MetaNexus Project Documentation

**1. Introduction**

**1.1 Project Overview**

MetaNexus is a **Metastore Viewer** designed for seamless metadata extraction from cloud storage services like **AWS S3** without requiring an external metastore. It supports multiple table formats, including **Parquet, Iceberg, Delta**.

**1.2 Problem Statement**

Organizations face challenges in managing scattered metadata across multiple storage platforms, leading to:

* Format incompatibilities
* Lack of visibility
* Difficulty in real-time tracking

**MetaNexus** solves these issues by offering a **unified metadata professional dashboard**, **real-time tracking**, and **AI-powered insights**.

**2. Proposed Solution**

**2.1 Key Features**

* **Direct Metadata Extraction:** Fetches metadata directly from cloud storage without requiring a metastore.
* **Multi-Format Compatibility:** Supports **Parquet, Iceberg, Delta, and Hudi**.
* **Table Visualization:** schema, partitions, and version history.

**2.2 Innovations**

* **Automated Data Quality Insights:** AI-powered insights for detecting schema inconsistencies and null values.
* **AI-Powered Chatbot:** Assists users with metadata-related queries.
* **Smart Metadata Summary:** Automatically generates structured summaries of metadata from various table formats (Parquet, Iceberg, Delta) for better data visibility and governance.

**3. Approach & Methodology**

**3.1 System Architecture**

MetaNexus follows a **three-layered architecture**:

1. **Data Ingestion Layer:** Directly fetches metadata from cloud storage.
2. **Processing Layer:** Standardized readers ensure compatibility across formats.
3. **User Interaction Layer:** Offers a Web UI, and an AI-powered chatbot.

**3.2 Key Challenges & Solutions**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| Multi-format support | Standardized readers for each format |
| Performance Concerns | Scans only metadata, not full datasets |

**4. Technologies & Implementation**

**4.1 Tech Stack**

* **Programming Languages:** Python, Next.js
* **Frameworks:** React, Flask
* **LLMs:** Mystral AI

**4.2 Implementation Flow**

1. **User Signup & Authentication**
   * User creates an account.
   * System generates an **ARN (Amazon Resource Name)** code.
   * User grants **IAM (Identity and Access Management)** role-based access.
2. **Adding S3 Buckets**
   * User logs into the MetaNexus platform.
   * User adds one or more **Amazon S3 buckets** for metadata analysis.
3. **Bucket-Specific Dashboard**
   * Each bucket gets a dedicated **dashboard** displaying:
     + **Row count**
     + **Storage size**
     + **Partition keys**
     + **Schema details**
     + **File format compatibility**
     + **Other metadata insights in graphical format**
4. **AI Chatbot Interaction**
   * Users can interact with an **AI-powered chatbot** for:
     + Metadata queries
     + Schema verification
     + Assistance in resolving storage-related concerns
5. **Generating Summary Reports & Insights**
   * Users can generate **Summary Reports** providing an overview of metadata
   * **Data-Driven Insights**

**5. Potential Impact**

**5.1 Target Audience**

* Data Engineers & Analysts
* Businesses Managing Large-Scale Data
* Cloud Storage Providers

**5.2 Benefits**

* Saves Time & Boosts Productivity
* Improves Data Governance
* Democratizes Data Access
* Reduces Cloud Storage Costs
* Eco-Friendly: Optimized Compute Usage

**6. References & Additional Resources**

**6.1 Research Paper Links:**

* <https://journal.code4lib.org/articles/11752>
* <https://www.scitepress.org/Papers/2020/101295/101295.pdf>
* <https://arxiv.org/pdf/1903.05838>

**7. Team Details**

* **Project Name:** MetaNexus
* **Problem Statement & ID:** PS 2 - Metastore Viewer for Parquet, Iceberg, Delta & Hudi Tables on S3
* **Team Name:** Glitch Club
* **College:** Vishwakarma Institute of Technology, Pune