\*\*Product Requirements Document: Medicaid Audit Report Analyzer\*\*

\*\*Version:\*\* 1.0

\*\*1. Introduction\*\*

\* \*\*1.1. Purpose:\*\* This document outlines the product requirements for the "Medicaid Audit Report Analyzer," a web application designed to help Medicaid auditors, program managers, and the public to summarize, aggregate, review, and query Medicaid-related audit reports.

\* \*\*1.2. Scope:\*\*

\* \*\*In Scope (Version 1.0):\*\*

\* Admin functionality for uploading PDF audit reports.

\* AI-powered structured data extraction from PDFs.

\* Admin review and editing of AI-extracted data.

\* Persistent storage of extracted data and PDF metadata in a relational database.

\* Mechanism for report deduplication.

\* RAG (Retrieval Augmented Generation) system:

\* Ingestion pipeline for parsing, chunking, and embedding report content.

\* Admin review of the chunking process.

\* Storage of vector embeddings.

\* Vector database viewer for admin inspection.

\* Publicly accessible RAG chat interface for querying the report corpus using semantic and BM25 search with reranking.

\* Display of AI-extracted keywords with counts, allowing users to filter reports by keyword.

\* Display of admin-curated "Featured Reports."

\* \*\*Out of Scope (Version 1.0):\*\* See Section 9.

\* \*\*1.3. Vision:\*\* To create an efficient and insightful tool that provides a bird’s-eye view of the Medicaid program landscape by making audit reports easily accessible, searchable, and understandable. The tool aims to help users stay up-to-date with current issues, identify trends, and target areas that might need special attention within the vast Medicaid program.

\*\*2. Goals & Objectives\*\*

\* \*\*2.1. Primary Goal:\*\*

\* Reduce the manual effort and time spent extracting consistent key information from diverse audit report PDFs.

\* Enable users to quickly find relevant information across a corpus of reports to identify trends, precedents, or specific details without re-reading entire documents.

\* Provide a centralized, queryable repository of public Medicaid audit report data.

\* \*\*2.2. Success Metrics:\*\*

\* High user engagement (measured by site usage analytics for the public RAG feature).

\* Admin efficiency in processing reports (measured qualitatively by the admin).

\* Accuracy and consistency of AI-extracted data (evaluated by the admin).

\* Number of reports successfully processed and ingested into the system.

\*\*3. Target Audience\*\*

\* \*\*3.1. Administrator (Single User - "You"):\*\*

\* \*\*Needs:\*\* Efficiently upload, process, and manage audit reports. Ensure data quality of AI extractions. Manage the RAG ingestion pipeline. Curate content (e.g., featured reports).

\* \*\*Technical Proficiency:\*\* High (developer/owner of the tool).

\* \*\*3.2. Public Users (e.g., Medicaid Program Managers, Auditors (from various agencies), Researchers, Journalists, General Public):\*\*

\* \*\*Needs:\*\* Easily search and query a large corpus of Medicaid audit reports. Understand key findings and trends without deep technical expertise in AI or data processing. Access structured summaries and original documents.

\* \*\*Technical Proficiency:\*\* Moderately proficient in technology and website use. Expect an intuitive, user-friendly interface.

\* \*\*User Count (Year 1 Estimate):\*\* A few hundred.

\*\*4. User Stories / Key Use Cases\*\*

\* \*\*4.1. Administrator:\*\*

\* As an Admin, I want to upload one or more PDF audit reports so that they can be processed by the system.

\* As an Admin, I want the system to use an AI model (e.g., OpenAI, Gemini) to extract pre-defined structured data from the uploaded PDFs.

\* As an Admin, I want to choose which AI model is used for extraction to compare performance or cost.

\* As an Admin, I want to review the AI-extracted structured data alongside the original PDF content before saving.

\* As an Admin, I want to edit, add, or delete any AI-extracted data to ensure accuracy and completeness.

\* As an Admin, I want the system to prevent duplicate report entries based on file hash and filename, offering options to replace or cancel.

\* As an Admin, I want the extracted data and PDF metadata to be saved transactionally to a relational database, ensuring data integrity.

\* As an Admin, I want to initiate the RAG ingestion process for uploaded reports, which includes parsing, chunking, and embedding.

\* As an Admin, I want to review the text chunks generated before they are embedded for the RAG system.

\* As an Admin, I want to view the contents of the vector database (chunks, associated metadata).

\* As an Admin, I want to mark specific reports as "Featured" to highlight them on the public website.

\* As an Admin, I want AI-extracted keywords to be stored, and I want to be able to review and curate these keywords.

\* As an Admin, I want to store AI processing metadata (model used, tokens, cost, time) for each extraction.

\* \*\*4.2. Public User:\*\*

\* As a Public User, I want to use a chat interface to ask natural language questions about the content of the Medicaid audit reports.

\* As a Public User, I want the RAG system to provide answers based on semantic search and BM25, with reranking for relevance.

\* As a Public User, I want the RAG answers to cite the source reports (and page numbers if possible) so I can verify the information.

\* As a Public User, I want to see the relevant text chunks that informed the RAG's answer.

\* As a Public User, I want to view a list of "Featured Reports" on the homepage.

\* As a Public User, I want to see a list of keywords extracted from reports, along with the count of reports associated with each keyword.

\* As a Public User, I want to click on a keyword and see a list of all reports tagged with that keyword.

\* As a Public User, I want to view the detailed structured summary (objectives, findings, recommendations, etc.) for any individual report.

\* As a Public User, I want to access the original PDF document for any report in the system.

\*\*5. Functional Requirements\*\*

\* \*\*5.1. PDF Ingestion & Processing (Admin):\*\*

\* 5.1.1. PDF Upload: Admin can upload single or multiple machine-readable PDF files.

\* 5.1.2. Deduplication:

\* System checks for duplicates based on PDF file content hash. If exact match, admin can cancel or replace.

\* If filename matches an existing report but content is different, warn admin and allow proceeding as a new report or canceling.

\* 5.1.3. PDF Parsing: System extracts text content from PDFs.

\* 5.1.4. PDF Storage: Original PDF files are stored securely (e.g., cloud storage or Replit filesystem for V1).

\* \*\*5.2. AI Structured Data Extraction (Admin):\*\*

\* 5.2.1. Model Selection: Admin can select from configured LLMs (e.g., OpenAI, Gemini) for extraction.

\* 5.2.2. Data Extraction: AI extracts predefined structured data fields (see Section 6.1) using Pydantic models with `instructor` library for schema enforcement. This includes lists for findings, recommendations, and objectives.

\* 5.2.3. Keyword Extraction: AI suggests relevant keywords from the report content.

\* 5.2.4. AI Metadata Logging: System logs model name, token counts, costs, and processing time for each extraction.

\* \*\*5.3. Review & Edit Interface (Admin):\*\*

\* 5.3.1. Display: Side-by-side view (or similar) of PDF content and AI-extracted fields.

\* 5.3.2. Editing: Admin can edit, add, or delete text in any extracted field. Lists (findings, recommendations, objectives, keywords) allow item-level editing, addition, or deletion.

\* 5.3.3. PDF Source Highlighting (Nice-to-have V1): If feasible with the chosen LLM, highlight the source text in the PDF corresponding to an extracted field.

\* \*\*5.4. Data Persistence (Admin):\*\*

\* 5.4.1. Transactional Save: All parts of a report's structured data (report metadata, findings, recommendations, objectives, keywords, AI log) are saved to the database in a single transaction. If any part fails, the entire operation is rolled back.

\* 5.4.2. Database Storage: Data stored in a PostgreSQL database as per the defined schema (see Section 6.3).

\* \*\*5.5. RAG System - Ingestion (Admin):\*\*

\* 5.5.1. Batch Processing: Option to process single or multiple reports for RAG ingestion.

\* 5.5.2. Chunking: Reports are parsed and divided into text chunks (target: 500 tokens with 200 token overlap, configurable).

\* 5.5.3. Chunk Review: Admin can review the generated text chunks per report before embedding. No manual editing of chunks in V1, only review.

\* 5.5.4. Embedding: Approved chunks are converted into vector embeddings using a chosen model.

\* 5.5.5. Vector Storage: Embeddings and associated metadata (report ID, page number, chunk text) are stored in a vector database (ChromaDB file-based for V1).

\* \*\*5.6. RAG System - Retrieval & Chat (Public User):\*\*

\* 5.6.1. Query Input: Users can submit natural language queries via a chat interface.

\* 5.6.2. Hybrid Search: System uses a combination of semantic search and BM25 keyword search.

\* 5.6.3. Reranking: A reranker model refines search results before feeding to the LLM.

\* 5.6.4. LLM Response Generation: An LLM generates a synthesized answer based on retrieved chunks.

\* 5.6.5. Source Citation: Answers must cite source reports (title, link to detail page, page number if available).

\* 5.6.6. Retrieved Chunk Display: Users can view the source text chunks that informed the AI's answer.

\* 5.6.7. No Chat History (V1): Each query is treated as a new, standalone interaction.

\* \*\*5.7. Public Website Features:\*\*

\* 5.7.1. Homepage: Displays Featured Reports, Top Keywords, and RAG search bar.

\* 5.7.2. Featured Reports Display: Shows admin-curated reports with title, audit organization, and overall conclusion snippet.

\* 5.7.3. Keyword Display & Filtering:

\* Displays keywords with report counts (e.g., tag cloud or list).

\* Allows users to click a keyword to view a list of associated reports.

\* 5.7.4. Report List Page: Displays reports filtered by keyword or other criteria (e.g., "Browse All"). Includes pagination.

\* 5.7.5. Report Detail Page: Displays all structured extracted data for a single report, associated keywords, and a link to the original PDF.

\*\*6. Data Requirements\*\*

\* \*\*6.1. Key Structured Data Fields for AI Extraction (per report):\*\*

\* `report\_title` (Text)

\* `audit\_organization` (Text)

\* `publication\_year` (Integer)

\* `publication\_month` (Integer)

\* `publication\_day` (Integer, Optional)

\* `objectives` (List of Text - each objective as a separate item)

\* `findings` (List of Text - each finding as a separate item, potentially with associated `financial\_impact\_for\_finding` (Numeric))

\* `recommendations` (List of Text - each recommendation as a separate item, optionally linked to a specific finding)

\* `overall\_conclusion` (Text)

\* `llm\_insight` (Text - AI-generated summary/insight)

\* `potential\_objective\_summary` (Text - AI-generated summary of objectives)

\* `original\_report\_source\_url` (Text, Optional)

\* `state` (Text, e.g., "NY", "CA")

\* `total\_financial\_impact` (Numeric - for the whole report)

\* `audit\_period\_start\_date` (Date)

\* `audit\_period\_end\_date` (Date)

\* `audit\_period\_description` (Text)

\* `extracted\_keywords` (List of Text)

\* \*\*6.2. PDF & System Metadata to Store:\*\*

\* `original\_filename` (Text)

\* `file\_hash` (VARCHAR(64) - SHA256)

\* `pdf\_storage\_path` (Text)

\* `file\_size\_bytes` (Integer)

\* `featured` (Boolean - Admin controlled)

\* `status` (VARCHAR - e.g., 'published', 'processing')

\* `processing\_status` (VARCHAR - e.g., 'pending', 'extracted')

\* `error\_message` (Text - if processing fails)

\* `created\_at`, `updated\_at` (Timestamps)

\* \*\*6.3. AI Processing Metadata (per extraction event):\*\*

\* `model\_name` (Text)

\* `input\_tokens`, `output\_tokens`, `total\_tokens` (Integer)

\* `input\_cost`, `output\_cost`, `total\_cost` (Numeric)

\* `processing\_time\_ms` (Integer)

\* `extraction\_status` (VARCHAR - e.g., 'SUCCESS', 'FAILURE')

\* `error\_details` (Text)

\* \*\*6.4. Database Schema Overview:\*\*

\* The system will use a PostgreSQL relational database. Key tables include:

\* `reports`: Main report metadata, AI-extracted summary info, file info.

\* `report\_objectives`: Stores individual objectives linked to a report.

\* `report\_findings`: Stores individual findings linked to a report, including specific financial impacts.

\* `report\_recommendations`: Stores individual recommendations, linked to a report and optionally to a finding.

\* `keywords`: Stores unique keyword strings.

\* `report\_keywords`: Junction table linking reports and keywords.

\* `ai\_processing\_log`: Logs metadata for each AI extraction attempt.

\* `vector\_chunks`: Stores text chunks, report ID, page number, and other metadata for RAG. (Actual vector embeddings managed by ChromaDB in V1).

\* `system\_activity\_log`: Logs important admin/system actions.

\* Relationships are enforced via foreign keys. Transactional integrity is critical for saves.

\* \*Detailed schema (SQL CREATE TABLE statements) is defined in a separate technical document.\*

\*\*7. Non-Functional Requirements\*\*

\* \*\*7.1. Usability:\*\*

\* Admin interface should be efficient for core tasks (upload, review, save).

\* Public interface must be intuitive, easy to navigate, and provide clear information.

\* Modern, clean, uncluttered UI design.

\* \*\*7.2. Performance:\*\*

\* Admin processing: AI extraction time is dependent on LLM, but system overhead should be minimized.

\* Public RAG queries should return results within a reasonable timeframe (e.g., < 5-10 seconds for typical queries).

\* Website pages should load quickly.

\* \*\*7.3. Security:\*\*

\* Standard web application security best practices (e.g., protection against common vulnerabilities like XSS, CSRF if forms are used beyond basic search).

\* Secure management of API keys and database credentials.

\* Since reports are public and no user accounts for public access, HIPAA compliance for this data is not a concern. If private data were ever introduced, this would change significantly.

\* \*\*7.4. Scalability (Initial):\*\*

\* System should handle a few hundred concurrent public users for RAG queries.

\* Database should handle storage and querying for thousands of reports.

\* Designed with potential to scale components (e.g., move from file-based ChromaDB to a managed vector DB if needed).

\* \*\*7.5. Maintainability:\*\*

\* Code should be well-documented, modular, and follow Python/TypeScript best practices.

\* Clear separation between backend, frontend, and AI services.

\* \*\*7.6. Reliability/Availability:\*\*

\* The public-facing site should aim for high availability.

\* Admin processing should be robust, with clear error handling and logging.

\*\*8. User Interface (UI) & User Experience (UX) Concepts\*\*

\* \*\*8.1. Overall Feel:\*\* Modern, clean, professional, trustworthy. Emphasis on clarity and ease of information discovery.

\* \*\*8.2. Key Public Pages:\*\*

\* \*\*Homepage:\*\* Prominent RAG search bar, section for "Featured Reports" (cards with title, org, conclusion snippet), section for "Top Keywords" (clickable pills/tags with counts).

\* \*\*Report List Page:\*\* Displays reports filtered by keyword (or other criteria). Options for sorting. Reports shown as list items or cards. Pagination.

\* \*\*Report Detail Page:\*\* Full structured data for a report (title, metadata, objectives, findings, recommendations, conclusion, LLM insight), associated keywords, link to original PDF.

\* \*\*RAG Chat Interface:\*\* Input field for questions, display area for conversation (question/answer format), clear source citations in answers, option to view supporting chunks.

\* \*\*8.3. Admin Interface (Initial - Potentially CLI/Scripts, evolving to Web UI):\*\*

\* Functionality for PDF upload.

\* Review/edit screen for AI extractions (side-by-side PDF and structured fields).

\* Interface to trigger RAG ingestion and review chunks.

\* Vector DB viewer (listing chunks, metadata).

\*\*9. Out of Scope for Version 1.0 / Future Considerations\*\*

\* \*\*Out of Scope for V1.0:\*\*

\* User accounts, login, or personalization for public users.

\* Advanced analytics dashboards or report generation features.

\* User annotations or commenting on reports.

\* Direct integration with other external state systems.

\* Sophisticated workflow management for report review beyond single admin.

\* OCR for scanned (non-machine-readable) PDFs.

\* Automated RAG chunk editing (only review in V1).

\* Conversation history for the public RAG chat.

\*\*10. Technology Stack (Initial Proposal)\*\*

\* \*\*Backend:\*\* Python (FastAPI recommended)

\* \*\*Frontend:\*\* TypeScript (React, Vue, or Angular)

\* \*\*Database:\*\* PostgreSQL (Cloud-hosted, e.g., Neon, Supabase)

\* \*\*AI Models:\*\* OpenAI APIs (GPT-4, etc.), Google Gemini APIs.

\* \*\*AI Orchestration/Tooling:\*\* `instructor` library, LlamaIndex.

\* \*\*Vector Database:\*\* ChromaDB (file-based for V1, managed by LlamaIndex).

\* \*\*PDF Parsing:\*\* PyMuPDF/fitz or pypdf2.

\* \*\*Deployment (Initial):\*\* Replit (for application hosting) + Cloud PostgreSQL. PDF storage via Replit filesystem or basic cloud object storage.

---

**I. Core UI Principles for a Modern Feel:**

1. **Clarity & Simplicity:** Information should be easy to find and understand. Avoid clutter.
2. **Visual Hierarchy:** Guide the user's eye to the most important information first.
3. **Intuitive Navigation:** Users shouldn't have to think hard about how to get around.
4. **Responsiveness:** The site must look good and work well on desktops, tablets, and mobile devices.
5. **Performance:** Fast loading times are crucial.
6. **Professional Aesthetics:** Clean typography, consistent styling, and a thoughtful color palette.

**II. Key Pages/Sections and UI Elements:**

**A. Homepage / Main Landing Page**

* **Purpose:** Introduce the site, provide immediate value (RAG search, featured content), and offer pathways to explore data.
* **Layout Ideas:**
  + **Top Navigation Bar (Sticky):**
    - Site Logo/Name
    - Links: "Home," "Browse Reports," "Explore Keywords," "About" (optional)
    - **Prominent RAG Search Bar:** This is a primary interaction. "Ask anything about Medicaid Audits..."
  + **Hero Section (Optional, but common for modern sites):**
    - Catchy headline (e.g., "Insights from Medicaid Audits at Your Fingertips")
    - Brief tagline explaining the site's purpose.
  + **Featured Reports Section:**
    - **Visuals:** Use cards for each featured report. Cards are great for summarizing content.
    - **Card Content:**
      * **Report Title** (Clear, prominent)
      * **Audit Organization**
      * **Publication Date** (formatted nicely, e.g., "Jan 2024" or "Q1 2024")
      * **Overall Conclusion** (Truncated to 2-3 lines with a "Read More..." link that takes to the full report page).
      * Optional: Small visual cue if it's a "featured" item (e.g., a ribbon or icon).
    - **Layout:** Display 3-4 featured reports horizontally. If more, consider a subtle carousel or a "View All Featured" link.
    - **Admin Control:** You'll need a way in your admin backend to mark reports as "featured."
  + **Explore by Keywords Section:**
    - **Title:** "Trending Topics" or "Explore by Keyword"
    - **Presentation Options for Keywords + Counts:**
      * **Tag Cloud:**
        + Keywords displayed, with font size proportional to their count.
        + Visually engaging. Can become cluttered if too many keywords or counts are very disparate.
        + Each keyword is clickable.
      * **Pill/Tag List:**
        + Keywords shown as styled pills/tags, each with the count next to it (e.g., [Improper Billing (27)]).
        + Can be sorted by count (most frequent first) or alphabetically.
        + Cleaner than a tag cloud if many keywords.
        + Display top N (e.g., 10-20) with a "View All Keywords..." link.
      * **Interactive Bar Chart (More advanced):**
        + Top N keywords as bars, length representing count. Hover to see exact count. Clickable bars.
    - **Interaction:** Clicking a keyword takes the user to a "Report List Page" filtered by that keyword.
  + **Call to Action for RAG (if not solely in header):**
    - Another prominent search bar for the RAG model, perhaps with example questions.
  + **Footer:**
    - Copyright, links to "About," "Privacy Policy," "Terms of Use" (even for public data, good practice).

**B. Report List Page (e.g., after clicking a keyword or "Browse Reports")**

* **Purpose:** Display a list of reports based on filters or a general browse.
* **Layout Ideas:**
  + **Header:** Clearly state the context (e.g., "Reports tagged with: [Selected Keyword]" or "All Reports").
  + **Filtering/Sorting Options (Sidebar or Top Bar):**
    - **Active Filters:** Show the currently applied keyword filter (and allow removal).
    - **Sort By:** Dropdown (Publication Date (Newest/Oldest), Report Title (A-Z), Relevance (if applicable)).
    - **Additional Filters (Future):** By State, by Publication Year range.
  + **Report Items (List View or Card View):**
    - **List View (more compact):** Each report as a row.
      * Columns: Report Title, Audit Organization, Publication Date, State (if applicable).
      * A short snippet of the overall conclusion.
    - **Card View (more visual):** Similar to featured reports cards, but potentially less prominent.
  + **Information per item:**
    - Report Title (clickable, links to Report Detail Page)
    - Audit Organization
    - Publication Date
    - A few associated keywords (pills)
  + **Pagination:** If many reports, use clear pagination controls.

**C. Report Detail Page**

* **Purpose:** Display all extracted information for a single report.
* **Layout Ideas:**
  + **Main Content Area:**
    - **Report Title (H1 Heading)**
    - **Key Metadata Block (clearly separated):**
      * Audit Organization
      * Publication Date
      * Audit Period
      * State (if applicable)
      * Original Report Source URL (if available, as a clickable link)
      * Link to download/view the original PDF (e.g., "View Original PDF Document")
    - **Structured Extracted Data (clearly labeled sections):**
      * **Overall Conclusion:** Full text.
      * **Objectives:** Displayed as a bulleted or numbered list of individual objective texts.
      * **Findings:**
        + Each finding as a distinct item (e.g., in its own box or clearly separated).
        + Display finding\_text.
        + If financial\_impact\_amount exists for the finding, display it clearly (e.g., "Financial Impact: $X,XXX.XX").
      * **Recommendations:**
        + Each recommendation as a distinct item.
        + Display recommendation\_text.
        + If linked to a specific finding, indicate this (e.g., "Recommendation for Finding #Y: ...").
      * **LLM Insight:** The AI-generated summary/insight.
      * **Total Financial Impact** (for the whole report, if available).
    - **Associated Keywords:**
      * Displayed as clickable pills/tags. Clicking them would take the user to the Report List Page filtered by that keyword.
  + **Sidebar (Optional):**
    - "Ask about this report" button that could pre-fill the RAG context to focus queries on this specific document.
    - Related reports (if logic can be developed for this).

**III. Styling and "Modern Feel" Specifics:**

* **Color Palette:** Choose a limited, professional palette. Often a neutral base (whites, grays) with one or two accent colors for calls to action, links, and highlights. For a government-related tool, blues and greens can convey trust and stability.
* **Typography:** Select clean, readable web fonts (e.g., sans-serif like Inter, Open Sans, Lato, Roboto). Ensure good contrast between text and background.
* **Whitespace:** Don't cram elements together. Ample whitespace makes the design feel breathable and less overwhelming.
* **Icons:** Use icons sparingly and consistently (e.g., for search, download, links). Libraries like Font Awesome or Material Icons are good resources.
* **Shadows & Borders:** Subtle use of box shadows or borders can help define elements like cards and buttons, adding depth.
* **Interactions:** Smooth, subtle hover effects on clickable elements. Avoid jarring animations.

**IV. Hammering out Details for Keywords & Featured Reports:**

**Featured Reports (on Homepage):**

* **How many?** 3-4 is a good starting point. Too many defeats the "featured" purpose.
* **Why these reports?** You (admin) will curate them. They might be recent, impactful, or cover a particularly relevant topic.
* **Visuals:**
  + Each report in a "card."
  + Card Header: Report Title
  + Card Body: Audit Organization: [Name] | Published: [Date]
  + Card Body: Overall Conclusion: [First 2-3 sentences]... [Read More](links to full report detail page)
  + Card Footer (Optional): Key Keywords: [Pill 1] [Pill 2](first few keywords as pills)

**Keywords + Counts (on Homepage or "Explore Keywords" Page):**

* **Challenge:** Presenting clearly and usefully.
* **Solution Idea - "Top Keywords" + "Browse All":**
  1. **Homepage - "Top Topics" Section:**
     + Display the Top 10-15 keywords as clickable pills, sorted by count (descending).
     + Each pill: [Keyword Text (Count)]
     + Example: [Prior Authorization (42)][Improper Billing (35)] [Dental Services (28)]
     + A "View All Keywords" link below this section.
  2. **"All Keywords" Page (when "View All Keywords" is clicked):**
     + A simple, searchable, sortable list.
     + **Layout:** Two columns: Keyword | Report Count
     + **Sorting:** Click column headers to sort alphabetically by keyword or numerically by count.
     + **Search Bar:** "Filter keywords..." to quickly find a specific keyword in a long list.
     + Each keyword in the list is clickable, taking to the Report List Page.

**Interaction Flow for Keywords:**

1. User sees "Top Topics" on homepage OR goes to "All Keywords" page.
2. User clicks a keyword pill/link, e.g., [Improper Billing (35)].
3. User is taken to the **Report List Page**.
   * Header shows: "Reports related to: Improper Billing (35 reports)"
   * The list/cards below show the 35 reports that have "Improper Billing" as a keyword.