**Project Documentation and Installation Guide**

**System Specifications:**

|  |  |  |
| --- | --- | --- |
| **Component** | **Recommendation**  **(Tested on this PC)** | **Minimum Requirements** |
| Operating System | Windows 10 / 11 (64-bit) | Windows 10 / macOS / Linux (64-bit) |
| Processor (CPU) | AMD Ryzen 9 / Intel i7 / i9 | |  | | --- | |  |  |  | | --- | | Intel i5 / AMD Ryzen 5 or higher | |
| RAM | 32 GB | Minimum 16 GB |
| GPU | NVIDIA RTX 3080 (10 GB VRAM) | NVIDIA GTX 1660 / RTX 2060 or higher |
| Python Version | Python 3.10 | Python 3.8+ |
| Storage | 1 TB SSD (fast read/write) | At least 10 GB free disk space |
| Dependencies | Streamlit, OpenAI SDK, ChromaDB, PyMuPDF, etc. | Same, via requirements.txt |
| Internet | Required for OpenAI API + model downloads | Required |
| Jupyter Notebook | Installed via Anaconda / pip | Optional (for visualization) |

**Installation Guide:**

1. **Clone or Download the Repository from** [**Here**](https://github.com/cspedalapu/ASKME--AI-Support-for-Knowledge-Management---Engagement)

*git clone https://github.com/your-repo-url/CampusGPT.git*

*cd ASKME*

1. **Create a Virtual Environment**

*python -m venv .venv*

1. **Activate the Environment**

***Windows:*** *source .venv/bin/activate*

1. **Install Dependencies**

*pip install -r requirements.txt*

1. **Set Environment Variables and Configure**

*Place your API key in environment file.*

1. **Run Streamlit UI Application**

*streamlit run app.py*

1. **Command-Line Test Query Interface**

*python pipeline/query\_terminal.py*

1. **Visualize Embedding Metrics (Optional)**

Run*visualize\_embedding.ipynb* for visualization. Make sure to assign the right path.

1. **Embedding New Data**

To embed and process raw data using recursive strategy:

*python pipeline/recursive\_embedder.py*

1. **Optional Development Tools**

Test chunk analysis, embedding vectors.

**Additional Notes:**

* All raw data files are placed in the data/ folder as shown in directory before embedding.
* All logs, performance analysis, preprocessing results and embedding outcomes save in output/ folder.

**Summary of ease of testing the files:**

|  |  |
| --- | --- |
| **Task** | **Command** |
| Run Streamlit App | streamlit run app.py |
| Terminal QA Testing | python pipeline/query\_terminal.py |
| Embed New Data | python pipeline/recursive\_embedder.py |
| Visualize Chunks & Embeddings | Open visualize\_embeddings.ipynb |
| Clean Input Data | python pipeline/save\_cleaned.py |

**Project/ File Structure:**

A screenshot of a computer

AI-generated content may be incorrect.

**Project File Overview:**

|  |  |
| --- | --- |
| **File / Folder** | **Purpose** |
| .env.example | To Store API Keys |
| requirements.txt | List of all Python dependencies required to run the project |
| run\_app.bat | Windows batch script to quickly launch the Streamlit app |
| app.py | User Interface to run on Streamlit |
| output/ | Stores Chroma vector DB, embedding logs, and analysis results |
| data/ | To store raw and processed data. |
| analysis/visualize\_embeddings.ipynb | Jupyter notebook for visualizing chunk distribution and embedding metadata |
| frontend/, other/ | Reserved for future frontend components or utility code |

**Pipeline Folder Breakdown (/pipeline/)**

|  |  |
| --- | --- |
| **File** | **Importance** |
| cleaner.py | Cleans raw documents (removes noise like headers/footers) |
| embedder.py | Embeds data using Sentence Transformers or other model |
| recursive\_embedder.py | Main script that applies RecursiveCharacterTextSplitter + embedding |
| save\_cleaned.py | Saves cleaned documents to a JSON file for further processing |
| loader.py | Loads raw files from folders for chunking/embedding |
| for\_chunk\_analysis.py | Computes chunk-level statistics and generates chunk metadata |
| search\_model.py | Loads a query, performs similarity search on vector store |
| rag\_engine.py | Full RAG engine that retrieves context and calls LLM to generate answers |
| query\_terminal.py | Command-line interface to test questions on embedded knowledge |
| test\_query\_recursive.py | Tests query performance on recursively chunked data |
| test\_query\_reranker\_recursive.py | Same as above, but adds a reranking step before final answer |
| test\_query\_compare\_recursive.py | Compares outputs of different query modes (vector-only vs rerank) |
| model\_router.py | Switches between OpenAI / local model based on config (future extensibility) |

**Output & Logs**

|  |  |
| --- | --- |
| **File** | **Details** |
| chroma\_store\_recursive/ | Persistent vector DB (used in querying and retrieval) |
| chunk\_analysis\_recursive.json | Stats like average tokens, max tokens, etc. |
| recursive\_vectors.json | Stores the embedding output for each chunk |
| recursive\_chunk\_log.json | Logs generated during recursive chunking |
| rag\_qa\_logs.json | |  | | --- | |  |   Stores all query-answer pairs for evaluation during testing on query |
| rag\_ui\_query\_log.json | Captures Streamlit-based user interaction logs |

**User Interface and Testing:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

Captures user interactions from the Streamlit interface and stores them in a backend .json file for future training and analysis.**A computer screen shot of a keyboard

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**