AuditPilot Project Guide

Version: 1.0
Generated: 2026-02-20

1. Project Summary

AuditPilot is a full-stack application for automated inspection report generation from uploaded files (images, audio, PDFs). It runs AI-based extraction/classification pipelines in the backend, stores results in PostgreSQL (Supabase), and exposes dashboards/reports in a Next.js frontend.

Core outcomes:
- Upload inspection files (image/audio/pdf)
- Automatically process each file with AI models
- Produce structured findings with severity/confidence
- Aggregate inspection-level risk and narrative summary
- Surface items requiring human review

2. Tech Stack

Frontend:
- Next.js 14 (App Router)
- React 18 + TypeScript
- Tailwind CSS + shadcn/ui + Radix UI
- Supabase auth client (`@supabase/ssr`, `@supabase/supabase-js`)

Backend:
- FastAPI
- SQLAlchemy ORM + PostgreSQL
- Supabase-hosted Postgres
- pgvector extension for 384-dim finding embeddings
- Hugging Face Inference API for multimodal AI tasks

AI models used:
- Image captioning: `Salesforce/blip-image-captioning-large`
- Audio transcription: `openai/whisper-large-v3`
- Classification: `facebook/bart-large-mnli`
- Summarization: `facebook/bart-large-cnn`
- Embeddings: `sentence-transformers/all-MiniLM-L6-v2`

Deployment:
- Frontend: designed for Vercel
- Backend: Render (`render.yaml`)

3. Monorepo Layout

Top-level:
- `app/` Next.js routes/pages
- `components/` UI and feature components

- `lib/` API client, Supabase clients, utilities
- `backend/` FastAPI service, DB models, routes, migrations, workers
- `types/` shared TypeScript types (currently minimal)
- `render.yaml` Render deployment spec for backend

4. Frontend Architecture

4.1 Route Map

- `/` landing page (`app/page.tsx`)
- `/login` email/password login via Supabase (`app/login/page.tsx`)
- `/signup` email/password signup (`app/signup/page.tsx`)
- `/auth/callback` auth code exchange route (`app/auth/callback/route.ts`)
- `/dashboard` inspection list + metrics + queues (`app/dashboard/page.tsx`)
- `/inspection/[id]/upload` upload files + processing status (`app/inspection/[id]/upload/page.tsx`)
- `/inspection/[id]` inspection report view (`app/inspection/[id]/page.tsx`)

4.2 Auth and Session Handling

- Middleware (`middleware.ts`) uses `lib/supabase/middleware.ts` to refresh/read session.
- Unauthenticated users are redirected to `/login`.
- Authenticated users visiting `/login` or `/signup` are redirected to `/dashboard`.
- API calls include `Authorization: Bearer <access_token>` pulled from Supabase session.

4.3 API Integration Layer

`lib/api/client.ts` wraps backend calls:
- Inspections: list/create/get/stats
- Files: list/upload
- Findings: list/stats/review queue

Upload path uses `XMLHttpRequest` to expose upload progress callbacks.

4.4 Main Feature Components

- `NewInspectionDialog`: create inspection
- `FileUpload`: drag-drop uploader with type/size constraints
- `FileStatusList`: polling file processing statuses
- `ReviewQueue`: low-confidence findings list
- `FindingsChart`: findings category pie chart
- `CostTracker`: estimated model-cost panel (currently mock logic)

5. Backend Architecture

5.1 Entry Point and Middleware

File: `backend/main.py`
- Registers CORS with `allow_origins=["*"]`
- Mounts route modules: files, findings, inspections, organizations

- Health endpoint: `GET /health`

5.2 Request Auth Model

File: `backend/app/core/auth.py`
- Expects Supabase JWT bearer token.
- Validates token via `supabase.auth.get_user()`.
- Maps user -> deterministic org UUID using MD5(user.id).
- Auto-upserts org record (`organizations` table) for that UUID.

Implication: API tenancy is user-isolated by derived org, not by explicit org selection in client requests.

5.3 Persistence and Data Access

- DB session factory: `backend/app/core/database.py`
- SQLAlchemy repositories:
  - `InspectionRepository`
  - `FileRepository`
  - `FindingRepository`

5.4 Asynchronous Processing Pipeline

Background processing starts after upload (`BackgroundTasks`):
1. File uploaded and DB record created with `pending`
2. Worker entry `process_file_background()` (`backend/app/workers/file_processor.py`)
3. File status set `processing`
4. File downloaded from local storage service
5. Pipeline by type:
   - Image: caption -> classify -> embed -> finding
   - Audio: transcribe -> classify -> embed -> finding
   - PDF: extract text -> classify -> embed -> finding
6. File status set `completed` (or `failed` with error)
7. Inspection progress updated
8. If all files done, `InspectionCompletionService.finalize()` computes:
   - risk level (max severity)
   - narrative summary (BART-CNN)
   - final inspection status (`review` if any finding needs review else `completed`)

5.5 Storage

File: `backend/app/services/storage_service.py`
- Current implementation is local filesystem storage.
- Base path: `${LOCAL_UPLOAD_DIR or tempfile}/auditpilot_uploads`
- `generate_presigned_url` returns local file path (not cloud URL).

6. API Reference

Authentication:

- Most endpoints require `Authorization: Bearer <Supabase access token>`.
- No explicit `X-Org-Id` header is required in the current backend implementation.

Endpoints:

Organizations:
- `POST /organizations` create org (dev/bootstrap)
- `GET /organizations/{org_id}` fetch org

Inspections:
- `GET /inspections` list org-scoped inspections
- `POST /inspections` create inspection
- `GET /inspections/{inspection_id}` get inspection
- `GET /inspections/stats` aggregate dashboard stats

Files:
- `POST /inspections/{inspection_id}/files` upload one or many files (multipart)
- `GET /inspections/{inspection_id}/files` list files
- `GET /files/{file_id}` file metadata and local download path

Findings:
- `GET /inspections/{inspection_id}/findings` list inspection findings
- `GET /findings/stats` org aggregate by category
- `GET /findings/review-queue` low-confidence findings across org

Operational limits:
- Upload size cap: 50MB per file
- Allowed mime/file types: image/jpeg/png/jpg, audio/mp3/m4a/wav, application/pdf

7. Database and Migrations

Migrations: `backend/migrations/000..007`

Extensions:
- `uuid-ossp`
- `vector`

Tables:
- `organizations`
- `users`
- `inspections`
- `files`
- `findings` (includes `embedding VECTOR(384)` + HNSW index)
- `human_reviews`
- `usage_logs`

Relationship highlights:
- Organization -> users, inspections
- Inspection -> files, findings, usage_logs

- File -> findings, usage_logs
- Finding -> human_reviews

8. Environment Variables

Frontend (`.env.local`, template in `.env.example`):
- `NEXT_PUBLIC_API_URL`
- `NEXT_PUBLIC_SUPABASE_URL`
- `NEXT_PUBLIC_SUPABASE_ANON_KEY`
- `NEXT_PUBLIC_APP_URL`
- `NEXT_PUBLIC_API_BASE_URL`
- `NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY` (legacy/not used in current auth path)
- `CLERK_SECRET_KEY` (legacy/not used in current auth path)
- `NEXT_PUBLIC_DEFAULT_ORG_ID` (legacy from old org-header approach)

Backend (`backend/.env`):
- `DATABASE_URL`
- `SUPABASE_URL`
- `SUPABASE_KEY` or `SUPABASE_ANON_KEY` (JWT validation client)
- `SUPABASE_SERVICE_KEY` (for service-level access if needed)
- `HF_API_TOKEN` (required by current HF client)
- `LOCAL_UPLOAD_DIR` (optional)

Important naming note:
- `render.yaml` currently defines `HUGGINGFACE_API_TOKEN`, while code reads `HF_API_TOKEN`.
- For production correctness, align variable names.

9. Local Development Guide

9.1 Frontend

```bash
npm install
cp .env.example .env.local
npm run dev
```

Frontend URL: `http://localhost:3000`

9.2 Backend

```bash
cd backend
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
# configure backend/.env
python apply_migrations.py
uvicorn main:app --reload
```

```

Backend URL: `http://localhost:8000`
Docs: `http://localhost:8000/docs`

## 10. Deployment

Render service spec (`render.yaml`):
- Runtime: Python 3.10.12
- Root dir: `backend`
- Build: `pip install -r requirements.txt`
- Start: `uvicorn main:app --host 0.0.0.0 --port $PORT`

Recommended deployment checks:
- Ensure DB extensions (`uuid-ossp`, `vector`) are enabled.
- Ensure env var names match code (`HF_API_TOKEN`).
- Set secure CORS origins (replace wildcard in production).
- Confirm writable local upload dir if using ephemeral disk.

## 11. Security and Reliability Notes

- Current CORS policy is open (`*`); tighten for production.
- `FileUpload`/backend validation enforces type and max size, but no malware scanning.
- Local storage is not durable across ephemeral deployments.
- Background tasks run in-process; high-volume workloads may require queue worker architecture.
- `console.log` in `lib/api/client.ts` leaks token presence state to browser console.

## 12. Known Gaps and Improvement Backlog

- Move storage to durable object store (S3/Supabase Storage).
- Introduce real async queue (Celery/RQ/Arq) for processing.
- Add observability: structured logs + tracing + error reporting.
- Add test coverage (unit/integration/e2e).
- Normalize env var naming across docs/config/code.
- Tighten auth/tenant model and formalize org membership model.
- Add usage log writes to support real cost tracking.

## 13. File-by-File Inventory

This appendix lists tracked project files for full repository orientation.

```text
AuditPilot-PRD-Complete-Tasks.md
README.md
app/auth/callback/route.ts
app/dashboard/page.tsx
app/globals.css
app/inspection/[id]/page.tsx
```
```

app/inspection/[id]/upload/page.tsx
app/layout.tsx
app/login/page.tsx
app/page.tsx
app/signup/page.tsx
backend/README.md
backend/app/__init__.py
backend/app/api/__init__.py
backend/app/api/routes/__init__.py
backend/app/api/routes/files.py
backend/app/api/routes/findings.py
backend/app/api/routes/inspections.py
backend/app/api/routes/organizations.py
backend/app/core/auth.py
backend/app/core/database.py
backend/app/core/supabase.py
backend/app/models/__init__.py
backend/app/models/file.py
backend/app/models/finding.py
backend/app/models/human_review.py
backend/app/models/inspection.py
backend/app/models/organization.py
backend/app/models/usage_log.py
backend/app/models/user.py
backend/app/repositories/__init__.py
backend/app/repositories/file_repository.py
backend/app/repositories/finding_repository.py
backend/app/repositories/inspection_repository.py
backend/app/schemas/__init__.py
backend/app/schemas/file.py
backend/app/schemas/finding.py
backend/app/schemas/human_review.py
backend/app/schemas/inspection.py
backend/app/schemas/organization.py
backend/app/schemas/usage_log.py
backend/app/schemas/user.py
backend/app/services/__init__.py
backend/app/services/audio_processor.py
backend/app/services/embedding_service.py
backend/app/services/hf_client.py
backend/app/services/image_processor.py
backend/app/services/inspection_completion_service.py
backend/app/services/job_tracker.py
backend/app/services/pdf_processor.py
backend/app/services/storage_service.py
backend/app/workers/__init__.py
backend/app/workers/file_processor.py
backend/apply_migrations.py
backend/main.py

backend/migrations/000_enable_extensions.sql
backend/migrations/001_create_organizations.sql
backend/migrations/002_create_users.sql
backend/migrations/003_create_inspections.sql
backend/migrations/004_create_files.sql
backend/migrations/005_create_findings.sql
backend/migrations/006_create_human_reviews.sql
backend/migrations/007_create_usage_logs.sql
backend/requirements.txt
components.json
components/features/CostTracker.tsx
components/features/FileStatusList.tsx
components/features/FileUpload.tsx
components/features/FindingsChart.tsx
components/features/NewInspectionDialog.tsx
components/features/ReviewQueue.tsx
components/features/StatsCard.tsx
components/ui/badge.tsx
components/ui/button.tsx
components/ui/card.tsx
components/ui/dialog.tsx
components/ui/dropdown-menu.tsx
components/ui/index.ts
components/ui/input.tsx
components/ui/label.tsx
components/ui/progress.tsx
components/ui/select.tsx
components/ui/table.tsx
components/ui/tabs.tsx
components/ui/toast.tsx
components/ui/toaster.tsx
docs/PROJECT_GUIDE.md
lib/api/client.ts
lib/hooks/use-toast.ts
lib/supabase.ts
lib/supabase/client.ts
lib/supabase/middleware.ts
lib/supabase/server.ts
lib/utils.ts
middleware.ts
next-env.d.ts
next.config.js
package-lock.json
package.json
postcss.config.js
render.yaml
tailwind.config.ts
tsconfig.json
tsconfig.tsbuildinfo

```
```