Phase 1: OCR and Text Extraction Pipeline (1900–1950)
Objective:
To extract clean machine-readable text from scanned PDF pages of Minneapolis city directories (1900–1950).

Pipeline Implementation:

1. PDF to Image Conversion
    - Converted multi-page city directory PDFs into high-resolution images (300 DPI) using `pdf2image` in Python.

2. Preprocessing for OCR:
    - Applied grayscale conversion, CLAHE contrast enhancement, Gaussian blurring, and adaptive thresholding.
    - Implemented vertical line detection to split multi-column pages (handled both 2-column and 3-column layouts dynamically).

3. Column Cropping:
    - Initially tested fixed-width splits and line-detection based cropping.
    - After facing inconsistency issues (false line detections, ad-columns), implemented tall-line height filtering and center-nearest or multi-line boundary-based cropping to generalize for both 2-column and 3-column layouts.

4. OCR Text Extraction:
    - Used `pytesseract` for text extraction from each cropped column.
    - Saved outputs as year-wise `.txt` files (e.g., `1906.txt`, `1912.txt`, etc.).

Challenges in Phase 1:

- Highly inconsistent scan quality between years (faded text, noisy backgrounds, varying contrast).
- Dealing with page layout variability (some years had ads on margins, others had double or triple column layouts)
- Excessive false-positive vertical line detections, requiring multiple iterations of filtering logic.
- Managing file naming consistency across over 50 years of data.

Phase 2: LLM-Assisted Address-Specific Resident Timeline Extraction (1807 Dupont Ave S)
Objective:
From the OCR outputs, extract only the records for residents of 1807 Dupont Ave S, year by year.

Pipeline Implementation:

1. Targeted Text Snippet Extraction:
    - Used regex-based pattern matching to locate all "1807" occurrences in each year's OCR text.
    - For each occurrence, extracted a 1000-character window (500 before and after) for LLM context narrowing.

2. LLM Verification and Extraction (Using Gemini 1.5 Pro API):
    - For each snippet:
        - Asked Gemini to verify if it truly referenced 1807 Dupont Ave S, handling OCR spelling variations.
        - Then asked Gemini to extract structured resident details: Name, Spouse, Occupation, Employer, and OCR-stated address line.
    - Used chunking and per-snippet prompting to reduce token cost and prevent LLM overload.

3. Aggregation:
    - Compiled year-wise results into structured JSON.
    - Saved both per-year JSONs and a combined full timeline file.

Challenges in Phase 2:

- OCR Noise: Address mentions often had OCR spelling errors (e.g., "Dupont av So", "Dupont ay s"), making regex and address filtering tricky.
- Gemini LLM Parsing Failures: Gemini sometimes returned non-JSON outputs or hallucinated answer formats, requiring retries and stronger prompt engineering.
- Token Limits: Some year OCR files were very large, needing careful chunking before sending to Gemini.
- Address Variations: Needed to fine-tune prompts to account for different address spellings (av, ave, avenue, south, s., etc.).

Future Advancements:

1. Fine-tuned OCR Models:
    - Explore using Google Document AI, or finetune Tesseract models on directory-specific fonts.
2. OCR Post-Correction:
    - Implement a spell-correction model trained on historical addresses and names to fix OCR errors before LLM prompting.
3. LLM Fine-tuning:
    - Fine-tune Gemini or OpenAI models with city directory-specific extraction tasks for higher JSON reliability.
4. Interactive Search Dashboard:
    - Build a Streamlit or React dashboard allowing historians to search any address across years.
5. Scalable LLM Query Handling:
    - Implement Gemini batch API calls with backoff, retry, and chunk result streaming for faster processing of thousands of snippets.
6. Multi-Address Batch Extraction:
    - Generalize pipeline to extract multiple target addresses in one run (not just 1807 Dupont Ave S).