SentralQ

Evidence Lakehouse Deployment Documentation

Version:	1.0.0
Date:	January 8, 2026
Status:	Production Active
Environment:	Vercel + Render.com

Live Deployment URLs

Frontend Application

URL: https://sentraiq.vercel.app/

- Platform: Vercel (Free Tier)
- Stack: React + TypeScript + Vite
- Status: ✓ Live & Connected

Backend API

URL: https://sentraiq.onrender.com

- Platform: Render.com (Free Tier)
- Stack: FastAPI + Python 3.11
- Status: ✓ Healthy & Responding

API Documentation

URL: https://sentraiq.onrender.com/docs

Interactive Swagger UI with live API testing and complete endpoint documentation

GitHub Repository

URL: https://github.com/Deep-Learner-msp/SentraIQ

Branch: main (auto-deploys to production)

System Architecture

Three-Tier Architecture

Component	Technology	Platform	Purpose
Frontend	React + Vite	Vercel	User Interface & Dashboard
Backend	FastAPI + Python	Render.com	REST API & Business Logic
Database	PostgreSQL	Render.com	Data Persistence
AI/ML	OpenAI GPT-5	Cloud API	Natural Language Processing

Data Flow

1. User interacts with React frontend hosted on Vercel
2. Frontend sends HTTPS requests to FastAPI backend on Render
3. Backend processes requests and queries PostgreSQL database
4. For natural language queries, backend calls OpenAI GPT-5 API
5. Results are returned through the stack to the user

Technology Stack

Backend Technologies

Component	Version	Purpose
FastAPI	0.109.0	Web framework for building REST APIs
Python	3.11	Runtime environment
SQLAlchemy	2.0.25	ORM and async database operations
PostgreSQL	Latest	Relational database
OpenAI SDK	1.10.0	GPT-5 integration for NL parsing
PyMuPDF	1.24.14	PDF document processing
Uvicorn	0.27.0	ASGI server
Pydantic	2.5.3	Data validation and settings

Frontend Technologies

Component	Version	Purpose
React	18	UI component library
TypeScript	5	Type-safe JavaScript
Vite	5	Build tool and dev server
Tailwind CSS	3	Utility-first CSS framework
Axios	Latest	HTTP client for API calls
React Router	6	Client-side routing

API Endpoints

```
Base URL: https://sentraiq.onrender.com/api/v1
```

1. Health Check

```
GET /health
```

Returns system health status and version information.

2. Dashboard Statistics

```
GET /api/v1/dashboard/stats
```

Returns aggregate statistics for the dashboard including total logs, documents, evidence objects, and assurance packs.

3. Ingest Log File

```
POST /api/v1/ingest/log
```

Content-Type: multipart/form-data

Parameters:

- file: Log file (required)
- source: SWIFT | ACH | SEPA | CARD (required)
- description: Log description (required)

4. Ingest Document

```
POST /api/v1/ingest/document
```

Content-Type: multipart/form-data

Parameters:

- file: PDF document (required)
- doc_type: POLICY | PROCEDURE | STANDARD | AUDIT_REPORT (required)
- description: Document description (required)

5. Evidence Telescope (Natural Language Query)

```
POST /api/v1/evidence/telescope
```

Content-Type: application/json

Accepts natural language queries and returns relevant evidence objects. Uses OpenAI GPT-5 for query parsing and intent extraction.

6. Generate Assurance Pack

```
POST /api/v1/assurance/generate
```

Content-Type: application/json

Generates tamper-evident assurance pack with evidence for specified compliance controls and date range.

Project Structure

SentraIQ/

```
■■■ backend/ Backend FastAPI application
■ ■■■ main.py Main application entry point
■ ■■■ config.py Configuration & settings management
■ ■■■ database.py SQLAlchemy models & DB initialization
■ ■■■ routers/ API route handlers
■ ■ ■■■ ingestion.py Layer 1: Data ingestion endpoints
■ ■ ■■■ evidence.py Layer 2: Evidence extraction
■ ■ ■■■ assurance.py Layer 3: Assurance pack generation
■ ■ ■■■ dashboard.py Dashboard statistics API
■ ■ ■■■ demo.py Demo data endpoints
■ ■■■ layers/ Business logic implementation
■ ■■■ ingest_layer.py Log/document processing logic
■ ■■■ evidence_layer.py Evidence object creation
■ ■■■ assurance_layer.py Assurance pack generation
■ ■■■ telescope.py OpenAI NL query parsing
■
■■■ frontend/
■ ■■■ sentraiq-dashboard/ React frontend application
■ ■■■ src/
■ ■ ■■■ components/ React UI components
■ ■ ■■■ services/ API client (api.ts)
■ ■ ■■■ types/ TypeScript type definitions
■ ■■■ vite.config.js Vite build configuration
■ ■■■ vercel.json Vercel deployment config
■
■■■ data/ Sample data files
■ ■■■ logs/ Sample log files for demo
■ ■■■ documents/ Sample policy documents
■
■■■ storage/ Runtime storage directories
■ ■■■ raw_logs/ Ingested log files
■ ■■■ raw_documents/ Ingested document files
■ ■■■ assurance_packs/ Generated assurance packs
■
■■■ requirements.txt Python dependencies
■■■ render.yaml Render.com deployment config
■■■ .python-version Python version (3.11.0)
■■■ DEPLOYMENT.md This documentation
```

Sample Data & Demo

Sample Log Files

Filename	Description	Use Case
swift_access_q3_2025.log	SWIFT terminal access logs	Access control evidence
ach_transactions_q3_2025.log	ACH payment transaction logs	Transaction monitoring
firewall_logs_q3_2025.log	Network firewall logs	Network security evidence

Sample Policy Documents

Filename	Description	Control Mapping
mfa_policy.pdf	Multi-Factor Authentication Policy	AC-001, IA-005
encryption_policy.pdf	Data Encryption Standards	CR-001, SC-013
access_control_procedure.pdf	Access Control Procedures	AC-002, AC-003
audit_report_q2_2025.pdf	Internal Audit Report	AU-001, AU-002

Demo Endpoints

`GET /api/v1/demo/logs` - List available demo log files

`GET /api/v1/demo/documents` - List available demo documents

Local Development Setup

Prerequisites

- Python 3.11
- Node.js 18+
- PostgreSQL (or SQLite for local development)
- OpenAI API Key (optional, for Telescope feature)

Backend Setup

```
git clone https://github.com/Deep-Learner-msp/SentraIQ.git
cd SentraIQ

# Create virtual environment
python3.11 -m venv venv
source venv/bin/activate # Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Create .env file
echo "DATABASE_URL=sqlite+aiosqlite:///./sentraiq.db" > .env
echo "OPENAI_API_KEY=sk-proj-your-key" >> .env
echo "ENVIRONMENT=development" >> .env

# Run backend server
uvicorn backend.main:app --reload --port 8000
```

Backend will be available at: http://localhost:8000

Frontend Setup

```
cd frontend/sentraiq-dashboard

# Install dependencies
npm install

# Run development server
npm run dev
```

Frontend will be available at: http://localhost:3000

Vite proxy automatically forwards API requests to http://localhost:8000

Deployment Process

Continuous Deployment

Both frontend and backend are configured for automatic deployment from the GitHub main branch.

Backend Deployment (Render)

1. Push changes to main branch
2. Render detects changes via GitHub webhook
3. Build command: pip install -r requirements.txt
4. Start command: uvicorn backend.main:app --host 0.0.0.0 --port $PORT
5. Health check on /health endpoint
6. Deploy to: https://sentraiq.onrender.com

Build time: 2-3 minutes

Frontend Deployment (Vercel)

1. Push changes to main branch
2. Vercel detects changes via GitHub webhook
3. Build command: npm install && npm run build
4. Deploy static files from /dist directory
5. Deploy to: https://sentraiq.vercel.app

Build time: 1-2 minutes

Environment Variables

Variable	Platform	Value	Required
DATABASE_URL	Render	Auto-configured by Render	Yes
OPENAI_API_KEY	Render	User provided	Optional
ENVIRONMENT	Render	production	Yes
VITE_API_URL	Vercel	https://sentraiq.onrender.com	Yes

Troubleshooting Guide

Issue	Symptom	Solution
Frontend CORS error	CORS errors in browser console	Verify VITE_API_URL in Vercel. Check CORS config in backend/main.py
Backend 500 errors	Internal server errors	Check Render logs. Verify DATABASE_URL. Validate OPENAI_API_KEY
Database connection fail	Connection refused errors	Verify PostgreSQL service running. Check DATABASE_URL format
Slow first request	30+ second response time	Render free tier spins down after 15min inactivity. First request wakes it up
Build failures	Deployment fails	Check requirements.txt syntax. Verify Python version (.python-version file)
Missing data	Empty dashboard	Ingest sample data using /demo endpoints or frontend UI

Monitoring & Logs

- **Backend logs:** https://dashboard.render.com/web/sentraiq-backend/logs
- **Frontend logs:** https://vercel.com/dashboard/deployments
- **Database metrics:** https://dashboard.render.com/d/sentraiq-db

Testing & Validation

API Testing with cURL

```
# Health check
curl https://sentraiq.onrender.com/health

# Get dashboard statistics
curl https://sentraiq.onrender.com/api/v1/dashboard/stats

# Upload log file
curl -X POST https://sentraiq.onrender.com/api/v1/ingest/log \
-F "file=@swift_logs.log" \
-F "source=SWIFT" \
-F "description=Test upload"

# Natural language query
curl -X POST https://sentraiq.onrender.com/api/v1/evidence/telescope \
-H "Content-Type: application/json" \
-d '{"natural_language_query":"Show me all MFA evidence"}'
```

Frontend Testing

1. Visit https://sentraiq.vercel.app/
2. Verify dashboard loads with three-layer visualization
3. Check that statistics are fetched from backend
4. Test Layer 1 (Ingestion) - Upload sample files
5. Test Layer 2 (Evidence) - Use Telescope for NL queries
6. Test Layer 3 (Assurance) - Generate compliance packs

Compliance Controls Mapping

Control ID	Name	Description	Keywords
AC-001	Multi-Factor Authentication	Enforce MFA for SWIFT terminal access	mfa, two-factor, 2fa, authentication
AC-002	Access Control	Restrict access to authorized personnel	access, authorization, permission, denied
CR-001	Data Encryption	Encrypt payment data in transit and at rest	encryption, tls, ssl, encrypted, cipher
AU-001	Audit Logging	Maintain comprehensive audit logs	audit, log, record, event, activity
IA-005	Authenticator Management	Manage authentication tokens/devices	token, authenticator, device, FIPS
SC-013	Cryptographic Protection	Use approved cryptographic mechanisms	AES-256, TLS 1.3, cryptographic

Supported Frameworks

- PCI-DSS (Payment Card Industry Data Security Standard)
- SWIFT CSP (Customer Security Programme)
- NIST 800-53 (Security and Privacy Controls)
- ISO 27001 (Information Security Management)
- SOC 2 (Service Organization Control)

Resources & Links

Resource	URL	Description
Frontend	https://sentraiq.vercel.app/	Live application dashboard
Backend API	https://sentraiq.onrender.com	REST API base URL
API Docs	https://sentraiq.onrender.com/docs	Interactive Swagger documentation
Health Check	https://sentraiq.onrender.com/health	System health status
GitHub Repo	https://github.com/Deep-Learner-msp/SentraIQ	Source code repository
Render Dashboard	https://dashboard.render.com	Backend & DB management
Vercel Dashboard	https://vercel.com/dashboard	Frontend deployment management

Support

For issues, feature requests, or questions:

- Create an issue: https://github.com/Deep-Learner-msp/SentraIQ/issues
- Review documentation: DEPLOYMENT.md in repository
- Check API docs: https://sentraiq.onrender.com/docs

SentraIQ Evidence Lakehouse

Hybrid Evidence Lakehouse for Payment Systems

Version 1.0.0 | January 8, 2026

Status: ✓ Production Deployment Active