Unriddle

Internship Task Documentation | Unriddle Technologies Pvt. Ltd.

Document Compiled by Amulya Kotagiri

Project Live at: https://project.amulyakotagiri.tech

Technical Report

1. Problem Statement

Digitizing handwritten documents—whether medical records, meeting notes, or historical manuscripts—poses significant challenges for accuracy and readability. Manual transcription is labor-intensive, error-prone, and does not scale.

2. Solution Overview

Frontend: A modern HTML/CSS/JS single-page application deployed on Vercel, featuring a styled file picker, template dropdown, loading states, and a responsive results panel.

Backend: FastAPI service handling file uploads, OCR extraction via Qwen2_5_VLForConditionalGeneration, and summarization via the Hugging Face pipeline("summarization", model="facebook/bart-large-cnn").

3. Data Flow:

Step 1 : User uploads PDF/image → FastAPI saves it to temp-data-rec dir on server temporarily till processing, and removes after sending response for efficient server storage.

Step 2 : OCR: Image is passed to Qwen2.5-VL, which extracts text paragraphs exactly.

Step 3 : Summarization: Extracted text is chunked (≤500 chars) and summarized; results concatenated.

Step 4 : Formatting: Summary is wrapped in the selected template header.

Step 5 : Response: JSON returned; frontend displays formatted summary with an option to download.

4. Tech Stack

Layer	Technology / Library
Backend	Python, FastAPI, PyTorch, Transformers, PIL
OCR Model	Qwen2.5-VL-3B-Instruct
Summarization	Hugging Face Transformers (BART-large-CNN)
Frontend	HTML5, CSS3, JavaScript (Fetch API)
Templating	Jinja2
Deployment	Vercel (frontend), Ngrok + Linux server (API)
CORS	FastAPI Middleware

5. Implementation Details

File Handling: Uploaded files are temporarily stored in temp-data-rec; auto-deleted after processing.

OCR Pipeline: Uses AutoProcessor.apply_chat_template to construct multimodal prompts.

Batches images into GPU-accelerated inference where available, decoding outputs to retain exact paragraph structure.

Summarization Pipeline: Text is sanitized (line breaks → spaces), split into ≤500 char chunks, then fed to BART summarizer with max_length=150, min_length=40.

Template Formatting: Three templates implemented via format_summary(summary, template_id), prepending headers like “Patient Visit Summary:”.

Frontend UX Enhancements: Custom file-picker button, dynamic filename display, submit-button spinner, and keyboard-accessible form controls.

Results panel uses monospace font and auto-scroll for readability.

6. Summary Template Support

Users select from three domain-specific templates:

1. Patient Visit Summary
2. Medical Case Notes
3. Historical Record Summary

Upon selection, the backend prepends the appropriate header. This modular design allows easy addition of new templates by updating format_summary.

7. Key Challenges & Solutions

Challenge	Solution
Handwritten OCR Accuracy	Adopted Qwen2.5-VL Instruct—fine-tuned for multimodal tasks—to maximize text fidelity.
Chunking Long Text for Summarization	Implemented dynamic chunking to avoid truncation while respecting model input limits.
CORS & Cross-Origin Requests	Configured FastAPI CORSMiddleware to allow secure, domain-restricted access from the Vercel frontend.
UX Responsiveness During Processing	Added button disablement and spinner UI to inform users of progress and prevent duplicate requests.
Deployment Networking	Used Ngrok to expose local FastAPI server securely; configured port-forwarding and HTTPS tunnels.
Custom SubDomain	Attaching subdomain gave trouble. Replacing ns records to vercel’s allowed correct cname record validation, further installing ssl certificate.

8.Performance Metrics:

OCR Latency: ~3 s/image on NVIDIA A5000 GPU 64 GB | 128 GB Ram

Summarization Latency: ~2 s per 500 char chunk.

Quality Checks: Spot-checked summaries for factual consistency against source text; iterated chunk sizes to optimize summary length and relevance.

9. Deployment & Live Demo | Deliverables

Frontend: Hosted at custom subdomain : https://project.amulyakotagiri.tech

Backend: FastAPI server on Ubuntu, exposed via Ngrok HTTPS tunnel (configured as /process/ endpoint) : https://41d5-182-156-3-86.ngrok-free.app/process/

Repository: All source code, documentation are available on GitHub:

https://github.com/amulyakotagiri/internship-task-amulyak.git

10. Future Enhancements

Bulk Processing: Batch multiple documents in one request.

Download Endpoints: Enable direct download of extracted text and summaries.

Model Fine-Tuning: Customize Qwen2.5-VL on domain-specific handwriting samples.

UI Polishing: Add real-time preview of templates and dark-mode support.

11. Conclusion

This project demonstrates an end-to-end pipeline—from handwritten OCR to AI-driven summarization—deployed as a production-ready web app. Its modular architecture, robust UX, and clear template support make it an ideal candidate for internships focusing on AI engineering, full-stack development, and user-centered design.