ReportIQ – Documentation Generator

Bhaskara Hemanth Karthikeya Ganti| Data Analyst Intern | June 9th to August 9th 2025

Project Objective:

The objective of the **ReportIQ** project was to develop a system/framework for generating **documentation** and wiki pages in a standard format, with the option of allowing users to input **contextual information** for fine-grained control. The system aimed to support exporting documentation in formats such as **Markdown** and **DOCX,** and export clean ER diagrams to enhance integration with platforms like **Azure Devops** and **Github**.

Impact:

This project streamlined documentation by significantly **reducing the manual effort** needed and simplified further development through a **modular** architecture. It improved standardization, enabling faster onboarding to complex projects. Clean, downloadable **ER diagrams** made the data easier to understand, and real-time documentation **streaming** in the app improved responsiveness. Overall, the solution addressed key challenges like **manual, time-intensive** documentation and work needed for ER diagram creation.

Work Accomplished:

I developed a user-friendly **Streamlit** interface that allows users to select file types, upload **Power BI** reports , and generate documentation seamlessly with additional context if needed. The system processes **pbip** files by extracting report.json and .tmdl files from zipped **Report** and **SemanticModel** folders. I used **Factory** design pattern, particularly in processing different files and setting their contexts to make it more modular and easy for future use**. Dynamic** system and user prompts are generated based on file contents and optional user input, and documentation is produced in real-time using a **completion model API**. I also generated ER diagram based on the info in **tmdl** files Users can **download** the output in Markdown and DOCX formats, and the architecture is designed to **easily support** additional file types like PDFs and CSVs.

Key Learnings and Takeaway:

During the internship, I gained hands-on experience in **designing modular systems** using abstract classes and the **Factory design pattern**. I learned how to apply **prompt engineering** techniques to craft effective system and user prompts for guiding LLMs. I learned about **Streamlit** technology, integrating file handling, API communication, and dynamic UI components. I developed logic to parse tmdl and json files to generate clean, downloadable ER diagrams. Working in an **Agile** environment through Azure DevOps, I participated in **sprint planning**, stand-ups, and task management, which improved collaboration and **project tracking.**

Challenges Faced:

One of the key challenges was completing the application using **minimal** external packages, especially for **DOCX export**, to maintain simplicity and control. **Validating** uploaded **pbip zip files** and their internal tmdl and json contents required careful handling. Managing API errors due to exceeding the **maximum context length** (128K) was another technical hurdle. Generating **clean ER diagrams** in a simple format and enabling downloads added complexity. Through looking for **efficient solutions**, **workarounds** and through the **guidance** of my **colleagues and seniors**, I was able to overcome these and deliver a modular and architecturally sound solution.