1. **User Completes Wizard:**

   - The system presents the AI Pilot Discovery Questionnaire (as seen in your prototype) to the user.
   - The user provides answers to the questions, including text descriptions, ratings (1-5), and selections (Yes/No, checkboxes).
   - Upon completion, the user submits the form.

2. **Write Raw Response Data to Database:**

   - On submission, the system captures all raw user responses from the wizard.
   - A new record is created in the `assessments` table to represent this specific assessment instance. Let's assume this generates an `assessment_id`.
   - Each individual answer is stored, likely in a new table (e.g., `assessment_responses`). This table should link back to the specific assessment (`assessment_id`), the user (`user_id`), the question asked (e.g., `question_id` or text), the raw answer (`response_text`, `response_score`, etc.), and a timestamp.

3. **Submit to AI for Processing:**

   - A trigger (manual or automated upon saving assessment responses) initiates an asynchronous job to process the assessment data using an AI model (like a Large Language Model). This job takes the `assessment_id` as input.
   - The AI performs the following sub-tasks based on the collected wizard responses and potentially linked data (like capabilities):
   - Identify Key Themes: Analyze qualitative responses (e.g., answers to Q2 on pain points, Q16 on benefits) to identify recurring business challenges and goals.
   - Extract and Rank Key Priorities: Extract explicit and implicit priorities from answers (Q2, Q7, Q8, Q16). Rank these priorities based on the "Value Potential" and "Ease of Implementation" scoring methodology outlined in the questionnaire, which aligns with the "Speed vs ROI" concept.
   - Match Priorities to Capabilities: Compare the ranked priorities against the `name` and `description` fields in the `ai_capabilities` table. Use semantic matching to find relevant capabilities. The AI should verify if the suggested capabilities logically address the specific pain points mentioned by the user (Q2).

- Verify Existing Tools: Check the user's answer to Q13 regarding existing tools. Compare these mentioned tools against the capabilities identified above to see if there's overlap or if current tools already address some needs.

- Recommend Tools: Based on the matched capabilities (that aren't already covered by existing tools), query a database of AI tools (needs to be created, see DB section below) filtered by `capability_id`. Recommend relevant tools, specifically highlighting open source options as requested, but potentially including commercial ones for comparison. Tool recommendations should ideally consider factors like implementation effort and business value, potentially stored in the `ai_capabilities` or a new `ai_tools` table.

- Provide Rollout Overview/Commentary: Generate a concise, actionable summary for implementation. This should incorporate insights from the ESI case study's post-90-day plan and the questionnaire's focus on 90-day pilots. It should emphasize quick wins, potential training needs, key success metrics (Q17), and considerations for change management.

- Generate Heatmap Data: Calculate "Speed" (Ease) and "ROI" (Value) scores for the recommended capabilities/features based on the assessment data and the scoring framework. Output this data (e.g., in JSON format) suitable for rendering a visual heatmap, prioritizing high-impact, quick-win features. This should factor in the comparison of tool types (open-source, cloud API, commercial) as part of the evaluation User Input.

- The AI processing job concludes by writing the structured results (themes, priorities, capabilities, tools, commentary, heatmap data) to a dedicated results table (e.g., `assessment_results`) linked to the `assessment_id`. The status in the `assessments` table is updated (e.g., to 'Complete').

Database and Schema Update Tasks

Based on the functional requirements and your existing schema, here are the necessary updates:

1. Create New Table: `assessment_responses`

- Purpose: Store individual raw answers from the wizard.

- Columns:

- `response_id` (Primary Key, auto-increment)

- `assessment_id` (Foreign Key referencing `assessments.id`)
- `user_id` (Foreign Key referencing `users.id`)
- `question_identifier` (Text or Integer, referencing the specific question)
- `response_text` (Text, for free-form answers)
- `response_numeric` (Numeric, for scale/score answers)
- `response_boolean` (Boolean, for Yes/No answers)
- `response_json` (JSONB, for multiple choice/checkboxes)
- `created_at` (Timestamp)

2. Create New Table: `ai_tools`

- Purpose: Catalog commercial and open-source AI tools.
- Columns:
- `tool_id` (Primary Key, auto-increment)
- `tool_name` (Text, not null)
- `description` (Text)
- `website_url` (Text)
- `license_type` (Text, e.g., 'Open Source', 'Commercial', 'Freemium')
- `primary_category` (Text, aligns with general function)
- `tags` (Text Array or JSONB)
- `created_at` (Timestamp)
- `updated_at` (Timestamp)

3. Create New Table: `capability_tool_mapping`

- Purpose: Link AI capabilities to the tools that can fulfill them (Many-to-Many relationship).
- Columns:
- `mapping_id` (Primary Key, auto-increment)
- `capability_id` (Foreign Key referencing `ai_capabilities.id`)

- `tool_id` (Foreign Key referencing `ai_tools.id`)
- `notes` (Text, optional, e.g., suitability notes)

4. Create New Table: `assessment_results`
   - Purpose: Store the processed output from the AI analysis.
   - Columns:
   - `result_id` (Primary Key, auto-increment)
   - `assessment_id` (Foreign Key referencing `assessments.id`, unique constraint recommended)
   - `identified_themes` (JSONB or Text Array)
   - `ranked_priorities` (JSONB, e.g., `{priority: "...", value_score: 4, ease_score: 5}, ...`)
   - `recommended_capabilities` (JSONB or Array of Foreign Keys referencing `ai_capabilities.id`)
   - `capability_rationale` (JSONB, explaining the matches)
   - `existing_tool_analysis` (Text, commentary on existing tools)
   - `recommended_tools` (JSONB or Array of Foreign Keys referencing `ai_tools.id`)
   - `rollout_commentary` (Text)
   - `heatmap_data` (JSONB, e.g., `{capability_id: X, speed: Y, roi: Z}, ...`)
   - `processing_status` (Text, e.g., 'Success', 'Failed')
   - `error_message` (Text, if failed)
   - `created_at` (Timestamp)
   - `completed_at` (Timestamp)
5. Update Existing Table: `assessments`
   - Add a `status` column (Text, e.g., 'New', 'Processing', 'Completed', 'Error') to track the state.
   - Ensure `user_id` and potentially `organization_id` Foreign Keys exist.

6. Update Existing Table: `ai_capabilities`

- Review the existing `implementation_effort` and `business_value` columns. Ensure they align with the ‘Ease’ and ‘Value’ dimensions needed for the scoring and heatmap. You might need to refine these or add specific numeric score fields if the current ‘text’ type isn’t sufficient for calculations.