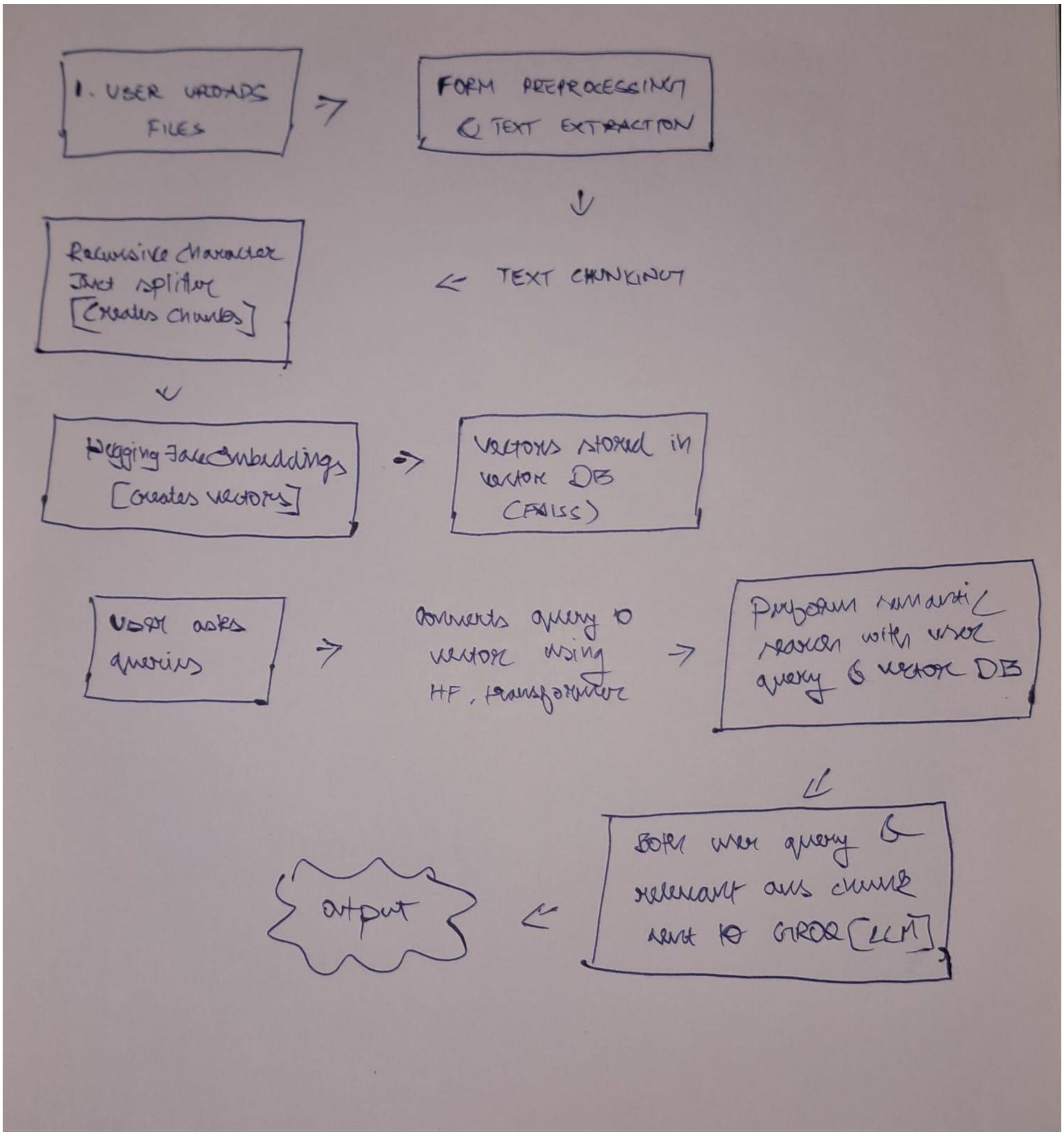
Workflow:


1. USER UPLOADS FILES
FORM PREPROCESSING & TEXT EXTRACTION
TEXT CHUNKING
Recursive Character Text splitter [Creates Chunks]
Hugging Face Embeddings [Creates vectors]
Vectors stored in vector DB (FAISS)
User asks queries
Converts query to vector using HF, transformer
Performs semantic search with user query & vector DB
Both user query & relevant ans chunk sent to GROQ [LLM]
output

Intelligent Form Agent Workflow

1. **File Upload**

- Users upload one or more files via **Jupyter FileUpload widget**.
- Supported formats:
  PDF, DOCX, CSV, Excel (XLS/XLSX), PNG/JPG/JPEG.
- Each uploaded file is temporarily saved to disk for processing.

---

2. File Type Detection

- load_document(file_name) determines the file type based on its extension.
- Supported types mapped to:
  - PDF → "pdf"
  - DOCX → "docx"
  - CSV → "csv"
  - Excel → "excel"
  - Images → "image"

3.Text Extraction

- **PDFs:** extracted using PyMuPDF (fitz) page by page.
- **DOCX:** extracted using python-docx from paragraphs.
- **CSV / Excel:** extracted using pandas, converted row-wise to text.
- **Images (PNG/JPG/JPEG):**
  - Preprocessed with grayscale + Gaussian blur + thresholding.
  - OCR performed via pytesseract.
  - Optional fallback OCR if preprocessed image returns empty text.

---

4. Content Aggregation

- Extracted text from each file is stored as a **Document object** (langchain.schema.Document).
- All documents collected into a list: all_content.

---

5. Text Chunking

- RecursiveCharacterTextSplitter splits each document into **chunks**:
    - Chunk size: 3000 characters
    - Overlap: 50 characters
- Ensures long documents are split for **embedding** and semantic search.

---

6. Embedding & Vector Store

- Chunks converted into **vector embeddings** via HuggingFaceEmbeddings:
    - Model: sentence-transformers/all-MiniLM-L6-v2 (CPU-friendly)
- Vectors stored in **FAISS local vectorstore** (vectorstore/db_faiss):
    - If store exists, it’s **loaded**.
    - If store doesn’t exist, it’s **created** from chunks.
- Skips empty text chunks to prevent vectorstore creation errors.

---

7. Semantic Search

- User queries are run through:
  vector_store.similarity_search(query, k=100)
- Returns the **top relevant chunks** as context for the LLM.

---

8. LLM Query & Answer Generation

- **Groq LLM** used with a **prompt template**:
    - Inputs: context chunks + user question
    - Output: concise, context-aware answer
    - If answer not found in documents, responds:

“I don't know from the current knowledge base.”

---

9. User Interaction / Jupyter UI

- **Query box:** user types questions.
- **Output area:** scrollable display for long answers.
- **Process button:** triggers vectorstore creation from uploaded files.
- **File upload widget:** allows multi-file upload.

---

10. Cleanup & Maintenance

- Old vectorstore and embeddings are deleted on:
    - atexit
    - OS signals (SIGINT / SIGTERM)
- Ensures clean state for repeated runs.
