OmniDash - Tech Spec (DRAFT)

Omni Dash

Technical Specification — YouTube MVP & Phased Platform Expansion

Prepared by: Second Life Software

Prepared for: Omni Dash

Version: 1.1

Document Type: Authoritative Technical Specification

Status: Implementation-Ready

---

1. Purpose of This Document

This document defines the **exact technical implementation** for the Omni Dash **YouTube-only MVP** and its phased expansion across additional platforms.

It serves as:

- The **single source of truth** for engineering execution
- A shared alignment artifact between Omni Dash and Second Life Software
- A technical appendix supporting the Statement of Work (SOW)

All decisions herein are **final, intentional, and scoped** to optimize for speed, safety, reliability, and cost efficiency.

---

2. MVP Product Scope (YouTube Only)

In Scope (Phase 1 MVP)

- Responsive **web application**
- AI-assisted and AI-autonomous comment engagement (YouTube only)

- **YouTube platform integration (required and sole platform for MVP)**
- Controlled rollout to **100–200 early users**
- Paid SaaS subscriptions (Stripe)
- Multi-layer AI guardrails with full auditability
- **Firebase Authentication** (creator login + onboarding)

Explicitly Out of Scope (Phase 1 MVP)

- Native mobile applications
- Direct messages (DMs)
- Enterprise workflows
- Instagram / X / any second platform integration
- More than one platform in MVP
- Unrestricted or unsupervised AI autonomy
- **LoRA / model training / fine-tuning**

---

3. Technology Stack (Final Decisions)

Frontend

- **React + TypeScript** (Next.js)
- Tailwind CSS

Authentication

- **Firebase Authentication**
  - Email/password (baseline)
  - Optional: Google OAuth as a follow-on if required by Omni Dash
- Firebase used for authentication only; user profile and application data remain in Postgres.

Backend API

- **Python FastAPI**
- Async-first architecture
- Stateless service design

Background Processing

- Python worker services
- Queue-based job execution
- Deterministic retry and backoff logic

Database

- **PostgreSQL**
- Primary relational datastore (Cloud SQL)

Caching & Rate Limiting

- **Redis**
- Used for:
    - reply deduplication
    - rate limiting
    - short-lived state locks

AI Inference

- Open-source LLM
    - llama 3.1
- Served via **vLLM**
- Dedicated GPU inference service

Hosting Provider

- **Google Cloud Platform (GCP)**

Deployment Model

- Web & API: **Cloud Run**
- Database: **Cloud SQL (Postgres)**
- Cache: Redis (managed or VM-hosted)
- GPU inference: **Compute Engine GPU VM**
- Object storage: **Cloud Storage**

This architecture isolates GPU cost while allowing all other services to scale to zero.

---

4. AI Model Selection

MVP Model (Phase 1)

- **Llama 3.1 8B Instruct**
- Quantized (4-bit or 8-bit)
- Temperature and token limits enforced

Model Serving

- **vLLM** inference server
- Request batching enabled
- Structured output enforced (JSON schema)

---

5. System Architecture Overview

Core Services

1. **Web Application**
    - Creator dashboard
    - Unified inbox
    - AI approval workflows
    - Analytics

- Subscription management

2. **Authentication Service**
    - Firebase Authentication for creator identity
    - Backend verifies Firebase JWT and maps to internal Omni Dash user records
3. **API Service**
    - Authentication & authorization
    - Platform orchestration
    - Guardrail enforcement
    - Data access layer
4. **Worker Services**
    - Comment ingestion
    - AI generation jobs
    - Reply posting
5. **AI Inference Service**
    - GPU-hosted LLM endpoint
6. **Admin Console**
    - User management
    - Kill switches
    - Platform connection status

---

6. Platform Connector Architecture (YouTube Only in MVP)

Each platform is implemented as a **modular connector** with a standardized interface:

- OAuth authentication

- Comment ingestion
- Normalization to internal schema
- Reply posting
- Rate-limit handling

Phase 1 MVP includes the YouTube connector only.

Instagram and X connectors are explicitly deferred to later phases.

---

7. Phased Delivery Plan

Phase 0 — Core Platform Foundation (Included in Phase 1 MVP Execution)

Objective: Establish platform-agnostic infrastructure and AI orchestration (even though only YouTube is implemented in MVP).

Deliverables

- Firebase Authentication integration (creator login + session)
- User/team model (creator + optional team members)
- Stripe subscription integration (paid access gating)
- Unified comment inbox foundation (platform-neutral internal schema)
- AI orchestration pipeline (generate → validate → approve/post)
- Guardrails framework (v1)
- Audit logging system
- Admin controls
- Observability & logging

Phase 1 — YouTube MVP (Scope-Committed)

Objective: Deliver production-ready MVP functionality **for YouTube only**, including Assisted and Autonomous AI modes with full guardrails and auditability.

Phase 1 Assumptions

- AI development tools (Copilot / Cursor / ChatGPT) may be used to accelerate implementation, but all architecture and safety decisions remain human-owned.
- **No LoRA / training / fine-tuning** in Phase 1.
- **One platform only:** YouTube.
- Web-first only.
- Production-ready (reliable and auditable), not demo-only.

Phase 1 Work Breakdown (No Hours)

1) Product & Technical Planning

Deliverables

- Finalize architecture and service boundaries (Web/API/Workers/Inference)
- Finalize data model (users, comments, threads, replies, audit logs, rulesets)
- Finalize guardrail policy thresholds and enforcement ordering
- Formalize platform connector contract (implemented for YouTube in MVP)

2) Frontend (Next.js Web App)

Deliverables

- Authentication + onboarding flows (Firebase Auth)
- Creator dashboard
- Unified inbox UI (threaded comments, filtering, search)
- Assisted Mode UX:
    - generate suggestions

    - approve/edit
    - post
- Autonomous Mode controls:
    - opt-in activation
    - reply limits
    - active hours
    - kill switch
- Analytics views (MVP)
- Production UI readiness:
    - error states
    - loading states
    - guardrail-block messaging

---

3) Backend API (FastAPI)

Deliverables

- Firebase JWT verification + internal user mapping
- Core data models + API endpoints
- Comment lifecycle state machine (ingest → eligible → draft → validate → approve/post → log)
- Creator voice profile system:
    - tone, length, emoji rules, phrase bans, examples
- Guardrail enforcement module:
    - eligibility, behavioral, safety checks
- Assisted Mode orchestration endpoints
- Autonomous Mode orchestration endpoints
- Analytics aggregation endpoints (MVP-level)

---

4) YouTube Integration

Deliverables

- OAuth connection flow
- Comment ingestion:
  - thread handling
  - pagination
  - polling cadence
- Reply posting
- Rate limit handling + retries
- Integration testing + edge-case handling

---

5) AI Inference & Prompting

Deliverables

- GPU inference service setup (vLLM + Llama 3.1 8B Instruct)
- Prompt templates:
  - system rules
  - voice profile injection
  - few-shot examples
  - structured JSON output
- Voice example handling:
  - capture approved replies
  - reuse as examples
- Safety validation:
  - post-generation checks
  - optional “judge” evaluation pass (MVP level)
- Prompt testing across known risky inputs and edge cases

---

6) Guardrails & Safety (Cross-cutting)

Deliverables

- Reply deduplication (no multiple replies per comment)
- Rate limiting (per user + global caps)
- Active hours enforcement
- PII detection and blocking
- Toxicity / unsafe input filtering
- Audit logging + traceability:
  - why a reply was allowed/blocked
  - what rules were applied
  - what mode posted it (assisted/autonomous)

---

7) Infrastructure & Deployment (GCP)

Deliverables

- Cloud Run deployment for web + API
- Cloud SQL provisioning and migrations
- Redis provisioning
- GPU VM provisioning for inference service
- Secrets management via GCP Secret Manager
- IAM / networking configuration
- Observability:
  - structured logs
  - error reporting
  - basic alerting

---

8) QA, Stabilization & Delivery

Deliverables

- Integration testing across:
    - YouTube ingest
    - reply posting
    - assisted approvals
    - autonomous posting flows
- Edge-case handling and bug fixes
- Deployment hardening and rollback procedures
- Client handoff:
    - walkthrough
    - runbook
    - admin training (kill switch, audit logs, platform connection troubleshooting)

Phase 2 — Instagram Integration (Deferred)

Objective: Add second platform using the existing connector framework.

Scope

- OAuth connection
- Comment ingestion
- Reply posting
- Shared inbox & analytics

Phase 3 — X (Twitter) Integration (Deferred)

Objective: Add third platform post-MVP.

Scope

- Mentions/replies ingestion

- Reply posting
- Enhanced throttling due to platform sensitivity

---

8. Guardrail System (Mandatory)

Omni Dash implements a **five-layer guardrail system** enforced deterministically by application logic.

Guardrail Layers

1. Eligibility
2. Behavioral
3. Voice & Content
4. Safety & Compliance
5. Control & Auditability

Guardrail Matrix

Layer	Description	Enforcement
Eligibility	Deduplication, topic blocks, toxic input	Code + classifiers
Behavioral	Rate limits, active hours, cooldowns	Code
Voice	Tone, length, banned phrases	Prompt constraints
Safety	PII, hate, claims, policy	Filters + classifiers
Control	Opt-in, kill switch, audit logs	UX + code

Mandatory Phase 1 MVP Guardrails

- Reply deduplication (no multiple replies per comment)
- Max replies per hour/day
- Active hours enforcement
- Autonomous Mode opt-in
- Kill switch

- Input & output safety checks
- Full audit logging

---

9. Comment Lifecycle State Machine

1. Comment ingested
2. Eligibility evaluated
3. AI reply generated
4. Post-generation safety validation
5. Assisted approval or autonomous post
6. Audit logged
7. Analytics updated

---

10. Creator Voice System

Creators configure a **Voice Profile** consisting of:

- Tone
- Length preferences
- Emoji usage rules
- Phrase bans
- Example replies

Voice profiles are applied at generation time via structured prompts and example injection.

---

11. LoRA / Model Tuning Strategy (Out of Scope for Phase 1)

Phase X MVP Position

- **No model training in MVP**

- Prompting + voice profiles only

Post-MVP (Phase 4+)

LoRA tuning will be introduced once:

- 300–1,000+ approved replies exist
- Measurable quality improvements are required
- Cost or latency optimization becomes necessary

LoRA adapters will be:

- Versioned
- Optional
- Reversible

---

12. Security & Reliability

- OAuth tokens encrypted at rest
- Firebase JWT verification on backend
- Least-privilege scopes
- Idempotent posting
- Retry with backoff
- Dead-letter queues
- Secrets via GCP Secret Manager

---

13. Cost Controls

- Serverless compute for non-GPU workloads
- Single GPU inference service
- Quantized model
- Request batching

- Autonomous mode volume caps
- Rate limiting per user

---

14. Deliverables Summary

This document defines:

- What will be built
- How it will be built
- In what order it will be delivered
- How safety, cost, and trust are enforced

Any changes require a formal amendment.

---

15. Approval

Proceeding with implementation confirms alignment with this technical specification.