Technical Design Document (TDD)

System Name

Automated Manhwa Crawling, OCR, and Translation Pipeline

Document Scope

This document translates the PRD into a concrete technical architecture, defining: - Modules and responsibilities - Data flow and interfaces - Agent design and prompt structure - Storage schemas - Error handling and extensibility points

This is a **research-grade, non-commercial system** intended for study and experimentation.

---

1. Architectural Overview

1.1 Design Principles

- **Pipeline-first**: each stage is isolated, testable, and replaceable
- **Vision-before-language**: spatial reasoning precedes LLM usage
- **LLMs as decision-makers, not detectors**
- **Human-inspectable artifacts at every stage**

1.2 High-Level Architecture

```
Streamlit UI
   ↓
Crawler ──→ Panel Extractor ──→ Stitcher
                                    ↓
                               OCR Pass #1
                                    ↓
                             Smart Split Engine
                                    ↓
                               OCR Pass #2
                                    ↓
                            OCR Metadata Store
                                    ↓
            ┌──────────────── Filter Agent (LLM)
            ↓
        Inpainting Engine
            ↓
      Translation Agent (LLM)
```

```
        ↓
Font & Layout Reconstruction
        ↓
Image Renderer & Composer
        ↓
   Streamlit Viewer
```

---

2. Module Breakdown

2.1 Streamlit UI Layer

Responsibilities

- Accept chapter URL input
- Display progress per pipeline stage
- Visualize intermediate artifacts
- Serve final translated output

Components

- `ui_input.py`
- `ui_progress.py`
- `ui_viewer.py`
- `ui_debug.py`

Interfaces

- Calls `pipeline.run(url, config)`
- Reads artifacts from workspace directory

2.2 Crawler Module

Responsibilities

- Load chapter pages
- Handle infinite scrolling
- Solve simple visual captchas
- Download only relevant assets

Components

- `crawler/browser.py` (Playwright / Selenium)
- `crawler/captcha_solver.py`
- `crawler/asset_filter.py`

Key Techniques

- DOM inspection for `<img>` tags
- CDN URL pattern filtering
- Size/aspect ratio heuristics

Output

```
List[PanelAsset]
PanelAsset:
  - panel_id
  - image_path
  - source_url
  - order_index
```

2.3 Panel Extraction & Stitching

Responsibilities

- Enforce ordering
- Normalize resolution
- Stitch panels vertically

Components

- `panels/extractor.py`
- `panels/stitcher.py`

Stitching Logic

- Sort by DOM order
- Pad width to max panel width
- Concatenate with no overlap

Output

```
StitchedImage:
  - image_path
  - panel_map: List[(y_start, y_end, panel_id)]
```

2.4 OCR Engine

OCR Pass #1 (Exploratory)

Purpose: identify text regions for safe splitting

OCR Pass #2 (Authoritative)

Purpose: produce final bounding boxes and text

Components

- `ocr/engine.py`
- `ocr/preprocess.py`
- `ocr/postprocess.py`

OCR Engine Requirements

- Vision-based (CNN/Transformer)
- Robust to stylized Hangul
- Outputs bounding boxes + confidence

Output Schema

```
OCRBox:
  - image_id
  - x, y, w, h
  - text
  - confidence
```

2.5 Smart Split Engine

Responsibilities

- Split stitched image into ≤100 panels
- Avoid cutting through text

Components

- `split/smart_split.py`

Algorithm

1. Detect candidate cut lines (horizontal whitespace)
2. Reject cut if intersects any OCRBox

3. Enforce ≥50px margin from nearest OCRBox
4. Balance panel heights

Output

```
List[SubPanel]:
  - subpanel_id
  - image_path
  - y_range
```

2.6 OCR Metadata Store

Responsibilities

- Persist OCR results
- Maintain image ↔ text mapping

Supported Backends

- CSV (default)
- SQLite (optional)

Tables / Files

ocr_boxes.csv - image_id - box_id - x, y, w, h - text - confidence

---

2.7 Filter Agent (Garbage Detection)

Role

Semantic classifier deciding whether OCR text is: - Dialogue / narration (KEEP) - SFX / watermark / noise (DROP)

Components

- `agents/filter_agent.py`

Inputs

- OCRBox data
- Bounding box size
- Optional image crop

Output

```
{
  "decision": "KEEP" | "DROP",
  "category": "dialogue" | "sfx" | "watermark" | "noise",
  "confidence": 0.0–1.0
}
```

Prompt Strategy

- Few-shot examples
- Conservative bias
- Strict JSON schema enforcement

2.8 Inpainting Engine

Responsibilities

- Remove kept text regions cleanly
- Preserve background

Components

- `inpaint/opencv_inpaint.py`
- `inpaint/mask_builder.py`

Techniques

- Mask dilation
- Edge-aware inpainting

2.9 Translation Agent

Role

- Translate Korean → target language
- Maintain tone, context, naming consistency

Components

- `agents/translation_agent.py`

Inputs

- Text to translate
- Neighboring dialogue
- Series metadata (optional)

Output

```
{
  "original": "…",
  "translated": "…",
  "tone": "casual/dramatic",
  "notes": "…"
}
```

Prompt Strategy

- Context window batching
- Style constraints
- Optional glossary injection

2.10 Font & Layout Reconstruction

Responsibilities

- Estimate font size and style
- Fit translated text into original box

Components

- `render/font_estimator.py`
- `render/layout_solver.py`

Heuristics

- Box height → font size
- Stroke width estimation
- Dynamic line wrapping

2.11 Image Rendering & Composition

Responsibilities

- Render translated text

- Bake into image
- Reassemble panels

Components

- `render/text_renderer.py`
- `render/composer.py`

---

3. Data & Workspace Layout

```
workspace/
  ├─ raw_panels/
  ├─ stitched/
  ├─ splits/
  ├─ ocr/
  ├─ filtered/
  ├─ inpainted/
  ├─ rendered/
  └─ final/
```

---

4. Error Handling & Recovery

- Per-panel try/catch
- Skip on failure, log error
- Resume from last successful stage

---

5. Extensibility Points

- Swap OCR engine
- Swap LLM provider
- Add RAG for series consistency
- Replace inpainting backend

---

6. Performance Expectations

- 1 chapter: 3–10 minutes
- OCR is dominant cost
- LLM calls batched per panel

---

7. Security & Legal Notes

- No credential storage
- Respect robots.txt where feasible
- Academic, non-commercial usage only

---

8. Summary

This TDD specifies a **modular, inspectable, agent-driven vision-language system** designed to translate manhwa chapters with minimal human intervention while preserving artistic integrity and research transparency.