**1. Project Summary**

**Project Title:** NASA OSDR Explorer

The NASA OSDR Explorer is an advanced data-driven web application designed to unlock the rich, complex datasets within NASA's Open Science Data Repository (OSDR). The project addresses the challenge of discovering hidden connections and gaining novel insights from vast amounts of scientific research data by transforming flat, tabular information into an intuitive, interactive visual network.

At its core, the application scrapes study metadata from the OSDR, processes it, and stores it in a dual-database system: MongoDB for robust data storage and querying, and Neo4j for powerful graph-based visualization. The user-facing interface, built with Streamlit, provides tools to not only search and filter studies but also to explore the intricate web of relationships between them—visualizing connections between research factors, biological organisms, and assay types.

Future development will integrate Google Cloud's Vertex AI to introduce two powerful features: AI-powered semantic search allowing natural language queries against the entire dataset, and a generative AI assistant capable of comparing and summarizing related studies to accelerate research and understanding.

**Tech Stack:** Python, Streamlit, MongoDB Atlas, Neo4j, Selenium, pyvis, Google Cloud (Vertex AI), GitLab.

**2. Glossary of Terms**

* **OSDR:** Open Science Data Repository. A collection of datasets from NASA-funded research, primarily focused on life sciences.
* **Streamlit:** A Python framework used to build and deploy data-centric web applications with minimal code. It is the foundation of our user interface.
* **MongoDB:** A NoSQL document database used as the primary data store for all scraped study information. We use MongoDB Atlas, the cloud-hosted version.
* **Neo4j:** A native graph database used to store and visualize the relationships between studies, organisms, factors, and assays.
* **Cypher:** The query language used to interact with the Neo4j graph database, similar to how SQL is used for relational databases.
* **Selenium:** A browser automation tool used by our web scraper to navigate the OSDR website and extract data.
* **Pyvis:** A Python library used to create and render the interactive network graphs from our Neo4j data.
* **Vertex AI:** (Planned Feature) Google Cloud's unified AI platform. We will use it for:
  + **Embeddings API:** To convert study text into numerical vectors for semantic search.
  + **Gemini API:** To power the generative AI model that compares and explains studies.
* **Vector Search:** (Planned Feature) An AI-powered search method that finds results based on conceptual meaning and context, not just keyword matching.

**3. Codebase Breakdown**

This is the structure of the core project files and their purpose.

* streamlit\_main\_app.py: The main entry point for the application. It defines the UI (tabs, sidebar, buttons), manages user session state, and orchestrates calls to all other modules.
* neo4j\_visualizer.py: A dedicated module that handles all interaction with the Neo4j database. It contains functions to run Cypher queries and generate the interactive graph visualizations using Pyvis.
* ingest\_to\_neo4j.py: A one-time utility script used to populate the Neo4j database. It reads data from a JSON file and builds the graph structure.
* scraper/: This directory contains all modules related to web scraping.
  + formatter.py: Defines the logic for parsing the HTML scraped from the OSDR website and formatting it into a clean data structure.
  + utils.py: Contains helper functions, including save\_to\_json and save\_to\_mongo.
* assets/: Contains static assets for the UI, primarily the custom SVG emoji icons.
* data/: The default output directory for the scraper. The osdr\_studies.json file is saved here.
* .env: A local file to store secret credentials like database connection strings and API keys. **This file should never be committed to Git.**
* gcp\_credentials.json: The secret key file for authenticating with Google Cloud Platform services. **This file must also be in .gitignore**.
* requirements.txt: A list of all Python libraries required to run the project.
* Dockerfile (To be created): A file that will contain instructions to build a container image of the application, making it ready for deployment on Google Cloud Run.

**4. How to Run the Application**

**As an Admin / Developer (Initial Setup)**

This covers the complete process of setting up the project from scratch on a local machine.

1. **Prerequisites:** Ensure you have Python 3.10+, Git, and Neo4j Desktop installed.
2. **Clone Repository:** git clone [your-gitlab-repo-url]
3. **Setup Environment:**
   * Navigate into the project directory: cd MONGO\_DB\_NASA\_OSDR
   * Create and activate a Python virtual environment:

Bash

python3 -m venv venv

source venv/bin/activate

* + Install all required libraries: pip install -r requirements.txt

1. **Configure Credentials:**
   * Create a .env file in the root directory.
   * Add your MONGO\_URI, NEO4J\_URI, NEO4J\_USER, and NEO4J\_PASSWORD to this file.
   * Follow the GCP setup steps to create and download your gcp\_credentials.json file and place it in the root directory.
2. **Start Databases:**
   * Ensure your MongoDB Atlas cluster is active.
   * Start your local database instance in the Neo4j Desktop application.
3. **Run the Full Data Pipeline:**
   * **Step A (Scrape Data):** Launch the Streamlit app (streamlit run streamlit\_main\_app.py). Go to the "Data Extract & Store" tab and click the "Fetch All... Studies" button. Wait for it to complete. This populates MongoDB and creates data/osdr\_studies.json.
   * **Step B (Ingest to Neo4j):** Stop the Streamlit app. Modify ingest\_to\_neo4j.py to point to data/osdr\_studies.json. Run the script from your terminal: python3 ingest\_to\_neo4j.py. Wait for it to complete.
4. **Run the Application:**
   * Launch the fully populated application: streamlit run streamlit\_main\_app.py.

**As a User (Interacting with the Deployed App)**

This describes how an end-user interacts with the final, deployed application.

1. **Access the App:** Open a web browser and navigate to the public URL provided by Google Cloud Run.
2. **Explore the Tabs:**
   * **🧬 Data Extract & Store:** This tab is primarily for admin use to populate the database. As an end-user, you can view the status of the last data fetch.
   * **📚 Study Explorer (MongoDB):** This is the main discovery tab. Use the filter boxes to search for studies by organism, experimental factor, or Study ID. Click "Search Studies" to see results. Click on any study to expand its details.
   * **🕸️ Knowledge Graph (Neo4j):** This tab is initially empty. To use it:
     1. First, find a study of interest in the "Study Explorer" tab.
     2. Click the "👁️ View Study Knowledge Graph" button for that study.
     3. Now, switch to the "Knowledge Graph" tab to see the interactive visualization for your selected study.
     4. Use the interactive query buttons below the graph to discover new connections and expand your view of the data.