ATUM DESK — Implementation Blueprint & Builder AI Execution Prompt

Version: 1.0 • Date: 2026-02-12 • Scope: End-to-end ticketing from external customer portal to internal master console, with optional local AI triage.

Purpose

ATUM DESK is a branded, production-shaped helpdesk/ticketing platform. Customers open tickets through an external portal. Tickets flow into an internal master console where a Support Manager accepts, assigns, and tracks SLA. Everything is auditable. No Docker. No external APIs. Local-first deployment.

Non-negotiables

• Brand parity: ATUM landing page layout + logo.svg + wordmark.svg + typography + glass theme must match the current ATUM project.

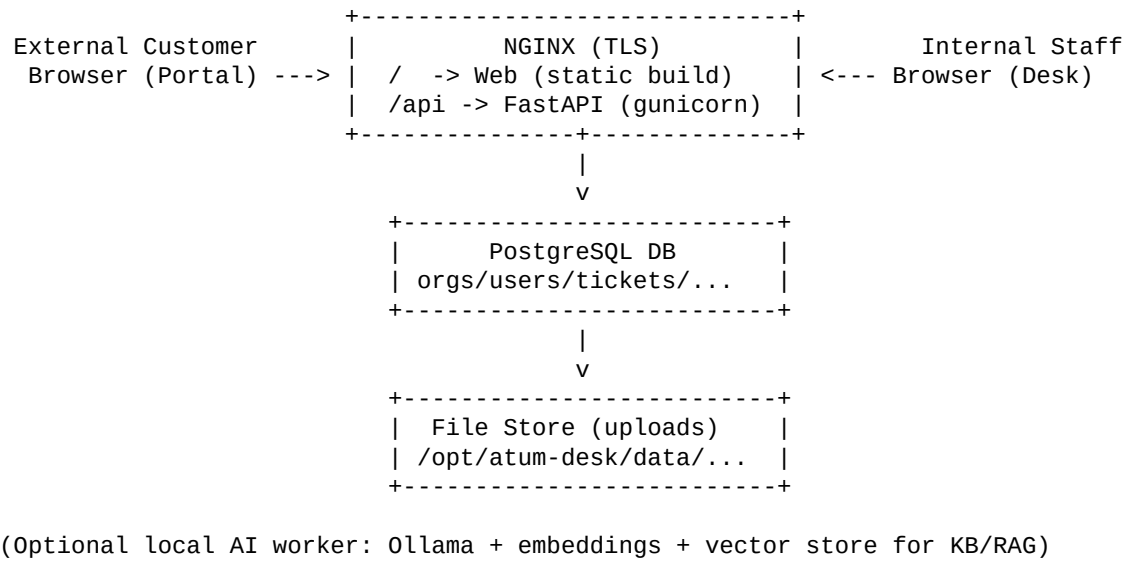
• Workflow parity: Manager acceptance gate before work starts; customer cannot manipulate internal status.

• Multi-tenant isolation: customers only see their org; internal staff see all (by role).

• Security: RBAC, audit trail, safe file uploads, rate limiting on auth.

• Deployment: bare-metal (systemd + nginx), offline-capable, no external API calls.

High-level architecture



Core workflows

This is the operational truth table of ATUM DESK. Implement it exactly.

Workflow	Who	Outcome
Create ticket	Customer	Ticket created in NEW state with attachments; notification fired
Accept ticket	Support Manager	Ticket moves to ACCEPTED; SLA clocks start; owner recorded
Assign ticket	Manager/Agent	Ticket assigned; technician works
Request info	Agent	Status WAITING_CUSTOMER; SLA pause rules apply
Resolve/Close	Agent/Manager	Resolution + root cause captured; ticket CLOSED

System deliverables checklist

■ Web UI: ATUM landing cloned as '/', plus /portal/* (customer) and /desk/* (internal) routes with ATUM glass styling.

■ Backend API: FastAPI with JWT auth, RBAC, tenant checks, ticket CRUD, comments, attachments, manager accept/assign/status transitions.

■ Database: PostgreSQL schema with migrations (Alembic).

■ Uploads: safe storage with size limits, sha256 hashing, allowed mime types, per-org access control.

■ Notifications: SMTP (no external API) for ticket created/accepted/status/comment events.

■ SLA engine: first response + resolution timers (start at ACCEPTED, pause on WAITING_CUSTOMER).

■ Audit: immutable append-only audit_log for key state transitions and sensitive actions.

■ Deployment: systemd services + nginx config + install script; single-command startup.

■ Docs: operator runbook + first-ticket walkthrough + troubleshooting.

Directory structure (from scratch repo)

```
atum-desk/
  api/
    app/
      main.py
      config.py
      auth/
      db/
      models/
      routers/
      services/
      middleware/
      utils/
    migrations/
    requirements.txt
  web/
    package.json
    vite.config.*
    src/
      atum/              # ATUM look rebuilt locally (tokens/components/layouts)
      pages/
      routes/
      api/
    public/
      brand/
        logo.svg
        wordmark.svg
  infra/
    nginx/atum-desk.conf
    systemd/atum-desk-api.service
    scripts/install.sh
  docs/
    RUNBOOK.md
    ACCEPTANCE_TESTS.md
```

Database schema (minimum)

```
CREATE TABLE organizations (
  id UUID PRIMARY KEY,
  name TEXT NOT NULL,
  status TEXT NOT NULL DEFAULT 'active',
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE users (
  id UUID PRIMARY KEY,
  org_id UUID REFERENCES organizations(id),
  email TEXT UNIQUE NOT NULL,
  name TEXT NOT NULL,
  role TEXT NOT NULL,
  password_hash TEXT NOT NULL,
  is_active BOOLEAN NOT NULL DEFAULT true,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE services (
  id UUID PRIMARY KEY,
  org_id UUID REFERENCES organizations(id),
  name TEXT NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

```
CREATE TABLE tickets (
  id UUID PRIMARY KEY,
  org_id UUID NOT NULL REFERENCES organizations(id),
  service_id UUID REFERENCES services(id),
  created_by UUID NOT NULL REFERENCES users(id),
  accepted_by UUID REFERENCES users(id),
  assigned_to UUID REFERENCES users(id),
  status TEXT NOT NULL,
  priority TEXT NOT NULL,
  category TEXT,
  title TEXT NOT NULL,
  description TEXT NOT NULL,
  accepted_at TIMESTAMPTZ,
  resolved_at TIMESTAMPTZ,
  closed_at TIMESTAMPTZ,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE ticket_comments (
  id UUID PRIMARY KEY,
  ticket_id UUID NOT NULL REFERENCES tickets(id),
  author_id UUID NOT NULL REFERENCES users(id),
  body TEXT NOT NULL,
  is_internal BOOLEAN NOT NULL DEFAULT false,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE ticket_attachments (
  id UUID PRIMARY KEY,
  ticket_id UUID NOT NULL REFERENCES tickets(id),
  uploaded_by UUID NOT NULL REFERENCES users(id),
  file_path TEXT NOT NULL,
  file_name TEXT NOT NULL,
  mime_type TEXT NOT NULL,
  size_bytes BIGINT NOT NULL,
  sha256 TEXT NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE sla_policies (
  id UUID PRIMARY KEY,
  org_id UUID REFERENCES organizations(id),
  priority TEXT NOT NULL,
  first_response_minutes INT NOT NULL,
  resolution_minutes INT NOT NULL
);

CREATE TABLE audit_log (
  id UUID PRIMARY KEY,
  actor_id UUID REFERENCES users(id),
  action TEXT NOT NULL,
  entity_type TEXT NOT NULL,
  entity_id UUID,
  old_value JSONB,
  new_value JSONB,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

API surface (minimum)

All endpoints are under /api and require JWT. Tenant and RBAC checks are enforced server-side.

Area	Endpoint	Notes
Auth	POST /auth/login	Returns JWT + user profile
Auth	GET /me	Session validation
Customer	POST /tickets	Create NEW ticket; attachments separate
Customer	GET /tickets	List tickets in org
Customer	GET /tickets/{id}	Detail (org-scoped)
Customer	POST /tickets/{id}/comments	Public comment
Customer	POST /tickets/{id}/attachments	Upload file (limits + hash)
Internal	GET /internal/tickets/new	Unaccepted queue
Internal	POST /internal/tickets/{id}/accept	Starts SLA; records accepted_by/accepted_at
Internal	POST /internal/tickets/{id}/assign	Assign to agent/team
Internal	POST /internal/tickets/{id}/status	State transitions with validation
Internal	POST /internal/tickets/{id}/comments	Internal note/comment
Admin	CRUD /orgs, /users, /sla	Optional MVP admin panel

Middleware & guards

• Auth middleware: verifies JWT, loads user, attaches to request context.

• Tenant guard: enforces org_id scope on customer endpoints.

• RBAC guard: role-based permission checks for internal/admin endpoints.

• Rate limiter: login + ticket create (to block abuse).

• Upload validator: allowed mime types, max size, sha256 hashing, path traversal protection.

• Audit writer: records state transitions and sensitive actions.

Optional local AI triage (Phase 2)

• Run entirely local (no external APIs).

• Use Ollama for LLM inference; pick a small instruct model that fits CPU/RAM (e.g., Qwen2.5 Instruct 3B-7B depending on hardware).

• Embeddings: a compact local embedding model; store vectors in Chroma/FAISS.

• Use cases: ticket summarization, category/priority suggestion, similar-ticket retrieval, KB/RUNBOOK retrieval (RAG).

• AI is advisory only; never auto-closes or auto-accepts.

requirements.txt (API)

```
fastapi==0.115.0
uvicorn[standard]==0.30.6
gunicorn==22.0.0
pydantic==2.9.2
pydantic-settings==2.5.2
python-jose==3.3.0
passlib[bcrypt]==1.7.4
python-multipart==0.0.9
sqlalchemy==2.0.35
alembic==1.13.2
psycopg[binary]==3.2.3
orjson==3.10.7
httpx==0.27.2
tenacity==9.0.0
structlog==24.4.0
```

Note: versions are pinned for stability. Builder AI may adjust to match the existing ATUM environment but must keep pinned constraints.

Acceptance test: first ticket end-to-end

```
1) Start services: postgres, api (systemd), nginx (serves web + /api proxy)
2) Seed: create org + customer_admin + manager + agent
3) Customer logs into /portal and creates a P2 ticket with an attachment
4) Ticket appears in /desk/inbox as NEW
5) Manager clicks ACCEPT -> ticket becomes ACCEPTED, SLA starts
6) Manager assigns to agent -> ASSIGNED
7) Agent posts internal note + public comment (internal must not be visible to customer)
8) Agent sets WAITING_CUSTOMER -> SLA pauses
9) Customer replies -> policy-driven transition back to IN_PROGRESS
10) Agent RESOLVES -> Manager CLOSES
11) Verify audit_log contains accept/assign/status changes + attachment access events
```