# User Stories and Task Breakdown for Purview/PowerBI

Implementation to include RAG functionality for Power BI reports using Purview and a vector database. Goal: to retrieve most relevant Power BI reports based on user queries, integrating it as an additional service alongside Data Insights and Enterprise Search agents.

## User Story

As a user, I want to retrieve the most relevant Power BI report based on my query so that I can access actionable insights quickly and without needing to search manually.

## Acceptance Criteria

* A Power BI RAG Service processes user queries to search for relevant Power BI reports.
* The service uses Purview to connect to Power BI dashboards and index their metadata (e.g., names, descriptions).
* A vector database stores embeddings for the indexed metadata (just metadata? Connect with Michelle)
* Upon receiving a user query, the service performs a similarity search on the vector database and retrieves the most relevant report.
* The retrieved report is returned as part of the Orchestrator Agent's final response.
* The service operates in parallel with sql agent and enterprise search agents.

## What Needs to Happen in the Code

### 1. Index Power BI Metadata into the Vector Database

* Connect to Purview for Metadata Extraction: Use Purview APIs to fetch Power BI report metadata, extracting relevant fields like report name, description, and report id.
* Generate Embeddings for Metadata: convert report names and descriptions into vector representations.
* Index Metadata in the Vector Database: Store embeddings in the vector database with corresponding metadata (e.g., report id, report name, description).
* Schedule Metadata Refresh: Implement a periodic process to refresh the vector database with new Power BI reports from Purview. (done somewhere else?)
* Test the Indexing Pipeline: Validate that metadata is correctly extracted, embedded, and indexed.

### 2. Implement Power BI RAG Search

* Integrate Query Parsing Logic: Parse the user query to identify intent related to Power BI reports. Or is this part of the routing in the orchestrator agent?
* Generate Query Embeddings: Use the same embedding model to generate embeddings for user queries. (Michelle working on?)
* Perform Similarity Search: Query the vector database using the query embedding to retrieve the most relevant reports.
* Handle No Matches: Add fallback logic to handle cases where no reports are sufficiently relevant.
* Format RAG Response: Structure the retrieved report information into a user-friendly format.
* Error Handling and Logging: Log all queries and responses for debugging.

### 3. Integrate Power BI RAG Service into the Orchestrator

* Extend Orchestrator Routing Logic: Modify routing logic to detect Power BI-related queries.
* Enable Parallel Execution: Allow the Power BI RAG Service to run in parallel with DI and ES agents. Outside Enterprise Search I guess but still important…
* Aggregate Power BI Results in Final Response: Combine Power BI RAG results with outputs from DI and ES agents.
* Pass-Through Logic for Single-Service Queries: Bypass DI and ES when the query is specific to Power BI reports.

### 4. Extend Unified Chat History Management

* Add Fields to Chat History Schema: Include fields like service: power bi rag, report id, report name, and description.
* Log Power BI Queries and Responses: Store each Power BI RAG query and its corresponding result in the chat history. (Be consistent about how this is being handeled in the sql agent if possible)
* Enable Contextual Follow-Ups: Ensure Power BI-related follow-up queries can leverage chat history to refine searches.

### 5. Testing and Validation

* Unit Tests for Power BI RAG Service: Test individual components like metadata extraction, embedding generation, and vector search.
* Integration Tests: Validate end-to-end functionality, from query input to the orchestrator's final response.
* Performance Testing: Test scalability of vector search with large metadata sets.
* User Acceptance Testing: Collect feedback to ensure results align with user expectations.

**Subtask Summary Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Task ID | Description |  | Estimated Time |
| 1.1 | Connect to Purview for Metadata Extraction |  | 4 hours |
| 1.2 | Generate Embeddings for Metadata |  | 3 hours |
| 1.3 | Index Metadata in Vector Database |  | 3 hours |
| 1.4 | Schedule Metadata Refresh |  | 2 hours |
| 2.1 | Integrate Query Parsing Logic |  | 3 hours |
| 2.2 | Perform Similarity Search |  | 3 hours |
| 2.3 | Handle No Matches |  | 2 hours |
| 2.4 | Format RAG Response |  | 3 hours |
| 3.1 | Extend Orchestrator Routing Logic |  | 3 hours |
| 3.2 | Enable Parallel Execution |  | 4 hours |
| 3.3 | Aggregate Power BI Results in Final Response |  | 3 hours |
| 4.1 | Add Fields to Chat History Schema |  | 2 hours |
| 4.2 | Log Power BI Queries and Responses |  | 2 hours |
| 5.1 | Unit Tests for Power BI RAG Service |  | 3 hours |
| 5.2 | Integration Tests |  | 4 hours |
| 5.3 | Performance Testing |  | 4 hours |

**Estimated Total Time**

|  |  |
| --- | --- |
| **Section** | **Estimated Time (Total)** |
| Metadata Indexing | 12 hours |
| RAG Query Execution | 11 hours |
| Orchestrator Integration | 10 hours |
| Unified Chat History | 4 hours |
| Testing and Validation | 11 hours |

**Grand Total:** **48 hours**