**ACL permission and Query Assistant User Manual**

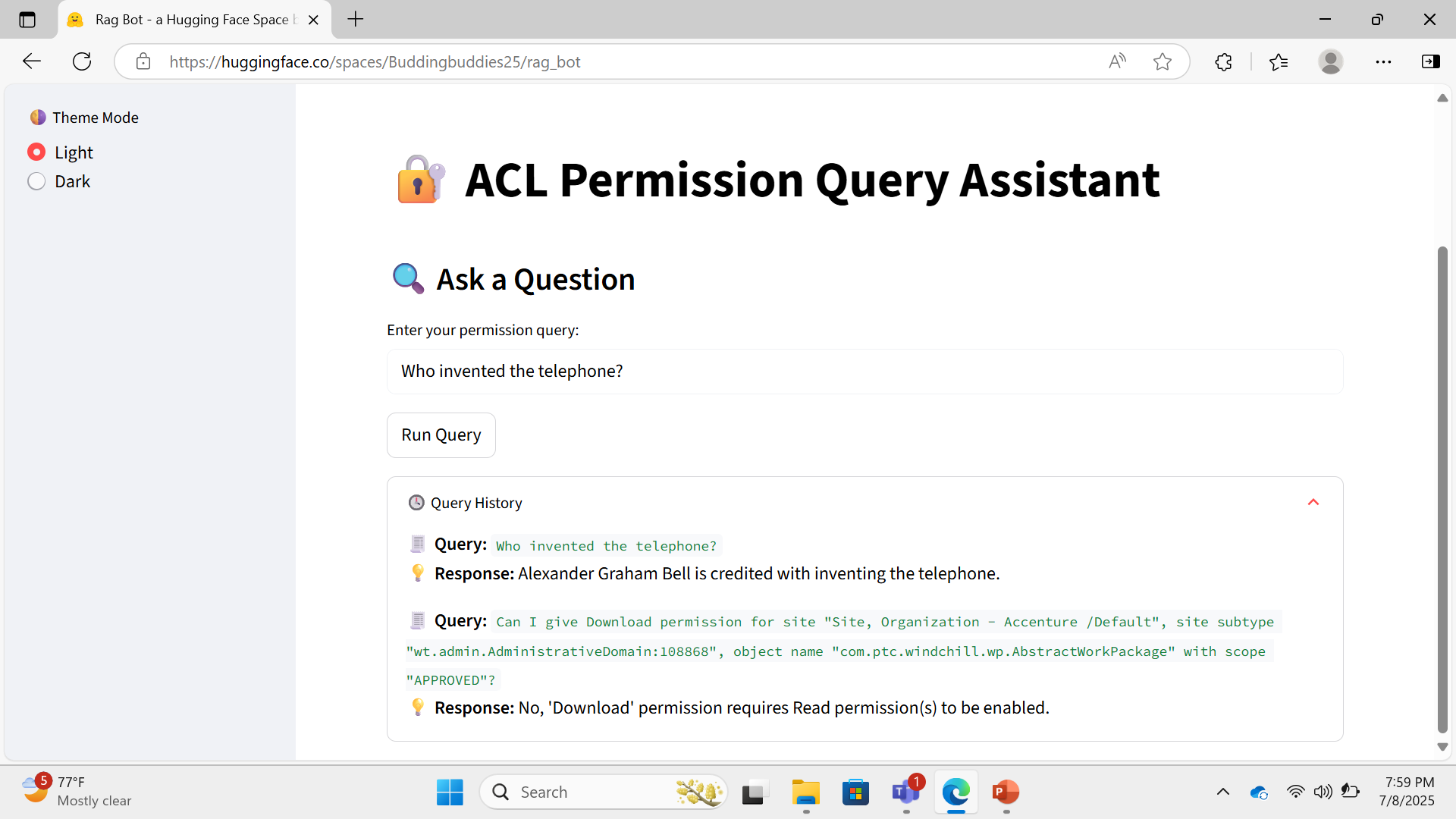
**Overview:-**

The ACL permission and Query Assistant is an AI-powered chatbot designed to help users interact naturally with complex access control data stored in an Excel spreadsheet. It allows users to ask questions about permissions, roles, and access scopes using simple language. The assistant intelligently combines:

* Semantic search of the Excel data using sentence embeddings and FAISS to find relevant entries.
* Local language model (Mistral) for understanding general questions and generating human-like responses.

This hybrid approach ensures accurate, context-aware answers for both specific permission queries and broad general knowledge questions — all running locally without requiring internet access for the AI model.

The assistant features a user-friendly web interface built with Streamlit, including light/dark modes and session history, making it easy to explore and audit access permissions interactively.

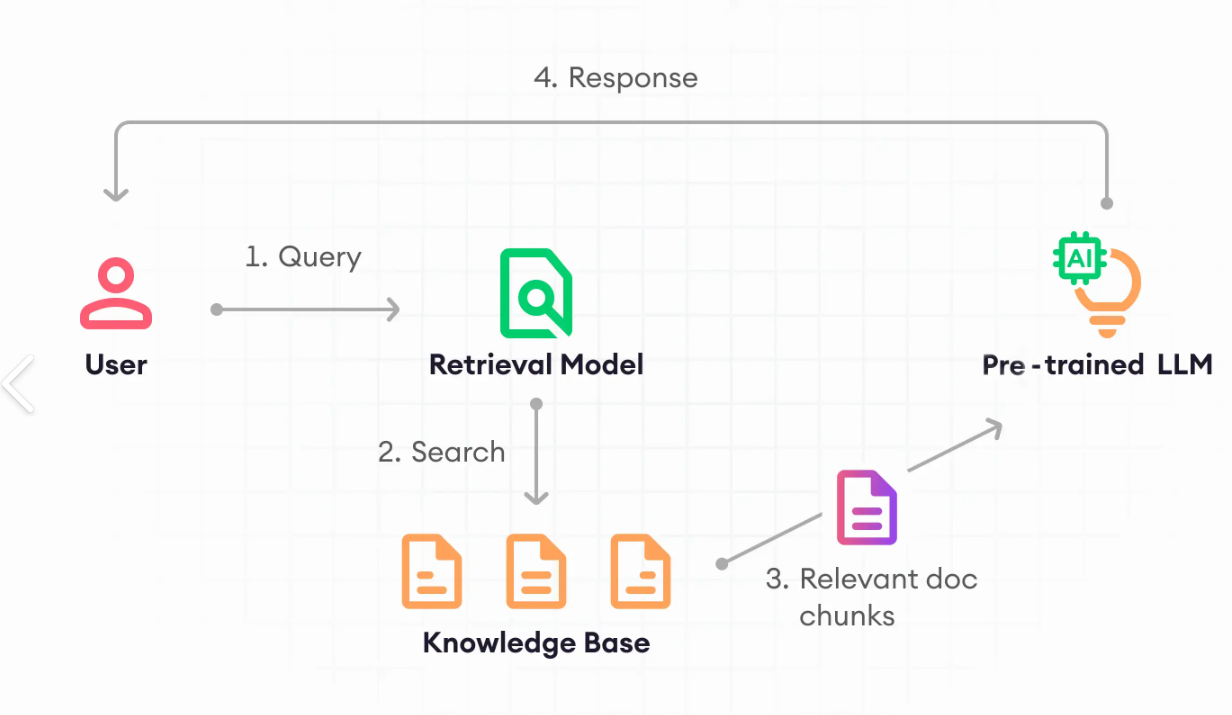


**RAG (Retrieval-Augmented Generation):-**

The ACL permission and Query Assistant uses a Retrieval-Augmented Generation (RAG) approach to answer questions effectively by combining two key components:

1. Retrieval:  
   When a user asks a query related to permissions, the system first searches relevant data from the Excel file. This is done by converting all Excel rows into numerical embeddings using a sentence transformer model, then quickly finding the closest matches using a FAISS vector index. This step ensures the assistant focuses on the most relevant permission records.
2. Generation:  
   The retrieved data snippets are then added into a prompt that is sent to a language model (like the Mistral model). The language model generates a natural language answer based on the context from the retrieved data combined with the user's question. For general questions unrelated to Excel data, the assistant directly uses the language model to generate responses.

This blend of retrieval and generation allows the assistant to provide accurate, context-aware, and fluent answers without needing to encode all knowledge inside the model itself.



**Folder Structure:-**

rag\_bot/

├── src/

│ ├── streamlit\_app.py # ✅ Streamlit UI interface

│ ├── rag\_excel\_bot.py # ✅ Core logic: FAISS, Excel parsing, query handling

│ └── mistral\_inference.py # ✅ Wrapper for running Mistral GGUF model

│

├── permissions.xlsx # ✅ Input Excel file containing permission data

├── requirements.txt # ✅ List of Python dependencies

└── README.md # ✅ Project overview and setup instructions

**Technologies Used:-**

|  |  |  |
| --- | --- | --- |
| **Component** | **Purpose** | **Notes on Deployment** |
| **pandas** | Excel data handling | Lightweight, no issue |
| **SentenceTransformers** | Embeddings for semantic search | Needs CPU, OK on free tiers |
| **faiss** | Fast nearest neighbor search | CPU only fine, works on free tiers |
| **transformers** | FLAN-T5 QA model inference | Small-medium model, can run on free tier if model size is small |
| **mistral\_inference** | Mistral 7B GGUF model inference | Heavy CPU inference, cannot run on free shared platforms directly |
| **Streamlit** | UI | Perfect fit for Streamlit Cloud or HF Spaces |
| **Others:** regex, pickle, torch, os | Support utilities | No deployment issues |

**Architecture Diagram:-**

A diagram of a computer

AI-generated content may be incorrect.

The assistant follows a Retrieval-Augmented Generation (RAG) architecture, which combines traditional search with modern AI language generation. Here's how the components work together:

1. User Input

* The user types a natural language query into the Streamlit interface.
* The system checks whether the query is related to permissions (Excel) or a general question.

2. Query Type Routing

* Excel-related query: Goes through semantic search using a sentence transformer and FAISS.
* General query: Directly passed to the Mistral language model for generation.

3. Retrieval (for Excel Queries)

* The query is embedded using a sentence transformer (all-MiniLM-L6-v2).
* FAISS (a fast vector search library) finds the most relevant rows from permissions.xlsx.

4. Prompt Augmentation

* The retrieved context is injected into a prompt.
* This prompt is passed to a language model (e.g., FLAN-T5 or Mistral).

5. Generation

* The language model generates a human-like, context-aware answer.

6. Response Display

* The final answer is shown in the Streamlit UI.
* It supports dark/light themes and query history.

**Sources utilized:-**

|  |  |  |
| --- | --- | --- |
| **Source Type** | **Details** | **Purpose in Methodology** |
| **Primary Source** | ✅ **Windchill Domain Policy Excel File** (permissions.xlsx) | Extracted raw permission data like roles, users, object names, and access rights. This dataset formed the **core knowledge base** of the assistant. |
| **Primary Source** | ✅ **User Queries** (entered via UI) | Captured and analyzed real-time user questions to test the assistant's ability to detect intent, classify query type (general vs permission-based), and return meaningful results. |
| **Secondary Source** | ✅ **PTC Windchill Documentation** | Used to understand domain-specific terms like ACL, Download, Modify, Reference Types, and permission inheritance logic. Helped design accurate filtering logic. |
| **Secondary Source** | ✅ **OpenAI Documentation (LLMs)** | Used for understanding how to integrate models like FLAN-T5 and Mistral for both factual and domain-specific queries. |
| **Secondary Source** | ✅ **GitHub Issues & Forums (e.g., Hugging Face, llama-cpp)** | Helped resolve technical issues like GGUF model loading, FAISS indexing, and deploying large models. These guided environment setup and inference logic. |

**Installation Steps:-**

- To set up the environment locally.

git clone <https://github.com/Buddingbuddies25/rag_bot.git>

cd rag\_bot.git

python -m venv rag\_env

source rag\_env/bin/activate   # or rag\_env\Scripts\activate on Windows

pip install -r requirements.txt

streamlit run src/streamlit\_app.py

- Model will auto-download from Hugging Face on first run.

- Excel file must be present at root level.

**Cloud Deployment:-**

1. Prepare Folder Structure

Ensure your folder is named rag\_bot and contains:

A screenshot of a phone

AI-generated content may be incorrect.

2. Create a Hugging Face Space

* Visit https://huggingface.co/spaces
* Click "Create New Space"
* Choose:
  + SDK: Streamlit
  + Hardware: CPU Basic
  + Visibility: Public or Private

3. Upload Files to Space

* Go to Files and Versions > Upload files
* Upload your entire rag\_bot/ content

4. Configure App File (if needed)

* Go to Settings > Repository structure
* Set the app file path: src/streamlit\_app.py

5. Requirements.txt

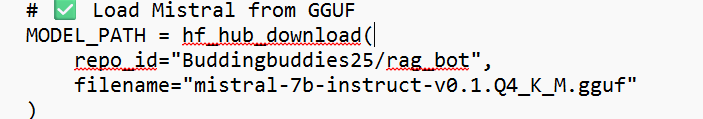
Ensure your requirements.txt includes:

A screen shot of a computer code

AI-generated content may be incorrect.

6. Load GGUF Model from Model Hub

Use the following inside rag\_excel\_bot.py:



7. Launch

* Hugging Face will auto-install dependencies and launch your app.
* Wait for the first-time download and build.
* After ~2–5 mins, your Streamlit app UI will appear.

**GitHub Push Instructions:-**

1. Initialize git in your project folder:

A close-up of a computer screen

AI-generated content may be incorrect.

1. Add your remote repository URL:



1. Add all files and commit:

A close-up of a white background

AI-generated content may be incorrect.

1. Push your code to GitHub:

A close up of words

AI-generated content may be incorrect.

1. Check your GitHub repository online to confirm files uploaded.

**Our Deployment:-**

[Rag Bot - a Hugging Face Space by Buddingbuddies25](https://huggingface.co/spaces/Buddingbuddies25/rag_bot)

**Example queries:-**

-How many entries have modify permission?  
-What are the unique values in Scope?  
-Can I give Download permission for site "Site, Organization - Accenture /Default", site subtype "wt.admin.AdministrativeDomain:108868", object name "com.ptc.windchill.wp.AbstractWorkPackage" with scope "APPROVED"?  
-Show permissions for site "Site, Organization - Accenture /Default", site subtype "wt.admin.AdministrativeDomain:108868", object name "com.ptc.windchill.wp.AbstractWorkPackage" with scope "REWORK"?  
-Count total number of read permissions  
-Find unique groups  
-List unique users  
-What permissions are assigned to scope "APPROVED"?  
-Which permissions are most used?  
-Which permissions are least used?

- What is artificial intelligence?

- How many legs does a cat have?

-Who invented Telephone?

-Explain French revolution.

**Troubleshooting:-**

|  |  |  |
| --- | --- | --- |
| **Issue** | **Error / Problem** | **Solution** |
| **App Not Starting** | FileNotFoundError: permissions.xlsx not found | Ensure permissions.xlsx exists in root folder or correct path. Upload it with your repo. |
| **Model File Not Found** | No such file or directory: mistral-7b-instruct.gguf | Use hf\_hub\_download() to load model from Hugging Face. Ensure .gguf model exists or is uploaded. |
| **Memory Exceeded** | Deployment crashes due to memory limit | Use smaller models (MiniLM, quantized Mistral), reduce load, or switch to Hugging Face CPU Basic. |
| **Permission Denied (Cache)** | PermissionError: huggingface/.cache | Set Hugging Face cache directory to /tmp with os.environ['TRANSFORMERS\_CACHE'] = '/tmp'. |
| **Streamlit Port or File Issues** | Invalid value: File does not exist: app.py or port errors | Use correct path: streamlit run src/streamlit\_app.py. Set --server.port=$PORT in cloud platforms. |
| **Blank Screen / Very Slow Load** | App loads slowly or shows nothing | Check logs, use st.spinner, avoid embedding large files at startup. |
| **Incorrect / Empty Query Result** | Excel-based queries return no data or wrong answer | Validate Excel headers, rebuild FAISS index, and regenerate embeddings. |
| **Hugging Face Space Not Loading** | Build takes too long or gets stuck | Avoid large files (>2GB), use CPU instead of GPU, simplify requirements.txt. |

**Output:-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**