I want to create a web app using react that will manage metadata for a solution that will generate automation scripts for Databricks.

Core Metadata (Supabase).

The core metadata will be managed and stored in a Supabase database. This metadata will have Workspace (User) that manages user Projects, Connections, Configurations, and Settings. The workspace should be connected to a git repo as soon as it is created. When a user signin and has no workspace a workspace should be created and the first step should be to connect the github repo.

Users who have admin rights can see all workspaces and projects.

When we select a project we will show a reactflow diagram that will allow the user to add nodes to the project. There must be a search that allow the user to search for all public nodes and filter it by node type. There should also be a grid view that allow the user to see all the nodes in the project in a grid view.

Node Metadata (Github)

Within a Project a user can import or create nodes. These nodes and nodeitems will have both a fully qualified name and a uuid to track lineage and accommodate renaming. All the uuids will be tracked in Supabase to ensure we can track the state of the nodes and identify any metadata corruption.

1. Nodes will be an Entity of a specific medallion layer like Raw (Landing), Bronze, Silver, Gold and within these layers the entity types can be Table, Staging, Persistent Staging, Data Vault, Data Mart and within these types there can be subtypes like Dimension, Fact, Hub, Link, etc.
2. The nodes will have nodeitems or columns/attributes. The attributes will have the normal metadata associated with it like datatype, transformations, description etc.
   1. NodeItems will track relationships to other entities. This is important to allow AI to assist with the query writing and to accelerate data vault.
   2. NodeItems will track mappings to both target objects and source objects. They can map one or many targets and one or many sources can map to it. This is important because data vault and data marts use different mapping approaches.

Import Metadata

1. The app will allow users to import metadata from files. This process will allow the user to drop sample files of these types **Supported file formats: .csv, .tsv, .tab, .json, .jsonl, .avro, .parquet, .txt, or .xml** onto the canvas and we need to derive the metadata from these files.
2. We need to be able to import metadata from MSSQL including Change Data Capture and Change Tracking databases.
3. Import metadata from rest services like Salesforce, Workday, ServiceNow and Google Analytics Raw Data
4. We should have the ability to use AI to discover and recommend descriptions for the columns and business names. We should be able to have the AI provide a confidence score of how certain it is that the results per node and nodeitem are correct.

Schema Modeling

Based on the nodes and the relationships tracked in the nodeitems we want to be able to create a entity relationship diagram using reactflow. This diagram will only be at the node level and when clicking on the node we will popup the node editor dialog. When we right click we can do other actions like add a relationship. Relationships will be drawn and when selected will show the relationship source and target and allow the user to change this information. We should have the ability for the model to be sent to AI for analysis and to identify

Data Vault Modelling.

If the project is configured for Data Vault when a node is selected or right clicked we should launch a new reactflow diagram that will be based on a accelerator that will use the source nodes metadata to derive hubs, satellites, links, and link sattelites based on preconfigured preferences. This will be a logical (preview) and when the user click apply it will create physical nodes based on this and add all mappings in the nodeitems. We will also have the ability to Point in Time (PIT) and Bridge nodes based on logic and node relationships. The PIT and Bridge can be defined and a Table, View or Materialised View. All of the above should be connected to AI to allow the user to ask AI for assistance and all AI results should be interpretable recommended changes displayed to user for approval and acceptance.

Dimensional Modeling

When we create a dimension, the user should be able to ask AI to assist in creating a dimension or fact. The AI should quickly scan the nodes and list sorted by the most likely source nodes at the top. The user should select the nodes they want to use and from the selection the AI should infer joins based on the relationships between the nodes. If it is not obvious the AI should ask the user to link the nodes based on nodeitums or columns. The user should them pick all the relevant columns they would like to include in the target table. When the user click apply it will create a node, nodeitem and add all the source to target mappings.

Macros

The user should have the ability to create reusable code fragments. These fragments will always be public and accessible to everyone.

Templating

The app will maintain all the jinja templates and template fragments in Supabase and users can make clones of these templates or fragments to their gitrepo to alter them if required. We will also allow the ability for users to override template fragments or inject fragments in selected places.