Product Requirements Document (PRD)

Product Title

Automated Manhwa Crawling, OCR, and Translation System

Document Status

Internal / Academic Use Only
Non-commercial, study-oriented project protected under Indonesian law

---

1. Background & Motivation

Manhwa (Korean webcomics) present unique challenges for automated translation: - Highly stylized Hangul fonts - Vertical scrolling layouts with panel-based composition - Embedded sound effects (SFX), watermarks, credits, and non-dialogue text - Anti-crawling mechanisms (lazy loading, CDNs, simple visual captchas)

Previous iterations of the system revealed two critical insights: 1. **Panel stitching before splitting** significantly improves OCR consistency and spatial context. 2. **Document-focused OCR engines perform poorly** on stylized manhwa typography, necessitating smarter, vision-aware OCR approaches.

This PRD defines a high-level system to automatically crawl, process, translate, and re-render manhwa chapters end-to-end, exposed through a Streamlit-based UI.

---

2. Goals & Non-Goals

2.1 Goals

- Fully automate manhwa chapter translation from a pasted URL
- Extract only relevant manhwa panels (no ads, UI, comments, or junk assets)
- Preserve visual fidelity (font style, size, layout)
- Use LLM-based agents for:
- Semantic garbage detection (SFX, watermarks, credits)
- Context-aware translation
- Provide a visual, inspectable output via Streamlit

2.2 Non-Goals

- Commercial deployment or monetization
- DRM circumvention beyond basic academic crawling
- Support for non-image-based manhwa formats (e.g., PDF, EPUB)
- Perfect typographic recreation (best-effort matching is sufficient)

3. User Personas

Primary User

- Researcher / student studying OCR, LLMs, or multimodal pipelines
- Comfortable with technical tooling but prefers a GUI for iteration

Secondary User

- Developer experimenting with agent-based LLM workflows
- Interested in vision-language interaction and document reconstruction

---

4. High-Level User Flow

1. User pastes a manhwa chapter URL into Streamlit UI
2. System crawls and loads all chapter content
3. Manhwa panels are extracted and stitched vertically
4. OCR is performed on stitched image
5. Image is split into sub-panels (safe-cut logic)
6. Per-panel OCR + bounding box indexing
7. LLM-based filtering (garbage vs essential text)
8. Inpainting of approved text regions
9. Translation + font/size estimation
10. Text re-rendered and baked into images
11. Final translated chapter displayed in Streamlit

---

5. Functional Requirements

5.1 Web Crawling & Ingestion

FR-1: URL-based crawling via Streamlit input
FR-2: Handle infinite scroll / lazy-loaded chapters
FR-3: Bypass simple visual captchas (typed numeric obfuscation)
FR-4: Restrict downloads to manhwa panel images only

Panel Extraction Strategies

- HTML class / ID heuristics
- Image dimension & aspect ratio filtering
- CDN path pattern matching
- Keyword exclusion (ads, avatars, emojis, UI assets)

5.2 Panel Stitching

FR-5: Vertically stitch extracted panels into a single long image
FR-6: Preserve original resolution and ordering
FR-7: Maintain mapping from stitched coordinates → original panel IDs

Rationale: maximizes OCR context and prevents dialogue fragmentation.

5.3 OCR Pipeline

Preliminary OCR

FR-8: Perform OCR on stitched image - Purpose: identify safe split zones

Smart Splitting

FR-9: Split stitched image into ≤100 sub-panels where: - Cut line does NOT intersect any OCR bounding box - Cut line is ≥50px away (above/below) from nearest OCR text

Final OCR

FR-10: OCR each sub-panel independently - Output bounding boxes, text, confidence - Reference image ID + coordinates

OCR Engine Requirements

- Robust to stylized Hangul fonts
- Vision-based (CNN/Transformer), not document-layout dependent
- Preferably supports character-level or dense text detection

---

5.4 Data Storage

FR-11: Persist OCR results with spatial metadata

Supported formats: - CSV (lightweight, inspectable) - SQL (SQLite/Postgres) for relational mapping

Required fields: - Image ID - Bounding box (x, y, w, h) - Detected text - OCR confidence - Panel index / chapter index

---

5.5 Garbage Detection (LLM Agent)

FR-12: Classify OCR text into: - Dialogue / narration (keep) - SFX - Watermark / credits - UI / noise

Approach

- Use Anthropic API (Claude) as a semantic filter agent
- Inputs:
- OCR text
- Bounding box size & position
- Local image crop (optional)

Prompt Engineering Considerations

- Few-shot examples of SFX vs dialogue
- Explicit instruction to be conservative (false negatives > false positives)
- Structured JSON output (KEEP / DROP + rationale)

5.6 Inpainting

FR-13: Inpaint only text regions approved by filter agent

Requirements: - Use OpenCV-based or diffusion-based inpainting - Respect background textures and gradients - Avoid bleeding into artwork edges

---

5.7 Translation (LLM Agent)

FR-14: Translate retained text using context-aware LLM

Inputs

- Original Korean text
- Neighboring dialogue (context window)
- Series metadata (if available)

Prompt Engineering

- Tone preservation (casual, dramatic, comedic)
- Consistent naming across chapter
- Optional glossary / RAG injection

5.8 Font & Layout Matching

FR-15: Estimate original font size and style

Heuristics: - Bounding box height → font size - Stroke thickness estimation - Sans/serif / handwritten classification

FR-16: Re-render translated text - Centered within original bounding box - Line wrapping constraints - Adaptive scaling if text expands

---

5.9 Image Recomposition

FR-17: Bake rendered text into image - Match original color (white/black/outline) - Preserve resolution and aspect ratio

FR-18: Reassemble panels into final chapter image(s)

5.10 Streamlit UI

FR-19: Streamlit-based interface providing: - URL input - Crawl progress visualization - OCR bounding box preview - Before/after image comparison - Final translated chapter viewer

FR-20: Debug & inspection tools - Toggle OCR boxes - Show filtered vs dropped text - View LLM decisions

---

6. Non-Functional Requirements

- Modular, pipeline-based architecture
- Deterministic re-runs (same input → same output)
- Graceful failure handling (skip panel, log error)
- Reasonable performance for 1 chapter (<5–10 min)

---

7. System Architecture (High-Level)

Crawler → Panel Extractor → Stitcher → OCR_1 → Smart Splitter → OCR_2
→ OCR DB → Filter Agent → Inpainter → Translation Agent
→ Font Matcher → Renderer → Streamlit UI

---

8. Risks & Mitigations

Risk	Mitigation
OCR fails on stylized fonts	Use vision-first OCR, multi-pass OCR
Over-aggressive garbage filtering	Conservative prompts, human-inspectable output
Site structure changes	Heuristic + pattern-based extraction
Translation overflow	Dynamic font scaling & reflow

9. Success Metrics

- % of dialogue correctly preserved
- Visual similarity to original layout
- Reduction in manual cleanup vs baseline
- End-to-end success rate per chapter

---

10. Future Extensions

- Multi-language output
- Interactive correction UI
- Fine-tuned OCR / filter models
- Full agent orchestration with memory
- Offline/local LLM support

---

11. Summary

This system is designed as a **research-grade, agent-driven, vision-language pipeline** for manhwa translation. The emphasis is on correctness, inspectability, and modularity rather than speed or commercial scalability, incorporating lessons learned from prior iterations and leveraging modern OCR + LLM capabilities.