Given current progress, the next most critical step is to **build out AI components for self-grading and response refinement**. This step aligns with your original plan and will significantly improve the app's correctness and usability by ensuring high-quality, relevant responses.

**Next Steps**

Here’s what to focus on, broken into workable steps:

**Step 1: Implement Self-Grading Logic**

**Purpose**: Score the relevance of retrieved results against the query to prioritize or refine responses.

**Actions:**

1. **Use Watson NLP**:
   * Integrate Watson Natural Language Processing (NLP) services to:
     + Perform semantic analysis of the query and passages.
     + Evaluate how closely the retrieved passages align with the query.
2. **Calculate Relevance Score**:
   * Develop logic to compare the query to the retrieved passage text.
   * Use a relevance metric like cosine similarity or keyword overlap.
3. **Output Refined Scores**:
   * Sort and prioritize the results in the frontend based on the calculated relevance.

**Step 2: Add Query Refinement and Error Handling**

**Purpose**: Ensure that the app gracefully handles irrelevant or low-confidence responses by refining the query or escalating to supplementary retrieval.

**Actions:**

1. **Query Rewriting**:
   * If the query returns low-confidence results:
     + Use Watson NLP to suggest more specific or alternative queries.
     + Automatically rewrite the query with additional context.
2. **Fallback Logic**:
   * Use a secondary data source (e.g., Google Search API or internal repositories) when Watson Discovery provides insufficient results.

**Step 3: Enhance Frontend for Grading and Refinement**

**Purpose**: Provide users with clear insights into the confidence and quality of the retrieved results.

**Actions:**

1. **Add Relevance Grading**:
   * Display a “Relevance Score” alongside each result in the frontend.
   * Use color-coded indicators (e.g., green for high relevance, yellow for moderate, red for low).
2. **Feedback Mechanism**:
   * Allow users to mark results as “relevant” or “not relevant.”
   * Use this feedback to refine future queries or improve data indexing.

**Step 4: Record and Log Results**

**Purpose**: Build a robust record of all queries and results for future optimization and troubleshooting.

**Actions:**

1. **Log All Queries**:
   * Expand the search\_history to store relevance scores and user feedback.
   * Save this data to a database or file for persistence.
2. **Visualize Metrics**:
   * Use a dashboard to display trends, such as:
     + Most frequent queries.
     + Average relevance scores.
     + Query success rates.

**Workable Plan**

**Iteration 1: Self-Grading**

* Implement Watson NLP to compute and display relevance scores.
* Integrate this scoring into the backend and frontend.

**Iteration 2: Query Refinement**

* Add fallback logic to rewrite queries for low-confidence results.
* Test using different input scenarios.

**Iteration 3: Feedback and Logging**

* Record user feedback and add a history dashboard.
* Implement a persistence layer for long-term storage.

**Tools to Use**

| **Task** | **IBM Cloud Service** |
| --- | --- |
| Text Analysis and Grading | Watson NLP |
| Query Refinement | Watson NLP + custom logic |
| Backend Development | Flask |
| Frontend Development | JavaScript + HTML/CSS |
| Persistence (Optional) | IBM Cloudant |
| Monitoring and Debugging | IBM Cloud Monitoring |