Sentinel: Fraud Detection AI Workbench

Comprehensive System Documentation

Architecture, Integrated Design, and User Manual

Contents

1. Introduction of Applications 5
2. Integrated Design 5
2.1 System Architecture & Data Flow 5
2.2 UI Design Strategy 6
3. Security Design 6
3.1 Login Implementations & RBAC 6
3.2 Data Security (SQL Anti-Injection) 6
3.3 Demo 7
3.3.1.1 Login Way 1 7
3.3.1.2 Login Way 2 7
3.3.2 Register 7
3.3.3 Admin Console – User Management 8
3.3.4 Admin Console - Role Management 8
4. UI1 - Home 8
4.1 UI Design 8
4.2 User Manual 8
4.3 Logic Workflow 9
4.4 Demo 9
5. UI2 - Data Hub 9
5.1 UI Design 9
5.2 User Manual 9
5.3 Logic Workflow 10
5.4 Demo 10
6. UI3 - SQL RAG 15
6.1 UI Design 15
6.2 User Manual 16
6.3 Logic Workflow 16
6.4 Demo 16
7. UI4 - Graph RAG 18
7.1 UI Design 18
7.2 User Manual 19
7.3 Logic Workflow 19

7.4 Demo 19
8. UI5 - Multimodal RAG 22
8.1 UI Design 22
8.2 User Manual 22
8.3 Logic Workflow 22
8.4 Demo 23
9. UI6 - Trends and Insights 28
9.1 UI Design 28
9.2 User Manual 28
9.3 Logic Workflow 28
9.4 Demo 28
10. UI7 - ML Workflow 30
10.1 Functionality Design 30
10.2 UI Design 30
10.3 User Manual 30
10.4 Logic Workflow 31
10.5 Demo 31
11. UI8 - LLM Fine Tuning 35
11.1 Functionality Design 35
11.2 UI Design 35
11.2 User Manual 35
11.3 Logic Workflow 36
11.4 Demo 36
12. UI9 - API Interaction Hub 41
12.1 UI Design 41
12.2 User Manual 41
12.3 Logic Workflow 41
12.4 Demo 41
13. UI10 - Admin Console 44
13.1 UI Design 44
13.2 User Manual 44
13.3 Logic Workflow 44
13.4 Demo 44

14. Show Demo 48
14.1 Description 48
14.2 Steps 48
14.3 Impacts to the System 48
15. Load Demo Data 49
15.1 Description 49
15.2 Steps and Location 49
15.3 Impacts to the System 49
15.4 Demo 50
16. Dependency References 51
16.1 FastAPI 51
16.2 Streamlit 51
16.3 SQLAlchemy 51
16.4 Neo4j (urllib3/neo4j-driver) 51
16.5 FAISS 51
16.6 SQLGlot 52
16.7 PyVis / NetworkX 52
16.8 BCrypt 52
16.9 Ollama 52
17. Glossary 52

1. Introduction of Applications

Sentinel Fraud AI Workbench Pro is a comprehensive, enterprise-grade platform engineered to detect, investigate, and mitigate financial fraud. It unifies Generative AI and traditional Machine Learning within a single, cohesive interface. The platform serves as a central hub for investigators, data scientists, and administrators to uncover hidden fraud rings, analyze massive datasets, and fine-tune models safely.

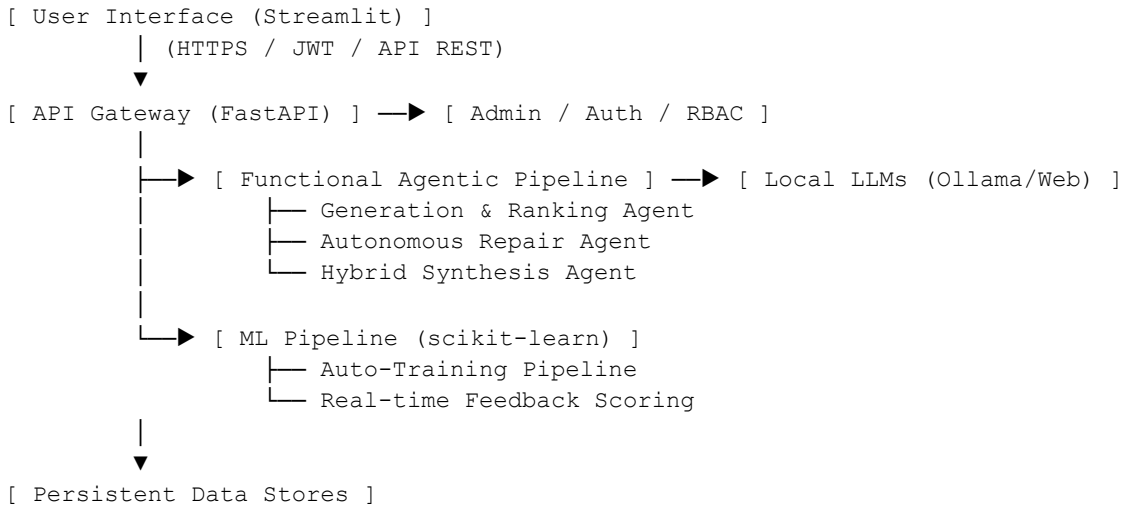
At its core, Sentinel leverages a decoupled client-server architecture. The backend operates robust, specialized Agentic Pipelines (combining prompt generation, SQL validation, autonomous error-recovery, and data synthesis) controlled by a high-concurrency FastAPI orchestrator. It provides three primary modalities of Retrieval-Augmented Generation (RAG): SQL RAG for structured transactional data, Graph RAG for interconnected entity analysis (tracking money laundering across accounts via Neo4j), and Multimodal RAG for unstructured evidence like documents, images, and audio utilizing FAISS vector indexing.

By integrating these capabilities with real-time Machine Learning pipelines (Random Forest, XGBoost) and persistent graph databases, Sentinel allows users to instantly transition from asking free-form natural language questions to deploying highly predictive fraud models.

2. Integrated Design

2.1 System Architecture & Data Flow

The architecture adheres to a strict decoupled pattern: a reactive Streamlit frontend communicates securely via JWT-authenticated REST APIs to a high-concurrency FastAPI backend. The backend acts as the central orchestrator, dynamically routing analytical requests through an ordered, functional Agentic Pipeline (SQL Generation, Candidate Ranking, Self-Recovery, and Reconciler agents), the supervised ML pipeline, or directly to the persistent data stores.



```
- SQLite (Raw structured data mappings)
- Neo4j (Entity graphs & taxonomic relations)
- FAISS (Vector embeddings for KB scaling)
```

2.2 UI Design Strategy

The UI architecture is entirely modularized under the `src/views/` directory. Each major feature (e.g., Data Hub, ML Workflow) is encapsulated in its own package, rendering state-aware tabs dynamically based on user context. The global application employs a 'Crimson Dark Mode' theme, utilizing conditional CSS injections to highlight critical alerts and interactive visual elements. Session State is rigorously managed to ensure that contextual data (such as a selected AI model, filter parameters, or a specific uploaded dataset) persists seamlessly as the user navigates fluidly between analytical tools.

3. Security Design

Security operates on a zero-trust model heavily focused on Role-Based Access Control (RBAC) and strict API perimeter defense.

3.1 Login Implementations & RBAC

Authentication uses standard OAuth2 architecture with JSON Web Tokens (JWT). Upon providing valid credentials, the backend (FastAPI) securely verifies password hashes utilizing BCrypt. It then issues a short-lived access token containing role scopes. The frontend caches this token locally and strictly attaches it to the header of every outbound API request.

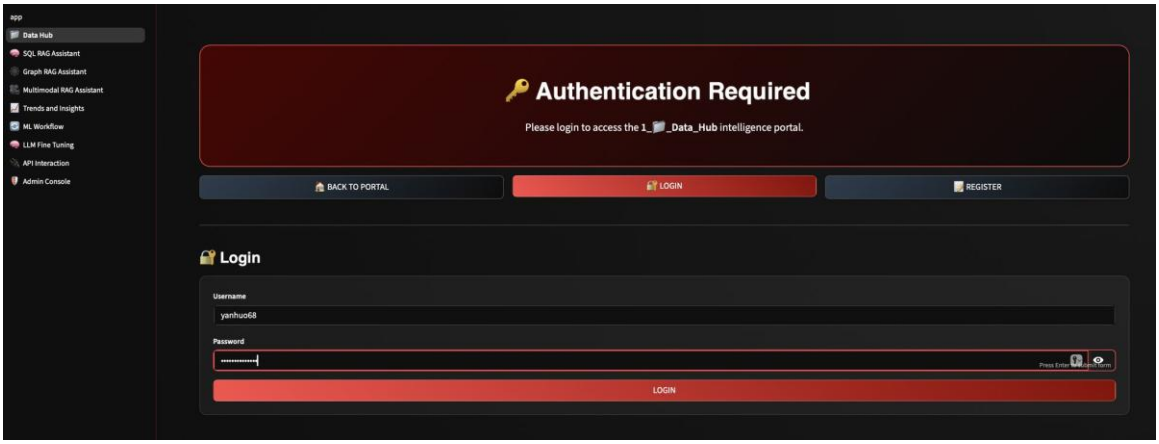
RBAC is enforced dual-layer. On the frontend, the UI constructs the navigation sidebar by querying the `/admin/permissions` API endpoint, hiding pages the user cannot access. On the backend, routes enforce access internally using parameterized dependency injection (`RequiresRole`), terminating unauthorized requests before any business logic executes.

3.2 Data Security (SQL Anti-Injection)

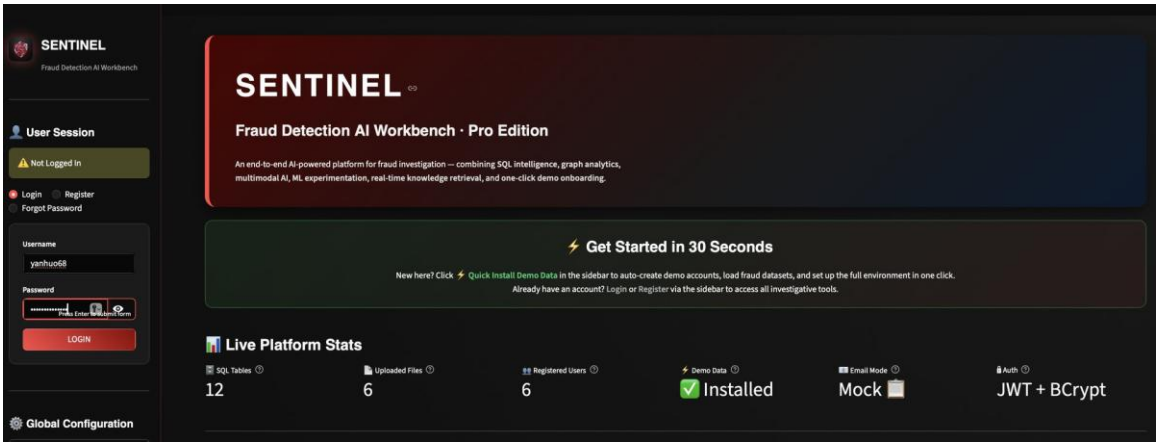
The SQL RAG module employs an explicit Abstract Syntax Tree (AST) validator utilizing the `sqlglot` dependency. It inherently rejects LLM-hallucinated queries attempting destructive DROP, DELETE, UPDATE, INSERT, or ALTER commands. It further restricts queries to explicitly approved reporting tables, fully neutralizing prompt injection vulnerabilities.

3.3 Demo

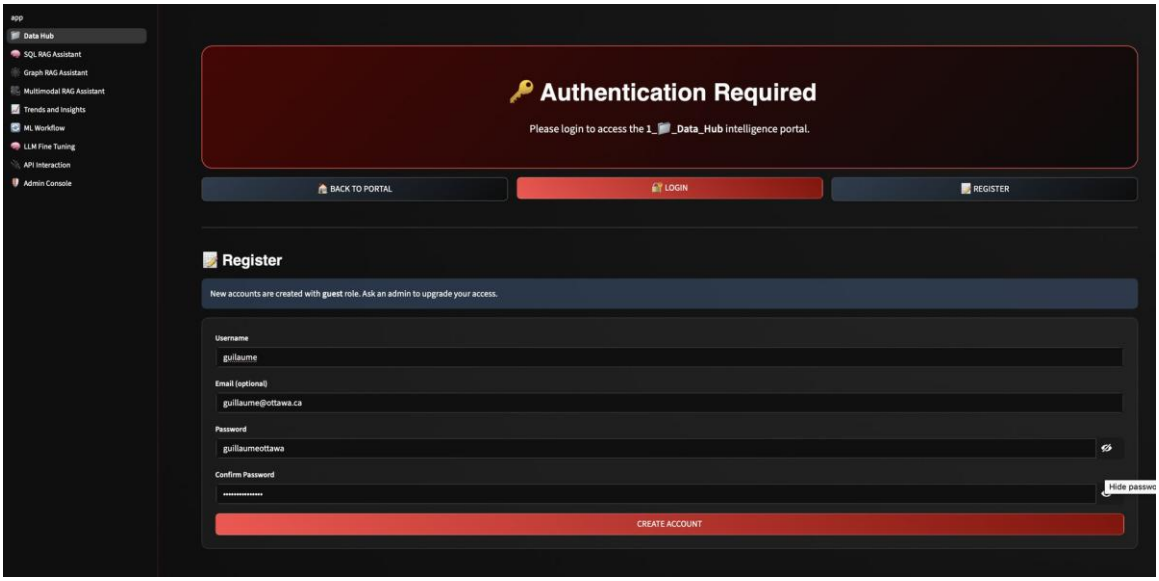
3.3.1.1 Login Way 1



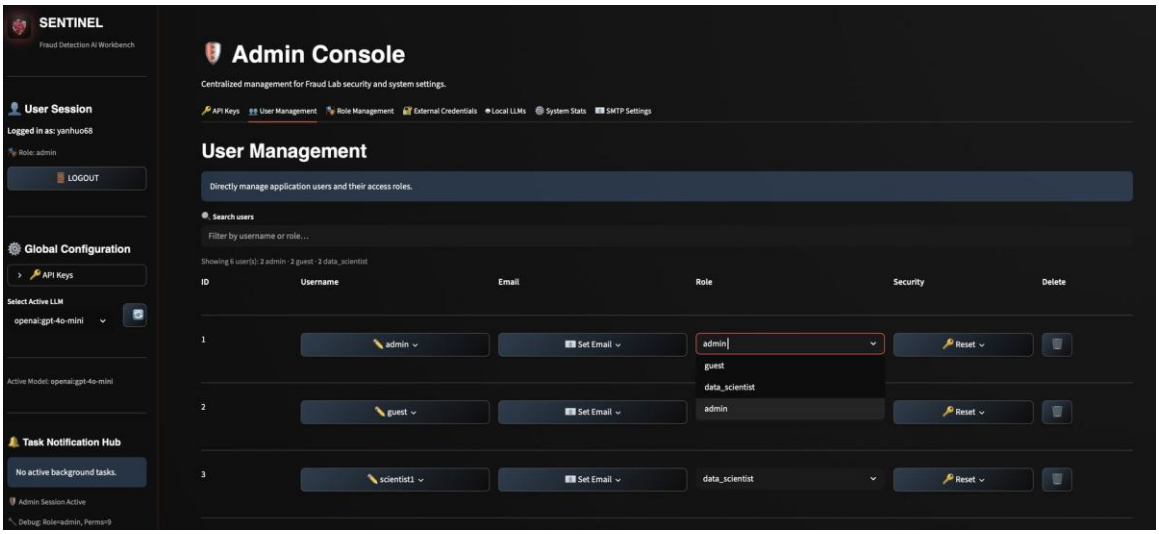
3.3.1.2 Login Way 2



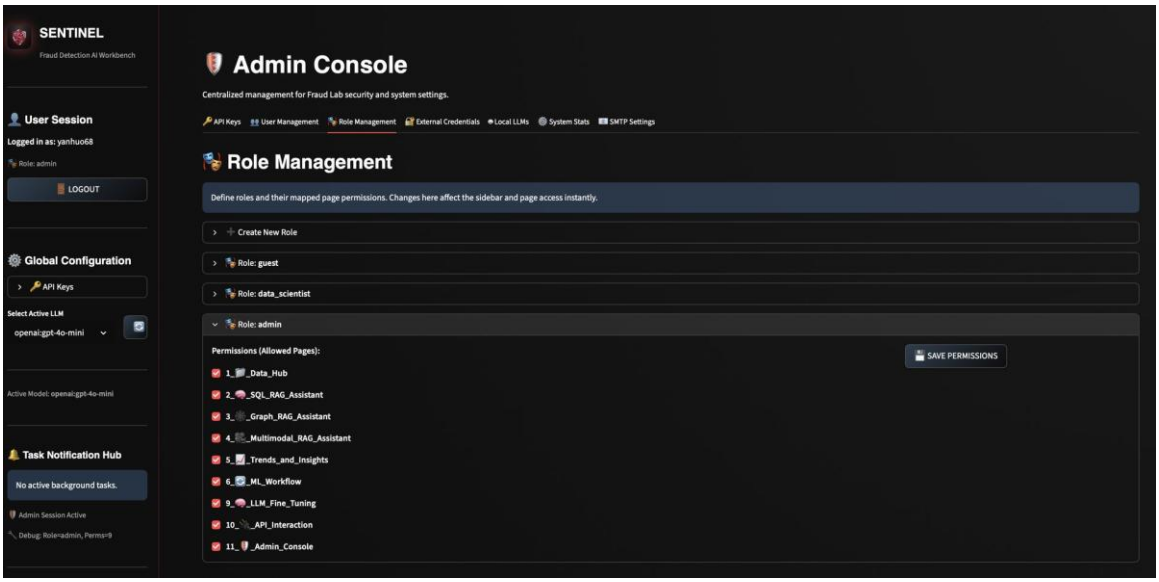
3.3.2 Register



3.3.3 Admin Console – User Management



3.3.4 Admin Console - Role Management



4. UI1 - Home

4.1 UI Design

The Home dashboard utilizes a responsive CSS grid layout mapping to three core investigative pillars: Intelligence, Visualization, and AI Core. It displays dynamic real-time system metrics (like active model count and Neo4j node connections) alongside action buttons governed by the authenticated user's RBAC role.

4.2 User Manual

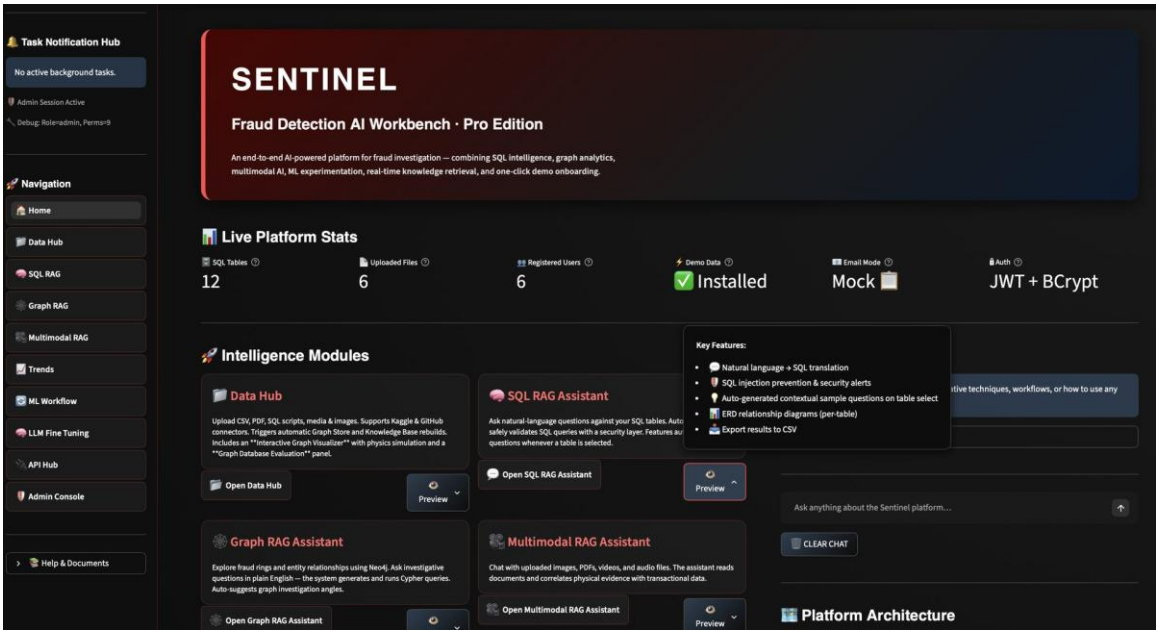
Step 1: Review the system health on the central dashboard immediately upon login.
Step 2: If the system returns zero metrics, click the 'Quick Install Demo Data' fast-action

button to securely populate the SQLite databases and Neo4j graph.
Step 3: Click any primary module tile to bypass standard sidebar navigation and jump directly into an investigation.

4.3 Logic Workflow

```
▶ [ Authenticate User JWT ]
   └──▶ [ Fetch System Metrics API ]
          └──▶ [ Authorize Role Dashboard UI ]
                 └──▶ [ Render Action Cards ]
```

4.4 Demo



When user initial comes to the system, above is home page. If user does not login, the page will redirect to Login page. If user has already logged in, user can click Preview to see key features or click Open xxx to go to desired page.

5. UI2 - Data Hub

5.1 UI Design

The Data Hub presents a tab-driven interface (Local Uploads, External Sources, Database Management) utilizing highly responsive file-uploader drop-zones and paginated dataframes. Crucially, it embeds a real-time progress bar array that visually indicates concurrent NLP embedding and graph ingest operations.

5.2 User Manual

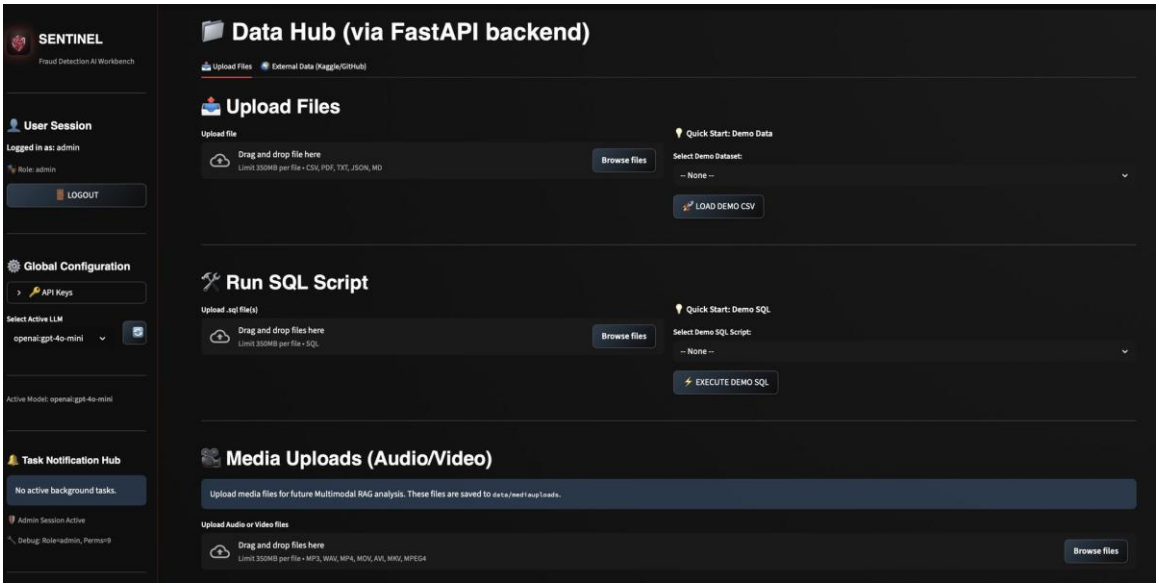
Step 1: Open the 'Local Uploads' tab and drag a structural CSV file into the drop-zone.
Step 2: Once uploaded, the backend auto-determines schema types. Click 'Rebuild

Knowledge Base' to slice text content and embed it into FAISS.
Step 3: Map the CSV columns to Node/Edge identifiers to trigger a Neo4j ingestion pipeline.

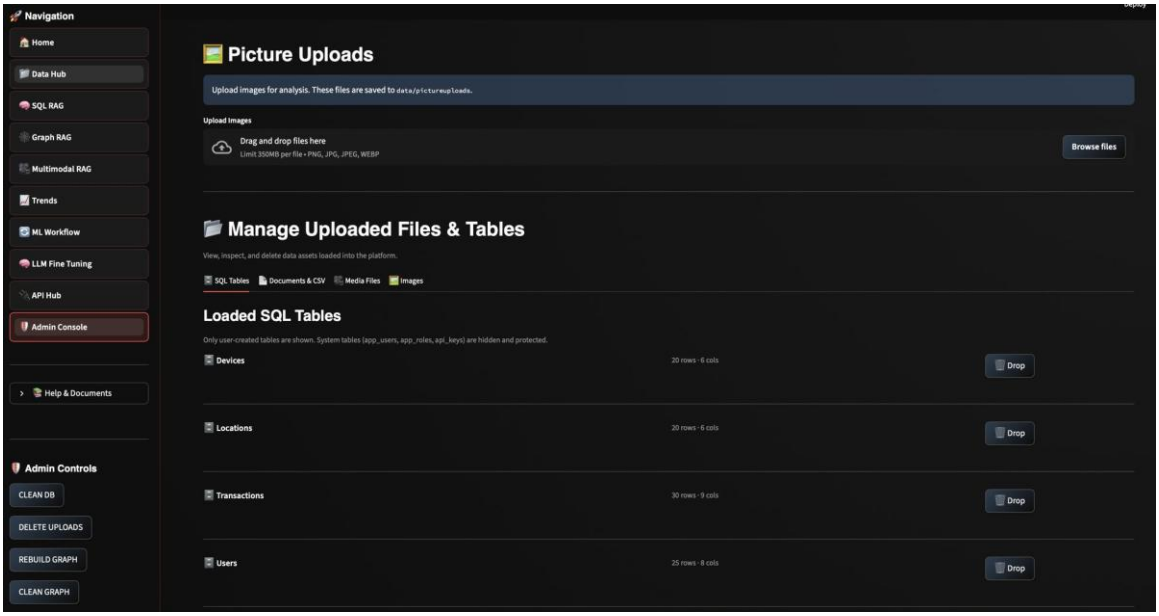
5.3 Logic Workflow

```
▶ [ Upload User CSV ]
  └──▶ [ Parse via Pandas ]
         ├──▶ [ Auto-Gen SQLite Table Schema ]
         ├──▶ [ Vectorize Text via NLP to FAISS ]
         └──▶ [ Map Structural Nodes into Neo4j Graph ]
```

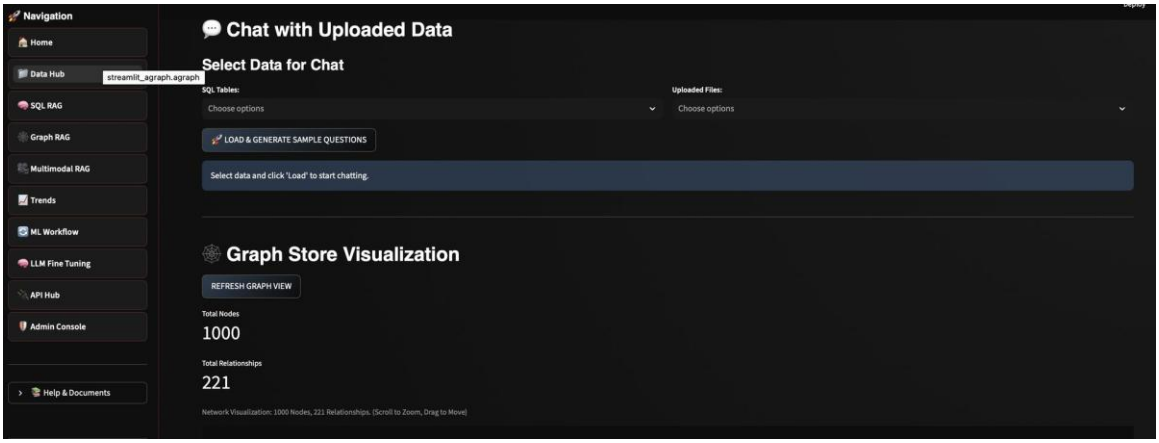
5.4 Demo



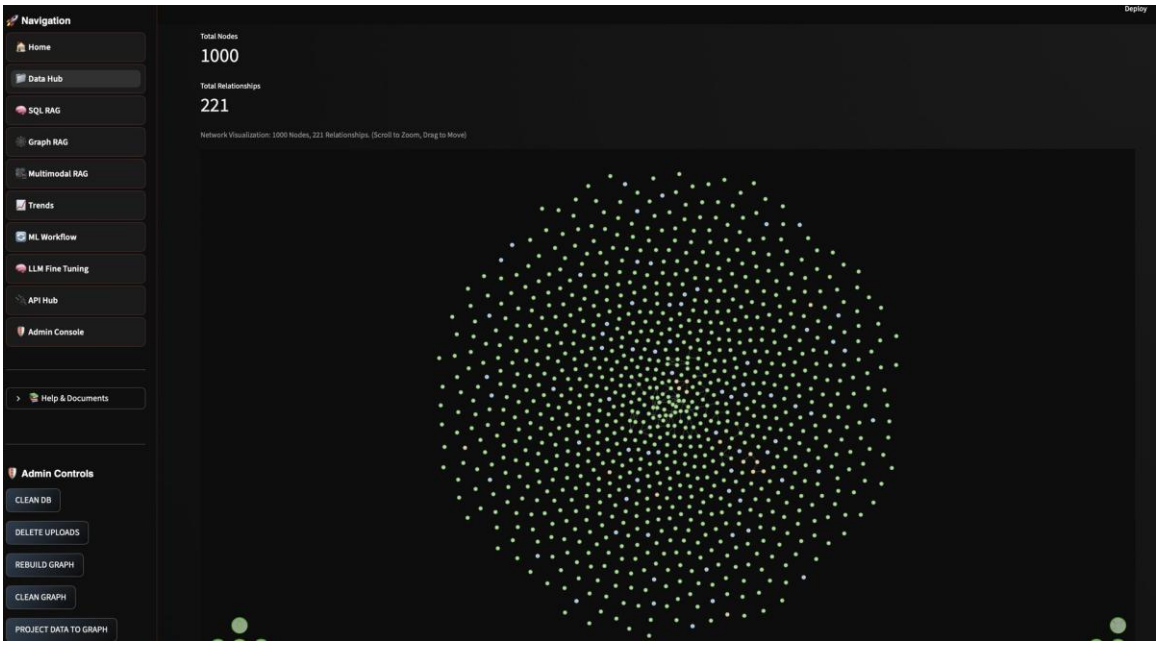
In the Data Hub, users can upload files (e.g. CSV/JSON/XML file, PDF file, media file (e.g. *.mp3, *.mp4, and so on), execute one or more SQL file(s). After uploading, for structure data can go to database, for unstructured data can go to vector store. After clicking Rebuild Graph button, all data no matter in database or vector store can be saved in graph store.



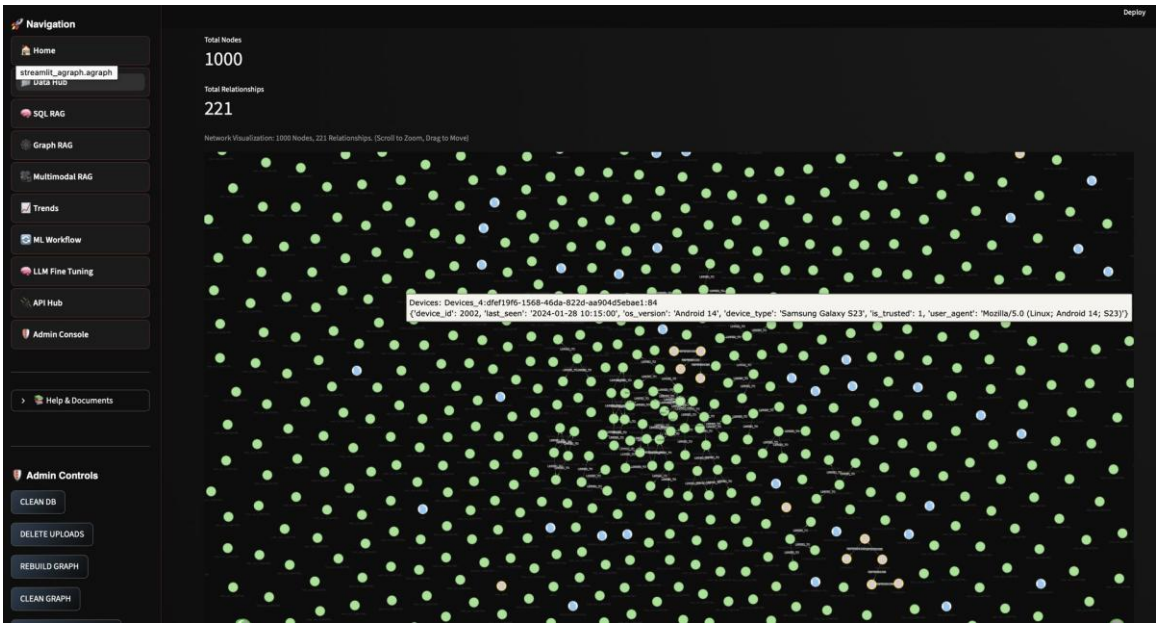
In the Data Hub, users can see the loaded tables and delete any one table.



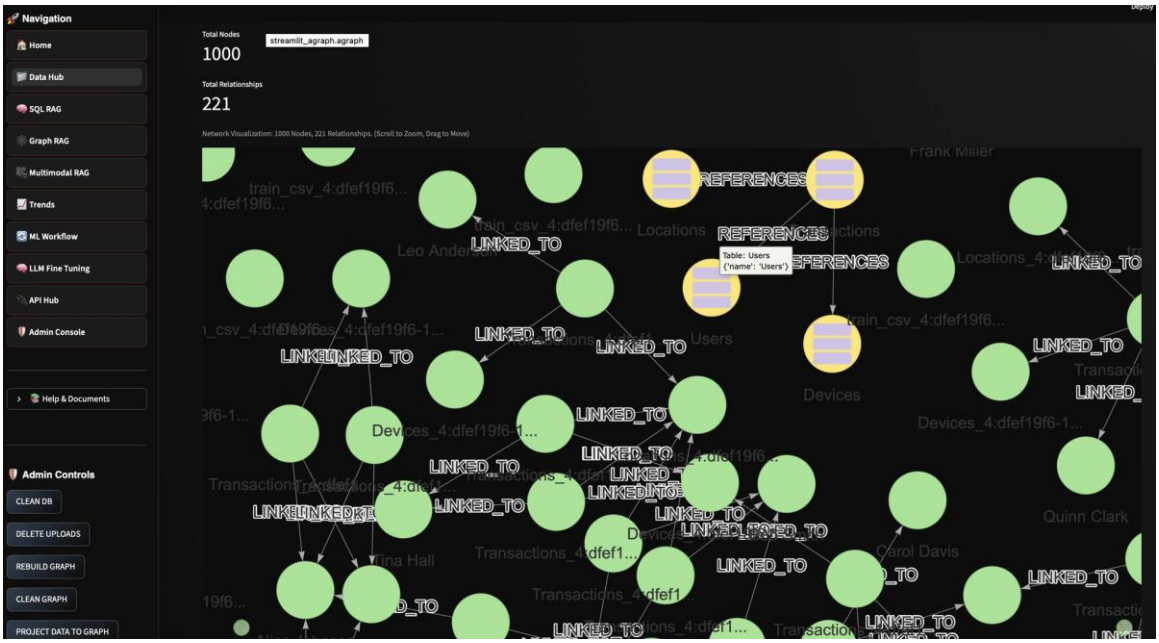
After uploading, user can ask a question in the chat using natural language. If user does not feel satisfied to the answer, user can manually click “Rebuild Graph” button in the sidebar, then after rebuilding graph store, user can ask the same questions again.



In Data Hub, users can see the whole graph and use zoom in/out and shift left/right to check the nodes.

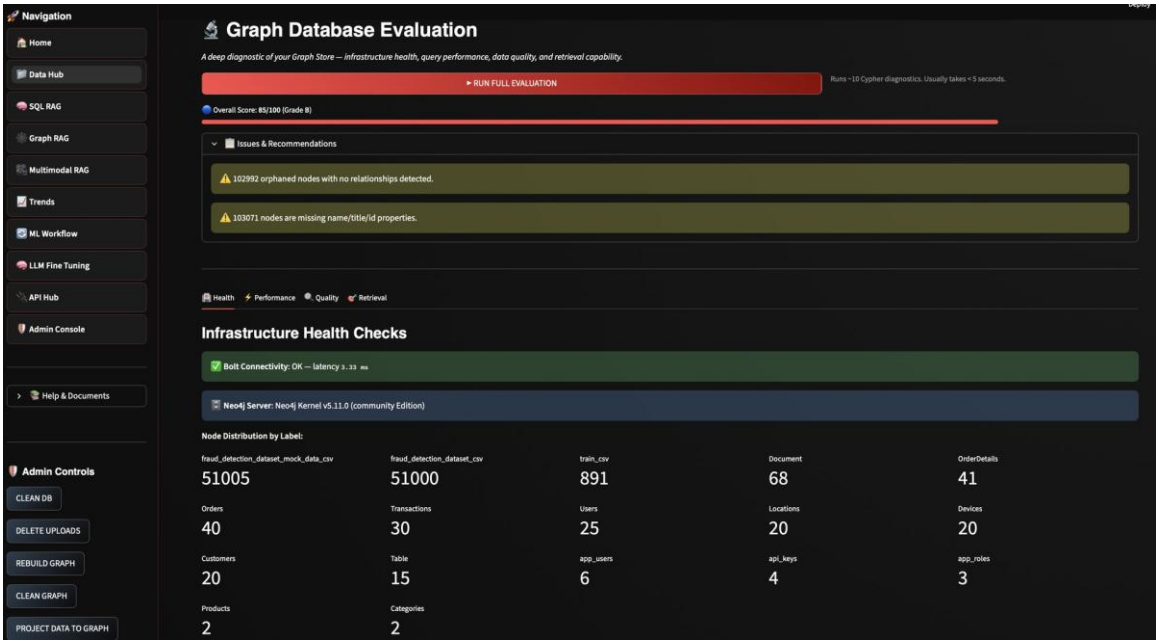


If users click zoom in, user can find the relationship between nodes.

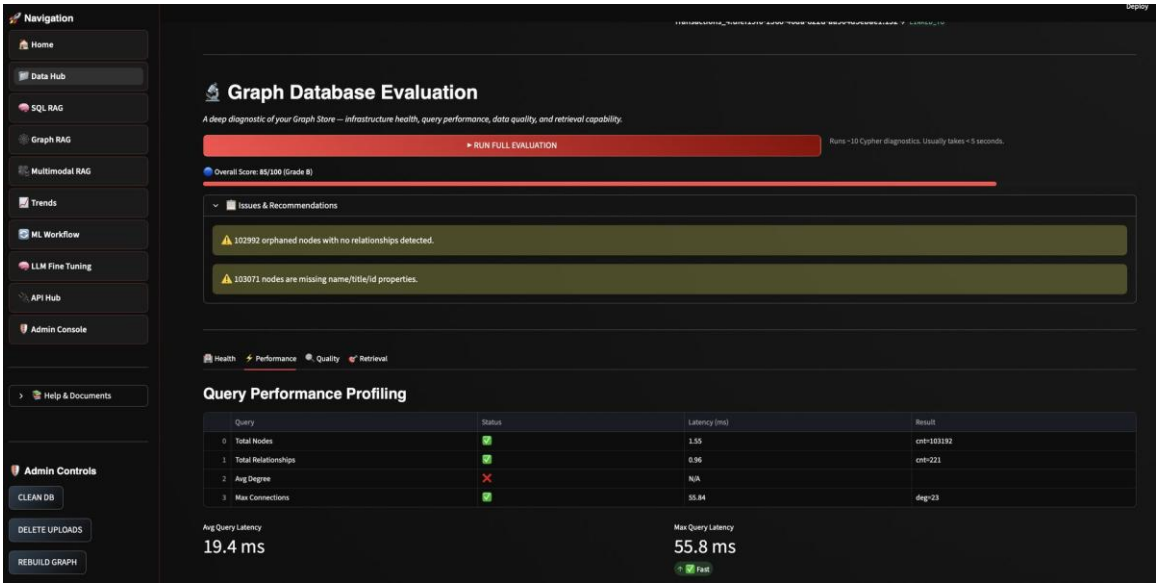


When user zooms deeper, user can see more detail of nodes.

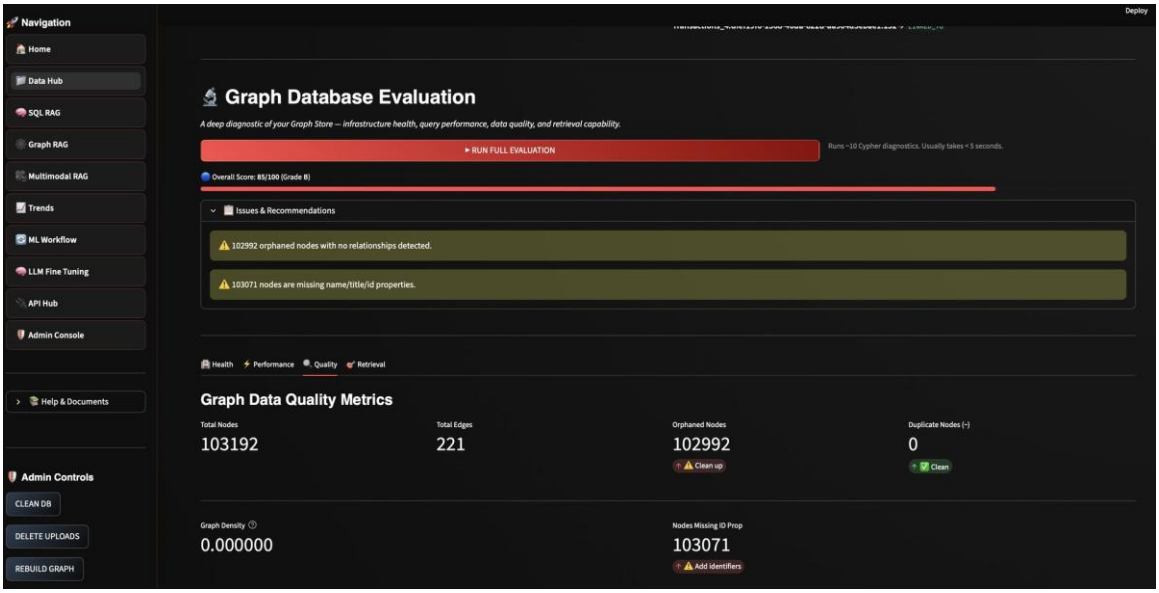
If user clicks on one node, user can see the node detail under the graph visualization.



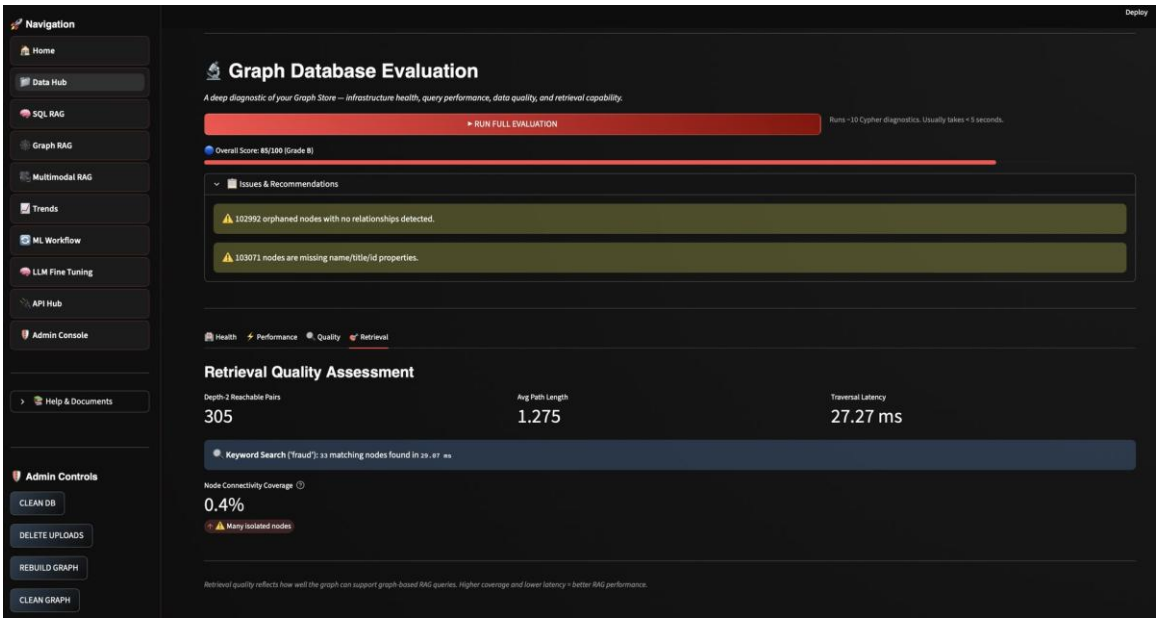
From Graph database Evaluation, users can see the details of Infrastructure Health Check.



From Graph database Evaluation, users can see Query Performance Profiling details.



From Graph Database Evaluation, users can click third tab to see graph Data Quality Metrics.



From Graph Database Evaluation, users can click fourth tab to see Retrieval Quality Assessment details.

6. UI3 - SQL RAG

6.1 UI Design

Constructed around a specialized split-screen IDE layout: the right-hand sidebar is rigidly fixed displaying dynamic active schema dictionaries and auto-generated sample questions. The primary left pane provides an endless-scroll conversational interface supporting dynamic rendering of Markdown, Pandas DataFrames, and Syntax-highlighted SQL blocks.

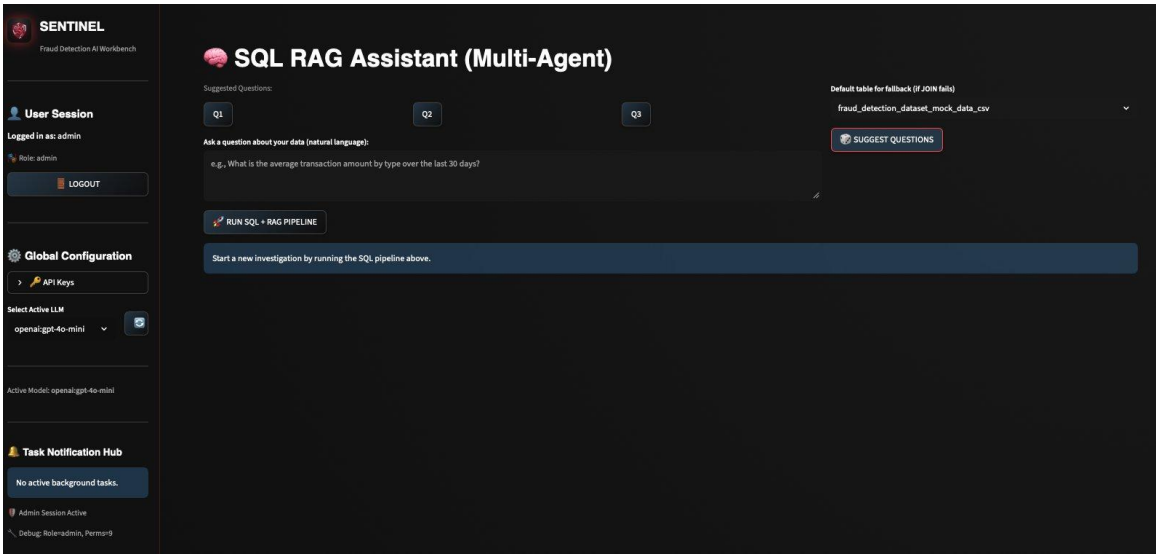
6.2 User Manual

Step 1: Select an active target database using the header dropdown selector.
Step 2: Type a natural language investigation metric (e.g., 'Return all accounts initiating over 5 transfers exceeding $10k in 48 hours').
Step 3: Review the AI's generated SQL, validate the query logic, and download the returned dataframe as an operational CSV.

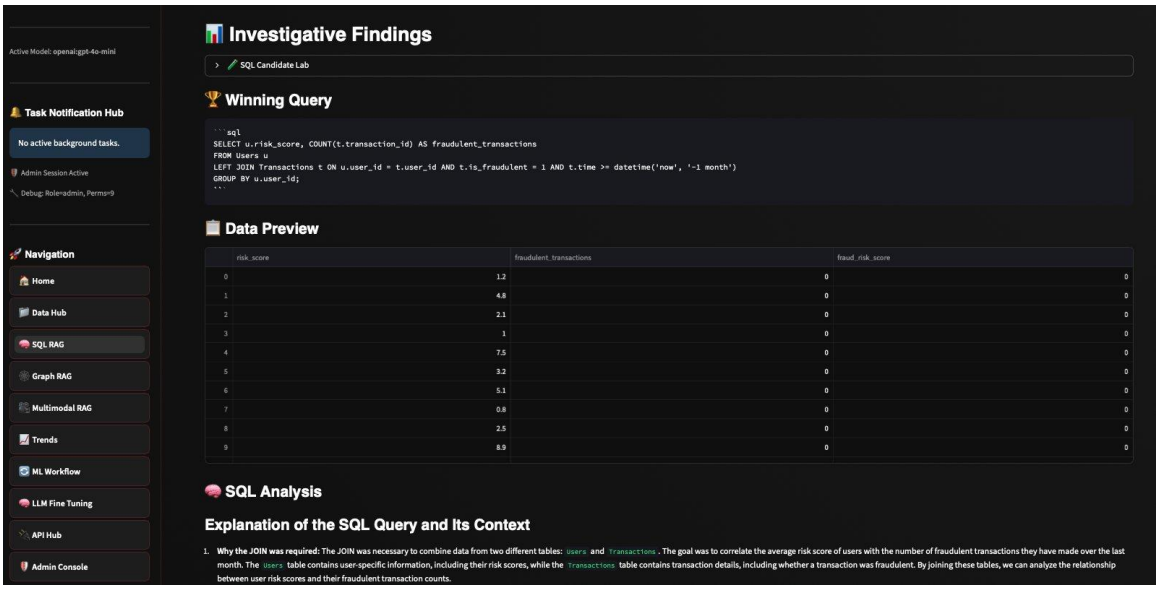
6.3 Logic Workflow

```
▶ [ Natural Language Prompt ]
  └──▶ [ Inject DB Schema Context ]
         └──▶ [ LLM Generates SQL ]
                └──▶ [ AST Security Validation ]
                       └──▶ [ SQLite Execution ]
                              └──▶ [ Render Pandas DataFrame ]
```

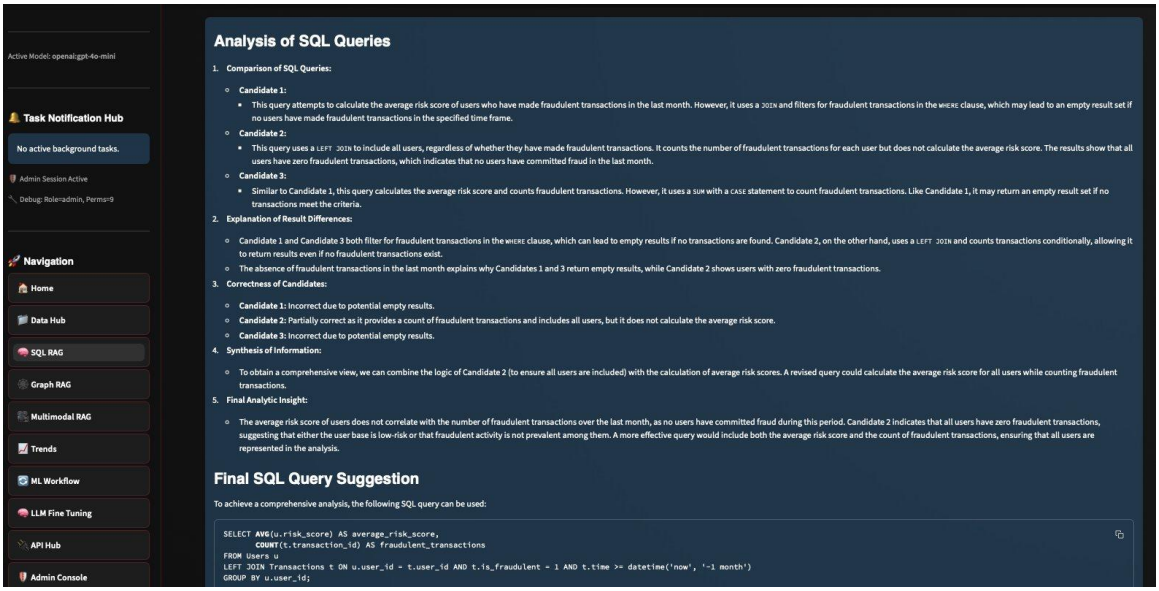
6.4 Demo



In SQL RAG page, users can input a question about structured data or click “Suggest Questions” button to generate sample questions by system. Users can click any one sample question; the sample question will automatically display in the input area. Then, users can click “RUN SQL + RAG PIPELINE” button to retrieve the results.

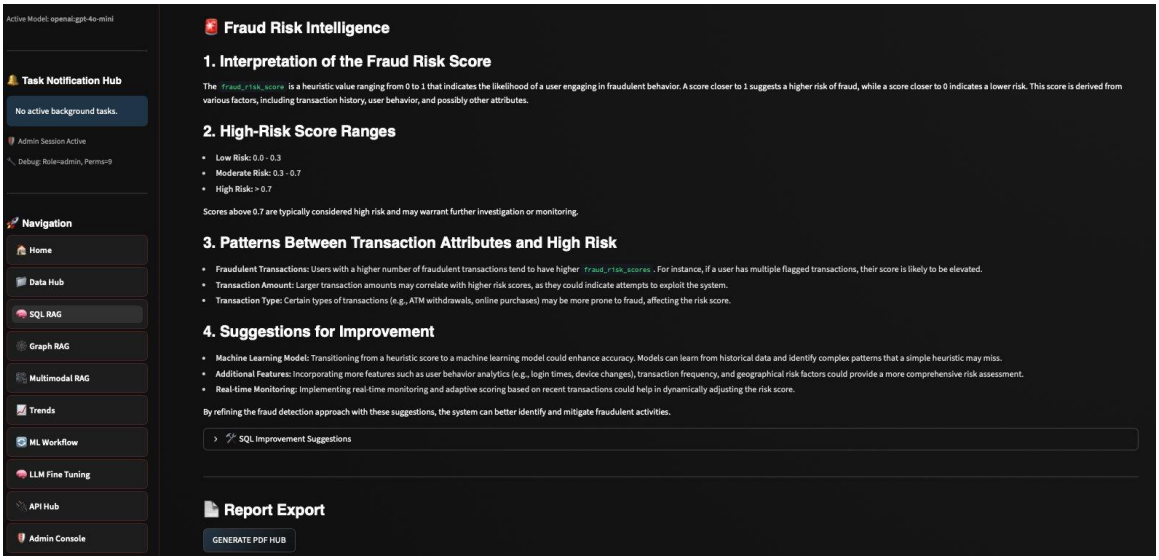


In the result of SQL RAG PIPELINE, it displays the most candidate of SQL to use query to the database and display data preview of the table.



This is a concrete analysis of SQL queries. Because LLM can generate at most 3 SQL queries and pick up the most sematic matched SQL query as the best candidate query to search data in the database.

This is a description of hybrid synthesis analysis report for user review.



In the last part, users can find Fraud risk intelligence analysis report.

7. UI4 - Graph RAG

7.1 UI Design

Mechanically mirrors the SQL RAG split-screen but substitutes relational schemas for intuitive Node/Edge taxonomy visualization. It integrates a physics-simulated PyVis network Javascript element directly via an interactive `st.components.v1.html` rendering block, situated immediately below the prompt intake.

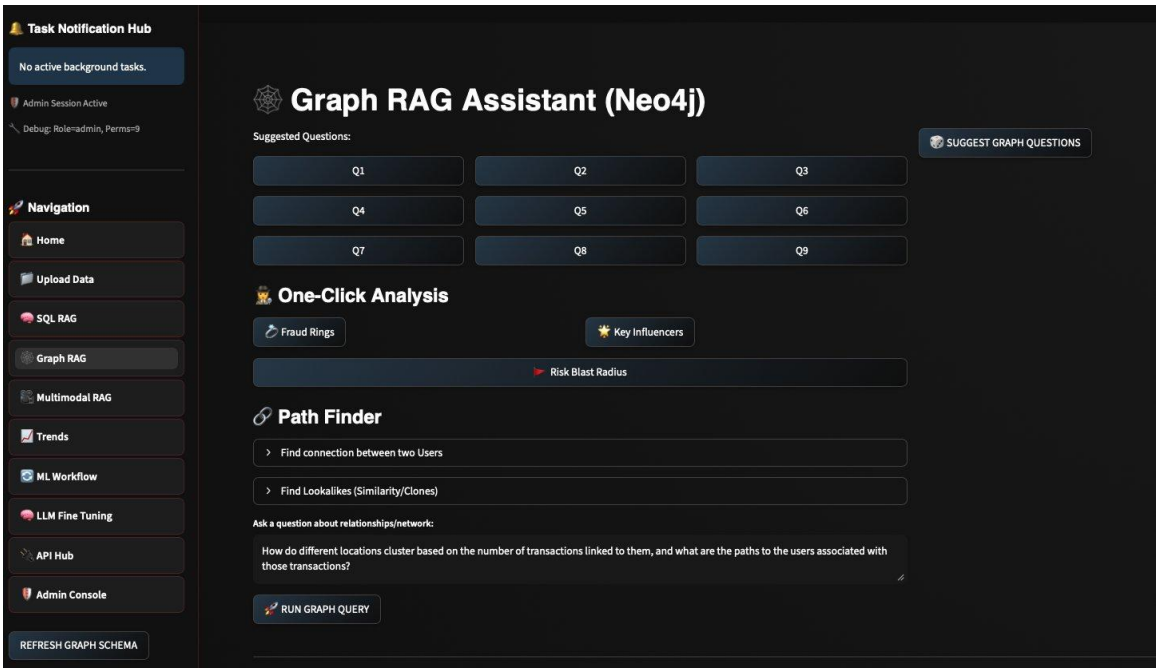
7.2 User Manual

Step 1: Choose an active Graph taxonomy matrix.
Step 2: Query the agent for overlapping attributes (e.g., 'Visualize the sub-graph involving IP Address 192.168.1.5').
Step 3: The interface retrieves the entity map. Manually drag nodes to rearrange the cluster layout, hover for deep property inspection, and review the AI-narrated summary of the identified fraud syndicate.

7.3 Logic Workflow

```
▶ [ User Natural Prompt ]
   └──▶ [ Fetch Graph Taxonomy Map ]
          └──▶ [ LLM Translates to Cypher ]
                 └──▶ [ Execute via Neo4j Driver ]
                        └──▶ [ Generate Physics Graph Elements ]
                               └──▶ [ Render HTML iframe PyVis ]
```

7.4 Demo

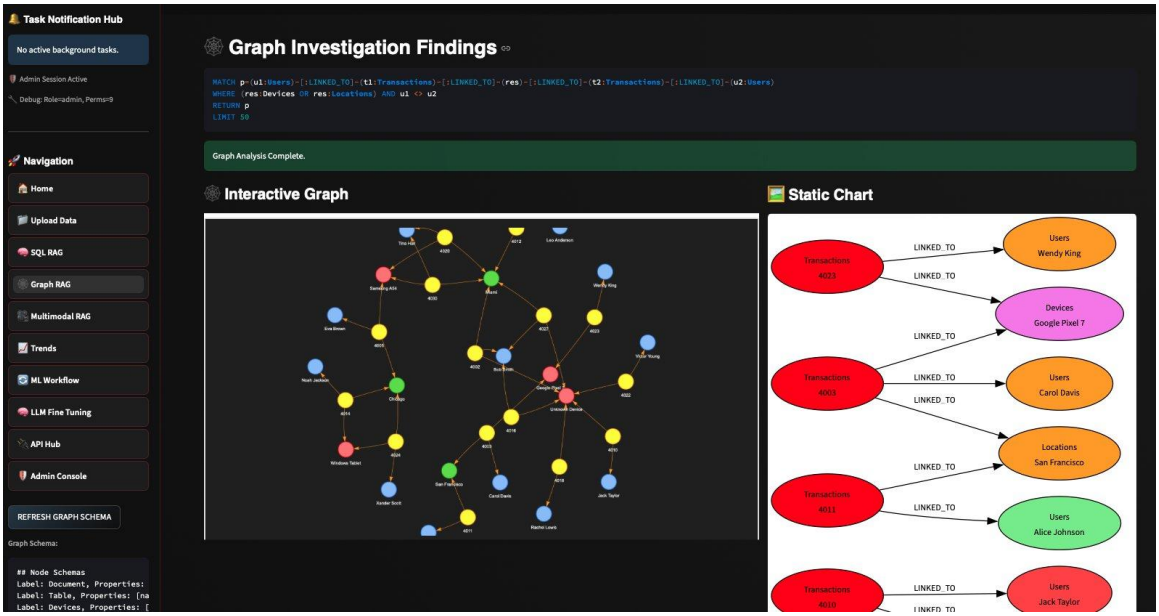


In **Graph RAG (Graph-based Retrieval-Augmented Generation)**, the system doesn't just retrieve similar chunks. It builds and reasons over a knowledge graph (entities + relationships). Users can have a few options to do graph RAG:

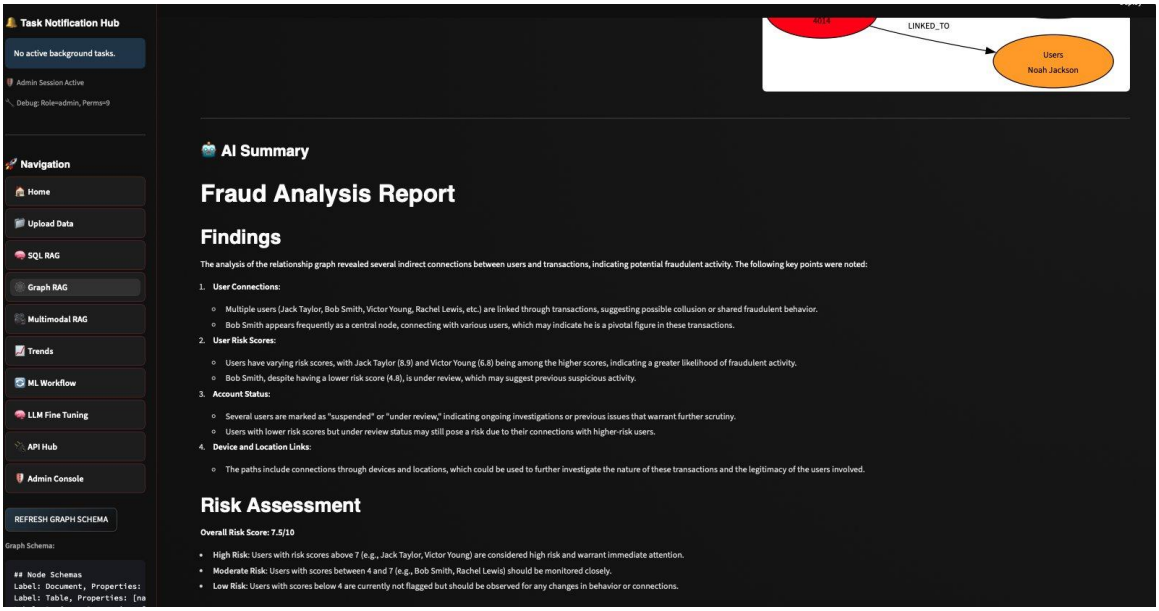
- Option 1: Users can input his/her own question.
- Option 2: Users can click "Suggest Graph Questions" to generate at most 9 questions then user can pick anyone question.
- Option 3: System provide one-click analysis which contains several buttons, user can click any one button
    - Fraud Rings

- Fraud rings = clusters of entities that collaborate in suspicious patterns.

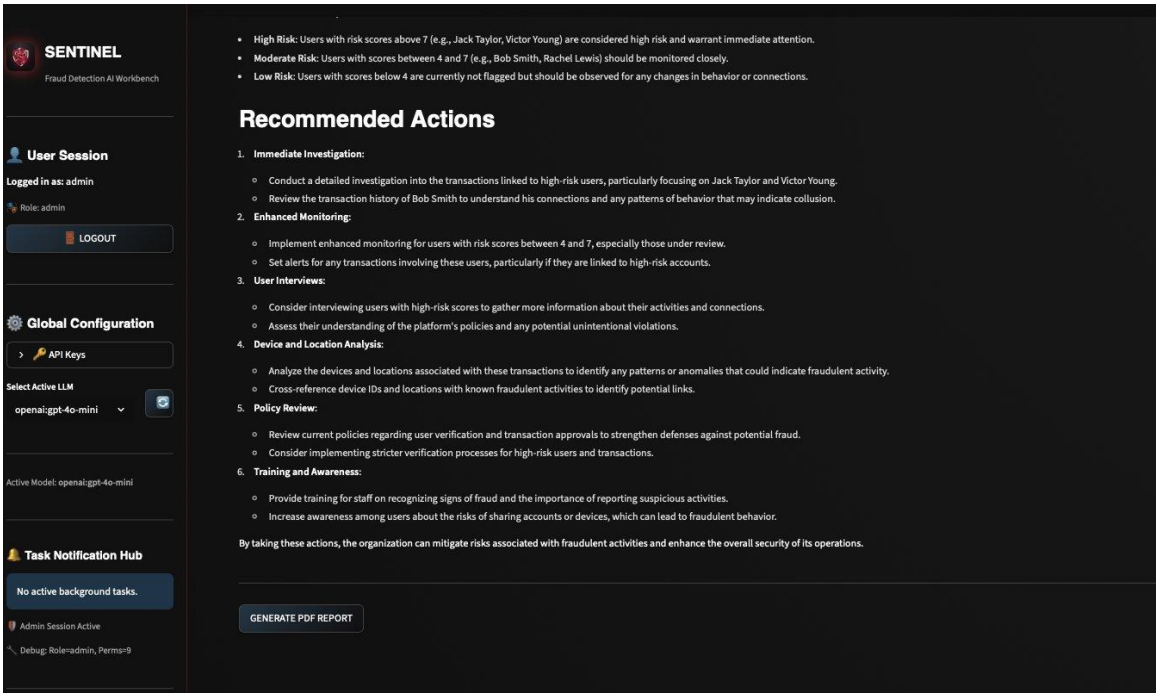
- Path Finder
  - Path Finder identifies the relationship path between two entities in the graph.
- Key Influence
  - This identifies which node (entity) has the most influence in a network.



In the Graph Investigation, first, users can see the best candidate of graph query language for graph query. Then, user can see Interactive Graph on the left side and static chart analysis on the right side.



After graph query, users can see AI insights of fraud analysis.



Users can see Recommended Actions.

Users can see top impacts to key influencers for fraud analysis report.

8. UI5 - Multimodal RAG

8.1 UI Design

Segmented cleanly via inner tabs dividing logical intake flows: Audio, Image, and Document ingestion. The interface incorporates built-in HTML5 media players enabling contextual review, alongside real-time transcription loading spinners.

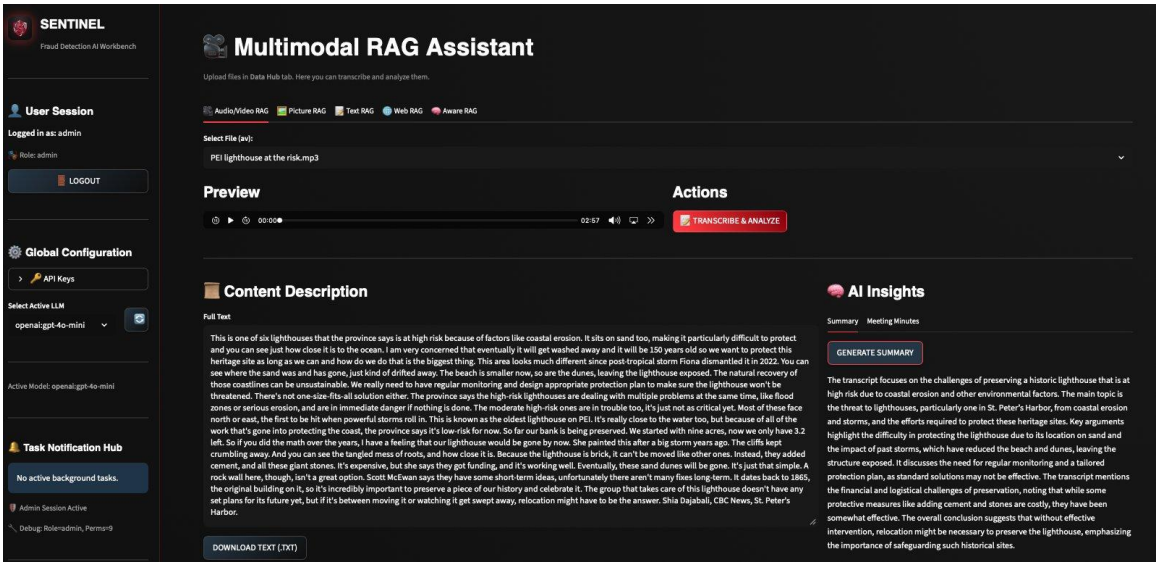
8.2 User Manual

Step 1: Upload a piece of localized evidence, such as a scanned invoice image or a suspicious WAV audio recording.
Step 2: Once uploaded, wait for the OCR or Whisper transcription block to finalize.
Step 3: Interact specifically with that media via the chat element, instructing the AI to 'Compare the extracted routing number to standard banking indices'.

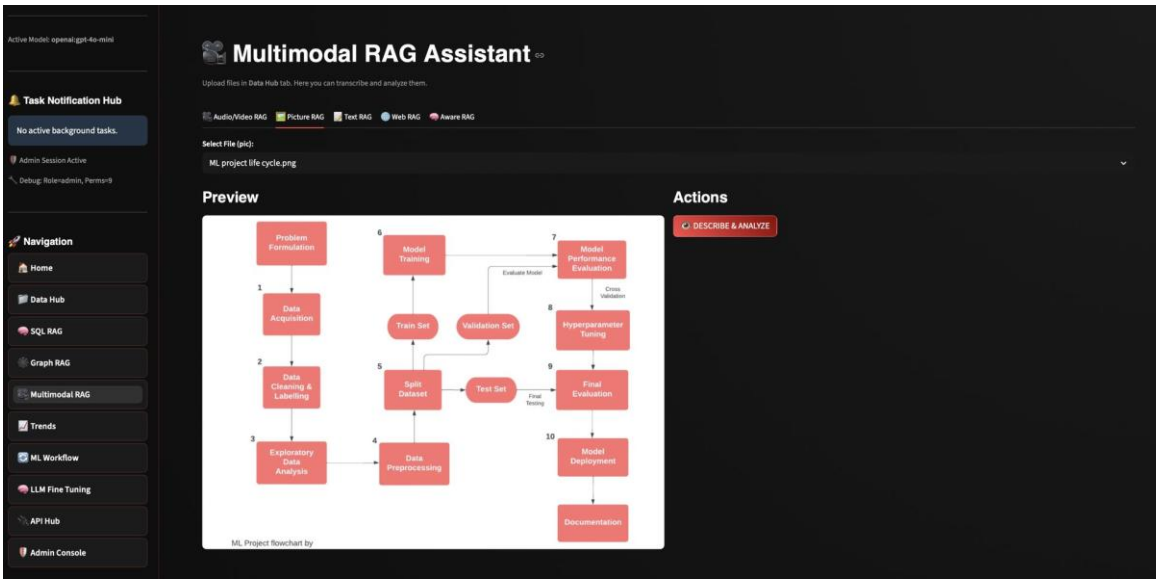
8.3 Logic Workflow

```
▶ [ Receive Binary Media Payload ]
  └──▶ [ Classify Media Modality ]
         ├──▶ (If Image) [ Extract Layout Extents via OCR ]
         └──▶ (If Audio) [ Transcribe via Whisper Node ]
                └──▶ [ Pipe Output to Multimodal LLM Agent ]
                       └──▶ [ Output Chat Response ]
```

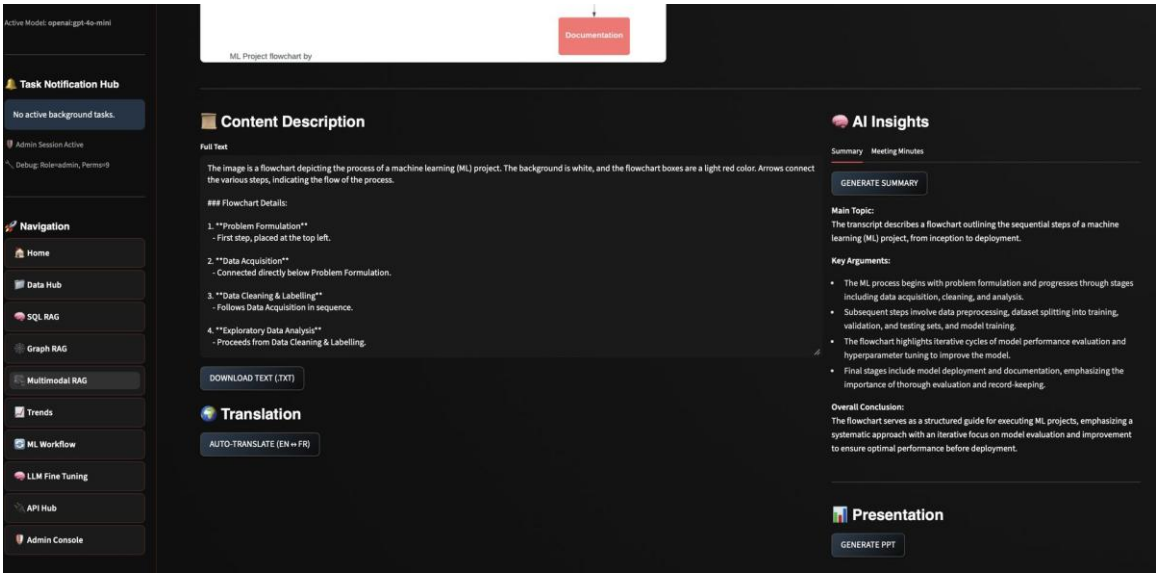
8.4 Demo



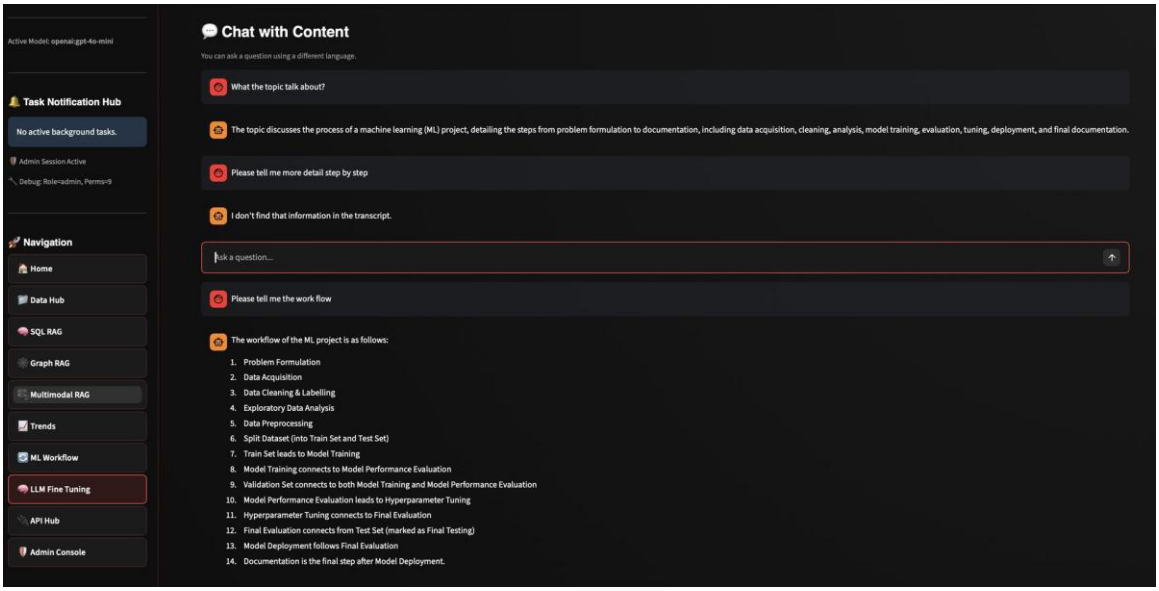
In Multimodal RAG, users can extract information from audio or video In above sample, user get content description from a mp3 file and can generate summary or meeting minutes.



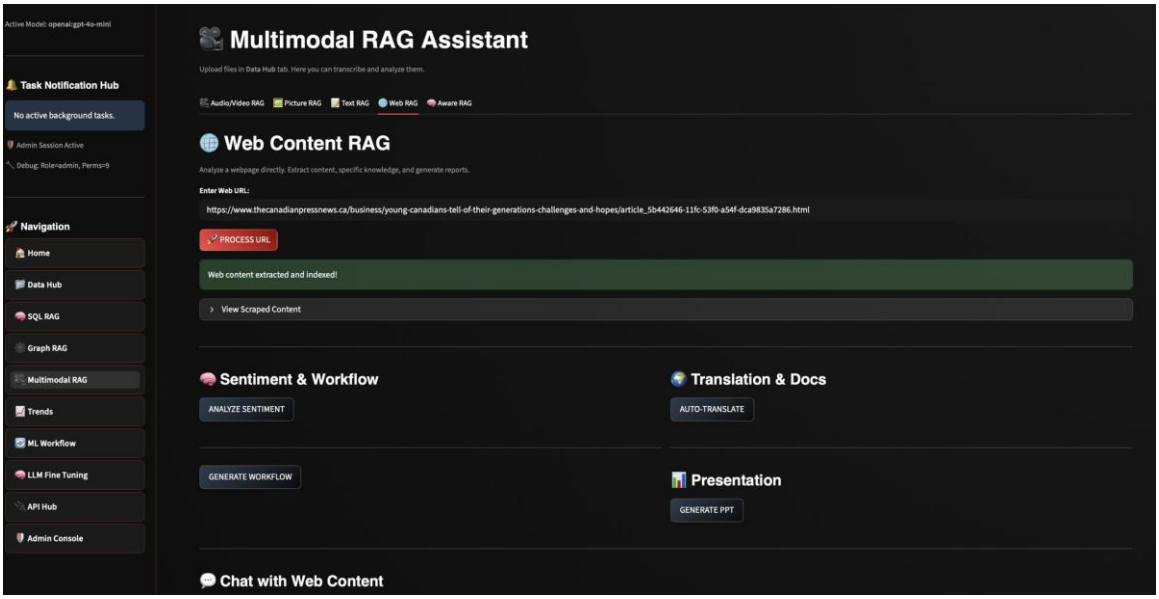
Based on a picture of workflow, user can use “Describe & Analyze” to get below information.



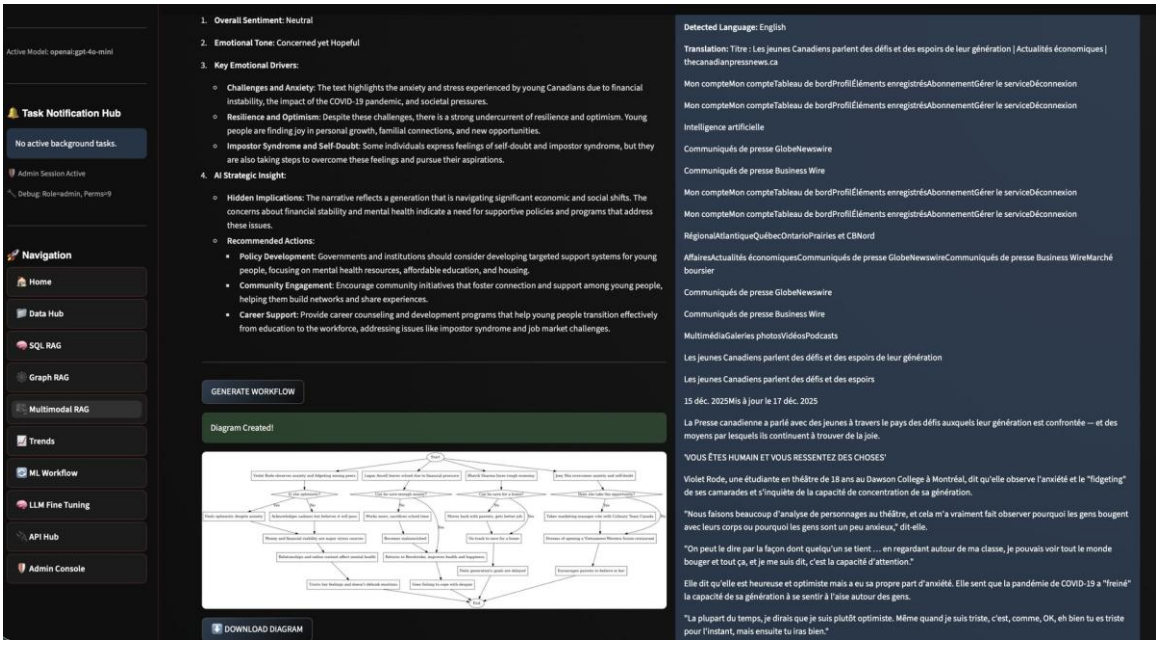
Based on detecting source language, user can translate to English and French. Users can generate PPT from summary or minute content.



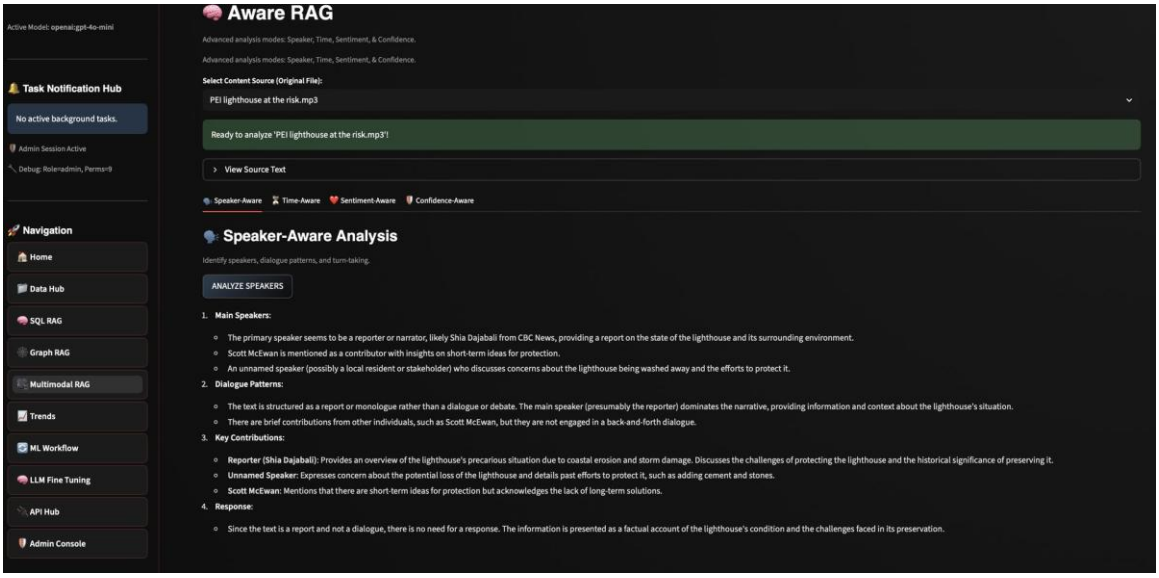
Based on graph data for multimodal, users can chat to get I information.



Based on Web content RAG, users can get sentiment analysis and draw the workflow shown below.



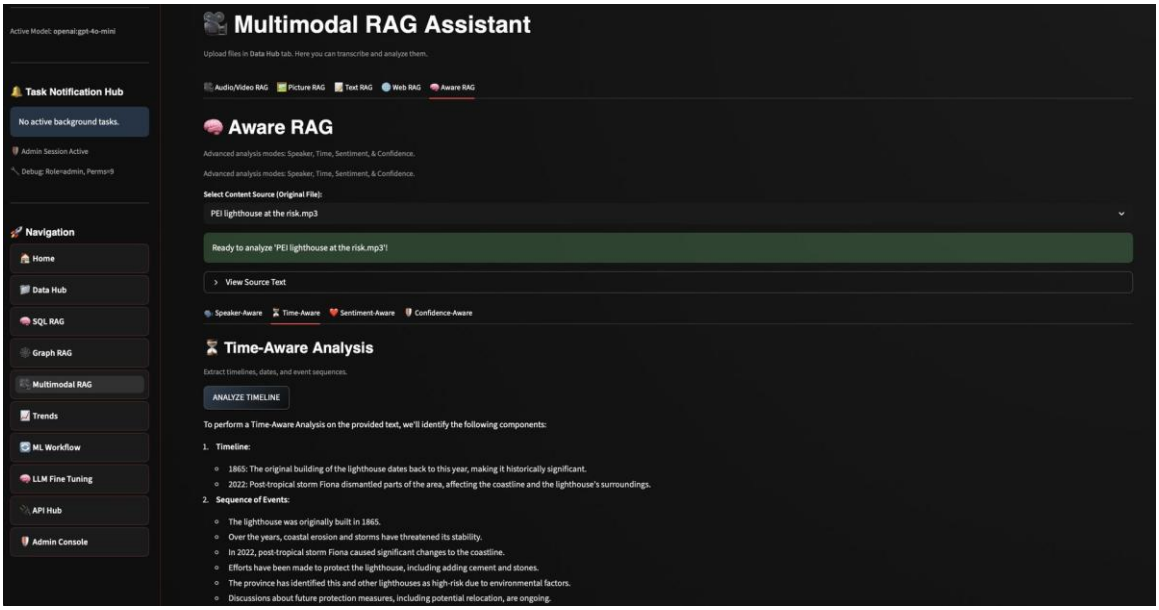
Based on extract Web information and source language, system can automatically translate to English and French.



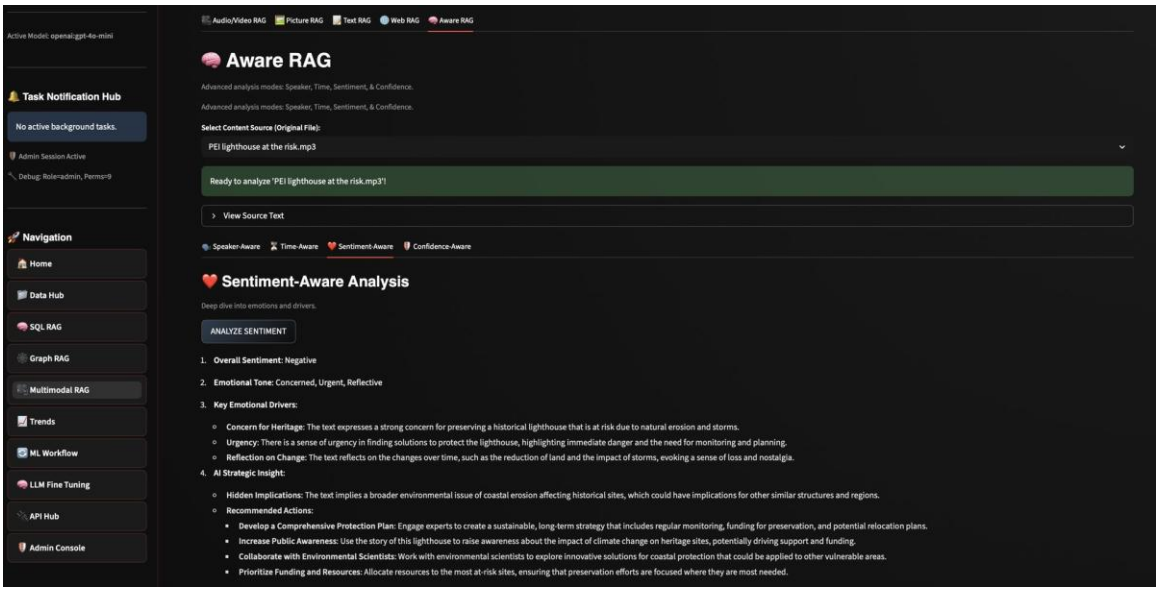
In Aware RAG, there are 4 options for users:

- Speaker Aware
- Time Aware
- Sentiment Aware
- Confidence Aware

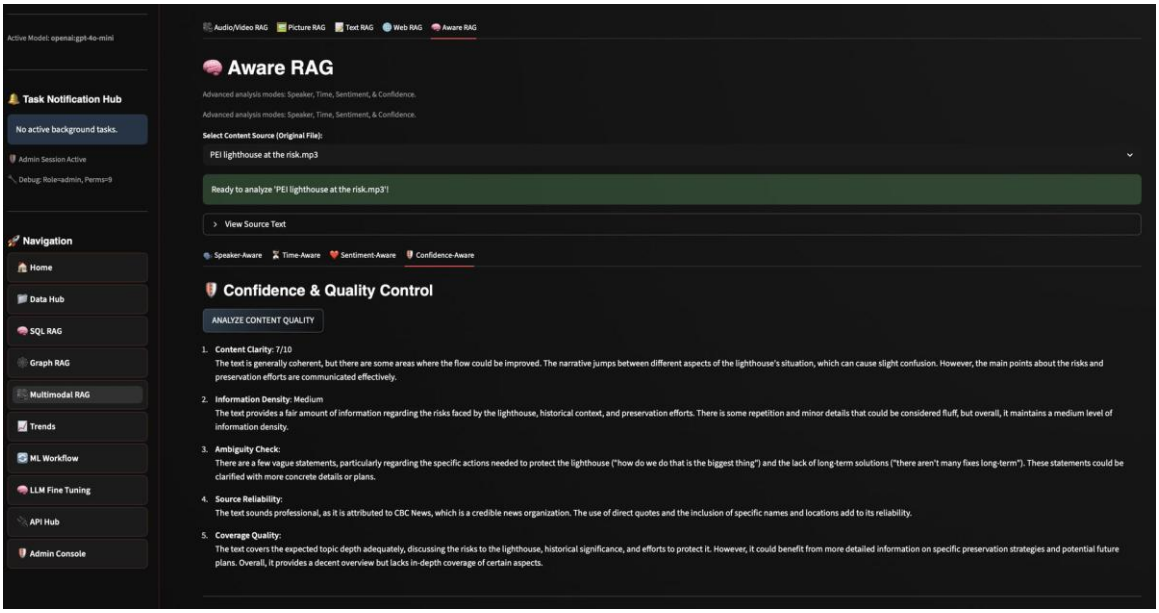
In Speak Aware, based on multimodal content, users can analyze speaker's purpose, dialogue patterns, key contributes, response, so on.



In Time-aware analysis, users can analyze multimodal timeline and sequence of events.



In Sentiment-Aware analysis, users can analyze sentiment of multimodal. For example, you can see overall sentiment active or negative, emotional tone, key emotional drivers and insights.



In Confidence Aware, users can analyze content quality, for example, content clarity, density, ambiguity check, source reliability and coverage quality, so on.

9. UI6 - Trends and Insights

9.1 UI Design

Serves as the primary BI reporting dashboard. Leverages visually reactive Plotly Express charts scaling automatically to container width. Employs advanced cascading drop-downs for dimensional X/Y axis parameterization, dynamic color grouping, and visual-type toggles (scatter, bar, histogram).

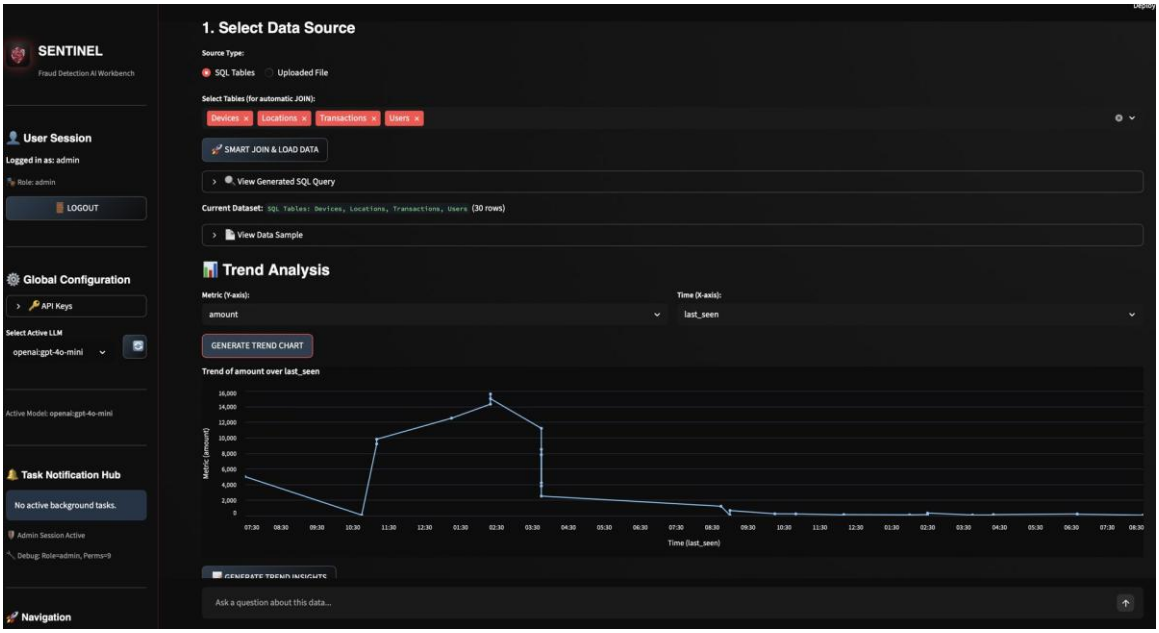
9.2 User Manual

Step 1: Select an ingested target dataset.
Step 2: Assign mathematical parameters to the X/Y axes to identify macro-trends visually (e.g., fraudulent volume plotted across transaction hours).
Step 3: Click 'Generate AI Insights'. The platform passes the statistical density bounds to the LLM to output a human-friendly narrative explaining the chart.

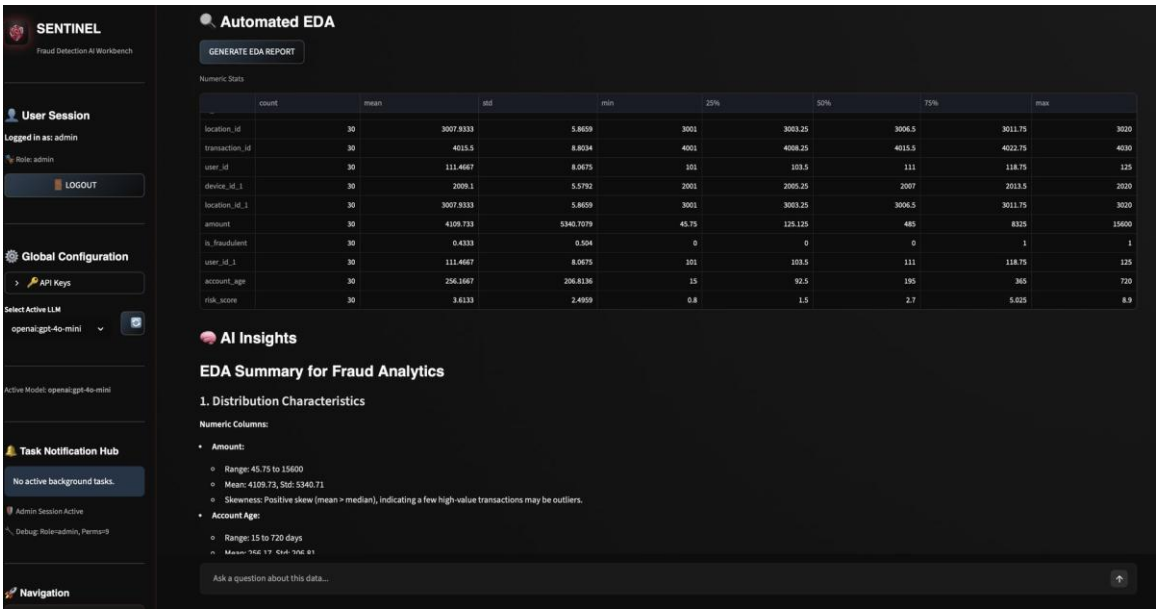
9.3 Logic Workflow

```
▶ [ Select Target Dataframe ]
  └──▶ [ Assign Interactive Chart Dimensionality ]
         └──▶ [ Generate Plotly Canvas ]
                └──▶ (Optional Trigger)
                       └──▶ [ Serialize Stats Payload ]
                              └──▶ [ Fetch LLM Chart Narrative Analysis ]
```

9.4 Demo

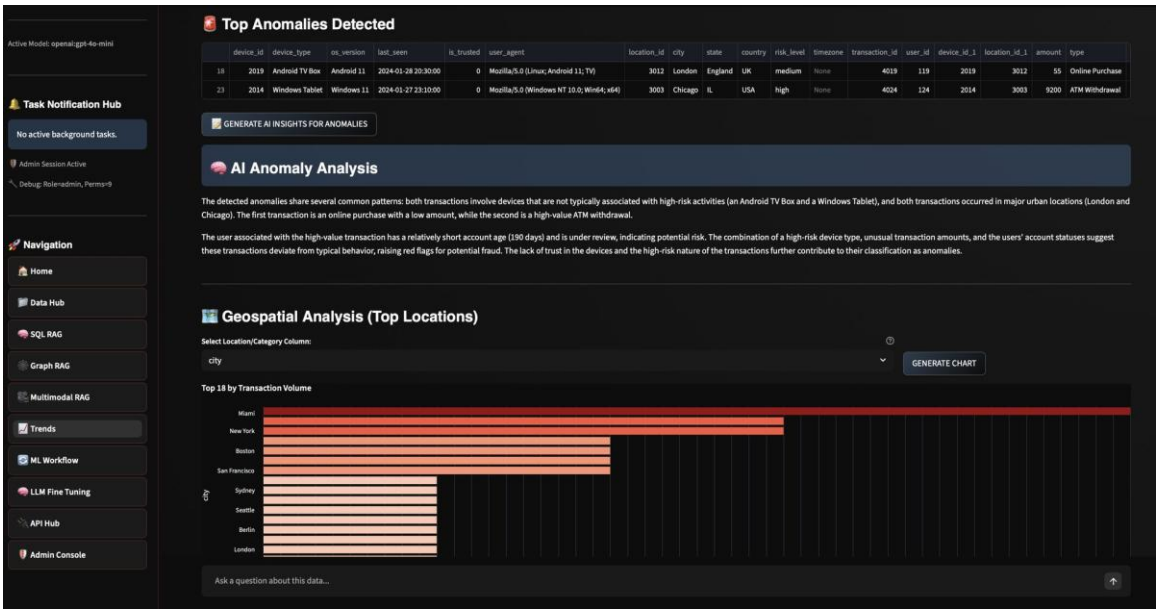


In Trends page, for multiple selected relational tables, after clicking “Smart join & data load” button, user can get Trend Analysis.

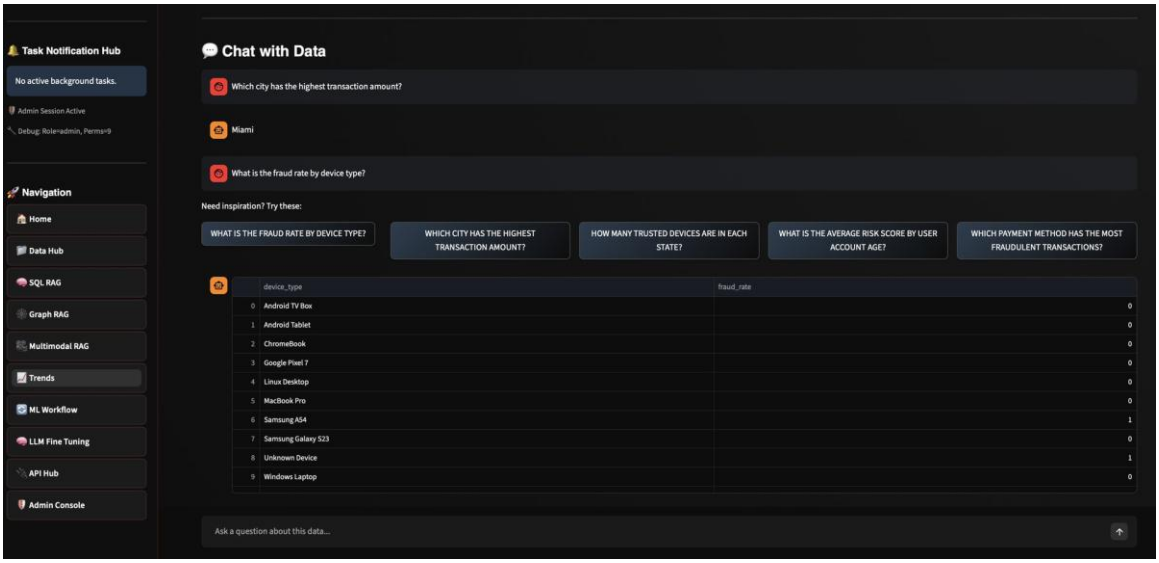


Based on relational data, users can get EDA analysis. EDA = Exploratory Data Analysis which is deeply explore data before building AI/ML models. EDA helps you:

- Discover patterns
- Detect outliers
- Understand feature relationships
- Validate assumptions
- Improve feature engineering



Users can see top anomalies detected analysis and based on location column, make the Geospatial analysis for the top locations.



Users can use natural language to chat with data or use system generated sample question to ask questions.

10. UI7 - ML Workflow

10.1 Functionality Design

Functionally designed as an automated, multi-tiered pipeline leveraging scikit-learn and imbalanced-learn. The system decomposes the complexity of traditional data science workflows into sequential state-managed modules. It actively mitigates fraud datasets' notorious class imbalance using SMOTE (Synthetic Minority Over-sampling Technique) before executing model training. Furthermore, game-theoretic SHAP (SHapley Additive exPlanations) values are automatically computed to provide strict regulatory transparency into exactly why the algorithmic decision tree flagged a specific transaction.

10.2 UI Design

The interface spans multiple horizontal tabs (Build, Train, Tune, Score, Deploy, Monitor) to gracefully orchestrate the end-to-end Machine Learning lifecycle without overwhelming the analyst. Parameters and hyperparameter tuning fields are housed in expandable accordions. Outputs include visually striking KPIs (Accuracy, F1-Score) and interactive matplotlib/plotly charts for SHAP dependency curves and Confusion Matrices.

10.3 User Manual

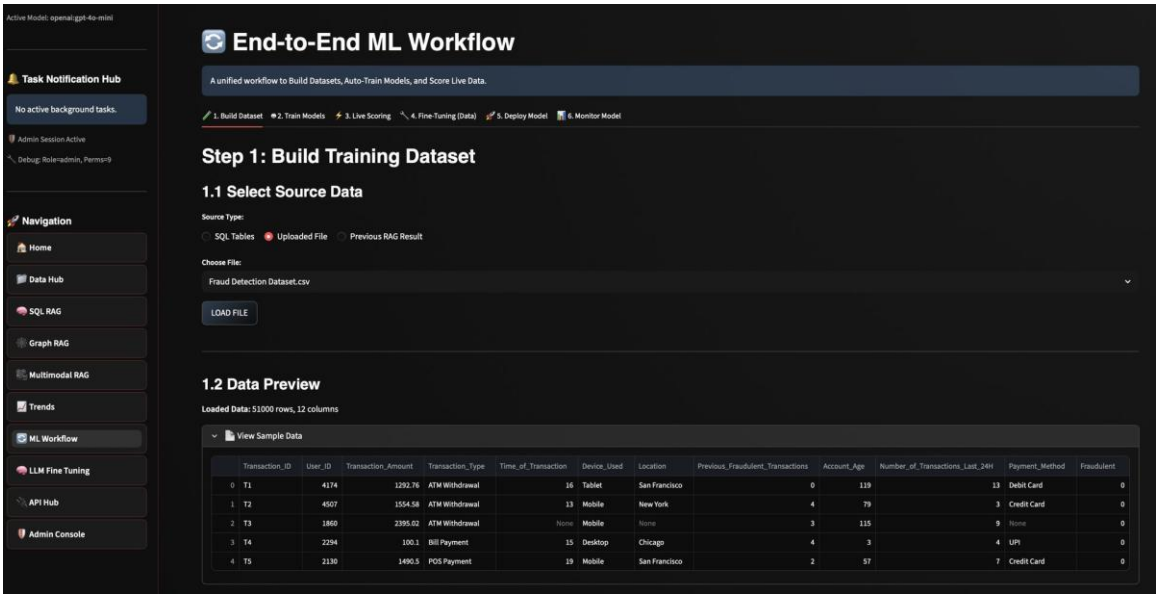
Step 1: In the 'Build Pipeline' tab, select a dataset, assign target variables, and configure automated feature engineering (imputation, categorical encoding, SMOTE for imbalance).
Step 2: In the 'Train Model' tab, choose an algorithm (e.g., Random Forest, XGBoost), configure the Train/Test split, and fit the model to view KPIs and SHAP feature importance.
Step 3: Move to 'Tune Model' for GridSearchCV hyperparameter tuning.

Step 4: Use 'Batch Scoring' to apply your serialized model to new datasets, and 'Monitor' to observe data drift and performance decay over time.

10.4 Logic Workflow

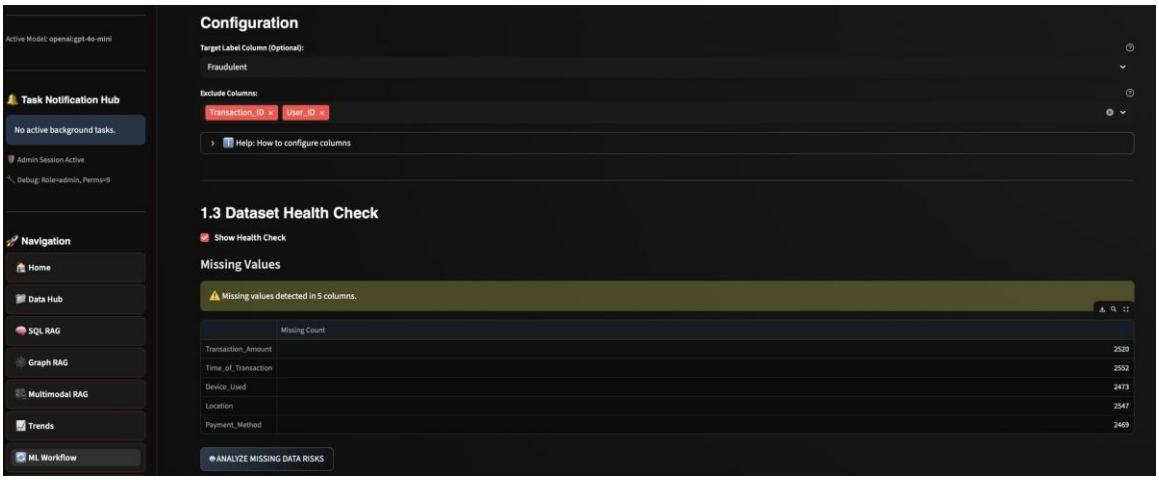
```
▶ [ Data Prep & Feature Eng (Build) ]
   └──▶ [ Split Data & SMOTE Balance ]
          └──▶ [ Train Classifier (Train/Tune) ]
                 └──▶ [ SHAP Interpretability Computations ]
                        └──▶ [ Serialize Model (.pkl) & Deploy ]
                               └──▶ [ Batch Scoring Execution ]
```

10.5 Demo

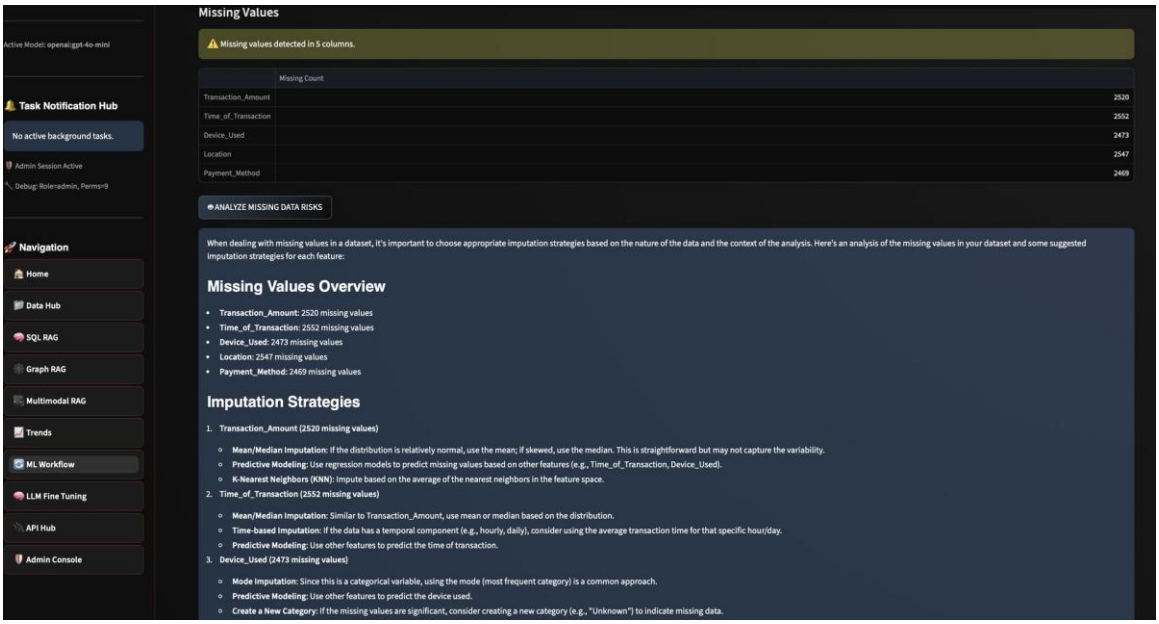


In the end-to-end ML workflow, there are 6 steps to implement in current system:

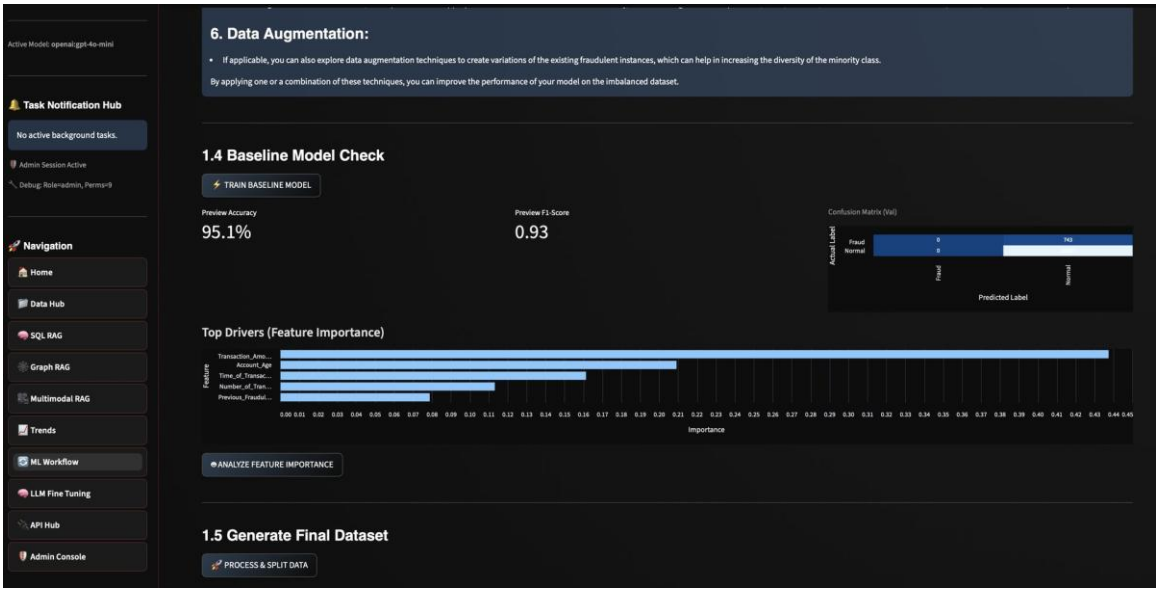
- Build dataset
- Train model
- Live Scoring
- Model fine tuning
- Deploy Model
- Monitor Model



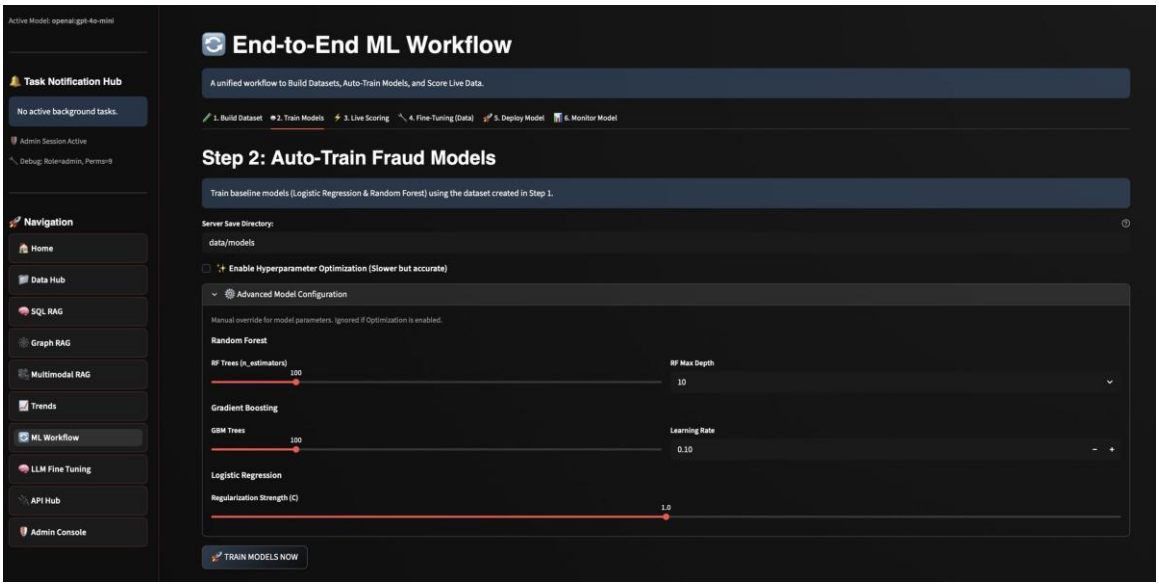
Before train data, users need to select target column and exclude columns. The user can do dataset health check.



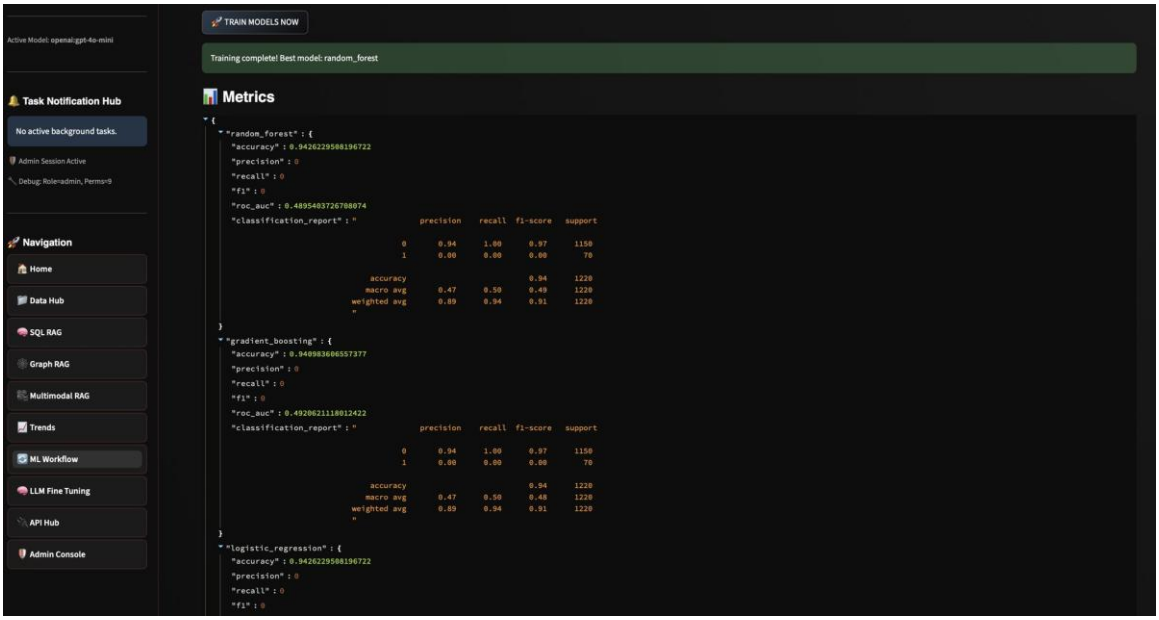
Users can see missing values insights after dataset health check.



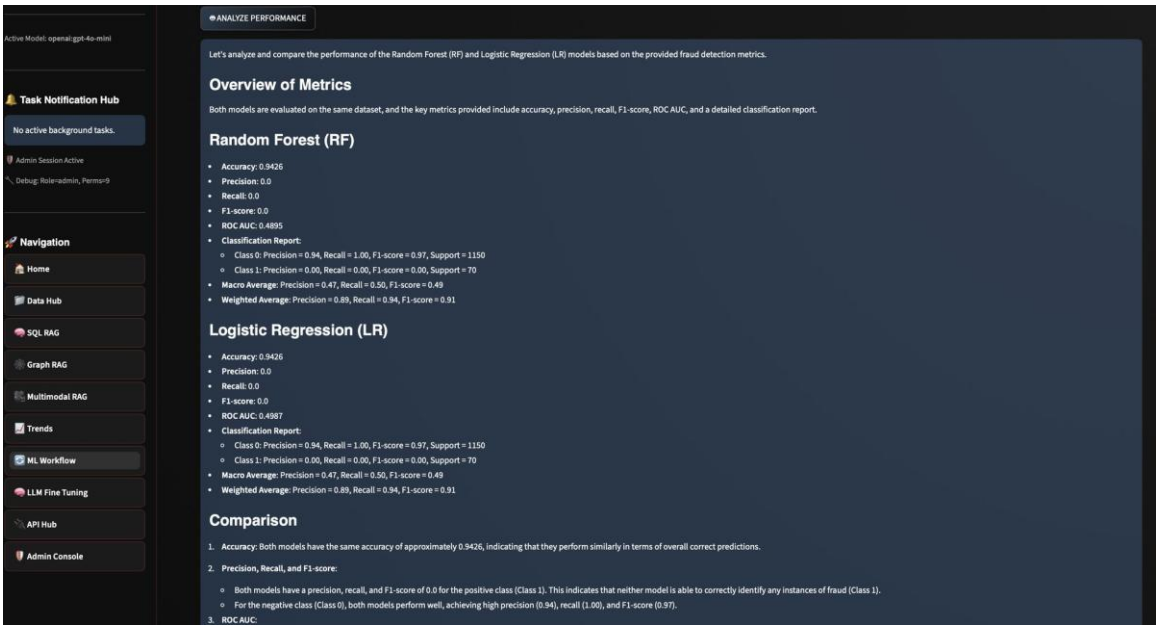
Users can do baseline model check to ensure current dataset features importance and then can execute to split and generate a dataset for training.



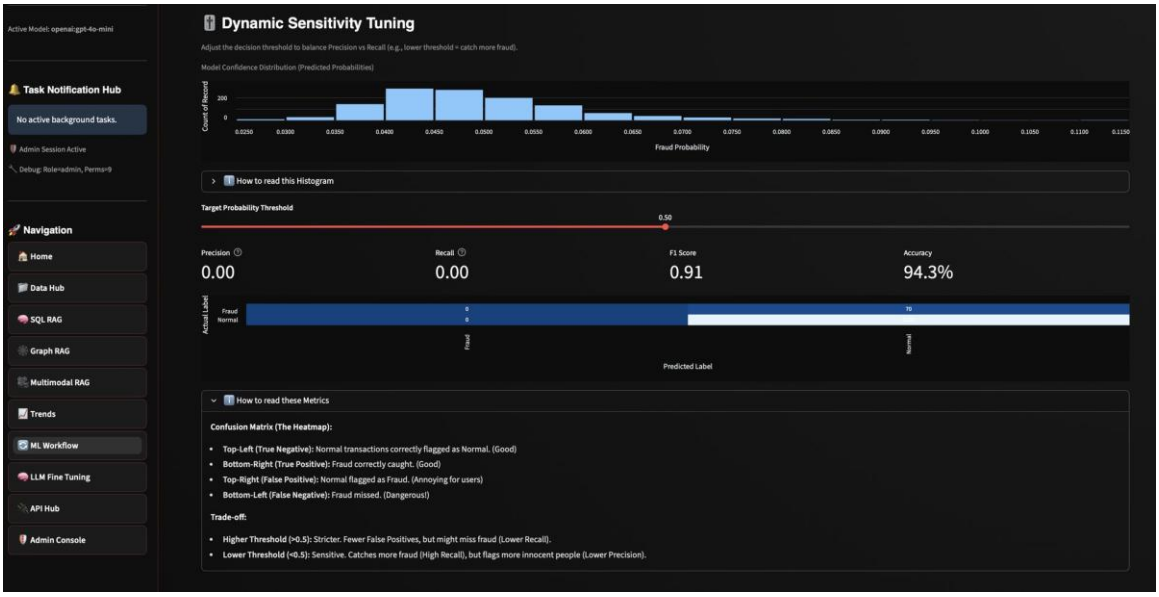
Before train the mode, users can manually adjust the model parameters and then click “Train Models now”.



After training the mode, users can see the model metrics.



There is overview of Metrices shown under neath.



Users can choose dynamically sensitive tuning.

11. UI8 - LLM Fine Tuning

11.1 Functionality Design

Designed to bring localized, secure Large Language Model optimization to the analyst without requiring cloud execution (preventing PII/data leakage). It utilizes Low-Rank Adaptation (LoRA) via Apple's MLX (or PyTorch) framework to modify model adapters in a fraction of traditional compilation time. The functional architecture isolates the dataset generation (Extracting historical investigator prompts or using teacher-LLM synthetics) from the multi-threaded inference testbed, ensuring empirical validation of model hallucination reduction.

11.2 UI Design

A highly structured 'Command Center' consisting of four distinct tabs: Collect, Review, Train, and Test. The UI balances technical command configurations (LoRA rank, Alpha, Batch limits) with intuitive data viewing capabilities. The 'Test' tab features a dual-chat split-pane, rendering identical subsequent prompts to both a 'Base Model' and a 'Fine-Tuned Model' concurrently for direct A/B visual comparison.

11.2 User Manual

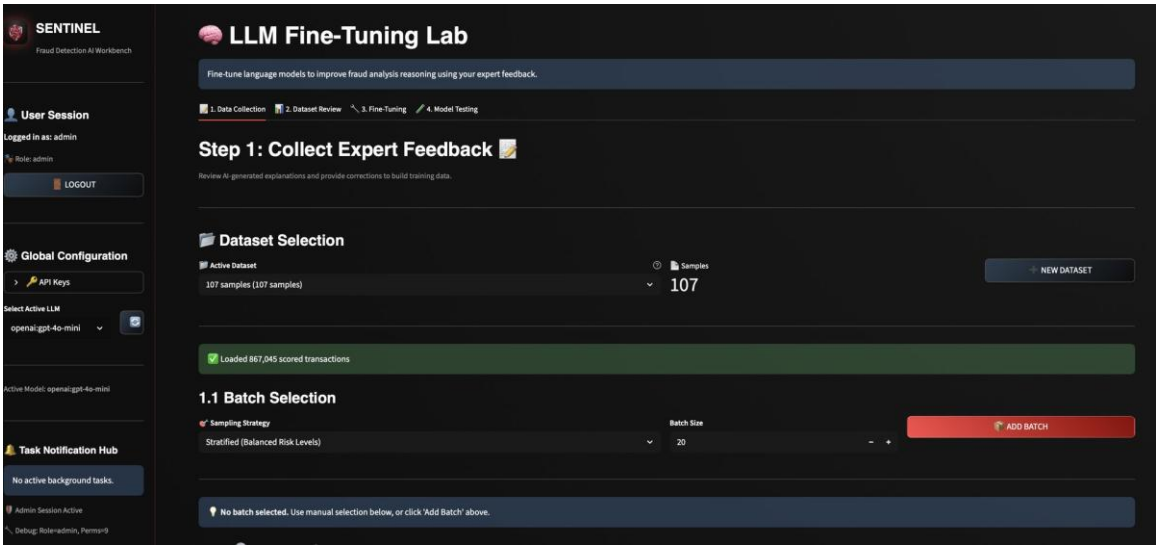
Step 1: Use the 'Collect Dataset' tab to securely aggregate Chat/RAG logs, or synthetically generate instructional JSONL datasets using an LLM teacher.
Step 2: In 'Review', curate the rows to ensure high-quality question/answer pairs and delete anomalous examples.
Step 3: In the 'Train' tab, select a Base Model (e.g., LLaMA), configure LoRA parameters, and execute the localized MLX training cycle.

Step 4: Launch the 'Dual-Chat Test' to interrogate both models simultaneously, verifying that the new heavily-weighted fraud knowledge adheres properly.

11.3 Logic Workflow

```
▶ [ Synthesize/Extract JSONL ]
  └──▶ [ Curate & Review Schema ]
         └──▶ [ Initiate MLX LoRA FineTuner Module ]
                └──▶ [ Output Custom Quantized Adapters ]
                       └──▶ [ Mount Dual Inference Endpoints ]
                              └──▶ [ Concurrent A/B Model Querying ]
```

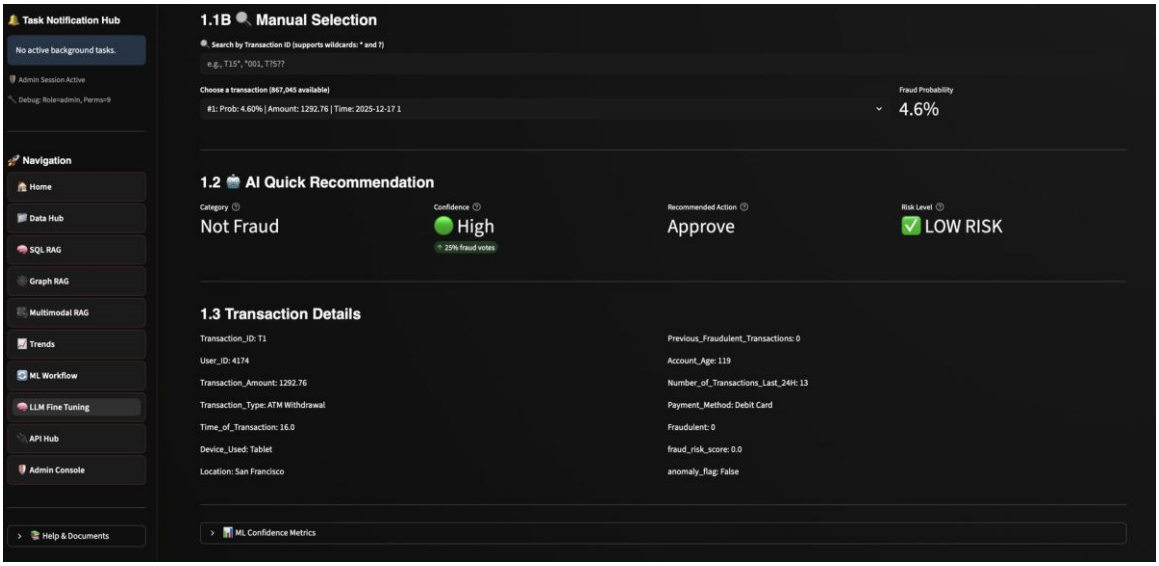
11.4 Demo



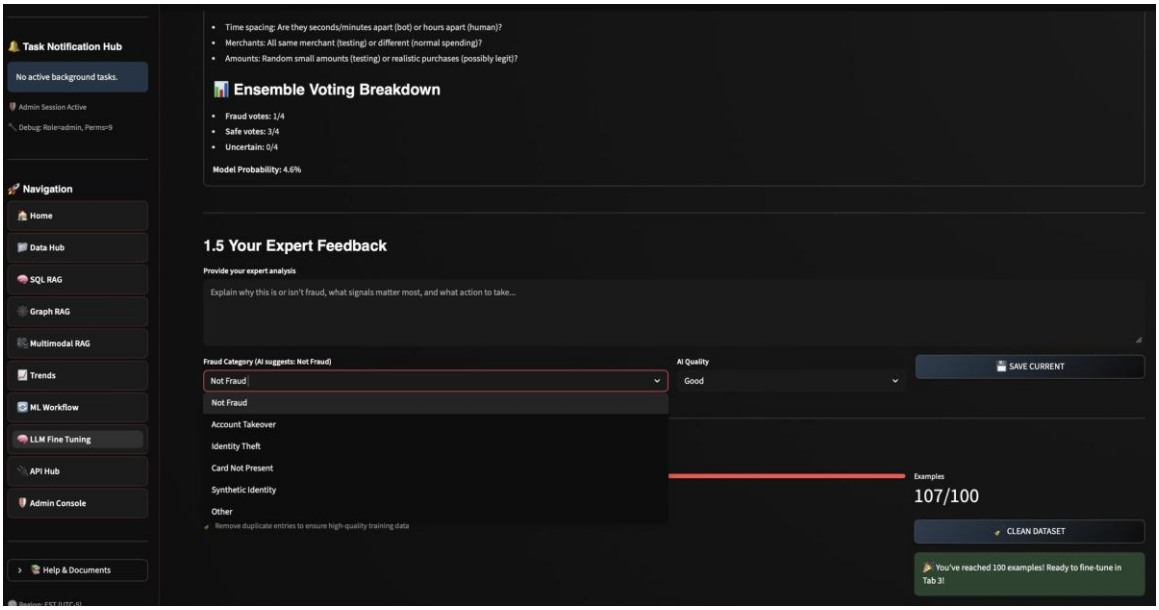
In LLM Tuning lab, there are 4 steps to do:

- Data collection
- Dataset review
- Fine tuning
- Model testing

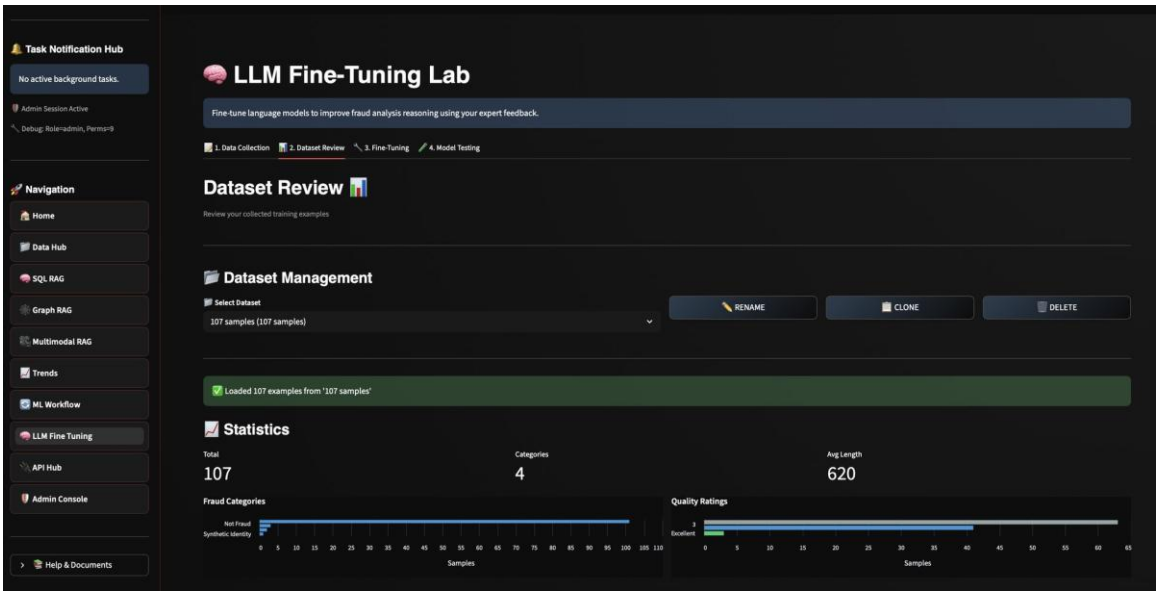
In the Data collection, user or data scientist needs to customize data entry one by one to evaluate risk degree.



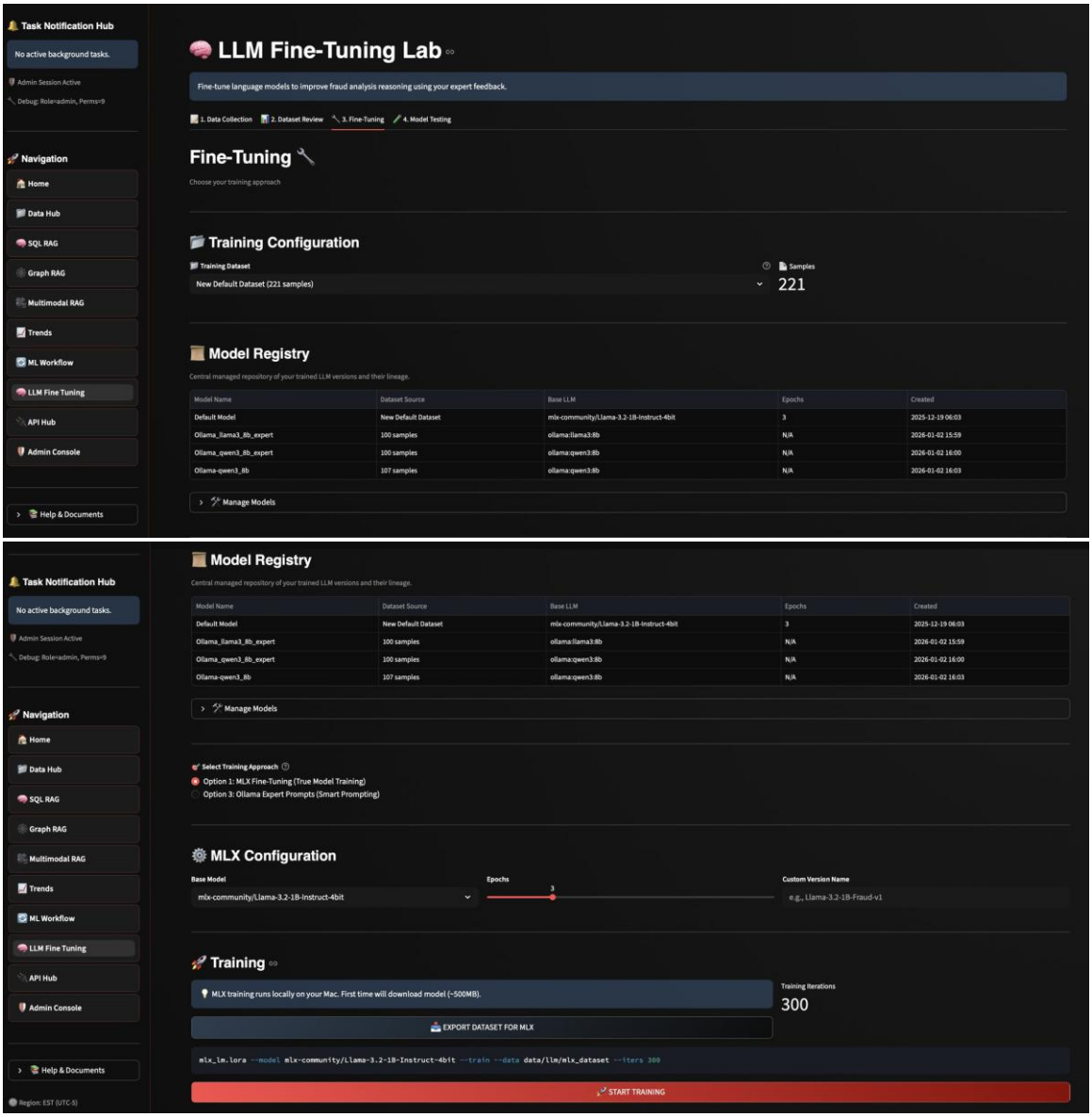
Based on selected data collection, AI gives suggestions for risk recommendations.



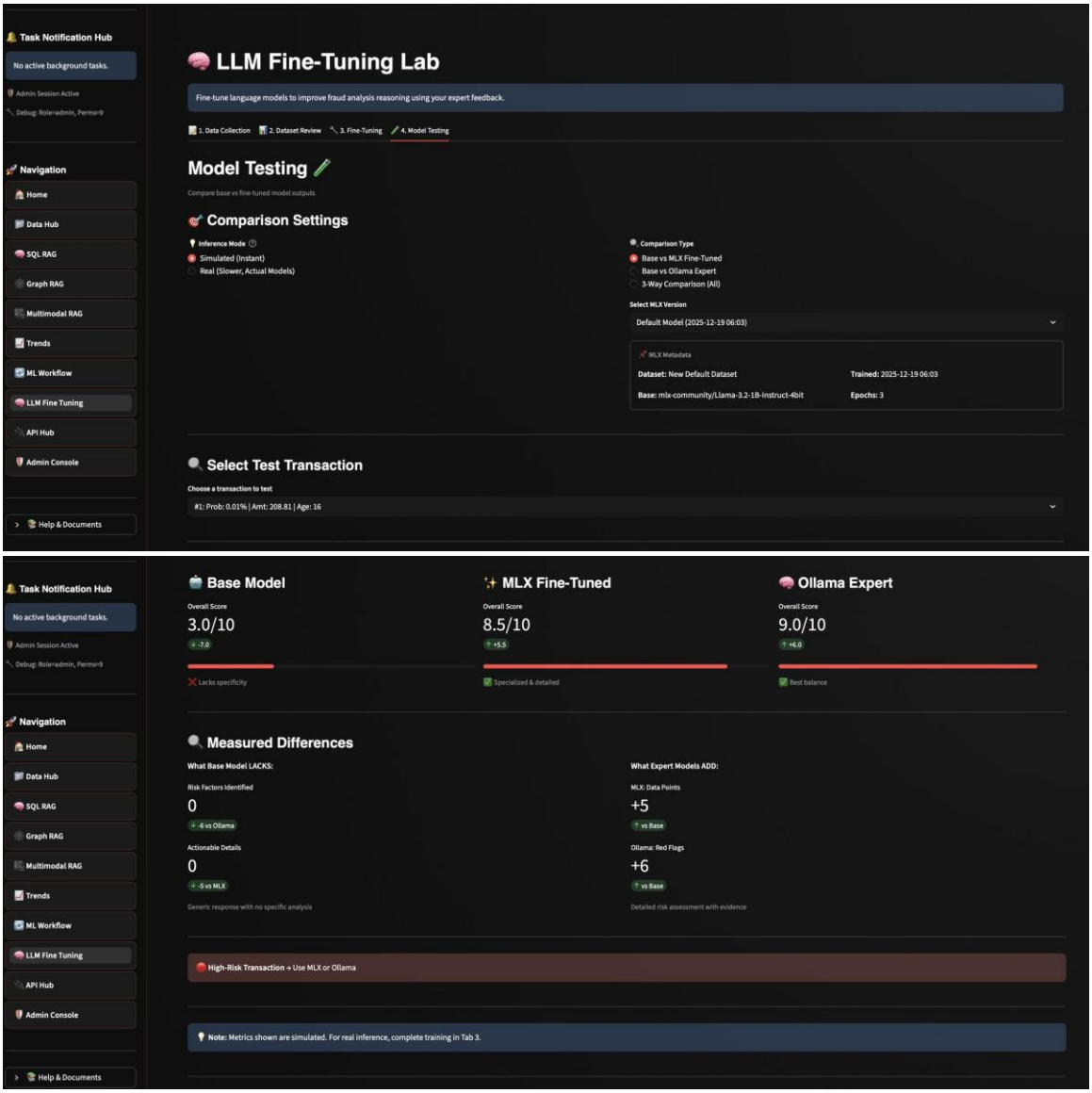
Based on knowledge, data scientist manually evaluates the risk scale and give his/her decision. Data scientist can scan data entry one by one from the data collection and give decision for fraud yes or not.



In data review, data scientist will review samples for LLM fine tuning.



User can start training LLM model. The LLM model can select local installed LLM. In the Admin-Console, administrator can install local LLM into Ollama.



After LLM has been fined tuning, it becomes a new model in the local installed LLM collection. Users can compare LLM in the different fine tuning mode. There are 3 compare modes:

- Base or original model
- MLX Fine-Tuned, which is cloud based LLM training
- Ollama Expert, which is a local LLM training

12. UI9 - API Interaction Hub

12.1 UI Design

A highly formalized developer portal aesthetic featuring terminal-dark code block structures with rapid 'Copy' clipboard logic. It seamlessly auto-embeds the native FastAPI Swagger documentation sandbox into an HTML frame.

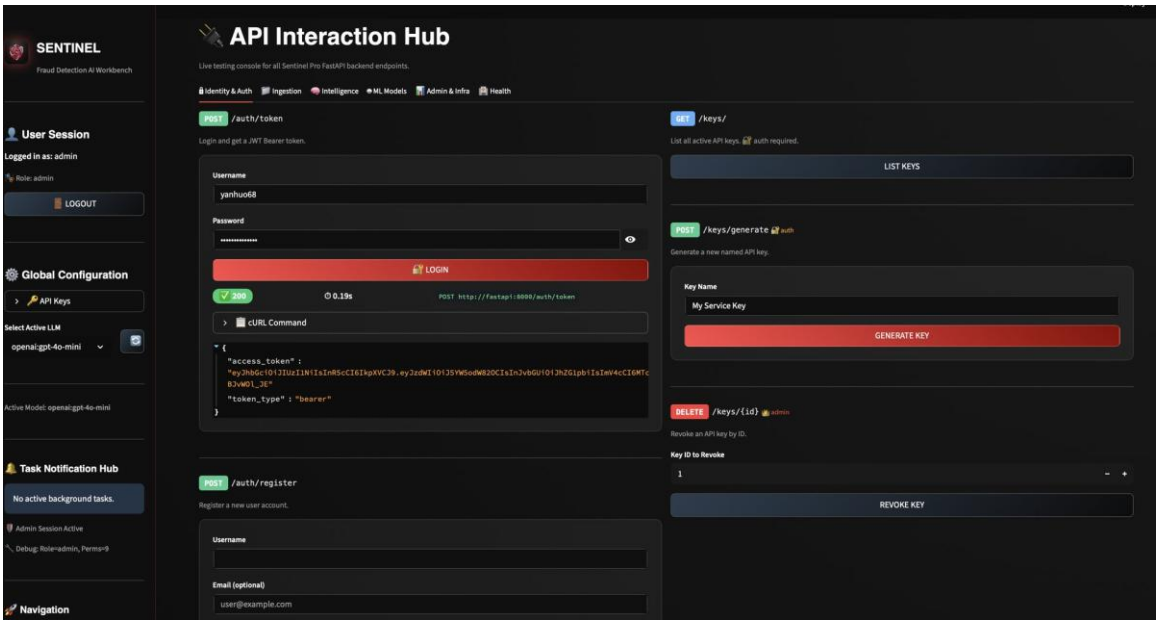
12.2 User Manual

Step 1: Administer the generation of a long-lived API execution token.
Step 2: Review the dynamically updating parameter requirements.
Step 3: Utilize the generated cURL/Python integration scripts to tie external banking infrastructure directly into Sentinel's scoring pipeline, pushing live transactions for millisecond ML evaluation.

12.3 Logic Workflow

```
▶ [ Generate Secure Request Token ]
   └──▶ [ Store Cryptographic Hash Server-Side ]
          └──▶ [ Retrieve Target Endpoint Schemas ]
                 └──▶ [ Render Integration Code Syntax ]
                        └──▶ [ Receive External Scoring JSON Post ]
```

12.4 Demo

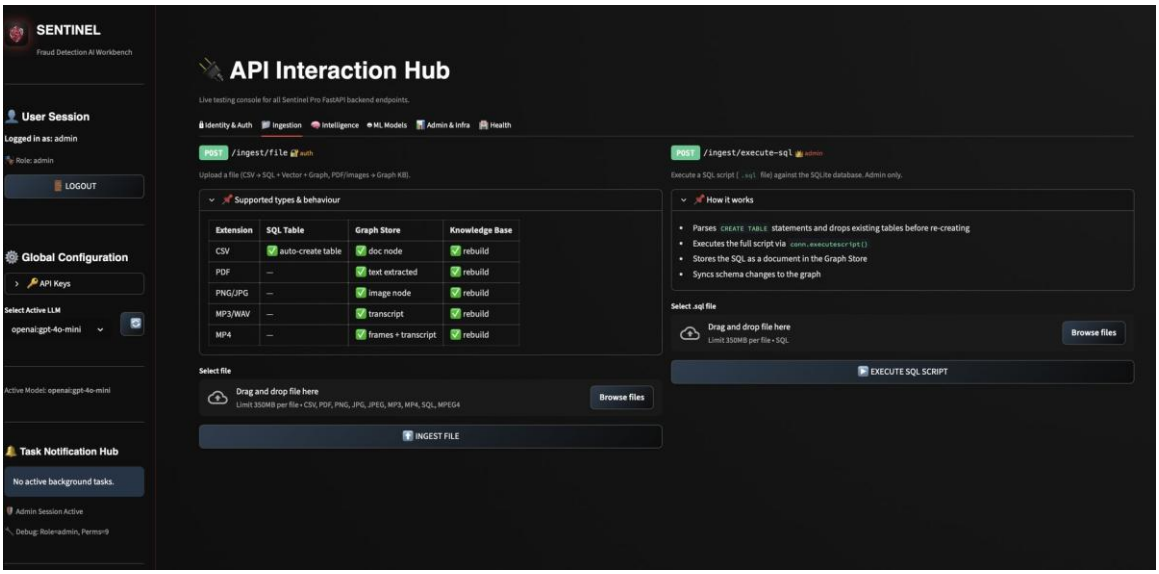


In API Interaction Hub, there are 6 tasks for user choosing:

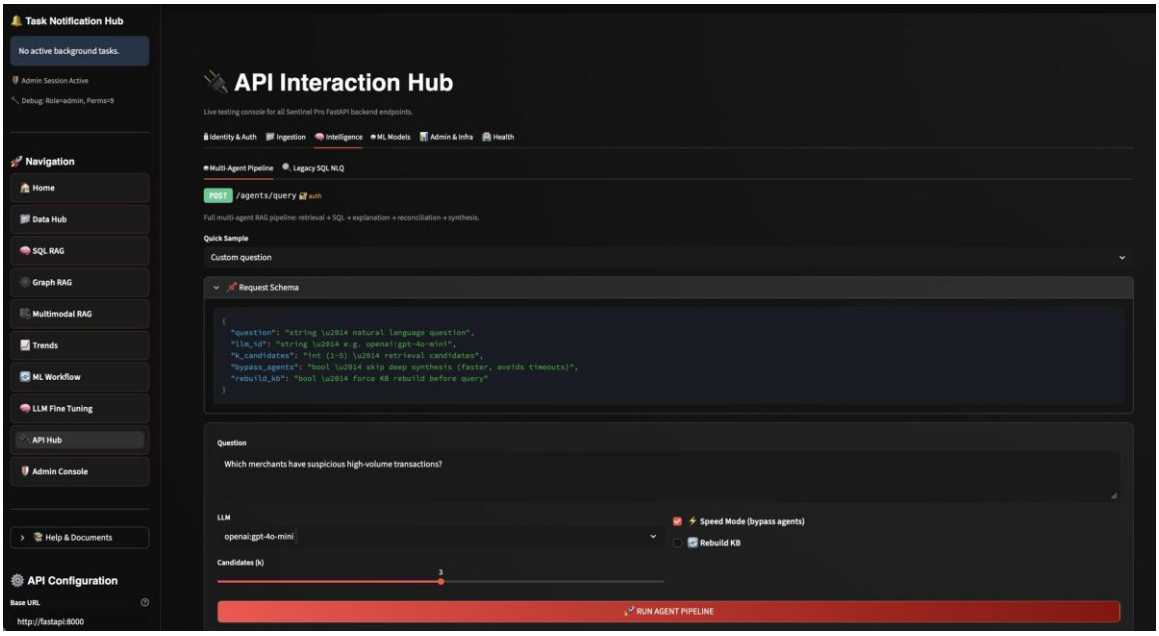
- Identity & Auth
- Ingestion
- Intelligence
- ML Models

- Admin & Infrastructure
- Health

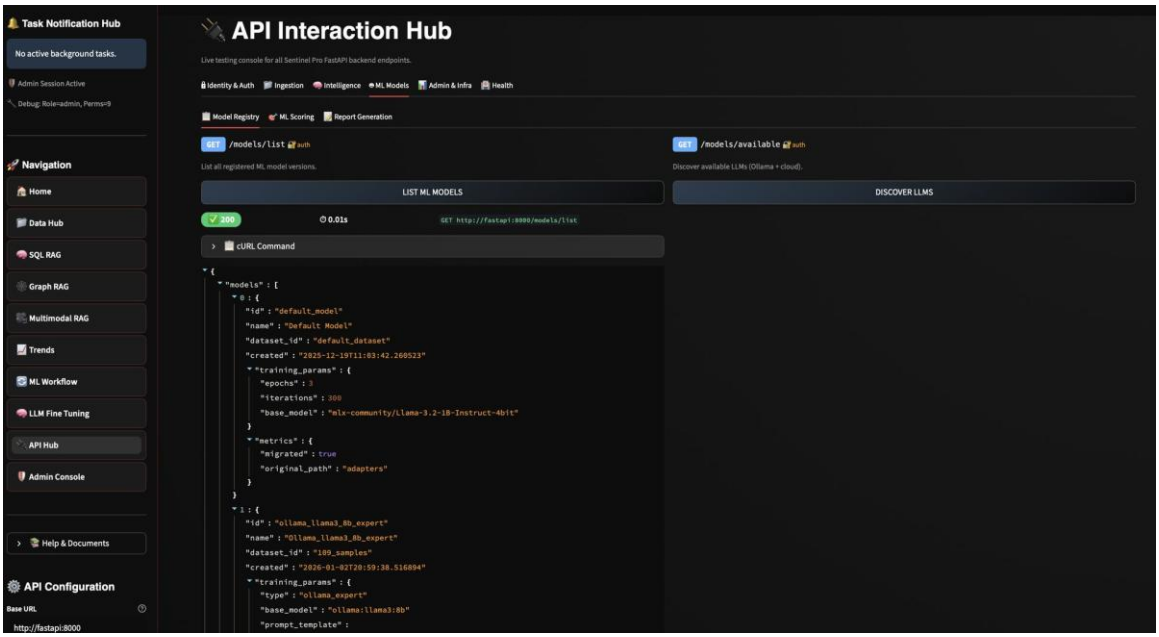
In Identity & Auth, user can test API connection for Identity or Auth. For example, for login, user simply input username and password and click “Login”, user can see cURL command and JSON response.



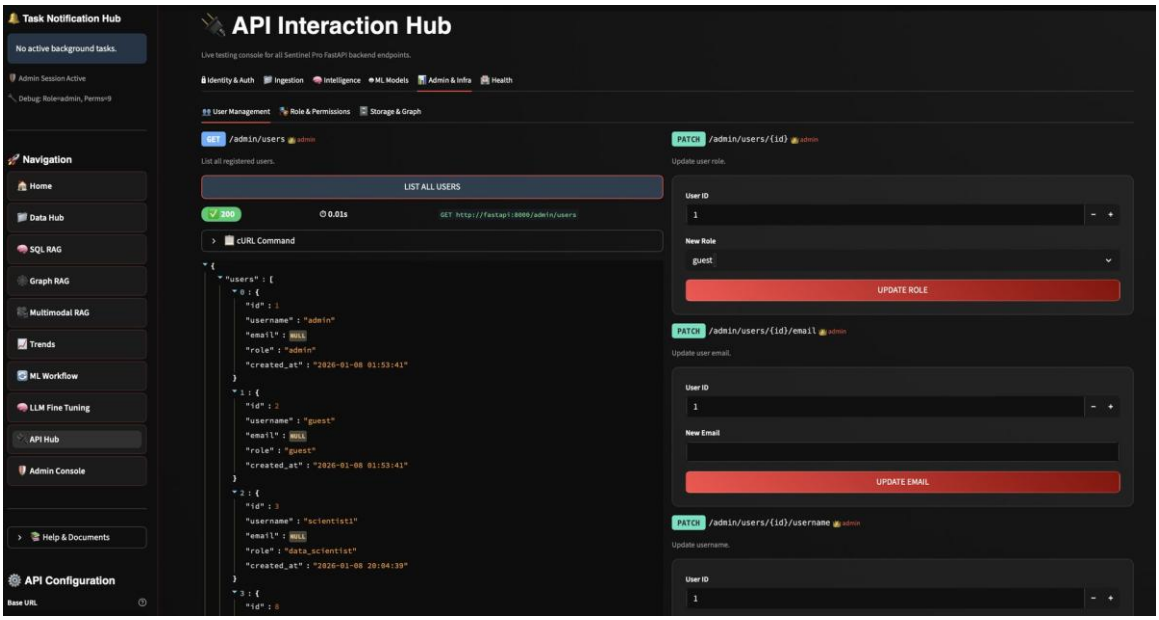
In Ingestion, users can test uploading data files or executing SQL files.



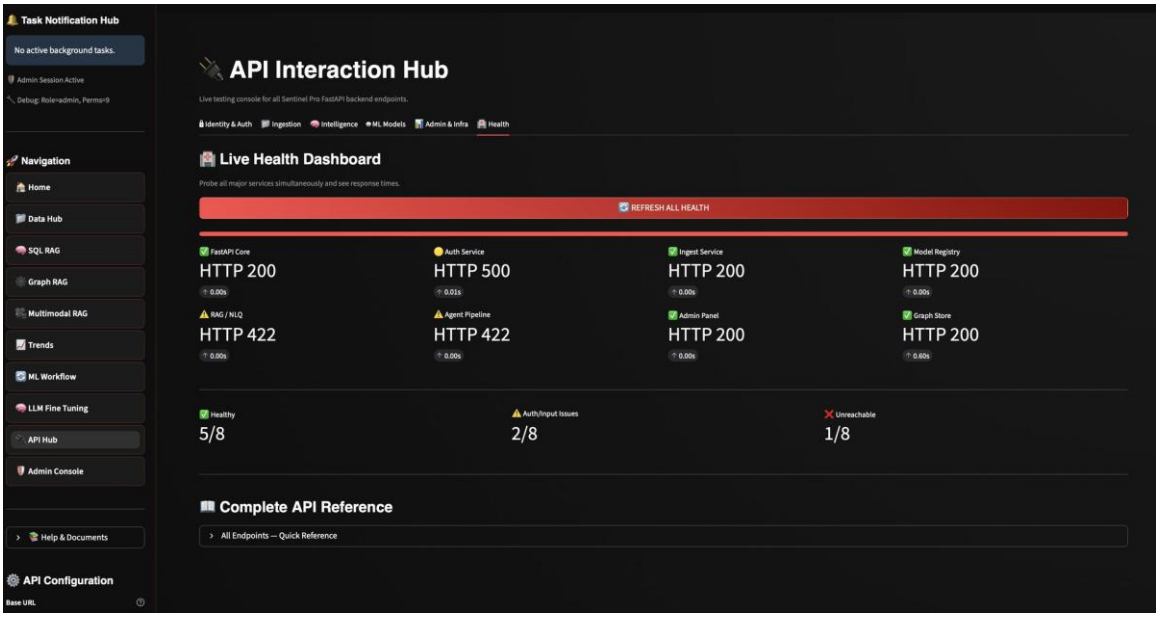
In Intelligence, users can test API connection for LLM. User simply input question to check JSON response.



In ML Models, users can test API connections for LLM, such as list ML models, or discover LLMs.



In Admin & Infra, user can test API connections for admin console and infrastructure management.



In the Health page, user can do live health check for API connections.

13. UI10 - Admin Console

13.1 UI Design

An operational control matrix exclusively protected by strict JWT RBAC scoping. Employs highly organized `st.data_editor` elements for bulk configuration mapping. All changes exhibit immediate hot-reloading behaviors across universal active sessions.

13.2 User Manual

Step 1: Only accessible by authenticated administrators.
Step 2: Provide specific OpenAI, DeepSeek, or Cloud API tokens; these map dynamically and overwrite `.env` states instantly.
Step 3: Provision new analyst user accounts, enforce mandatory password resets, and adjust viewing permissions per UI component via toggles.

13.3 Logic Workflow

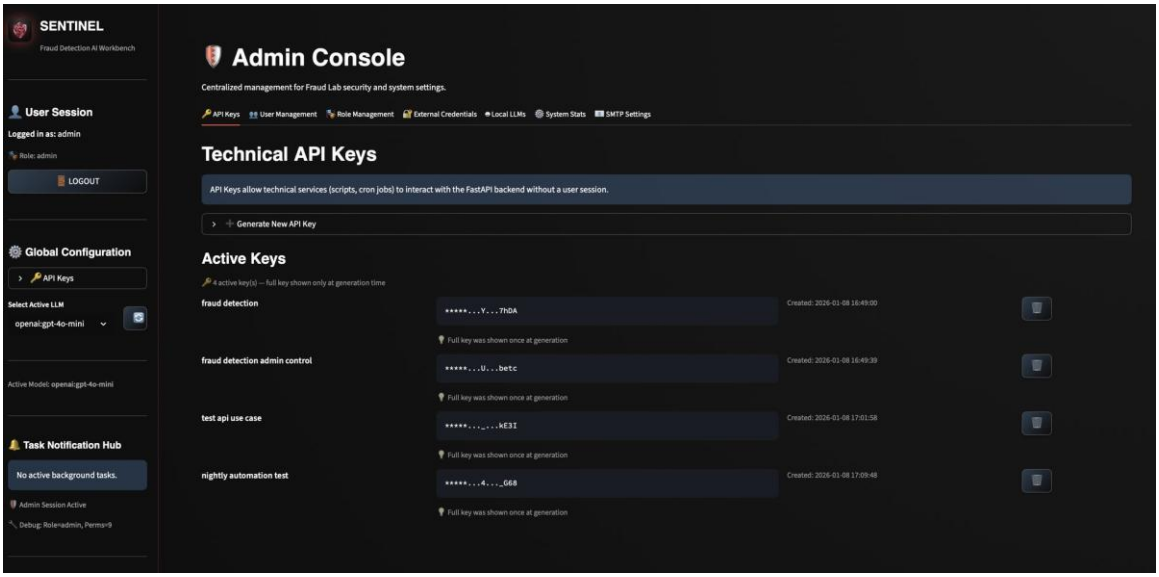
```
▶ [ Validate Admin Scope Requirements ]
  └──▶ [ Load Modular UI Tabs (Keys, Roles, SMTP) ]
         └──▶ [ Detect User Grid Modifications ]
                └──▶ [ Patch Backend Databases ]
                       └──▶ [ Force System Hot-Reload Event ]
```

13.4 Demo

In Admin Console, there are 7 tabs to manage below system settings:

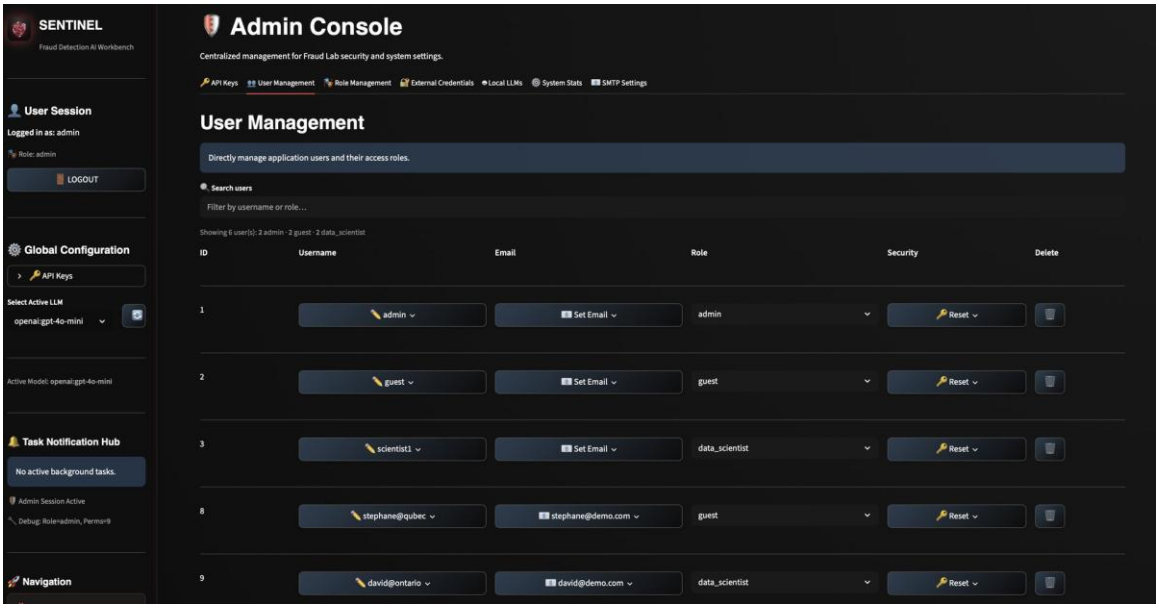
- API Keys
- User Management

- Role Management
- External Credentials
- Local LLMs
- System Stats
- SMTP Settings

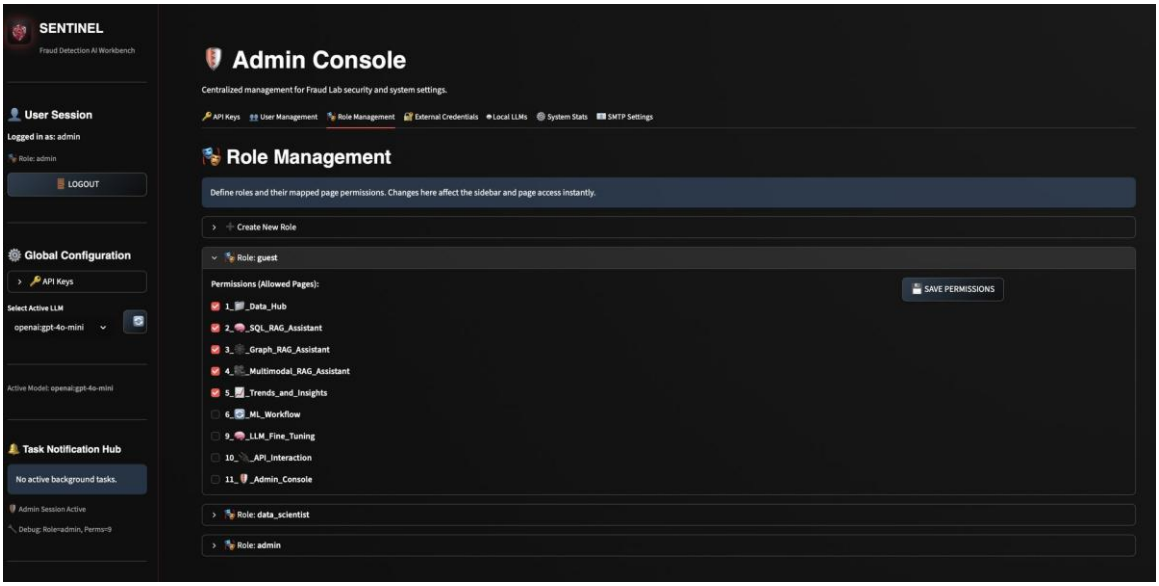


In API keys page, admin can generate new API key and delete existing API key(tokens).

ann

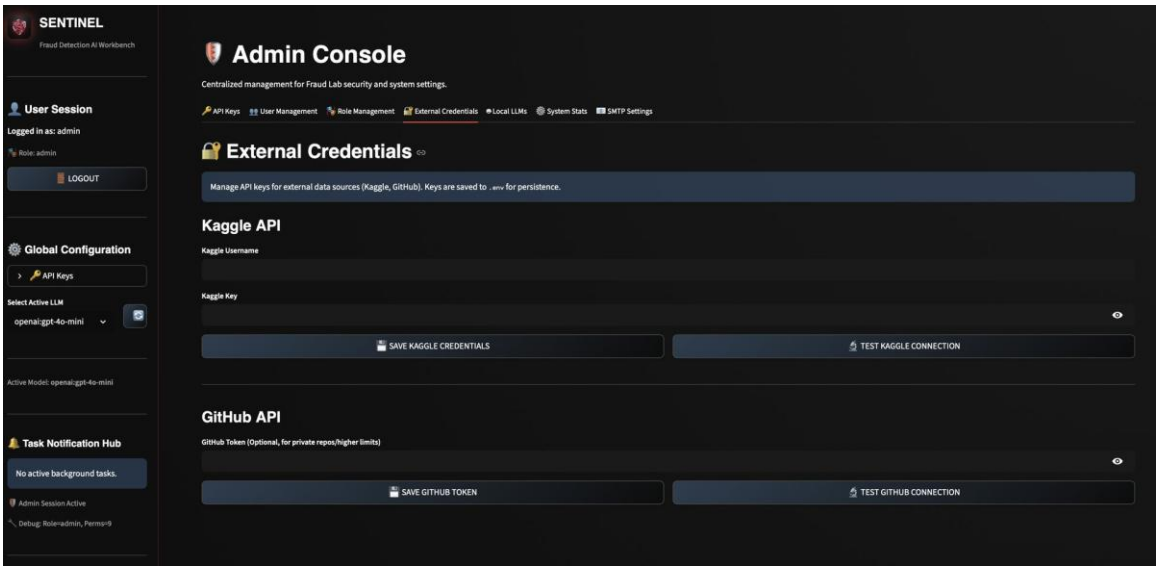


In User Management page, admin can change username, password, role and email address. Also, admin can delete user.

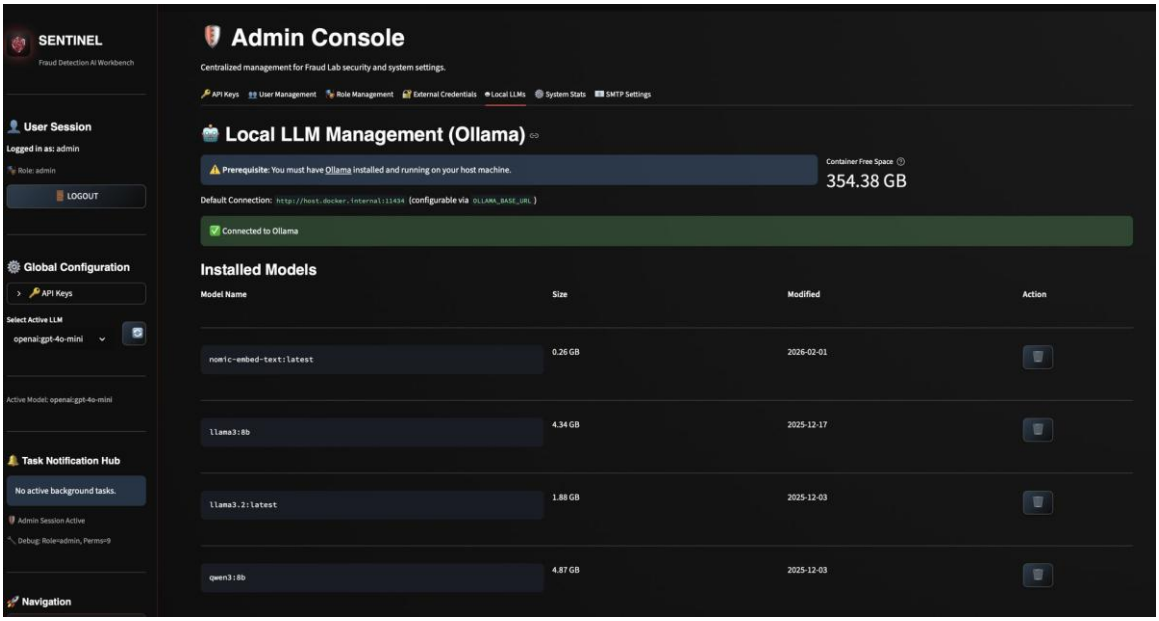


In Role Management page, admin set which page the role can access. System has 3 default roles:

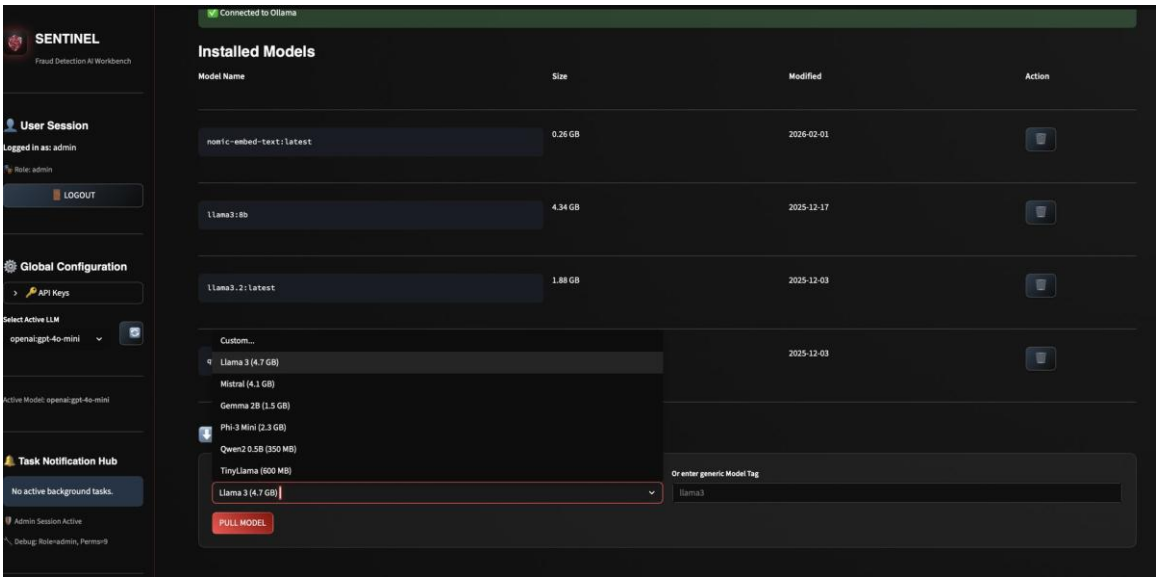
- Guest
- Data_Scientist
- Admin



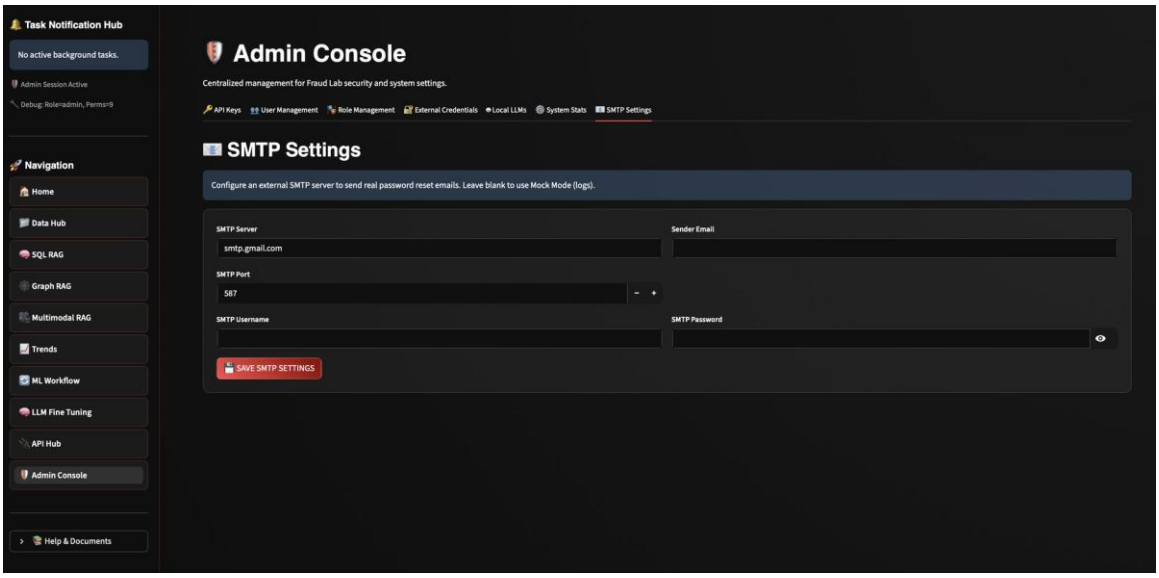
In External Credentials page, admin can set credentials for Kaggle and GitHub API access token. After completing setting, admin can save and test connection.



In Local LLMs page, admin can review and delete local installed LLMs.



Admin can install LLM with small billion parameters at the local Ollama. System default defines a list of LLMs for download and installation.



Admin can setup SMTP for sending email if user get notification for reset or forget password.

14. Show Demo

14.1 Description

A dedicated function located in the sidebar designed to present an interactive, pre-populated demonstration of the platform's capabilities without requiring user data upload.

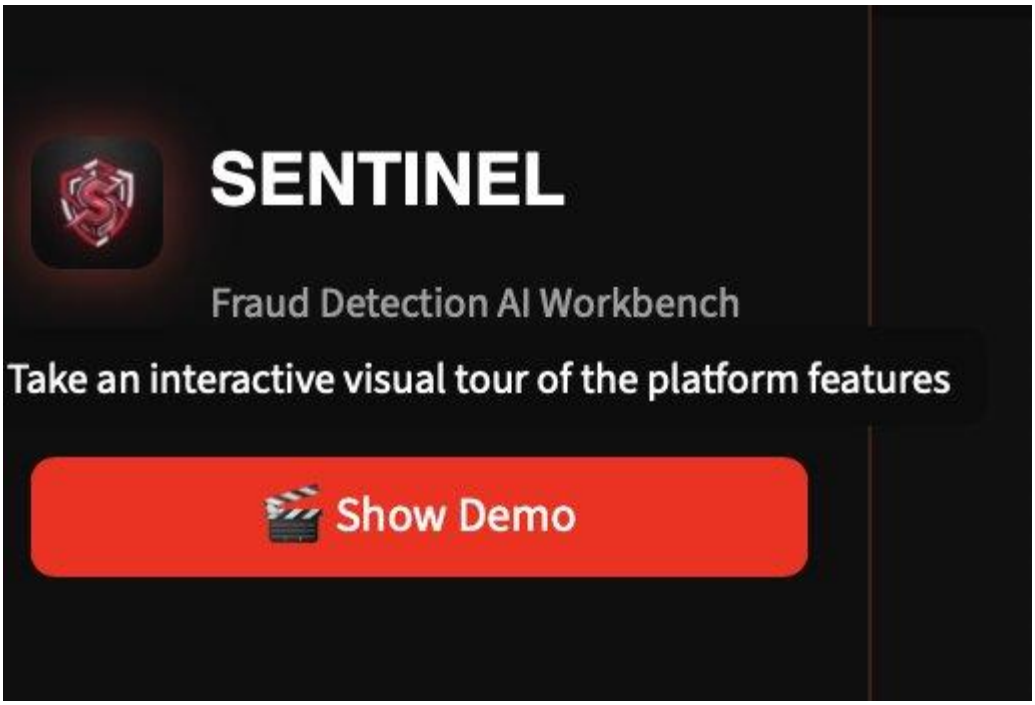
14.2 Steps

Step 1: Locate the 'Show Demo' toggle or button in the main application sidebar.
Step 2: Activate the demo mode to generate and load synthetic demonstration data into the application.
Step 3: Interact with the populated charts, tables, and graphs to explore the platform's features safely.

14.3 Impacts to the System

When activated, the Show Demo function bypasses live data fetching and database queries, injecting static synthetic payloads directly into the Session State. The system operates entirely in-memory for the demo session, temporarily suspending external API calls and persistent writes, ensuring that no experimental changes corrupt the underlying Neo4j or SQLite production data stores.

14.4 Demo



In the sidebar, there is a button to provide an interactive visual tour of the platform features.

15. Load Demo Data

15.1 Description

A privileged administrative action designed to bootstrap the system by seeding the underlying databases with a comprehensive suite of interconnected, synthetic fraud data. This allows for immediate, full-scale testing of the platform's analytical capabilities without requiring manual data ingestion.

15.2 Steps and Location

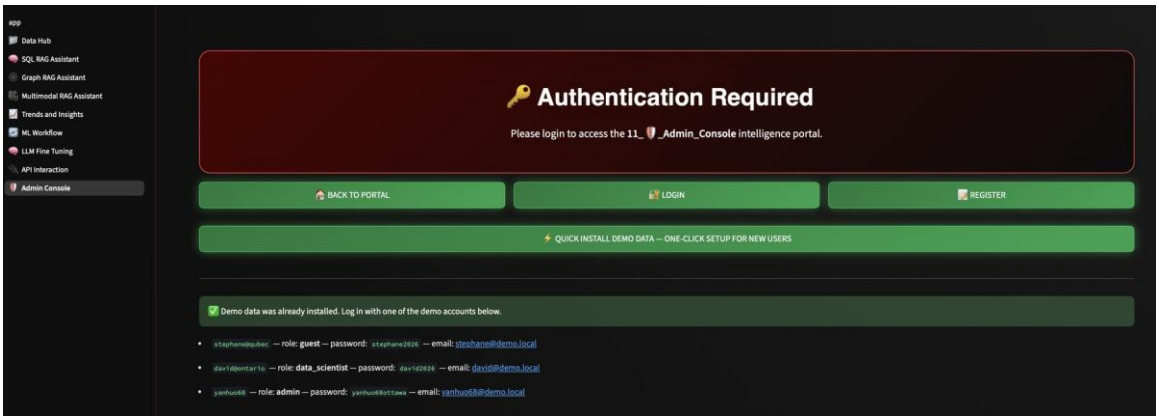
Step 1: Authenticate as a user with 'Administrator' privileges.
Step 2: Navigate to the 'Admin Console' via the main sidebar.
Step 3: Locate the 'Load Demo Data' button within the Database Management or Provisioning section of the console.
Step 4: Click the button and confirm the action to initiate the initialization pipeline.

15.3 Impacts to the System

When executed, this function triggers a multi-stage background pipeline that profoundly alters the system state:
• Relational Database (SQLite/PostgreSQL): Injects thousands of synthetic transaction records, user profiles, and account details representing typical and fraudulent activity patterns.

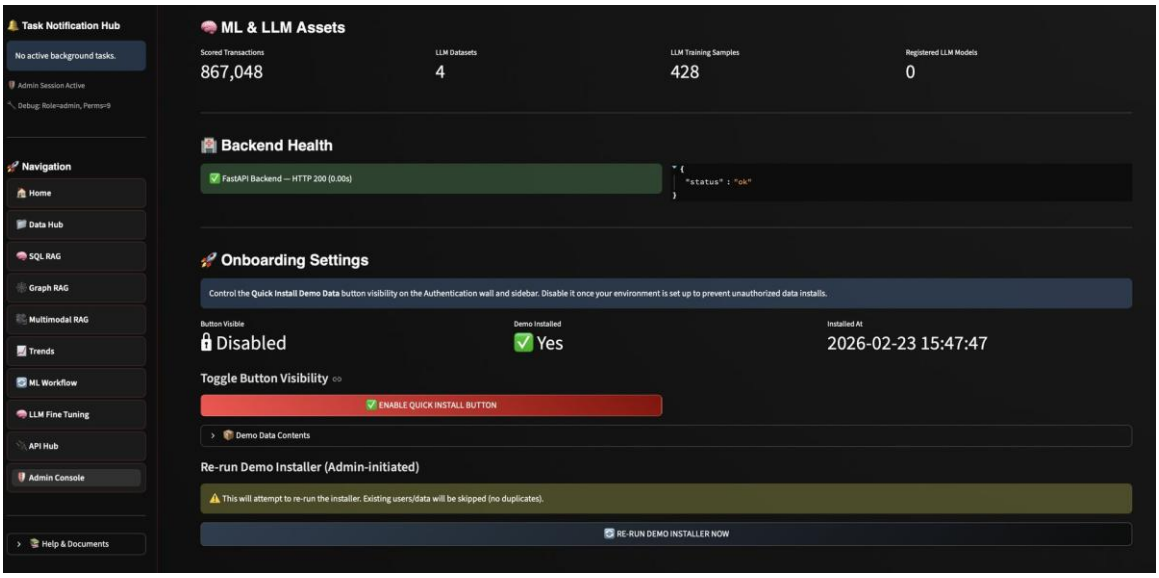
• Graph Database (Neo4j): Constructs a complex web of interconnected nodes (e.g., Users, IP Addresses, Devices, Bank Accounts) and semantic edges (e.g., 'TRANSFERRED_TO', 'LOGGED_IN_FROM'), explicitly modeling known fraud syndicates for Graph RAG analysis.
• Vector Store (FAISS): Embeds pre-generated textual scenarios, simulated case notes, and policy documents, populating the semantic index to enable context-aware similarity searches and Document RAG functionalities.
Warning: This action writes directly to persistent storage and may overwrite or conflict with existing sandbox data.

15.4 Demo



When user initial comes to system, system provides a convenient way to load pre-prepare dataset, including below:

- Create 3 users
  - Each user has a different role. The role includes guest(least privilege), data_scientist and admin.
- Upload fraud-detect-dataset.csv
- Execute 4 relational tables into the SQLite



In the Admin Console, admin can reset Load Demo button enabled after user has already clicked this button.

16. Dependency References

16.1 FastAPI

Utilized as the core asynchronous backend API router. Chosen for its automatic OpenAPI integration, high throughput, and strict Pydantic data validation model.

16.2 Streamlit

The primary frontend reactive visualization framework. Chosen to accelerate ML tool delivery by eliminating complex React/Vue boilerplate while maintaining interactive graphs.

16.3 SQLAlchemy

An Object Relational Mapper (ORM). Facilitates database-agnostic modeling, ensuring Sentinel can map securely across simple flat SQLite files or enterprise PostgreSQL clusters.

16.4 Neo4j (urllib3/neo4j-driver)

A dedicated graph database. Integral for calculating multi-hop relationships (fraud rings) that deeply struggle in standard row-based SQL structures.

16.5 FAISS

Facebook AI Similarity Search. A highly optimized C++ vector library wrapped in Python enabling instantaneous semantic similarity retrievals across billions of embedded document chunks.

16.6 SQLGlot

An incredibly efficient SQL parser. Necessary for parsing dangerous LLM-generated strings into AST representations to programmatically guarantee the absence of destructive DROP/ALTER statements before engine execution.

16.7 PyVis / NetworkX

Python-native graph visualization suites. Translates raw underlying neo4j edge coordinates into human-readable, physics-simulated node clusters seamlessly overlaid in Streamlit.

16.8 BCrypt

A computationally heavy cryptographic hashing algorithm explicitly chosen to counteract brute-force hardware attacks when storing registered investigator passwords.

16.9 Ollama

An underlying local inference engine wrapper. Enables Sentinel to rapidly pull, mount, and execute quantized open-weight models locally safely entirely off-network.

17. Glossary

RAG (Retrieval-Augmented Generation):

A methodology where an LLM is paired with an external knowledge retrieval mechanism (like Vector DBs) to ground its responses in factual, enterprise-specific context, eliminating hallucination.

Knowledge Graph:

A structured taxonomy of concepts and relations (Nodes and Edges). Essential for depicting complex fraud rings where individuals, IP addresses, and bank accounts overlap.

LLM (Large Language Model):

A massive neural network trained on vast quantities of text. Capable of translation, summarization, logical bridging, and code compilation (e.g., text-to-SQL logic).

Fine-Tuning (LoRA):

Low-Rank Adaptation. A computationally efficient mathematical shortcut to retrain a massive LLM on specific behavior without altering its fundamental parameter structure.

RBAC (Role-Based Access Control):

A security paradigm where access logic is denied inherently unless explicitly permitted via a matrix mapped to the authenticated user's registered job function.

JWT (JSON Web Token):

A cryptographically signed, compact URL-safe execution credential passed between the client and server to verify session integrity asynchronously.

Vector Embeddings:

Array structures representing the profound semantic meaning of text/images as mathematical distances, allowing search algorithms to identify context similarities regardless of explicit keyword matches.

SHAP (SHapley Additive exPlanations):

A game-theoretic ML model evaluation technique that quantifies and visually explains the exact logic path an algorithmic decision tree utilized to calculate a fraud-score output.