🏛️ German Address Change System

Complete Architecture & Technical Documentation

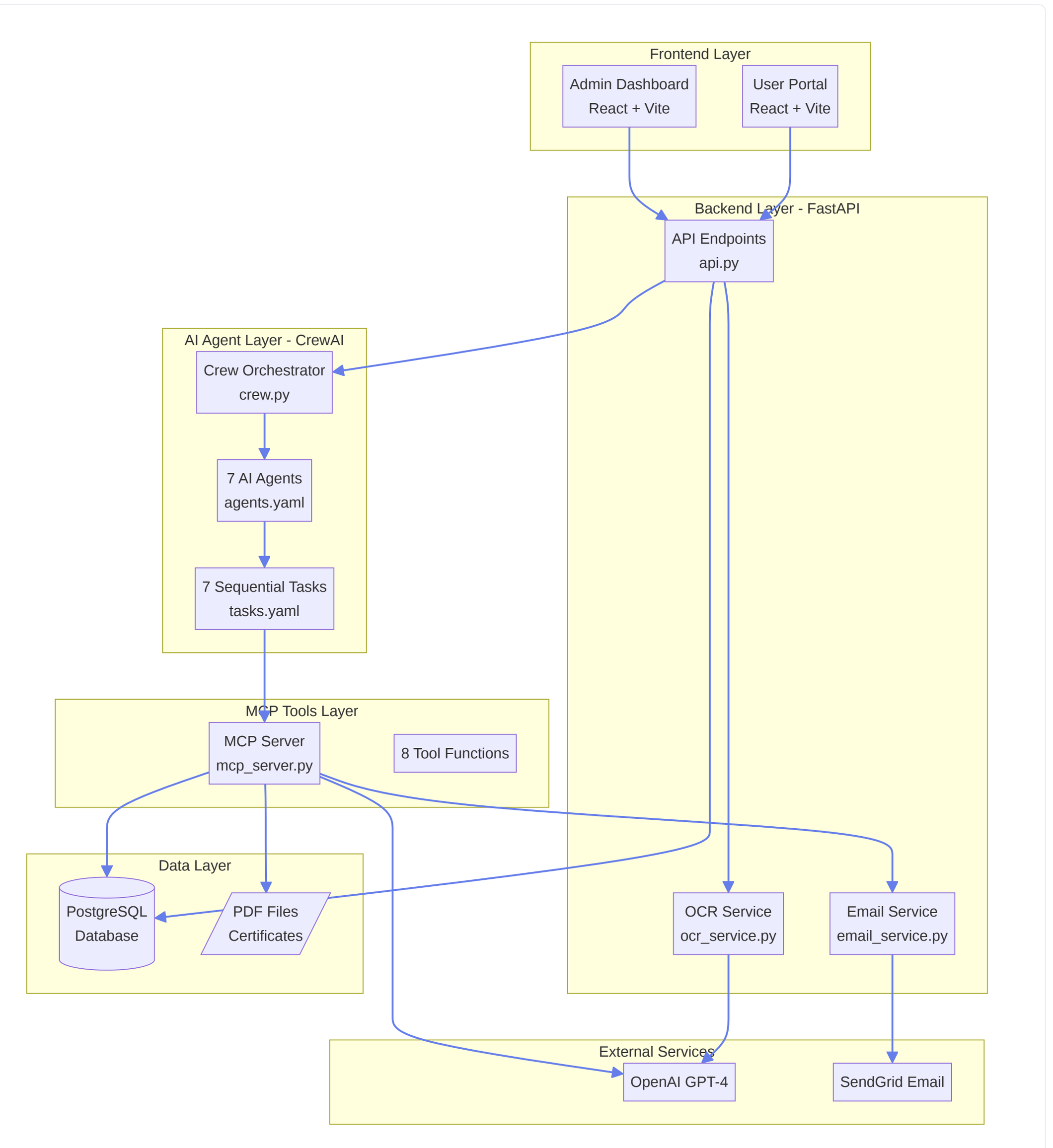
📋 System Overview

This system automates the German public administration's address change workflow using AI agents, OCR, and Human-in-the-Loop (HITL) verification.

Key Features

- ✅ **Automated OCR** - GPT-4 Vision extracts data from PDF documents
- ✅ **AI Agent Workflow** - 7 sequential tasks executed by specialized agents
- ✅ **HITL Quality Check** - Human review for low-confidence addresses
- ✅ **Real-time Status Updates** - Frontend polls for workflow progress
- ✅ **Automated Certificate Generation** - PDF creation and email delivery

🏗️ System Architecture

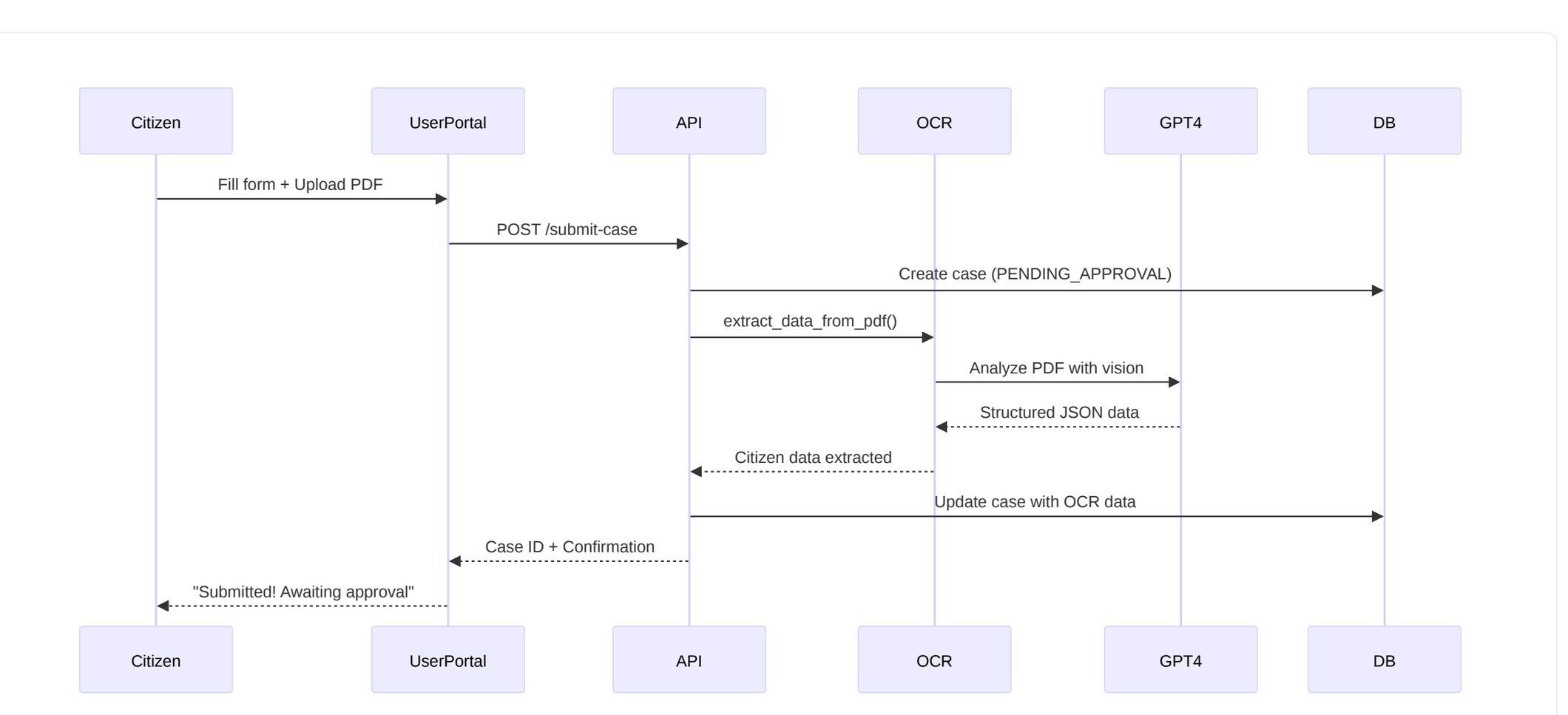


Technology Stack

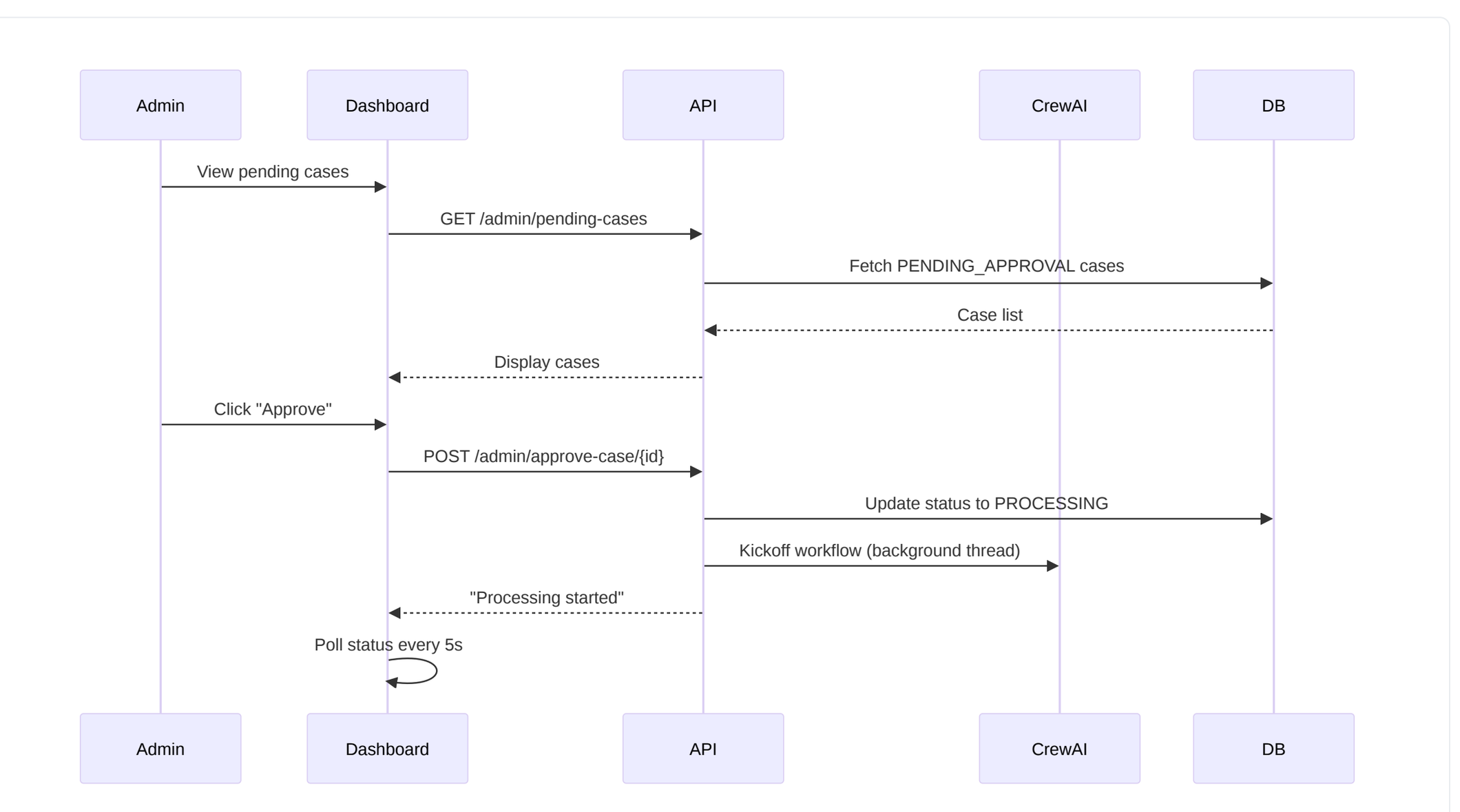
Layer	Technology	Purpose
Frontend	React 18 + Vite	UI framework
Styling	Vanilla CSS	Glassmorphic design
Backend	FastAPI + Uvicorn	REST API server
AI Orchestration	CrewAI	Multi-agent workflow
AI Model	OpenAI GPT-4	OCR + Address normalization
Tools	MCP Protocol	Agent-to-function interface
Database	PostgreSQL	Data persistence
PDF Generation	ReportLab	Certificate creation
Email	SendGrid	Email delivery
Deployment	Docker Compose	Containerization

📊 Complete End-to-End Workflow

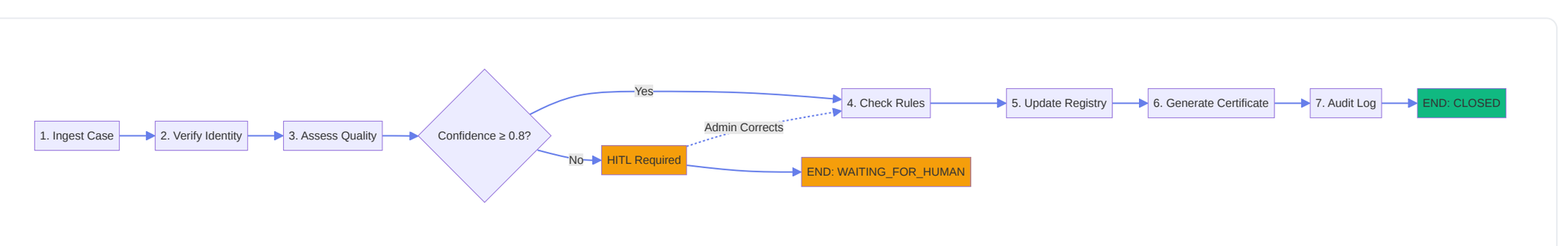
Phase 1: Case Submission (User Portal)



Phase 2: Admin Approval



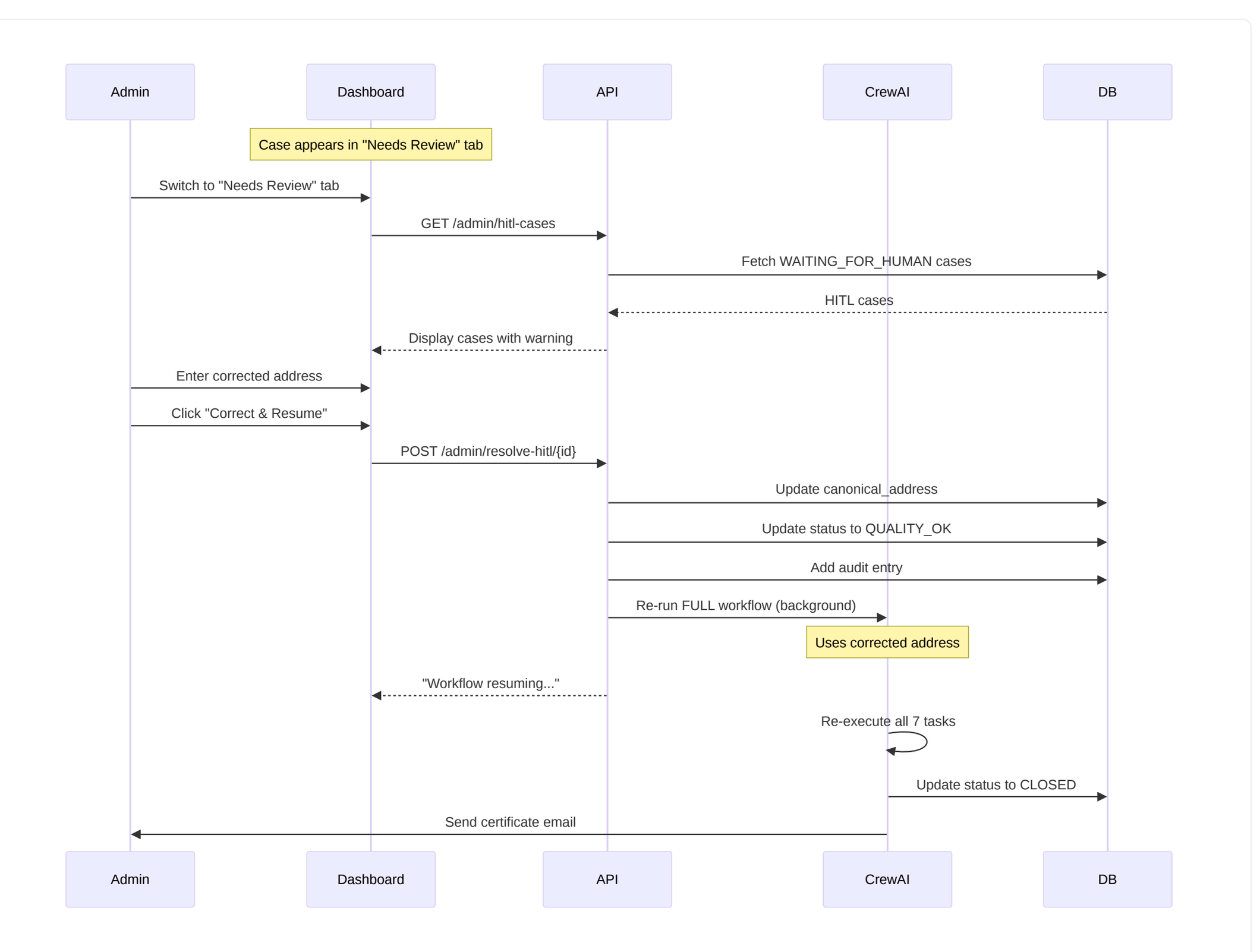
Phase 3: AI Agent Workflow (7 Tasks)



Task Breakdown:

1. **Ingest Case** - Fetch case data from database
2. **Verify Identity** - Validate citizen name and DOB
3. **Assess Quality** - Normalize address, check confidence
   - If confidence < 0.8 → Trigger HITL (pause workflow)
   - If confidence ≥ 0.8 → Continue normally
4. **Check Business Rules** - Validate move-in date, documents
5. **Update Registry** - Store new address in citizen registry
6. **Generate Certificate** - Create PDF, send email
7. **Audit Log** - Fetch complete audit trail

Phase 4: HITL Resolution



🔧 Technical Implementation

Project Structure

```
automation/
├── frontend/                       # React frontend
│   ├── src/
│   │   ├── pages/
│   │   │   ├── UserPortal.jsx        # Citizen form
│   │   │   └── AdminDashboard.jsx    # Admin interface
│   │   ├── App.jsx
│   │   ├── App.css                   # Glassmorphic styles
│   │   └── main.jsx
│   ├── package.json
│   └── vite.config.js
│
├── src/automation/                 # Backend Python
│   ├── api.py                      # FastAPI endpoints
│   ├── crew.py                     # CrewAI orchestration
│   ├── ocr_service.py              # PDF OCR
│   ├── email_service.py            # SendGrid integration
│   ├── db.py                       # PostgreSQL ops
│   ├── main.py
│   │
│   ├── config/
│   │   ├── agents.yaml             # Agent definitions
│   │   └── tasks.yaml              # Task definitions
│   │
│   └── tools/
│       └── mcp_server.py           # MCP tool implementations
│
├── docker-compose.yml
├── Dockerfile.backend
├── Dockerfile.frontend
└── render.yaml
```

Key Components

1. MCP Server (Tool Implementations)

Location: `src/automation/tools/mcp_server.py`

Contains 8 tool functions that AI agents can call:

- `ingest_case()` - Fetch case data
- `verify_identity()` - Validate citizen info
- `assess_quality()` - Normalize address, compute confidence
- `check_business_rules()` - Validate business logic
- `update_registry()` - Store address in registry
- `generate_certificate()` - Create PDF + send email
- `get_audit_log()` - Fetch audit trail

2. CrewAI Connection to MCP

Location: `src/automation/crew.py`

Tools are bound to agents like this:

```
from .tools.mcp_server import assess_quality, check_business_rules

@agent
def quality_confidence_officer(self) -> Agent:
    return Agent(
        config=self.agents_config['quality_confidence_officer'],
        tools=[assess_quality],  # ← Tool binding
        verbose=True
    )
```

3. Database Operations

Location: `src/automation/db.py`

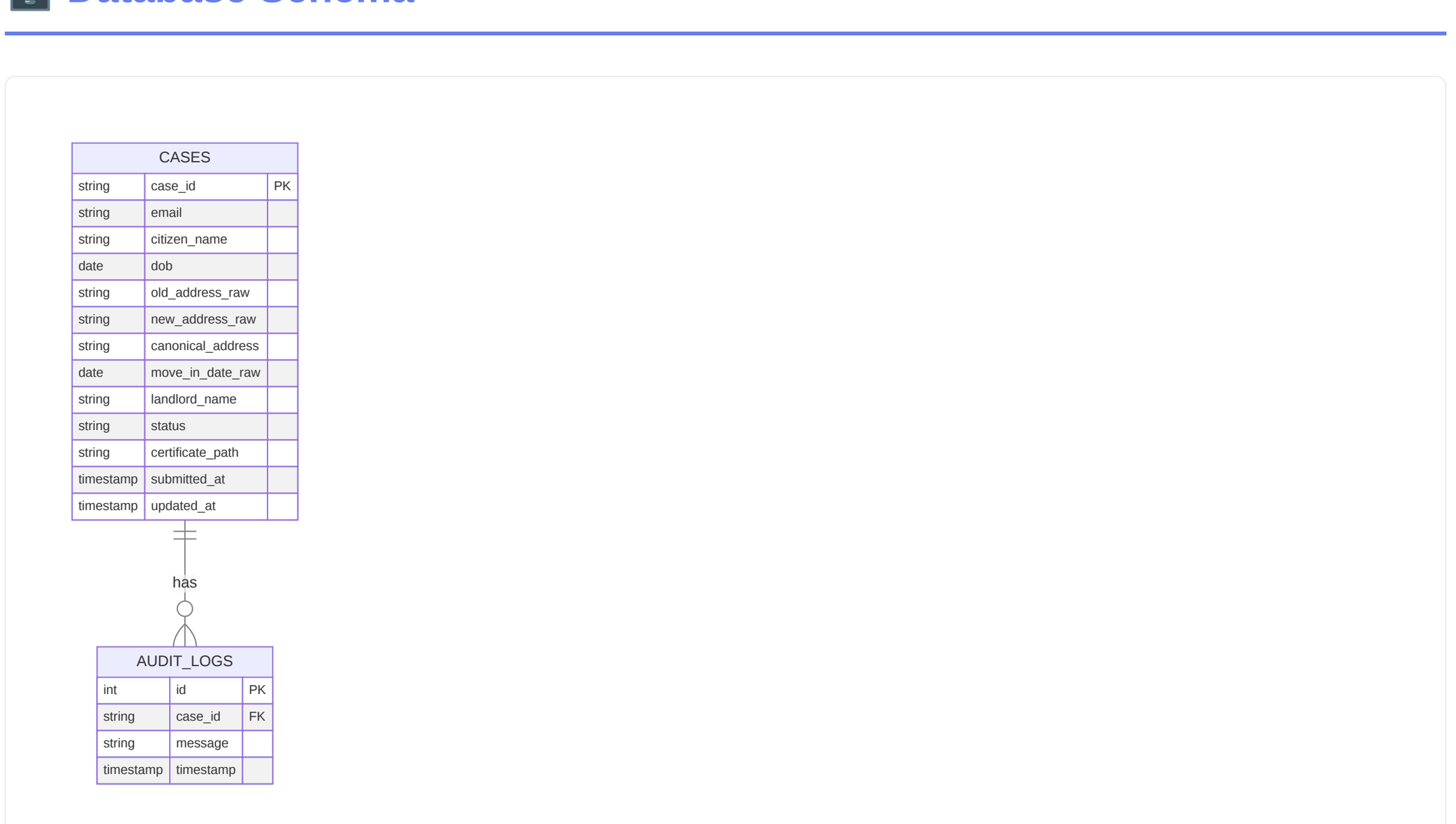
PostgreSQL connection and CRUD operations:

- `get_conn()` - Context manager for connections
- `update_case_status()` - Change case status
- `set_canonical_address()` - Store normalized address
- `add_audit_entry()` - Log workflow events
- `fetch_case_by_id()` - Retrieve case details

4. Frontend Components

Component	Location	Purpose
User Portal	`frontend/src/pages/UserPortal.jsx`	Citizen submission form
Admin Dashboard	`frontend/src/pages/AdminDashboard.jsx`	Case management + HITL resolution
Styles	`frontend/src/App.css`	Glassmorphic dark theme

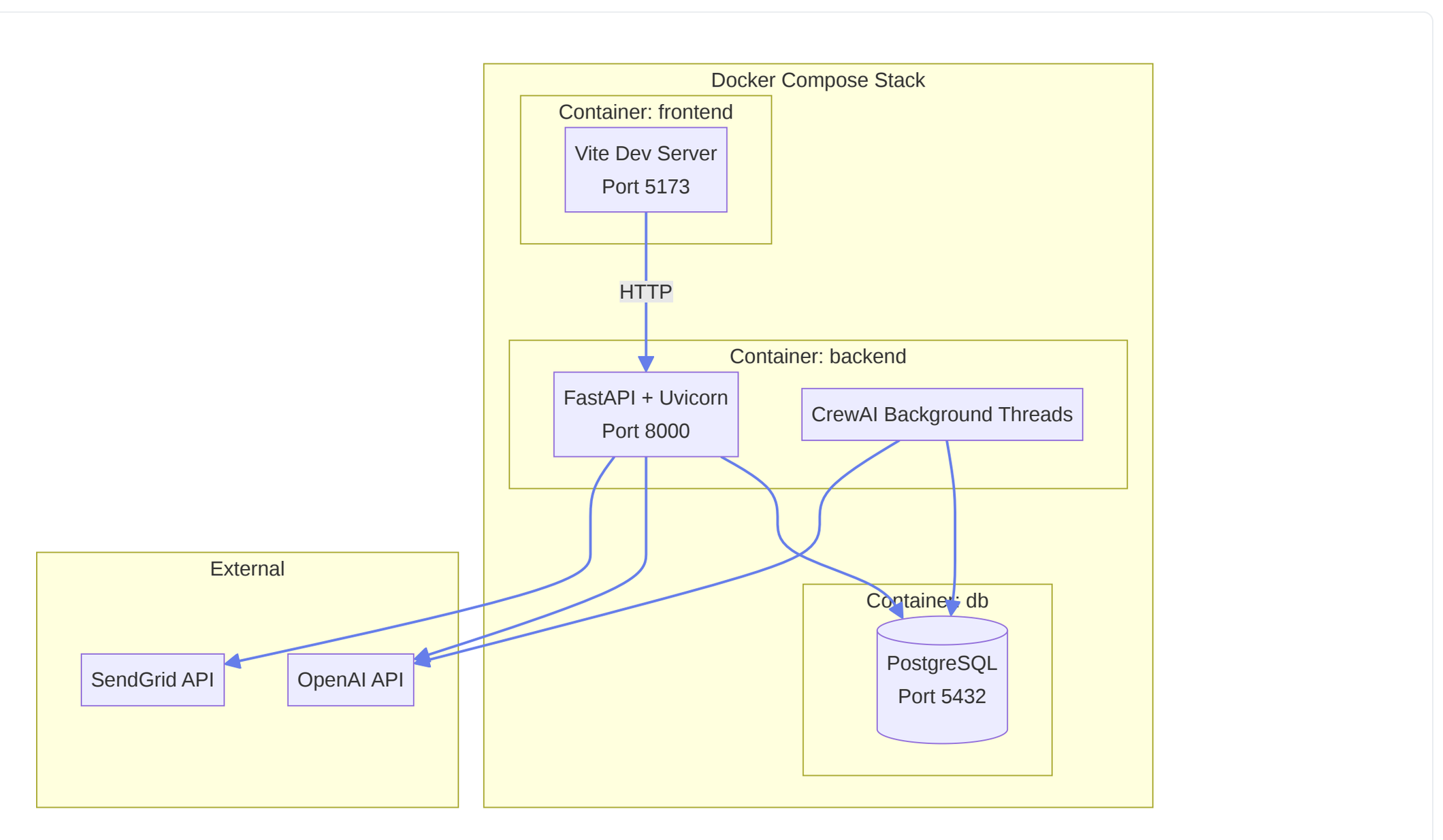
🗄️ Database Schema



Case Status Flow

1. PENDING_APPROVAL - Initial submission
2. PROCESSING - Admin approved, workflow running
3. WAITING_FOR_HUMAN - HITL triggered (low confidence)
4. QUALITY_OK - HITL resolved, re-processing
5. CLOSED - Workflow completed successfully
6. ERROR - Workflow failed

🚀 Deployment Architecture



Environment Variables

```
# Database
DATABASE_URL=postgresql://user:pass@db:5432/address_db

# AI Services
OPENAI_API_KEY=sk-...

# Email
SENDGRID_API_KEY=SG...

# Feature Flags
USE_DEMO_DATA=false

# Frontend
VITE_API_URL=http://localhost:8000
```

Key Design Decisions

1. HITL Workflow Pause Strategy

Challenge: CrewAI doesn't natively support mid-workflow pausing

Solution: Tools check case status before executing. If `WAITING_FOR_HUMAN`, return "skipped". Resume by re-running full workflow with corrected data.

2. Workflow Resumption

Initial approach: Create partial workflow starting from quality check

Problem: Agents couldn't find case_id in context

Final solution: Re-run FULL workflow from beginning with corrected address

3. Status Polling

Problem: "Processing" status stayed even after completion

Solution: Frontend polls case status every 5s, updates message when status changes

🎯 Testing Workflow

To Test HITL:

1. Submit case with abbreviated address: `"Musterstr 12A, 12345 KL, Deutschland"`
2. Approve case
3. Wait for "Needs Review" notification
4. Navigate to "Needs Review" tab
5. Enter corrected: `"Musterstraße 12A, 12345 Kaiserslautern, Deutschland"`
6. Click "Correct & Resume"
7. Verify workflow completes and certificate sent ✅

To Test Normal Flow:

1. Submit case with full address: `"Hauptstrasse 22, 67655 Kaiserslautern, Deutschland"`
2. Approve case
3. Verify immediate completion ✅
4. Check email for certificate