AURA: A Direct Response to the Problem Statement

AURA is the definitive blueprint for building a prototype that directly addresses the core business problem: the time-consuming, manual, and repetitive nature of the financial statement review process at the Adani Group.[1] The entire plan is structured to build the exact "autonomous review agent" envisioned in the challenge.[1]

Mapping AURA to the Six Core Capabilities

The problem statement specifies six core capabilities the AI agent must possess.[1] The AURA roadmap is designed to deliver each one in a logical, phased approach.

1. Data Ingestion & Consolidation

- **Problem Statement Requirement:** The agent must collect data from spreadsheets, assign user responsibilities from a master matrix, handle manual uploads for non-SAP entities, and manage an automated email workflow for the review process.[1]
- **AURA's Solution:**
  - **Phase 1: Foundation & Data Pipeline** is entirely dedicated to this. The plan uses **Pandas** to ingest the `trial_balance.csv` and `responsibility_matrix.csv` files, directly implementing the required data consolidation.[1]
  - The **Streamlit** UI, built in the same phase, provides the file uploader which serves as the mechanism for "manual trial report uploading".[1]
  - The plan makes a strategic hackathon decision to **simulate the automated email workflow** with clear status updates in the UI. This acknowledges the requirement while freeing up critical development time for the high-impact AI features that are more crucial for a winning prototype.[1]

2. Automated Report Generation

- **Problem Statement Requirement:** The agent must generate reports on the status of GL accounts, major variances compared to the previous period, and a graphical presentation of GL hygiene, using both tables and charts.[1]
- **AURA's Solution:**
  - **Phase 2: Validation & Automated Reporting** directly addresses this. The roadmap includes tasks to develop backend functions for "variance analysis" and "review status tracking."
  - The technology stack specifies using **Streamlit** with **Plotly** to create the required interactive dashboard, including bar charts for variances and pie charts for review status, fulfilling the "tabular and visual summaries" requirement precisely.[1]

3. Validation & Compliance Checks

- **Problem Statement Requirement:** The agent must perform critical data quality checks, most importantly ensuring that the "total of trial report is Nil" (i.e., debits equal credits).[1]
- **AURA's Solution:**
  - The plan incorporates a professional-grade solution by selecting **Great Expectations (GX)**.[2, 3] **Phase 2** includes the specific task of creating a GX "Expectation Suite" to programmatically

validate that the sum of the debit column equals the sum of the credit column. This demonstrates a mature, "unit tests for data" approach that directly satisfies this key compliance check.[4, 1]

4. Continuous Learning Loop

- **Problem Statement Requirement:** The agent must improve its accuracy with human feedback, using the example of a user correcting a line item, which the model then learns for the future.[1]
- **AURA's Solution:**
  - This is a central feature of **Phase 3: The Intelligence Layer**. The plan details a tangible user journey where a user corrects a misclassified GL account in the UI. This action triggers the backend to retrain a **Scikit-learn** classification model, creating a demonstrable, closed-loop system that provides clear "evidence of learning from user corrections".[5, 6, 1]

5. Agentic Behaviour

- **Problem Statement Requirement:** The agent should function like a "reporting assistant," autonomously compiling data and proactively surfacing insights, such as, "Expense for the current period is X times the expenses of previous month".[1]
- **AURA's Solution:**
  - The selection of **LangChain** in the technology stack is specifically for this purpose.[7, 8] **Phase 3** focuses on structuring the backend functions as "Tools" that a LangChain agent can orchestrate.
  - **Phase 4** then implements the "proactive insight" feature by having the agent display a contextually relevant summary on the main dashboard, directly mirroring the example in the problem statement.[1]

6. Interactive Visualization

- **Problem Statement Requirement:** The solution must feature a dashboard with drill-down capabilities and a conversational interface to query reports with natural language (e.g., "Show me year-over-year Profit of Adani Ports").[1]
- **AURA's Solution:**
  - **Phase 4: UI & Conversational Interface** is dedicated to building this exact experience. The plan specifies using **Streamlit** to create a polished dashboard with drill-down functionality on charts.[9, 10]
  - Crucially, it details the implementation of a chat interface (`st.chat_input`) that connects directly to the LangChain agent, allowing a manager to ask natural language questions and receive data-driven answers, precisely as requested in the problem statement.[1]

Mapping AURA to the Expected Hackathon Output

The problem statement concludes with four clear deliverables for the prototype.[1] The AURA roadmap is scoped to deliver on all four points.

Expected Output from Problem Statement [1]	How AURA Delivers This

Expected Output from Problem Statement [1]	How AURA Delivers This
1. Upload or connect to sample datasets (CSV/Excel).	**Phase 1** of the roadmap implements a Streamlit file uploader for exactly this purpose.
2. AI Agent ingests, consolidates, and generates draft reports.	The entire workflow across **Phases 1, 2, and 3** is designed to demonstrate this end-to-end process, from data ingestion with Pandas to report generation via the LangChain agent.
3. Dashboard or conversational interface presents key numbers, charts, and anomalies.	**Phase 4** focuses entirely on building this user interface with Streamlit, including the dashboard, charts for anomalies (variances), and the conversational chatbot.
4. Evidence of learning from user corrections.	The **Phase 3** implementation of the Scikit-learn model, which retrains based on user input from the UI, provides tangible and demonstrable evidence of this learning loop.

In summary, the **AURA** plan is a direct, comprehensive, and strategic blueprint for building a winning prototype that meets every technical requirement and deliverable outlined in the original problem statement.