MCP calls; link to receipts.
- **Metrics**: Prometheus → Grafana (latency, throughput, errors, budgets, queue depth).
- **Logs**: Elastic (structured JSON).
- **SLO targets**: deterministic p95 ~800 ms; research p95 < 12 s; TTFC < 2 s.

12. Security & Compliance

- Least-privilege credentials; egress allowlists; PII tagging/redaction; audit logs.
- Role-based access with policy gates; approval workflow for sensitive actions.
- Data isolation via schemas/namespaces; encryption at rest and in transit.

13. Failure Modes & Mitigations

- Provider outages → router failover, cached results, exponential backoff.
- Job queue pile-ups → autoscale workers, DLQs, idempotent handlers.
- Cost spikes → per-tenant budgets, throttling, small-model routing, nightly reports.
- Data drift in RAG → scheduled re-ingest; versioned vault; eval alarms.

14. Testing & Evaluations

- Unit/integration tests for deterministic tools; schema validators for ChatBlocks.
- Agent evals with golden sets + rubric scoring; regression dashboards; red-team prompts.

- IaC: `validate-bundle` checks (fmt/tflint/OPA) and sandbox plan/apply.

15. Roadmap (Next 90 Days)

1. **MCP**: SSE + multiplexing; policy gates + approvals; example Unity/Blender servers prod-ready.
2. **Eval**: exportable reports; CI gate on golden-set deltas; cost/latency budgets.
3. **Runway**: validate-bundle + Open-PR; drift detection/auto-remediation; marketplace.
4. **SRE**: CloudWatch/Cloud Logging/OTel adapters; on-call timeline; ChatOps actions.
5. **Productization**: tenant provisioning flow, budgets, policy profiles; docs + Quickstarts.

16. Open Questions

- How far to take the IDP SRE cockpit vs deferring to vendor UIs?
- Standardize a tiny **rules DSL** for policy engine?
- Add a **meta-agent** to plan multi-tool tasks across tenants/projects?

17. Appendices

A) ChatBlocks Contract (abridged)

```
{
  "type": "table|markdown|chart|job|form|n8n-table",
  "content": {},          // varies by type
  "meta": {"title":"...","trace_id":"...","citations":[...]},
  "stream": true
}
```

B) Example Runway Output (shape)

```
{
  "bundle_id": "runway-9633c8cba9",
  "files": [{"path":"main.tf","content":"..."}],
  "readme": "How to apply",
  "checksums": {"main.tf": "sha256:..."},
  "correlation_id": "..."
}
```

C) MCP Tool Call (shape)

```
{
  "server_id": "unity-cloud-1",
  "tool": "scene.compose",
  "args": {"assets": ["s3://..."], "timeline": [{"clip":"..."}]},
  "policy": {"approval_required": true}
}
```