# Backend Architecture: Intelligent Time & Productivity Tracker (Supabase)

## 1. High-Level Architecture

- Data Layer: PostgreSQL on Supabase with RLS. - APIs: Supabase REST & Realtime + Edge Functions for AI, uploads, webhooks. - Auth: Supabase Auth (email/password + SSO, org roles). - Files: Supabase Storage (voice, screenshots, transcripts). - Queue/Async: Supabase Queue or pg_cron for background AI tasks. - Search/AI: pgvector for embeddings & classification results. - Analytics: Materialized views for dashboards. - Offline: PWA local cache + sync. - Figma: Connects via Edge Functions, not direct DB.

## 2. Multi-Tenancy & Identity

- Organizations, Profiles, Memberships (user ↔ org roles). - Teams & Departments for internal scoping. - Roles: owner, admin, manager, member, viewer. - RLS ensures users only see their org data.

## 3. Data Model (Core Tables)

- organizations, profiles, memberships. - projects, teams, departments. - work_modes, task_categories, activity_types. - time_logs: main record (comment, times, activity, project, etc.). - ai_extractions: model outputs per log. - time_log_embeddings: vector embeddings for semantic recall. - attachments: voice/image files. - user_day_summaries: rollups per user/day.

## 4. Row-Level Security

- Enable RLS on all tables. - Policies for select/insert/update scoped to user's org membership. - Example: users can insert their own logs, admins can edit all org logs.

## 5. Edge Functions Needed

1. log.create → store log, enqueue classification. 2. ai.classify → run transcription/OCR + LLM, update structured fields. 3. log.finalize → confirm processed log. 4. upload.presign → return pre-signed URLs for uploads. 5. insights.rollup → nightly job for summaries. 6. webhooks.ingest → ingest events from Slack/Jira/Calendar.

## 6. Realtime & Views

- Realtime channels on time_logs per org. - Materialized views for weekly/monthly summaries. - Nightly refresh or trigger-based.

## 7. AI & NLP Layer

- Use LLM with strict JSON schema prompts. - Disambiguate with fuzzy matching & embeddings. - Store embeddings in pgvector. - Risk models: burnout (workload), utilization (focus vs meetings).

# 8. Storage Layout

- voices/{org_id}/{log_id}/... - images/{org_id}/{log_id}/... - exports/{org_id}/... - All private, pre-signed URL access only.

# 9. Offline & Sync Strategy

- Local cache in IndexedDB. - Optimistic UI with temp UUIDs. - Conflict resolution: server wins via updated_at + version. - Background sync retries.

# 10. Analytics Path

- Phase 1: rollups in Postgres. - Phase 2: stream to OLAP warehouse (BigQuery/ClickHouse). - Supabase remains source of record.

# 11. Security & Keys

- Never expose service role in Figma. - Use Supabase anon key with RLS. - Figma plugin calls Edge Functions only. - Auth via JWT + org scoping.

# 12. Implementation Steps

1. Create schema + RLS policies. 2. Build Edge Functions for log + AI. 3. Seed default vocab (work modes, categories). 4. Materialized views for insights. 5. Connect Figma Make → Edge Functions (upload + log).