You are a AI/ML engineer , you are working on creating a agentic ai application that does

A self-service portal that lets multiple teams log in, select AI models, databases, storage, secrets managers, and enterprise data sources → provisions complete, secure, and standardized environments in minutes using pre-approved configurations.

Step1: Create DDL scripts for tables with sample data -- Give me DDL Scripts

|  |  |
| --- | --- |
| artifact\_type | |
| artifact\_type\_id | artifact\_type |
| 1 | database |
| 2 | secrets manager |
| 3 | AI models |
| 4 | Storage |

|  |  |  |
| --- | --- | --- |
| artifacts | |  |
| artifact\_id | artifact\_name | Artifact\_type\_id(FK to artifact\_type table) |
| 1 | supabase | 1 |
| 1 | redshift | 1 |
| 1 | databricks | 1 |
| 1 | snowflake | 1 |
| 2 | doppler | 2 |
| 2 | aws secret manager | 2 |
| 2 | azure secret manager | 2 |
| 3 | claude | 3 |
| 3 | gpt | 3 |
| 3 | misral | 3 |
| 3 | llama | 3 |
| 4 | s3 | 4 |
| 4 | supabase | 4 |
| 4 | blob | 4 |

|  |  |
| --- | --- |
| environment | |
| environment\_id | environment\_name |
| 1 | dev |
| 2 | qa |
| 3 | prod |

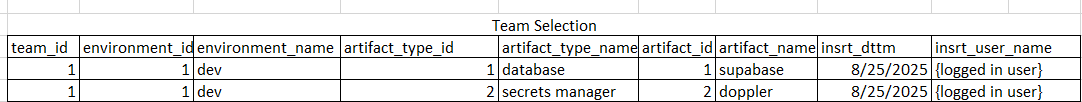
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Teams | | | | | |
| team\_id | team\_name | team\_pointofcontact | team\_distributionlist | username | pwd |
| 1 | team\_1 | john snow | [team\_1@mycorp.com](mailto:team_1@mycorp.com) | team\_1 | 1team! |
| 2 | team\_2 | ed | [ed@mycorp.com](mailto:ed@mycorp.com) | team\_2 | 2team! |

Step2 : I will run them in supabase and I have already created a supabase secret key

Step3: We need a portal which can be deployed in streamlit , it should have these features

1. Should allow multiple teams to login : Login’s are present in the table : Teams
2. When user login the webpage should have 5 drop down one for each (a) environment (b)artifact\_type (c) artifact

User should select atleast 1 from each drop down and entry should be made into the table : team selection as we see in the table below



DDL :

-- ===== 0) Safety: create extensions if needed (optional)

-- Supabase usually has pgcrypto; uncomment if you want UUIDs later.

-- CREATE EXTENSION IF NOT EXISTS "pgcrypto";

-- ===== 1) Core lookup tables

DROP TABLE IF EXISTS public.team\_selection CASCADE;

DROP TABLE IF EXISTS public.artifacts CASCADE;

DROP TABLE IF EXISTS public.artifact\_type CASCADE;

DROP TABLE IF EXISTS public.environment CASCADE;

DROP TABLE IF EXISTS public.teams CASCADE;

CREATE TABLE public.artifact\_type (

artifact\_type\_id smallint PRIMARY KEY,

artifact\_type text NOT NULL UNIQUE

);

INSERT INTO public.artifact\_type (artifact\_type\_id, artifact\_type) VALUES

(1, 'database'),

(2, 'secrets manager'),

(3, 'AI models'),

(4, 'storage');

CREATE TABLE public.environment (

environment\_id smallint PRIMARY KEY,

environment\_name text NOT NULL UNIQUE

);

INSERT INTO public.environment (environment\_id, environment\_name) VALUES

(1, 'dev'),

(2, 'qa'),

(3, 'prod');

-- Teams table – for demo: plaintext password (consider hashing in real use)

CREATE TABLE public.teams (

team\_id smallint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

team\_name text NOT NULL UNIQUE,

team\_pointofcontact text,

team\_distributionlist text,

username text NOT NULL UNIQUE,

pwd text NOT NULL

);

INSERT INTO public.teams (team\_name, team\_pointofcontact, team\_distributionlist, username, pwd) VALUES

('team\_1', 'john snow', 'team\_1@mycorp.com', 'team\_1', '1team!'),

('team\_2', 'ed', 'ed@mycorp.com', 'team\_2', '2team!');

-- ===== 2) Artifacts catalog

CREATE TABLE public.artifacts (

artifact\_id bigserial PRIMARY KEY,

artifact\_name text NOT NULL,

artifact\_type\_id smallint NOT NULL REFERENCES public.artifact\_type (artifact\_type\_id) ON UPDATE CASCADE ON DELETE RESTRICT,

CONSTRAINT uq\_artifact UNIQUE (artifact\_name, artifact\_type\_id)

);

-- Databases (artifact\_type\_id = 1)

INSERT INTO public.artifacts (artifact\_name, artifact\_type\_id) VALUES

('supabase', 1),

('redshift', 1),

('databricks', 1),

('snowflake', 1);

-- Secrets managers (artifact\_type\_id = 2)

INSERT INTO public.artifacts (artifact\_name, artifact\_type\_id) VALUES

('doppler', 2),

('aws secrets manager', 2),

('azure secret manager', 2);

-- AI models (artifact\_type\_id = 3)

INSERT INTO public.artifacts (artifact\_name, artifact\_type\_id) VALUES

('claude', 3),

('gpt', 3),

('mistral', 3),

('llama', 3);

-- Storage (artifact\_type\_id = 4)

INSERT INTO public.artifacts (artifact\_name, artifact\_type\_id) VALUES

('s3', 4),

('supabase', 4),

('blob', 4);

-- ===== 3) Team selections (one row per chosen artifact)

-- 1) Drop old table (or keep it if you want history)

-- DROP TABLE IF EXISTS public.team\_selection CASCADE;

-- 2) Header (one row per interaction/click)

CREATE TABLE IF NOT EXISTS public.team\_selection\_batch (

selection\_key bigserial PRIMARY KEY, -- <-- same number for the whole interaction

team\_id smallint NOT NULL REFERENCES public.teams(team\_id) ON DELETE CASCADE,

environment\_id smallint NOT NULL REFERENCES public.environment(environment\_id),

insrt\_dttm timestamptz NOT NULL DEFAULT now(),

insrt\_user\_name text NOT NULL

);

-- 3) Detail (one row per chosen artifact)

CREATE TABLE IF NOT EXISTS public.team\_selection\_detail (

selection\_detail\_id bigserial PRIMARY KEY,

selection\_key bigint NOT NULL REFERENCES public.team\_selection\_batch(selection\_key) ON DELETE CASCADE,

artifact\_type\_id smallint NOT NULL REFERENCES public.artifact\_type(artifact\_type\_id),

artifact\_id bigint NOT NULL REFERENCES public.artifacts(artifact\_id)

);

-- 4) Flat view that matches your Excel-style report

CREATE OR REPLACE VIEW public.v\_team\_selection\_flat AS

SELECT

b.selection\_key,

b.team\_id,

b.environment\_id,

e.environment\_name,

d.artifact\_type\_id,

at.artifact\_type AS artifact\_type\_name,

d.artifact\_id,

a.artifact\_name,

b.insrt\_dttm,

b.insrt\_user\_name

FROM public.team\_selection\_batch b

JOIN public.team\_selection\_detail d ON d.selection\_key = b.selection\_key

JOIN public.environment e ON e.environment\_id = b.environment\_id

JOIN public.artifact\_type at ON at.artifact\_type\_id = d.artifact\_type\_id

JOIN public.artifacts a ON a.artifact\_id = d.artifact\_id

ORDER BY b.selection\_key, d.artifact\_type\_id;