Annual Report Section Mapping Automation

1. Problem Statement

Analysts manually map fragmented, noisy text extracted from annual report PDFs to predefined sections such as Overview, Business Strategy, Risk Factors, Management Discussion, and ESG. The text is often broken across rows, contains gibberish, and requires manual PDF cross-checking.

Goal: Build an automated, scalable pipeline that extracts clean, page-aware text from PDFs and semantically maps it to predefined sections with high accuracy.

---

2. Solution Overview (Pipeline)

```
PDF
 → Layout-aware Text Extraction
 → Cleaning & De-fragmentation
 → Paragraph Chunking
 → Semantic Section Classification
 → Structured Output (Excel / JSON / DB)
```

---

3. Project Folder Structure

```
annual-report-mapper/
│
├── data/
│   ├── raw_pdfs/
│   ├── extracted_text/
│   ├── processed_chunks/
│   └── output/
│
├── config/
│   └── section_definitions.json
│
├── src/
│   ├── 01_pdf_extraction.py
│   ├── 02_text_cleaning.py
│   ├── 03_chunking.py
│   ├── 04_section_classification.py
│   └── 05_export_results.py
```

```
│
├── notebooks/
│   └── exploration.ipynb
│
├── requirements.txt
└── README.md
```

---

4. Section Definitions (config/section_definitions.json)

Each section is defined semantically, not by exact keywords.

Example:

```
{
  "Overview": "Company background, core operations, high-level summary of performance",
  "Business Strategy": "Long-term goals, growth plans, competitive positioning, strategic priorities",
  "Risk Factors": "Operational, financial, regulatory, market and external risks",
  "Management Discussion": "Management commentary on performance, results, and outlook",
  "ESG": "Environmental, social, and governance initiatives, sustainability disclosures"
}
```

---

5. Phase 1 – PDF Text Extraction (src/01_pdf_extraction.py)

Objective: Extract text blocks with page numbers and layout awareness.

Tool: PyMuPDF (fitz)

Output Schema:

```
{
  "page": 12,
  "block_id": 3,
  "text": "Our business strategy focuses on long-term growth..."
}
```

Key considerations: - Preserve page numbers - Ignore images and tables initially - Capture block-level text

---

6. Phase 2 – Text Cleaning & De-fragmentation (src/02_text_cleaning.py)

Cleaning Rules

- Remove headers, footers, page numbers
- Remove repeated company name / report title
- Remove gibberish symbols and OCR noise

De-fragmentation Logic

- Merge lines if:
- Previous line does not end with punctuation
- Next line starts with lowercase
- Normalize whitespace

Output: Clean, readable paragraphs

---

7. Phase 3 – Paragraph Chunking (src/03_chunking.py)

Objective: Convert cleaned text into logical paragraph-level chunks.

Chunk structure:

```
{
  "chunk_id": "P12_04",
  "page": 12,
  "text": "Our business strategy focuses on expanding digital channels...",
  "char_length": 312
}
```

Benefits: - Better semantic understanding - One-to-many mapping between pages and sections

---

8. Phase 4 – Semantic Section Classification (src/04_section_classification.py)

Approach 1: Embedding Similarity (Baseline)

- Convert section definitions and chunks to embeddings

- Use cosine similarity
- Assign section with highest similarity above threshold

Approach 2: LLM-based Classification (Recommended)

Prompt-based classification with confidence scoring.

Expected Output:

```
{
  "chunk_id": "P12_04",
  "section": "Business Strategy",
  "confidence": 0.91
}
```

Low-confidence chunks can be flagged for manual review.

---

9. Phase 5 – Export & Reporting (src/05_export_results.py)

Final outputs: - Excel for analysts - JSON for downstream systems - Optional: SQLite / Parquet for BI tools

Excel Schema: | Section | Page | Text | Confidence | |------|------|------|------------|

---

10. Future Enhancements

- RAG-based question answering on annual reports
- Power BI dashboard for section-wise insights
- Streamlit UI for upload & review
- Multi-year report comparison

---

11. Success Criteria

- ≥90% reduction in manual effort
- High section-mapping accuracy
- Page-level traceability to original PDF
- Scalable to hundreds of reports

---

12. Next Implementation Steps

1. Implement PDF extraction script
2. Validate text quality vs Excel
3. Build cleaning & merging logic
4. Add semantic classification
5. Export analyst-ready output

This project is designed to be production-ready, not a demo.