Combined Application: Normal Chat, Plus (RAG), and Custom Assistant

Author: Niclas Svanström
Date: 2025-02-22

---

1. Introduction

This document describes an application that unifies three distinct services into a single, coherent platform. Each service addresses a different user need:

1. **Normal Chat**: A ChatGPT-like interface allowing users to switch between various models.
2. **Plus (RAG)**: An enhanced chat environment with **file search** capabilities (RAG) for retrieving information from user-uploaded documents.
3. **Custom Assistant**: A tool that lets users build their own task-specific assistants, sharing the same file storage as the Plus environment.

A key design goal is to **share files across all three apps**, simplifying management and improving user experience. This setup is a **work in progress**, continuously evolving to incorporate new features and improvements. Deployment relies on a **GitHub integration** that automatically updates the live environment whenever new code is pushed.

---

2. Normal Chat

The **Normal Chat** feature offers a **standard ChatGPT-like interface**. Users can type questions and receive conversational responses, with the following highlights:

- **Multiple Model Selection**
    - Users can seamlessly switch between different language models (e.g., GPT-4o, GPT-4o mini, or custom models) for varied responses or performance.
- **Prompt Flow Integration**
    - All interactions are linked to a Prompt Flow that provides insights into how well the chat model is performing.
    - This makes it easier for developers to fine-tune prompts or monitor quality.
- **Web Search Connection**
    - The chat can leverage a web search API for additional real-time information.
    - Enhances the chatbot’s capability to respond with up-to-date data (e.g., news, weather, or general web queries).

---

3. Plus (RAG)

The **Plus** version expands on the Normal Chat by adding **file search** (RAG) capabilities—allowing the user to query and retrieve insights from their uploaded documents. Key aspects:

1. **Retrieval-Augmented Generation (RAG)**
    - Users upload documents that are then indexed in a vector store.
    - When querying, the system retrieves relevant document chunks to ground answers in the user's own data, increasing accuracy.
2. **Assistant API Integration**
    - Employs the OpenAI Assistant API for easy orchestration and synergy with other parts of the platform.
    - Simplifies future expansions: for instance, hooking into other specialized tools or endpoints.
3. **No Code Interpreter**
    - The Plus environment focuses solely on **file-based retrieval**.
    - It does not include the code execution functionality that is available in the third app.
4. **Azure Blob Storage**
    - All files are stored in Azure Blob Storage to ensure they're future-proof and available across the entire platform.
    - This also allows the user to potentially reuse or share documents in other services later.

---

4. Custom Assistant

The **Custom Assistant** feature (described in a previous project) is integrated as the **third app** within this ecosystem. It allows users to:

- **Create Their Own Assistants**
    - Each assistant can have custom instructions or specialized domain knowledge.
- **Support for File Search and Code Interpreter**
    - Assistants can tap into the same RAG flow to retrieve data from the shared file store.
    - The Code Interpreter (Python environment) is accessible here for data analysis, generating plots, or running small scripts.

- **Shared File Access**
    - All files (e.g., PDFs, text documents) uploaded by the user to Blob Storage are instantly available for any custom assistant.
    - Promotes consistency and eliminates the need to re-upload documents across multiple services.

---

5. Shared File Storage & User Experience

A central design choice is to **share file storage** across all three applications. By storing all uploaded files in a single Blob Storage container:

- **Single Source of Truth**
    - Users only upload a file once. Any of the three apps can retrieve or reference that file as needed.
- **Unified Interface**
    - A single interface manages uploads, versioning, or metadata tagging, ensuring easy organization.
- **Future-Proof**
    - If new features require advanced usage of these files (e.g., in the custom Assistant or advanced search), no migration or re-indexing is necessary.

---

6. Continuous Development & GitHub Deployment

This application is **actively developed** with new features and enhancements added regularly. To streamline updates:

1. **GitHub Repository**
    - Code for all three integrated apps resides in a single (or monorepo-style) GitHub repository.
2. **Automatic Deployment**
    - Any push to the designated production branch triggers a continuous deployment pipeline.
    - No manual deployment is needed—this ensures quick iteration and user access to new features.
3. **Safe Rollbacks**
    - If an update introduces issues, reverting a Git commit automatically reverts the live environment, minimizing downtime.

---

7. Conclusion

This **Combined Application** merges a Normal Chat interface, a “Plus” environment (RAG with file search), and a Custom Assistant builder—**all sharing the same file storage backend**. By leveraging the OpenAI Assistant API, Azure Blob Storage, and a uniform GitHub-based deployment pipeline, the platform delivers a seamless user experience across diverse functionalities. It remains under active development, continuously adding new capabilities and refining existing ones to serve a broad range of user needs.

Contact Information

- **Name**: Niclas Svanström
- **Email**: Niclas.svanstrom@hotmail.com