Intelligent Form Agent — Workflow Documentation

Github :- https://github.com/AbdurRahman22224/Intelligent-Form-Agent

Video Demo :- https://drive.google.com/file/d/1ROvQi84aFawzthiMV1nm4AgvXkIxQY40/view?usp=sharing

Name : Abdur Rahman

Reg No : 2022UGPI007

Email : abdurrahman22224@gmail.com

Tech Stack

1. Pyhton
2. Streamlit
3. Pytesseract
4. google-generativeai (Gemini API)
5. Pillow

This document captures the end-to-end journey of the project, highlighting key decisions, experiments, and implementation milestones in chronological order.

1. Notebook Prototyping & Research

1. **Environment bring-up on Colab**
    - Installed `google-generativeai`, `pymupdf`, `pytesseract`, `Pillow`, and `regex`.
    - Configured the `GENAI_API_KEY`.
    - Verified OCR dependencies (PyMuPDF for PDFs, Tesseract for text extraction).
2. **OCR experiments**
    - Built `pdf_first_page_to_pil` helper to rasterize the first page of PDFs via PyMuPDF.
    - Created `ocr_bytes_to_text` to run Tesseract on either PDFs or image uploads.
    - Validated extraction on sample job application and loan forms.
3. **Gemini prompt experiments**
    - Crafted the original `call_gemini` wrapper with retries, truncation-on-retry, and multi-path response decoding (text/candidates / raw JSON).
    - Developed the unified prompt (`UNIFIED_SYSTEM`) to support both single-form Q&A and multi-form comparisons with explicit JSON schemas.
    - Added a dedicated summary prompt prototype in the notebook.

2. Repository & File Structure

1. **Skeleton layout**

```
src/
  llm/            # PyMuPDF + Tesseract pipeline
  ocr/            # File storage helpers
  qa/             # Unified LLM orchestration
  utils/          # Gemini wrapper + prompts
app.py            # Streamlit interface
docs/
  DESIGN.md
  QUICKSTART.md   # step-by-step setup/run instructions
test_gemini.py    # Manual call validation
setup_check.py    # Environment verifier
data/
  forms_db/       # Persistent storage
```

3. OCR & Storage Layer

1. **`src/ocr/ocr.py`**
    - `ocr_file()` automatically detects PDF vs image, renders the first page if needed, and passes a PIL image to Tesseract.
    - Normalizes line breaks and trims whitespace for consistent downstream consumption.
2. **`src/utils/storage.py`**
    - `save_form()` persists both raw uploads and `ocr_text.txt` under `data/forms_db/<uuid>/`.
    - `load_all_forms_with_names()` returns `{ form_id: { filename, ocr_text } }`, ensuring the UI can show human-friendly file names.
    - Added `get_form_filename()` helper used internally for filename lookups.

4. LLM & Prompt Infrastructure

1. **`src/llm/gemini.py`**
    - `call_gemini()` handles API key loading, prompt concatenation, retries, optional truncation on retry, and multiple fallback strategies for extracting model output.
    - `UNIFIED_SYSTEM` prompt: instructs Gemini to produce strict JSON for both single-form answers and multi-form aggregations, with confidence levels and evidence.
    - `SUMMARY_SYSTEM` prompt: dedicated summarization contract returning a concise summary, key fields, warnings, and form type.
2. **`src/qa/unified.py`**
    - `_label_and_truncate_forms()` prepares a labelled bundle of OCR text per form while capping length to avoid token overrun.
    - `unified_form_query()` constructs the prompt, invokes `call_gemini()`, and runs a robust JSON parsing pipeline (direct parse, `<JSON>...</JSON>`, Markdown code fences, regex extraction).
    - Returns `{ success, result, raw }` so the UI can render friendly output or inspect raw JSON.

5. Streamlit UI (`app.py`)

1. **Upload Page**
    - Drag-and-drop up to 3 PDFs/images, immediate OCR processing, and preview of the extracted text.
    - On upload completion, invokes `save_form()` to persist the source file + OCR text.
2. **Ask Questions Page**
    - Loads all stored forms and exposes a multi-select dropdown so users can target specific files.
    - "Ask Question" button calls `unified_form_query()`; results are rendered with:
        - Answer text + confidence badges.
        - Evidence snippets with real file names.
        - Optional raw JSON toggle for debugging.
    - "Generate Summary" button builds a summary prompt (single vs multi-form aware) and uses `call_gemini(SUMMARY_SYSTEM, ...)`.
    - Summaries display structured details (summary paragraph, key fields, warnings, form type) with an optional raw JSON view.

6. Diagnostics & Tooling

1. `setup_check.py`
    - Verifies Python dependencies, Tesseract installation, `.env` presence, folder structure, and points to `test_gemini.py` as a final verification step.
2. `test_gemini.py`
    - Minimal script that feeds a sample OCR block + question into `call_gemini()`.
    - Confirms the API key, model configuration, and JSON parsing are functional.

7. Final Cleanup & Documentation

1. **Documentation refresh**
    - Updated `README.md`, `QUICKSTART.md`, `DESIGN.md`, and `PROJECT_HISTORY.md` to describe the final architecture and tooling (with README now deferring to the Quick Start guide for run instructions).
    - Removed references to unused test scaffolding, kept instructions focused on `streamlit run`, `setup_check.py`, and `test_gemini.py`

.

2. **Code hygiene**
    - Eliminated unused exports, placeholder scripts, and debug prints.
    - Verified `setup_check.py` and `test_gemini.py` run successfully in the final environment.

Next Steps (Future Enhancements)

1. Multi-page PDF OCR or per-page selection.
2. Optional vector search for large repositories.
3. Fine-tuned summarization per form type.
4. Automated regression tests around `call_gemini` (mocking or cached responses).
