🚀 Smart Company Discovery Assistant

🧩 Scenario

Your company manages internal knowledge (Q&A) in a **PostgreSQL** database.

You're building a small internal tool — **Smart Company Discovery Assistant** — that allows internal teams to:

1. Manage company Q&A information.
2. Ask natural-language questions and get LLM-powered answers based on the Q&A database.

🎯 Objective

Build a multi-service system consisting of:

- A **Go-based backend** (main web app + UI).
- A **Flask-based AI service** for LLM processing and embeddings.
- A **PostgreSQL database** for data storage.

⚠️ **NOTE:** You may implement the UI using either **Go templates** or **React / Next.js**, whichever you prefer.

✅ Project Deliverables Checklist

🧱 Core System Setup

- Go service (`/cmd` or `/app` folder) with clean structure and modular packages.
- Flask microservice for LLM operations.
- PostgreSQL schema and migrations (SQL scripts or `init.sql`).

- `.env` or config file for environment variables (DB, Flask URL, API keys, etc.).
- README file with clear setup and run instructions.

⚠️ **NOTE:** Your submission **must include clear setup and run instructions** in the README. If the reviewers are **unable to run your project successfully after following your README** (without additional clarification or manual fixes), **your submission will be disqualified**.

Please verify that:

- All required services (Go API, Flask service, PostgreSQL) start correctly.
- Environment variables and configuration steps are clearly documented.
- Any initialization commands or migrations work as described.

🗂️ 2. Knowledge Base (Q&A) Management (Go)

Create a `qa_pairs` table:

```
CREATE TABLE qa_pairs (
  id SERIAL PRIMARY KEY,
  question TEXT,
  answer TEXT
);
```

- Implement frontend pages or forms (HTML or React) for:
  - **Create** new Q&A entry.
  - **Edit / Update** an existing Q&A entry.
  - **Delete** a Q&A entry
- Validate inputs (non-empty question and answer).
- Include a confirmation before deleting an entry.

🤖 3. Flask LLM Service

- Flask endpoint: `POST /generate-answer`.

Input format:

{ "question": "What is your refund policy?" }

- Use **vector embeddings** to find the top 3 most relevant Q&A pairs from PostgreSQL.
    - Can use any model for embeddings (e.g., OpenAI, HuggingFace, etc.).
- Construct a clear prompt combining the retrieved Q&A context with the question.
- Generate an answer using **any LLM model** of choice.

Return a structured response:

{ "answer": "Refunds are processed within 5 business days.", status: 200 }

🔗 4. Go → Flask Integration

- Create API endpoint in Go: `POST /api/ask`.
- When a user submits a question:
    1. Forward it to the Flask `/generate-answer` endpoint.
    2. Receive and parse the response.
    3. Return the result as JSON or render it in the UI.
- Handle network or timeout errors gracefully.

🧭 5. UI Requirements

Navigation

- Show a nav bar or a side bar with links to navigate between pages easily.

Ask Question Page

- Page `/ask` with:
    - Input field for typing a question.
    - Button to submit.
    - Area to display AI-generated answer.
- Should call `/api/ask` internally and display the result.

Q&A Management Page

- Form(s) to **create**, **edit**, and **delete** Q&A pairs.
- Display existing Q&A entries in a table. Implement:
    - **Pagination**
    - **Sorting** (e.g., by name or company)
    - **Search** (by company or email)
- Ensure basic form validation and layout organization.

💡 **Frontend Options:**
You can build the UI using either:

- Go's built-in `html/template` system, or
- React / Next.js frontend with REST API calls to the Go backend.

Styling can be minimal — focus on clear structure and usability.

⚙️ 6. Configuration & Setup

- Use environment variables for:
    - Database connection string

    - Flask service base URL
    - LLM API keys
- Include `.env.example` for reference.
- Instructions to initialize DB, run both services, and seed data.

🧠 Bonus (Optional)

- Docker Compose setup to run Go, Flask, and PostgreSQL together.

🧰 How To Submit

- Upload your code in a **github repo** and share the link
- Write clear instructions related to deployment and running the application (including any ENV variables) in a **README** file.