# Proposed Plan

Here is a concise and structured plan for documenting the 6\_output.py module, following the same tone, segmentation, and clarity standards you've emphasized throughout this project.

**✅ Documentation Structure Plan for 6\_output.py → output.md**

**📘 Table of Contents**

1. [Overview](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#overview)
2. [System Architecture](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#system-architecture)
3. [Input Data Format](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#input-data-format)
4. [Core Functions](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#core-functions)
   * [main()](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#main)
   * [generate\_pdf\_report()](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#generate_pdf_report)
   * [create\_metadata\_table()](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#create_metadata_table)
   * [create\_answer\_section()](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#create_answer_section)
   * [create\_citations\_section()](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#create_citations_section)
5. [Styling and Design Choices](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#styling-and-design-choices)
6. [Error Handling](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#error-handling)
7. [Usage Examples](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#usage-examples)
8. [Appendix: Input JSON Schema](https://chatgpt.com/g/g-p-67fe726484c8819191dd1403485129af-ds205-group-project/c/68384556-2ad0-8002-b7fb-acc12c0110e3#appendix-input-json-schema)

**🔍 Visual + Code Snippet Plan (By Section)**

**1. Overview**

Short and to the point:

This module generates color-styled PDF fact sheets from LLM-generated JSON outputs. It organizes country-specific climate policy answers and citations into a readable document.

✅ No code snippet here.

**2. System Architecture**

📊 Mermaid Diagram:

graph TD

A[LLM JSON File] --> B[load\_llm\_response\_data()]

B --> C[generate\_pdf\_report()]

C --> D1[create\_metadata\_table()]

C --> D2[create\_answer\_section()]

C --> D3[create\_citations\_section()]

C --> E[ReportLab PDF Output]

✅ Mention:

* Uses ReportLab for layout and color styling
* Data source: data/llm/\*.json
* Output directory: outputs/factsheets/

**3. Input Data Format**

🗒️ Describe the expected structure of the LLM-generated JSON files.

* Top-level metadata
* Keyed questions object, each with:
  + question\_number
  + llm\_response → answer, citations, metadata

📌 Visual Suggestion: table or partial example block for structure:

{

"metadata": {...},

"questions": {

"question\_1": {

"llm\_response": {

"question": "...",

"answer": {"summary": "...", "detailed\_response": "..."},

"citations": [...]

}

}

}

}

**4. Core Functions**

📋 Top Table:

| **Function** | **Description** |
| --- | --- |
| main() | Orchestrates processing for all JSON files |
| generate\_pdf\_report() | Assembles all report sections |
| create\_metadata\_table() | Summarizes country metadata |
| create\_answer\_section() | Formats each Q&A block |
| create\_citations\_section() | Collates all citations across questions |

**main()**

llm\_files = get\_available\_llm\_files()

for json\_path in llm\_files:

country\_name, response\_data = load\_llm\_response\_data(json\_path)

generate\_pdf\_report(response\_data, output\_path, styles, country\_name)

* Scans folder, loads data, generates output filename
* Calls core PDF logic

**generate\_pdf\_report()**

doc = SimpleDocTemplate(output\_path, pagesize=letter)

elements = []

elements.append(create\_metadata\_table(...))

elements.extend(create\_answer\_section(...))

elements.extend(create\_citations\_section(...))

doc.build(elements)

🗒️ Builds the full document from content blocks.

**create\_metadata\_table()**

* Uses Table() to format:
  + Country
  + Citation Count
  + Timestamp
  + Question Count

✅ Code snippet (summary):

data = [["Country", country], ["Citations", citations\_count], ...]

table.setStyle(TableStyle([...]))

**create\_answer\_section()**

* Loops over all questions
* Handles:
  + Plain strings
  + Nested or malformed JSON
  + Escaped characters

✅ Example code logic:

summary = answer.get("summary")

detailed = answer.get("detailed\_response")

elements.append(Paragraph(summary, styles["SummaryStyle"]))

📌 Handles malformed responses defensively.

**create\_citations\_section()**

* Groups by country
* Removes duplicate citations
* Includes how\_used and similarity scores

✅ Example:

paragraph = Paragraph(f"[{i}] {content}", styles["CitationStyle"])

if similarity:

elements.append(Paragraph(f"Relevance Score: {similarity:.3f}", ...))

**5. Styling and Design Choices**

🗒️ Short bullet points:

* ReportLab styles are modular
* Consistent color schemes (COLORS dict)
* Rounded corners, clean borders
* Summary/detailed responses clearly differentiated

✅ Code snippet (style config):

ParagraphStyle(name='SummaryStyle', backColor=COLORS['summary\_bg'], ...)

**6. Error Handling**

✅ Code examples:

try:

country\_name, response\_data = load\_llm\_response\_data(json\_path)

except Exception as e:

print(f"[6\_OUTPUT] Error processing {json\_path}: {e}")

* Handles: missing files, malformed content, parse errors
* Logs clear output in console

**7. Usage Examples**

**CLI**

python 6\_output.py

* Processes all data/llm/\*.json files
* Creates 1 PDF per country in outputs/factsheets/

**Output format:**

[6\_OUTPUT] PDF report generated: outputs/factsheets/Singapore\_climate\_policy\_factsheet\_20250529\_162500.pdf

**8. Appendix: Input JSON Schema**

🗒️ Include a clean block showing the expected minimum:

{

"metadata": {

"country\_name": "Singapore",

"timestamp": "...",

...

},

"questions": {

"question\_1": {

"llm\_response": {

"question": "...",

"answer": {

"summary": "...",

"detailed\_response": "..."

},

"citations": [

{

"doc\_id": "...",

"content": "...",

"how\_used": "...",

"cos\_similarity\_score": 0.98

}

]

}

}

}

}

Would you like me to generate the final output.md file based on this plan now?

# Specifying Subsections

Here is a refined, section-by-section implementation plan for documenting 6\_output.py in output.md, designed for clarity, minimalism, and professional polish. This version integrates your prior feedback about:

* Clarity over verbosity
* Intuitive flow with justifications
* Tables to summarize structure
* Mermaid diagrams where helpful (fully supported by GitHub)

**✅ Final Section-by-Section Plan for output.md**

**1. 📘 Overview**

**Purpose:** Short explanation of what the script does, in one clean paragraph.

This module reads LLM-generated responses (in JSON) for each country and converts them into professionally formatted, citation-backed PDF fact sheets using ReportLab. It aggregates summaries, detailed answers, and citations under styled sections.

✅ **No code snippet.**  
✅ Keep concise. Avoid any generic statements or over-contextualization.

**2. 🧱 System Architecture**

📊 **Mermaid Flow Diagram** (top of section):

graph TD

A[LLM JSON File] --> B[load\_llm\_response\_data()]

B --> C[generate\_pdf\_report()]

C --> D1[create\_metadata\_table()]

C --> D2[create\_answer\_section()]

C --> D3[create\_citations\_section()]

D1 --> E[PDF File via ReportLab]

🗒️ This flow clearly shows:

* The lifecycle from raw data → full document
* Modular processing of metadata, Q&A, citations

✅ Add a sentence beneath explaining that each helper function populates a styled content block into the PDF.

**3. 🧾 Input Data Format**

🗒️ Brief text: LLM responses are structured as JSON with:

* top-level metadata
* a dictionary of questions, each containing:
  + question\_number, query\_text, llm\_response, citations

📋 **Mini JSON Snippet:**

{

"metadata": { "country\_name": "Singapore" },

"questions": {

"question\_1": {

"llm\_response": {

"question": "...",

"answer": {

"summary": "...",

"detailed\_response": "..."

},

"citations": [...]

}

}

}

}

**4. 🧩 Core Functions**

📋 **Top Summary Table**:

| **Function** | **Description** |
| --- | --- |
| main() | Entry point – scans files and orchestrates report generation |
| generate\_pdf\_report() | Builds the PDF layout from section components |
| create\_metadata\_table() | Adds country, question count, timestamp summary |
| create\_answer\_section() | Adds each Q&A with error handling and styling |
| create\_citations\_section() | Collates and formats citation content |

**4.1 main()**

📌 **Key Snippet:**

llm\_files = get\_available\_llm\_files()

for json\_path in llm\_files:

country\_name, response\_data = load\_llm\_response\_data(json\_path)

generate\_pdf\_report(response\_data, output\_path, styles, country\_name)

🗒️ Notes:

* Error logging on file failures
* Prints progress per country
* Generates output filenames using country + timestamp

**4.2 generate\_pdf\_report()**

📌 **Key Snippet:**

elements = []

elements.append(create\_metadata\_table(...))

elements.extend(create\_answer\_section(...))

elements.extend(create\_citations\_section(...))

doc.build(elements)

🗒️ Justify:

* Linear construction of the PDF flow using ReportLab's Platypus engine
* Each helper injects styled content blocks

📊 Diagram optional here — already shown in System Architecture

**4.3 create\_metadata\_table()**

📌 **Minimal Snippet:**

data = [

['Country', country\_name],

['Question Count', ...],

['Citations Count', ...],

['Generated', timestamp]

]

🗒️ Notes:

* Styled with alternating background colors
* One of the only tabular sections in the PDF (good use of ReportLab Table)

**4.4 create\_answer\_section()**

📌 **Key Snippet:**

for question\_num, question\_data in sorted(response\_data["questions"].items()):

answer = llm\_response.get("answer", {})

summary = answer.get("summary", "")

detailed = answer.get("detailed\_response", "")

🗒️ Explain:

* Handles malformed nested JSON strings gracefully
* Cleans escape characters
* Defaults to fallback message if summary is missing

📌 Optional mini table or logic flow:

| **Scenario** | **Handled With** |
| --- | --- |
| Well-formed summary/detailed | Direct styling |
| JSON-encoded strings | json.loads() and cleanup |
| Missing fields | Fallback paragraph warning |

**4.5 create\_citations\_section()**

📌 **Key Snippet:**

for citation in citations:

content = citation.get('content', 'No content')

how\_used = citation.get('how\_used', 'Referenced')

🗒️ Explain:

* Deduplicates content by string
* Groups by country
* Adds relevance score and “how used” statement below each citation

**5. 🎨 Styling and Design**

🗒️ Short bullet summary:

* All PDF styles are defined once in create\_pdf\_style()
* Named styles: CustomTitle, CustomHeading, SummaryStyle, etc.
* Background colors help distinguish types (answers vs citations)
* Text alignment, padding, spacing are customized for clarity

📌 **Key Snippet:**

styles.add(ParagraphStyle(name='SummaryStyle', backColor=COLORS['summary\_bg']))

📊 Optional table of styles and purpose:

| **Style Name** | **Used For** | **Background Color** |
| --- | --- | --- |
| SummaryStyle | Short answers | Light teal |
| DetailStyle | Long explanations | Light blue |
| CitationStyle | Source chunks | Light yellow |

**6. 🛡 Error Handling**

📌 **Key Snippet:**

except Exception as e:

print(f"[6\_OUTPUT] Error processing {json\_path}: {e}")

🗒️ Graceful handling of:

* Malformed JSON
* Missing files or fields
* Summary/detail fallback

**7. 🚀 Usage Examples**

# Generate reports for all countries in the data/llm folder

python 6\_output.py

✔️ Outputs go to:

/outputs/factsheets/<country>\_climate\_policy\_factsheet\_<timestamp>.pdf

No CLI arguments needed – fully self-contained.

**8. 🧾 Appendix: Input Schema (for Reference)**

📌 **Minimal Valid Example:**

{

"metadata": { "country\_name": "Singapore" },

"questions": {

"question\_1": {

"llm\_response": {

"answer": {

"summary": "...",

"detailed\_response": "..."

},

"citations": [...]

}

}

}

}

🗒️ This format is assumed. If structure changes, the module may break silently — worth noting in your documentation.

Let me know when you're ready for me to turn this into the actual output.md file.