Technical Interview Project: AI Agent for Workflow State Generation

Role: Infrastructure/Backend/DevOps Applicant

Submission Deadline: August 6, 2025-----Project Goal

The primary goal of this project is to develop an AI agent capable of generating workflow `state` and `blocks` (as defined in the provided schema and example `state` object) based on data that would reside in database tables equivalent to `workflow_blocks_rows` and `workflow_rows`.

This project aims to assess your skills in backend development, data processing, infrastructure design, and your ability to leverage AI/ML concepts for practical application.

Project Description:
You are tasked with building an AI agent (or a system that leverages AI techniques) that can construct the `state` object for a workflow, specifically populating the `blocks` section of that `state`. The agent should fetch information from PostgreSQL tables (`workflow_blocks_rows` and `workflow_rows`) which are structured similarly to the provided CSV files.

The `workflow` table schema and an example `state` object (including a `starter` block) are provided in the attached document. Your agent should be able to intelligently interpret the data from these database tables to generate a valid `state` object, focusing on the `blocks` and their properties (e.g., `id`, `type`, `name`, `position`, `subBlocks`, `outputs`, `enabled`, etc.).Provided Resources

1. **`Technical document`**: This document contains:
    - The schema for the `public.workflow` table.
2. **`workflow_blocks_rows.csv`**: This file serves as a guideline for the structure of a PostgreSQL table (`workflow_blocks_rows`) containing information about individual workflow blocks.
3. **`workflow_rows.csv`**: This file serves as a guideline for the structure of a PostgreSQL table (`workflow_rows`) containing higher-level workflow information.

Requirements

1. **Backend Application:** Develop a backend application (e.g., in Python, Node.js, Go, Java, etc.) that will house your AI agent.
2. **Database Interaction:** The application should be able to connect to a PostgreSQL database (e.g., Supabase) and query data from tables named `workflow_blocks_rows` and `workflow_rows`. You should assume these tables are populated with data.
3. **AI Agent Logic:** Implement the core logic of the AI agent. This agent should:
    - Query the necessary data from the `workflow_blocks_rows` and `workflow_rows` tables.

    - Intelligently map database columns to the properties within the `state` and `blocks` JSON structure.
    - Generate a complete and valid `state` JSON object, paying close attention to the `blocks` array.
    - Handle different block types (at a minimum, be able to generate the `starter` block as shown in the example).
    - Consider how to handle dynamic `subBlocks` and `outputs` based on block types.
4. **Database Persistence:** Once the workflow state is generated, push the resulting `workflow` row (including the generated `state` JSON) to a `public.workflow` table in a Supabase (or any PostgreSQL equivalent) database. You should create appropriate tables in the database to store this data, aligning with the provided schema.
5. **Error Handling:** Implement robust error handling for scenarios like missing database records, invalid data, or unparseable inputs.
6. **Containerization (DevOps Aspect):** Containerize your application using Docker. Provide a `Dockerfile` and instructions for building and running the container.
7. **Deployment Considerations (Discussion Point):** Discuss how you would deploy this application in a production environment (e.g., Kubernetes, serverless, etc.), including considerations for scalability, reliability, and security.

Deliverables

1. **`README.md`:** A comprehensive `README.md` file with the following sections:
    - **Project Overview:** A brief description of the project.
    - **Setup and Installation:** Clear instructions on how to set up and run your application locally.
    - **Usage:** How to use your API (if applicable) and trigger the state generation process.
    - **Database Setup:** Instructions for setting up the Supabase/PostgreSQL database and table schemas (`workflow_blocks_rows`, `workflow_rows`, and `public.workflow`). Include DDL for these tables.
    - **Design Choices:** Explanation of your technical decisions, including the choice of language, frameworks, and how your AI agent logic works.
    - **Testing Strategy:** How you approached testing your application.
    - **Deployment Considerations:** Your thoughts on deploying this solution to production.
    - **Assumptions:** Any assumptions you made regarding the database data structure or the `state` object.
    - **Future Improvements:** Ideas for enhancing the agent or the system.
2. **Sample Data (SQL Inserts):** Provide SQL insert statements for a few sample rows in your `workflow_blocks_rows` and `workflow_rows` tables that your agent can process and use to generate a workflow state.

Evaluation Criteria

Your project will be evaluated based on the following:

- **Correctness and Completeness:** Does the AI agent correctly generate the `state` and `blocks` JSON based on the database input, adhering to the provided schema and example?
- **Code Quality:** Readability, maintainability, modularity, and adherence to best practices.
- **System Design:** Scalability, robustness, and efficiency of your backend application.
- **AI Agent Logic:** The intelligence and extensibility of your approach to mapping database data to JSON structure.
- **Database Integration:** Correctness of database interaction (reading from and writing to PostgreSQL).
- **Testing:** Quality and coverage of your unit tests.
- **DevOps Practices:** Correct use of Docker, clarity of deployment considerations.
- **Documentation:** Clarity, thoroughness, and organization of your `README.md`.
- **Problem-Solving:** Your approach to handling potential ambiguities or complexities in the data mapping.

Submission Instructions

Please submit your README by **August 6, 2025**. We will schedule a call as soon as we've reviewed your submission so we can discuss your implementation.

Good luck! We look forward to seeing your innovative solution.

Schema:

```
create table public.workflow (
  id text not null,
  user_id text not null,
  workspace_id text null,
  folder_id text null,
  name text not null,
  description text null,
  state json not null,
  color text not null default '#3972F6'::text,
  last_synced timestamp without time zone not null,
  created_at timestamp without time zone not null,
  updated_at timestamp without time zone not null,
  is_deployed boolean not null default false,
  deployed_state json null,
  deployed_at timestamp without time zone null,
  collaborators json not null default '[]'::json,
  run_count integer not null default 0,
  last_run_at timestamp without time zone null,
  variables json null default '{}'::json,
  is_published boolean not null default false,
  marketplace_data json null,
  constraint workflow_pkey primary key (id),
  constraint workflow_folder_id_workflow_folder_id_fk foreign KEY (folder_id) references workflow_folder (id) on delete set null,
  constraint workflow_user_id_user_id_fk foreign KEY (user_id) references "user" (id) on delete CASCADE,
  constraint workflow_workspace_id_workspace_id_fk foreign KEY (workspace_id) references workspace (id) on delete CASCADE
) TABLESPACE pg_default;
```

```
create table public.workflow_blocks (
  id text not null,
  workflow_id text not null,
  type text not null,
  name text not null,
  position_x numeric not null,
  position_y numeric not null,
  enabled boolean not null default true,
  horizontal_handles boolean not null default true,
  is_wide boolean not null default false,
  advanced_mode boolean not null default false,
  height numeric not null default '0'::numeric,
  sub_blocks jsonb not null default '{}'::jsonb,
  outputs jsonb not null default '{}'::jsonb,
  data jsonb null default '{}'::jsonb,
  parent_id text null,
  extent text null,
  created_at timestamp without time zone not null default now(),
  updated_at timestamp without time zone not null default now(),
  constraint workflow_blocks_pkey primary key (id),
  constraint workflow_blocks_workflow_id_workflow_id_fk foreign KEY (workflow_id) references
workflow (id) on delete CASCADE
) TABLESPACE pg_default;

create index IF not exists workflow_blocks_workflow_id_idx on public.workflow_blocks using
btree (workflow_id) TABLESPACE pg_default;

create index IF not exists workflow_blocks_parent_id_idx on public.workflow_blocks using btree
(parent_id) TABLESPACE pg_default;

create index IF not exists workflow_blocks_workflow_parent_idx on public.workflow_blocks
using btree (workflow_id, parent_id) TABLESPACE pg_default;

create index IF not exists workflow_blocks_workflow_type_idx on public.workflow_blocks using
btree (workflow_id, type) TABLESPACE pg_default;
```