Supporting "File Drop" Users in PolicyGPT

Recommended Approach: Smart Uploader + Adaptive Drafting

1. Accept messy inputs (multi-file upload)

Enable users to upload:

- Rough notes
- Meeting transcripts
- Old business cases or templates
- Slide decks or background docs

Use PolicyGPTs knowledge file upload or FastAPI `upload_input_files` route. Store these in a staging folder (e.g., per project + gate).

2. Preprocess with Insight Extractor pipeline

Before drafting, summarize and label the uploaded content:

- Convert files to plain text
- Chunk and embed each section or paragraph
- Use GPT to extract:
  - Project goals
  - Drivers and rationale
  - Constraints or assumptions
  - Stakeholders, risks, timelines
  - Relevant decisions, options, metrics

Result: a structured **project profile** in YAML or Markdown containing extracted insights.

3. Generate outline automatically from insight profile

Instead of user providing an outline manually, GPT auto-generates a tailored outline based on the insight summary.

Example:

> Based on the uploaded meeting transcript and briefing deck, here is a draft outline for the Gate 1 Business Case...

4. Chunk-based drafting from outline

PolicyGPT still uses a section-by-section generation process, based on the auto-generated outline and the insight profile.

Each section is drafted in isolation:

- Prompt = [outline section] + [relevant extracted insights] + [user files as needed]
- Inject only the relevant chunks into the context (avoid token overflow)
- Commit each section as its generated

5. Provide a draft pack summary

At the end, PolicyGPT can:

- Show a summary of what was generated
- Offer links to full drafts + editable files
- Flag low-confidence sections or missing inputs

6. Enable human review and round-tripping

If the user wants to tweak anything externally:

- Allow them to edit documents in Drive or Word
- GPT can fetch the revised section later and integrate or summarize changes
- Offer a Redline and Review mode for comparison

Token-Safe Architecture

- Never inject all files at once use chunking + retrieval
- Dont generate the whole doc in one go draft per section
- Accepts messy inputs, but produces structured, coherent outputs

Implementation Components to Add

| Feature | Description |

|---|---|

| `upload_user_files()` | Accepts multiple file types and routes them to preprocessing |

| `extract_insights()` | Converts raw input into structured YAML profile |

| `generate_outline_from_insights()` | Creates a smart, customized outline from the project profile |

| `section_generator()` | Drafts each section separately |

| `summarize_and_flag_gaps()` | QA summary and unresolved questions |

| `fetch_edited_section()` | Accepts revised content from the user for reintegration |

Why This Works

This approach:

- Respects the users workflow
- Produces coherent, review-ready docs
- Provides transparency into what was generated
- Avoids hallucinations and token overflows