Corrado RAG — Architecture Document

Overview

Product: Local RAG system for document Q&A with intelligent ingestion

Core Innovation: "Chips" — contextual metadata embedded directly into chunk text, making document identity and context part of the vector space rather than a fragile filter layer.

MVP Scope:

- Single-user localhost app (no auth)
- Manual file upload (leases as test case)
- Chat interface with conversation context
- Supabase for storage + pgvector

Tech Stack:

- Next.js (App Router)
- React
- Supabase (Postgres + pgvector + Storage)
- OpenAI text-embedding-3-small
- Claude Sonnet
- Vercel (deployment)

File Structure

```
corrado-rag/
│
├── src/
│   │
│   ├── app/
│   │   ├── page.tsx
│   │   ├── layout.tsx
│   │   ├── chat/page.tsx
│   │   ├── upload/page.tsx
│   │   └── api/
│   │       ├── chat/route.ts
│   │       ├── upload/route.ts
│   │       └── conversations/route.ts
│   │
│   ├── components/
│   │   ├── ChatWindow.tsx
│   │   ├── MessageInput.tsx
│   │   ├── FileDropzone.tsx
│   │   └── ProcessingStatus.tsx
│   │
│   ├── file-client/
│   │   ├── extractor.ts
│   │   ├── cleaner.ts
│   │   ├── classifier.ts
│   │   ├── chips.ts
│   │   ├── chunker.ts
│   │   ├── embedder.ts
│   │   ├── orchestrator.ts
│   │   ├── get-template.ts
│   │   ├── save-document.ts
│   │   └── save-chunks.ts
│   │
│   ├── chat-client/
│   │   ├── retrieval.ts
│   │   ├── prompt.ts
│   │   ├── llm.ts
│   │   ├── orchestrator.ts
│   │   ├── get-history.ts
│   │   └── save-message.ts
│   │
│   ├── supabase.ts
│   │
│   └── types/
│       └── index.ts
│
├── supabase/
│   └── migrations/
│       └── 001_initial_schema.sql
│
├── .env.local
├── package.json
└── README.md
```

File Descriptions

app/

File	Purpose
page.tsx	Landing page, redirects to /chat
layout.tsx	Root layout wrapper
chat/page.tsx	Chat interface
upload/page.tsx	File upload interface with classification display
api/chat/route.ts	Handles chat messages
api/upload/route.ts	Receives files, triggers pipeline
api/conversations/route.ts	Get and delete conversations

components/

File	Purpose
ChatWindow.tsx	Displays message thread
MessageInput.tsx	Text input for chat
FileDropzone.tsx	Drag-and-drop upload
ProcessingStatus.tsx	Shows pipeline progress and classification result

file-client/

File	Purpose
extractor.ts	Routes file to correct text extraction method
cleaner.ts	Normalizes whitespace and fixes OCR errors
classifier.ts	LLM determines file type
chips.ts	Fetches template, LLM extracts chips from text
chunker.ts	Splits text, prepends chips to each chunk
embedder.ts	Calls OpenAI embedding API
orchestrator.ts	Runs full ingestion pipeline
get-template.ts	Fetches chip template from DB by file type
save-document.ts	Writes document record to DB
save-chunks.ts	Writes chip-chunks and embeddings to DB

chat-client/

File	Purpose
retrieval.ts	Embeds query, searches for matching chunks
prompt.ts	Assembles system message, chunks, history
llm.ts	Calls Claude API
orchestrator.ts	Runs full chat flow
get-history.ts	Fetches full conversation history
save-message.ts	Writes new message to DB

root src/

File	Purpose
supabase.ts	Initializes Supabase connection
types/index.ts	All shared type definitions

Database Schema

```
sql
```

```sql
-- Enable pgvector extension
create extension if not exists vector;

-- File type templates (chip schemas live here)
create table file_type_templates (
  id uuid primary key default gen_random_uuid(),
  type_name text unique not null,
  chip_fields jsonb not null,
  extraction_prompt text,
  created_at timestamptz default now()
);

-- Seed lease template
insert into file_type_templates (type_name, chip_fields) values (
  'lease',
  '["property_address", "unit_number", "tenant_name", "landlord", "lease_start", "lease_end", "monthly_rent", "security_depo
);

-- Seed misc template (fallback)
insert into file_type_templates (type_name, chip_fields) values (
  'misc',
  '["document_title", "date", "parties_involved", "summary"]'
);

-- Documents table
create table documents (
  id uuid primary key default gen_random_uuid(),
  original_name text not null,
  clean_name text,
  file_type text references file_type_templates(type_name),
  file_url text,
  full_text text,
  status text default 'pending',
  uploaded_at timestamptz default now(),
  processed_at timestamptz
);

-- Chip-chunks table
create table chip_chunks (
  id uuid primary key default gen_random_uuid(),
  document_id uuid references documents(id) on delete cascade,
  content text not null,
  chunk_index int,
  embedding vector(1536),
  created_at timestamptz default now()
);

-- Vector similarity index
create index on chip_chunks using ivfflat (embedding vector_cosine_ops);

-- Conversations table
create table conversations (
  id uuid primary key default gen_random_uuid(),
  created_at timestamptz default now()
);

-- Messages table
create table messages (
  id uuid primary key default gen_random_uuid(),
  conversation_id uuid references conversations(id) on delete cascade,
  role text not null,
  content text not null,
  created_at timestamptz default now()
);
```

Pipeline Flows

Ingestion (file-client)

```
┌─────────────────────────────────────────────────────────────
│                                                  │
│  Upload                                          │
│    │                                             │
│    ▼                                             │
│  extractor.ts ──────────── Detects file type, extracts raw text   │
│    │                                             │
│    ▼                                             │
│  cleaner.ts ─────────────── Normalizes whitespace, fixes OCR errors   │
│    │                                             │
│    ▼                                             │
│  classifier.ts ─────────── LLM returns file type (lease, misc, etc.)   │
│    │                                             │
│    ▼                                             │
│  get-template.ts ──────── Fetches chip fields from DB   │
│   │                                         │
│   ▼                                         │
│ chips.ts ──────────── LLM extracts chip values from full text        │
│   │                                         │
│   ▼                                         │
│ chunker.ts ────────── Splits text, prepends chips to each chunk       │
│   │                                         │
│   ▼                                         │
│ embedder.ts ──────── Embeds each chip-chunk via OpenAI               │
│   │                                         │
│   ▼                                         │
│ save-document.ts ──── Writes document record                        │
│ save-chunks.ts ────── Writes chip-chunks with embeddings            │
│                                             │
└─────────────────────────────────────────────
```

Query (chat-client)

```
┌─────────────────────────────────────────────
│                                             │
│ User Message                                │
│   │                                         │
│   ▼                                         │
│ retrieval.ts ──────── Embeds query, vector search, returns 5 chunks  │
│   │                                         │
│   ▼                                         │
│ get-history.ts ────── Fetches full conversation                     │
│   │                                         │
│   ▼                                         │
│ prompt.ts ─────────── Assembles system + chunks + history + query    │
│   │                                         │
│   ▼                                         │
│ llm.ts ────────────── Sends to Claude Sonnet, receives response      │
│   │                                         │
│   ▼                                         │
│ save-message.ts ───── Stores user message and assistant response     │
│   │                                         │
│   ▼                                         │
│ Return to UI                                │
│                                             │
└─────────────────────────────────────────────
```

Configuration Parameters

Parameter	Value	Notes
Chunks per query	5	Top 5 most similar
Conversation context	Full	Entire conversation included
Chunk size	300-500 words	Target range
Chip repetition	TBD	May repeat chips 2-3x to boost importance
Embedding model	text-embedding-3-small	1536 dimensions
LLM	Claude Sonnet	claude-sonnet-4-20250514
Vector index	ivfflat	cosine similarity

Chip-Chunk Format

Each chunk stored in the database looks like this:

```
[DOCUMENT CONTEXT]
Property Address: 1234 Main Street, Denver, CO 80202
Unit Number: 204
Tenant Name: John Smith
Landlord: Corrado Properties LLC
Lease Start: January 1, 2024
Lease End: December 31, 2024
Monthly Rent: $2,150
Security Deposit: $2,150

[CONTENT]
The Tenant agrees to pay rent on the first day of each calendar month.
Late payments will incur a fee of $50 if not received within 5 days of
the due date. Tenant shall not sublet the premises without written
consent from the Landlord...
```

The chips become part of the embedding, so semantic search naturally considers document identity alongside content.

Environment Variables

```
# .env.local

NEXT_PUBLIC_SUPABASE_URL=your_supabase_url
NEXT_PUBLIC_SUPABASE_ANON_KEY=your_supabase_anon_key
SUPABASE_SERVICE_ROLE_KEY=your_service_role_key

OPENAI_API_KEY=your_openai_key

ANTHROPIC_API_KEY=your_anthropic_key
```

Next Steps

1. Initialize Next.js project
2. Set up Supabase project and run migrations
3. Build file-client pipeline (start with extractor → cleaner → classifier)
4. Build basic upload UI to test pipeline
5. Add chips extraction and chunking
6. Build chat-client pipeline
7. Build chat UI
8. Test end-to-end with sample leases

Open Questions

- Chunk overlap: Should chunks repeat 50-100 words for continuity?
- Chip repetition count: How many times to repeat chips in each chunk?
- Context window limit: What's the max token count before truncating history?
- "View Source" UX: Modal with full text, or download original file?