The University of Hong Kong

Faculty of Engineering

Department of Computer Science

2024 - 2025

COMP 4801 Final year project

Interim Report

An Enhanced Note Assistance Application with Handwriting Recognition Leveraging LLM

Deng Jiaqi 3035832490

Gu Zhuangcheng 3035827110

Xie Changhe 3035770575

Zhou Zihan 3035772640

Supervisor: Prof. Luo Ping

Date of submission: 01/26/2025

Contents

Abstract

This project presents an innovative note-taking assistant application that transforms the note-taking process through advanced handwriting recognition, sketch conversion, and note question-answering (QA) capabilities. Leveraging large language models (LLMs) and optical character recognition (OCR) technologies, the application converts handwritten drafts into organized, searchable digital notes. It transforms rough sketches into clean formats like Markdown and enables users to query their notes for deeper insights. This tool enhances the efficiency and effectiveness of organizing and understanding notes for users.

1 Background

1.1 Overview of LLM Capabilities in Document Tasks

Multimodal LLMs will serve as the foundation of our app, providing the ability to integrate OCR and question-answering capabilities, enabling automated content extraction and context-based querying. We explored recent advancements in general-purpose models like GPT-4, Kosmos-2, Qwen2-VL, LLaMA3, and InternVL2 (Section 1.1.1) for their versatility in handling diverse document inputs but found limitations in detailed text recognition. Therefore, we also examined specialized models (Section 1.1.2), like Vary and GOT, which focus on overcoming these challenges with improved document layout understanding and text precision, making them a better fit for our application's requirements.

General Purpose LLM **GPT-4** [1] is a versatile multimodal model capable of handling text and image inputs. Its strength lies in structured document analysis, complex reasoning, and context-aware generation, yet it falls short in precise text recognition and parsing when dealing with dense, text-heavy documents. **Kosmos-2** [11] introduces grounding mechanisms that link text to visual regions, enabling strong spatial reasoning and multimodal comprehension. However, it lacks the fine-grained OCR accuracy needed for detailed text extraction. **Qwen2-VL** [14] leverages dynamic resolution adjustment and specialized embeddings to handle complex visual inputs, performing well in vision-language tasks but still struggling with dense textual data. **LLaMA3** [4] emphasizes multilingual support and scaling, integrating multiple modalities like image and speech. However, its image understanding is limited to high-level features, making it less suitable for detailed OCR tasks. **InternVL2** [3] employs a hierarchical fusion mechanism for better text-image integration but lacks the precision required for small-font and densely packed text. These models, despite their impressive capabilities, are primarily optimized for high-level multimodal understanding and reasoning, making them less effective in OCR-specific tasks that demand detailed layout understanding and precise text extraction.

Although these models represent significant progress in document-related tasks, their OCR capabilities are constrained by the composition of their training data and their architectural designs. Most of these models are optimized for high-level multimodal understanding and lack the fine-grained text recognition and layout understanding needed for effective document-level OCR tasks. This limitation becomes especially pronounced in text-intensive conditions, where traditional OCR models tend to outperform these general-purpose LLMs.

Specialized LLMs for Document Tasks To address the shortcomings of general-purpose LLMs in OCR and document understanding, several specialized models have been developed to handle complex document layouts and dense text more effectively.

Donut (OCR-free Document Understanding Transformer) [6] bypasses traditional OCR by directly transforming document images into structured text using a transformer-based architecture. This reduces OCR errors and increases processing speed but is limited to straightforward document parsing and struggles with complex reasoning tasks. **Nougat** [2] focuses on academic documents, recognizing complex formulas and structured text within PDFs and converting them into markup language. While ideal for scientific texts, its specialization in structured layouts limits its applicability in diverse document types. **Vary** [15] expands the vision vocabulary for vision-language models, enhancing OCR accuracy for non-English and visually rich documents. It effectively handles fine-grained visual perception but is limited in complex document QA tasks. **GOT** [16] introduces a unified end-to-end model that supports diverse document elements (e.g., text, tables, formulas), offering highly versatile OCR capabilities but lacking advanced contextual reasoning for in-depth document QA.

These specialized models excel in handling OCR-specific tasks but lack the comprehensive document QA abilities found in more generalized models. They focus primarily on text extraction and layout parsing, limiting their usefulness for tasks that require deep semantic understanding and complex question-answering capabilities.

1.2 Limitations of Existing Applications

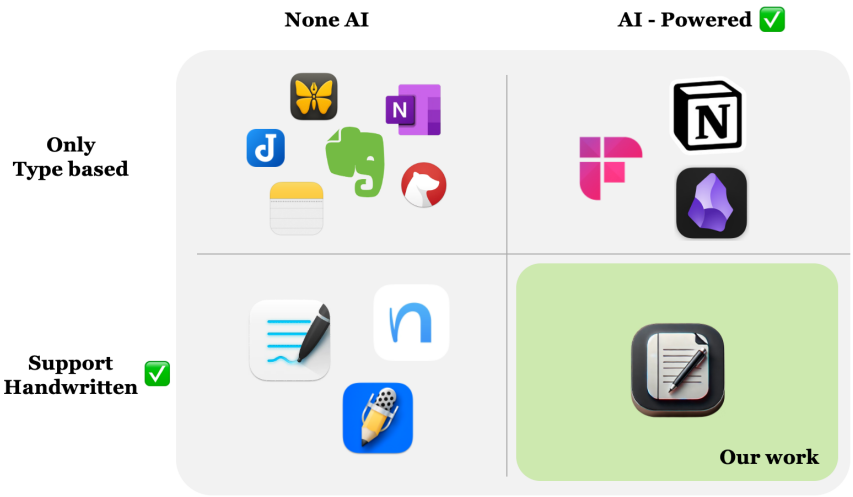


Figure 1.1: Comparison of Current Note-Taking Applications Based on AI Capabilities and Input Support

Limitations of Existing Note-Taking Applications Current note-taking applications, as illustrated in fig. 1.1, are limited in leveraging these advanced AI capabilities. Most tools focus on basic functions like text transcription and keyword search without effectively handling diverse inputs such as handwritten

notes, sketches, and complex visual data. In addition, they lack context-aware querying and intelligent content categorization, making it difficult for users to organize and interact with their notes, especially when dealing with a combination of text, diagrams, and annotations.

Challenges in Integrating LLM Tools with Note-Taking Workflows Based on industry analysis, existing LLM-based applications require users to switch between multiple tools and windows, such as GPT applications and note-taking software, leading to workflow disruptions and reduced productivity. Additionally, the generic prompts provided by most existing LLM applications are not tailored for specific note-taking scenarios, which further limits their effectiveness.

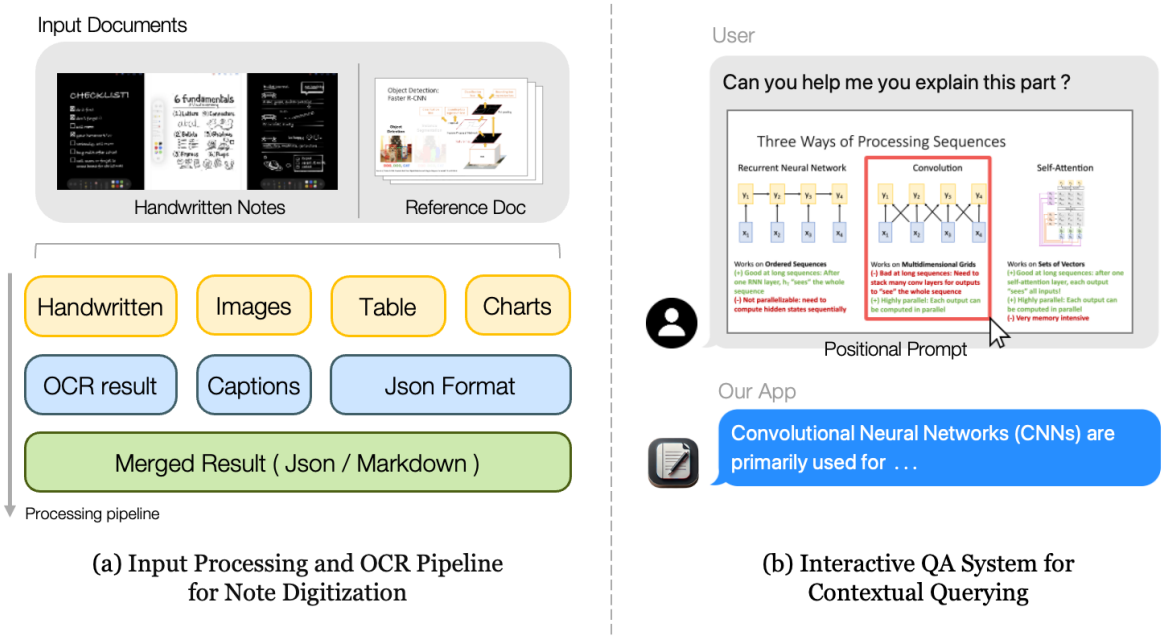
2 Objectives

The primary objective of this project is to build an intelligent note-taking tool that utilizes the capabilities of multimodal LLMs to create a more structured and interactive note management experience. As illustrated in Figure 2.1, the app will process user input in two main stages: first, converting handwritten notes and reference documents (e.g., lecture notes) into a digital format using OCR capabilities, and then using RAG to enhance QA functions, providing insightful answers to users' questions. This approach combines advanced text recognition and contextual understanding, making the app a versatile tool for organizing and querying complex information.

In general, this project aims to develop an intelligent note-taking tool that combines the strengths of general-purpose and specialized LLMs to offer a more interactive and structured note-management experience. The specific objectives are:

- **Develop a Cross-Platform User Interface**
  - Build a responsive and intuitive UI compatible with both PC and mobile devices.
  - Support diverse input formats, including text, handwriting, and diagrams.
- **Implement Multimodal LLM Integration**
  - Utilize Multimodal Large Language Models (MLLMs) for OCR, extracting information from handwritten notes, and other referencing documents.
  - Allow users to query their notes using natural language inputs, providing contextually relevant responses and deeper insights based on the content of the notes.
  - Create a comprehensive dataset specific for note-taking use cases and fine-tune the model accordingly.
- **Create a Structured Digital Note System**

- Convert unstructured handwritten notes and drafts into searchable digital formats.
- Implement advanced search and categorization features to effectively organize notes.



(a) Input Processing and OCR Pipeline for Note Digitization

(b) Interactive QA System for Contextual Querying

Figure 2.1: Workflow of the Intelligent Note-Taking Application

By achieving these objectives, the project will deliver a robust AI-driven note-taking application that not only digitizes and organizes content but also offers an interactive experience through enhanced information retrieval and contextual understanding capabilities.

3 Methodology

This methodology shown in fig. 3.1 encompasses the end-to-end process for developing enhanced note assistant applications powered by large language models. The methodology spans two main tracks: Model Development and App Development.

The Model Development track focuses on creating the underlying LLM that will drive the note assistant's capabilities. It involves dataset collection and preparation to curate the training data, followed by supervised fine-tuning to specialize the base LLM on the note assistance task. The trained model then undergoes evaluation to assess its performance before being deployed.

In parallel, the App Development track handles building the user-facing application that interfaces with the LLM backend. This encompasses both backend development, such as creating the API gateway and setting up data processing pipelines, as well as frontend development to design the user interface and

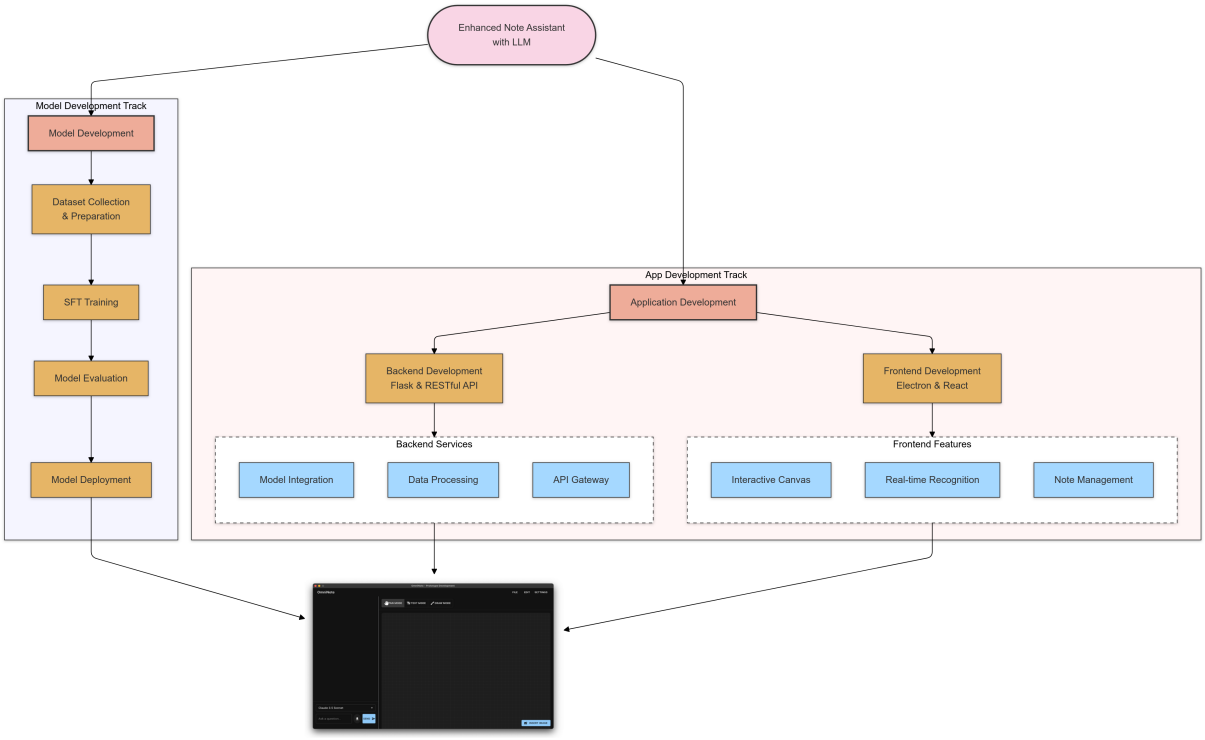


Figure 3.1: Methodology Overview

interactive features. Key frontend components include the Interactive Canvas for rich note-taking and handwriting recognition and note management to enable seamless capturing and organizing of information.

3.1 Model Development

Model Achitecture Our model development approach centers around a shared neural encoder that learns both visual and textual features to power document understanding.

The architecture shown in fig. 3.2 begins with document input, which can contain a mix of unstructured text, structured form-like content, and visual elements like tables and figures This raw document data first passes through a feature extraction module that integrates the visual and textual features. The representation then flows into a hidden states generation component that produces a dense vector encoding of the document. Separate decoder heads are employed for each target task, such as document understanding via question answering, element detection and localization, or document indexing and search. By sharing the same encoder backbone while learning task-specific output layers, we can optimize the model for a wide range of document processing objectives while benefiting from transfer learning and reduced architectural complexity.

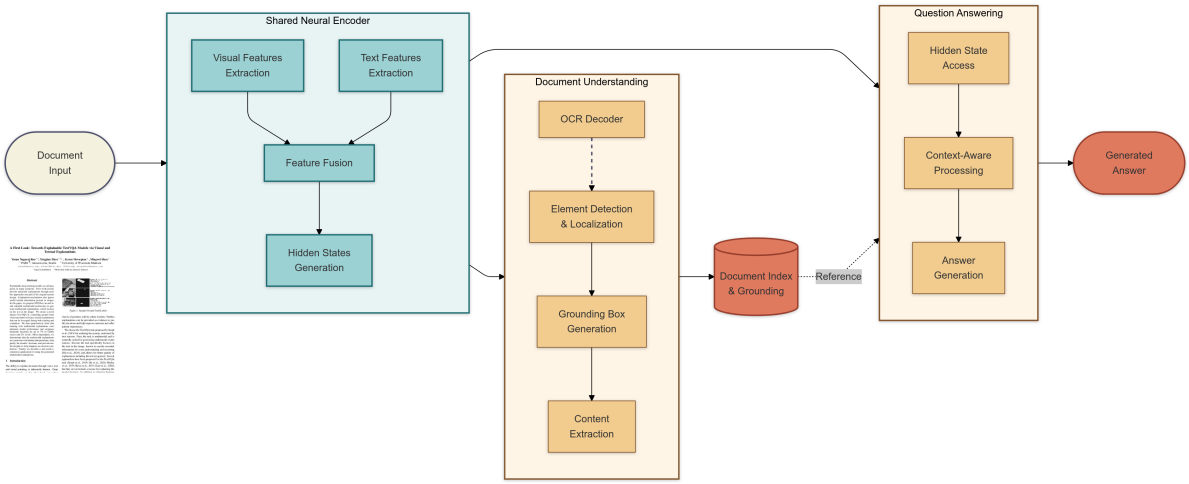


Figure 3.2: The designed architecture of model workflow

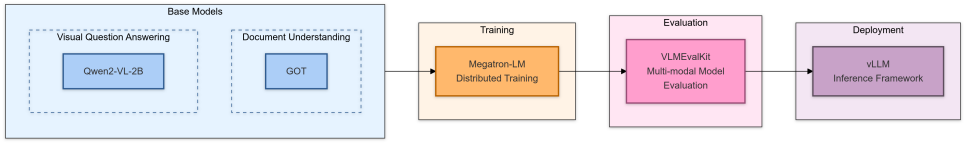


Figure 3.3: Finetuning Pipeline

Model Finetuning For model finetuning, we start with two base models: Qwen2VL-2B for visual question answering and GOT for document understanding. The whole workflow is shown in the Figure 3.3.

For the framework that we used: Megatron is a powerful distributed training system that enables efficient finetuning of large language models. By leveraging Megatron, we can process substantial amounts of domain-specific data while maintaining fast training cycles and resource efficiency. The finetuned model is evaluated using VLMEvalKit, a comprehensive evaluation suite designed for assessing the performance of Vision-Language Models. Once the model has been finetuned and validated, we integrate it into our Visual Language Model (vLLM) inference framework. The vLLM framework provides a unified API for deploying and serving the model in production environments.

3.2 Dataset

To train a model with our target use-case, we need to collect our own dataset. The OmniNote-1M dataset is designed to support a wide range of tasks in document parsing and visual question answering within note-taking scenarios. It comprises 180k samples for Page OCR, 172k samples for Text Spotting, and 160k samples for Text Recognition, facilitating robust document understanding. (the prompt for these tasks can refer to Section 3.2.3) Additionally, the dataset includes tasks related to Visual Question

Answering, which are categorized as follows: 443k samples for Natural Image VQA, 39k samples for Document VQA, and 40k samples for Chart VQA. Together, these datasets cover a diverse array of scenarios, ranging from structured document parsing to unstructured visual inputs.

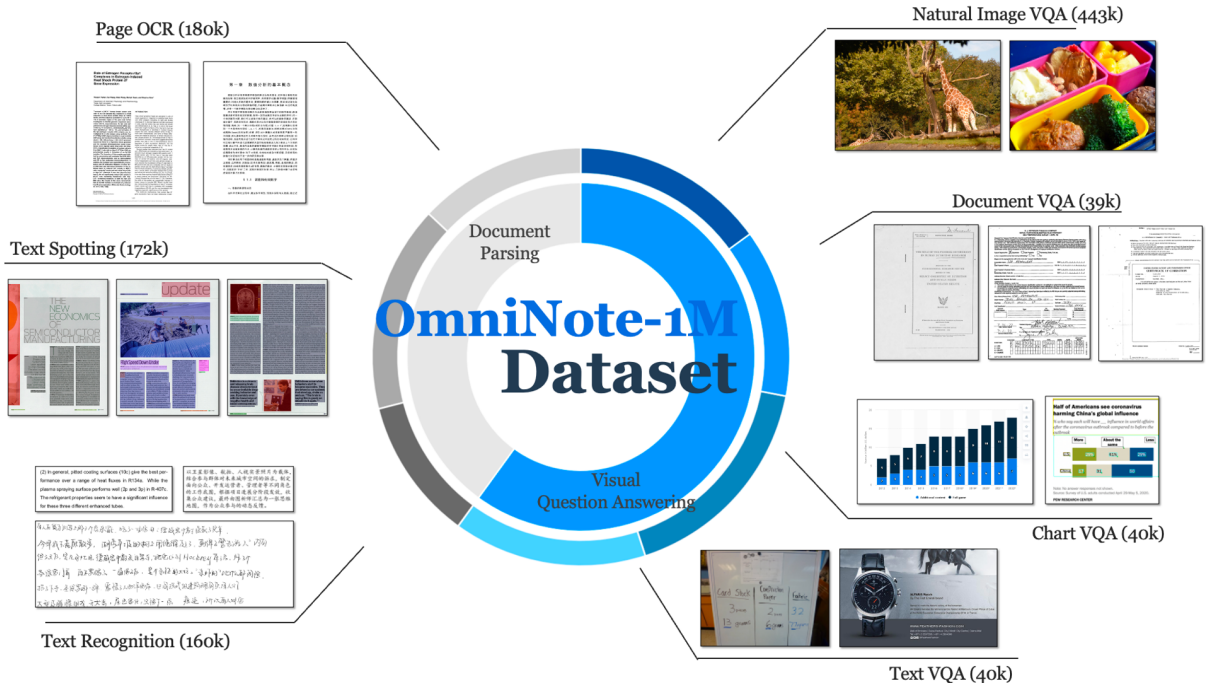


Figure 3.4: Dataset Overview of OmniNote-1M

Dataset Statistics The OmniNote-1M dataset was constructed using a combination of public datasets, synthetic data, and self-collected data. The document parsing subset consists of self-collected data from sources such as arXiv and Common Crawl, which are used for Page OCR and Text Spotting. The Text Recognition task also includes synthesized handwritten data, sourced from public datasets like CASIA-HWDB [7] (for Chinese) and IAM [10] (for English). These handwritten datasets are specifically designed for single-line recognition. For our use case, multiple single-line instances were randomly stacked together to create a paragraph-level images, simulating handwritten notes. The VQA tasks leverage public datasets such as VQA v2 [5], LRV-Chart [8], and TextVQA [12], ensuring a broad range of natural and document image types. This diverse combination of datasets supports both structured document tasks and general VQA challenges.

Dataset Preprocessing To ensure the high quality and diversity of the OmniNote-1M dataset, a rigorous filtering pipeline was applied to the self-collected data.

The first step involved utilizing layout detection model based on YOLO-v10 [13], which was fine-tuned for document layout feature extraction. This allowed us to effectively analyze the structural elements

Task	Name	Dataset Type	Source	Image type	Subtask	Quantity
Document Parsing	OmniNote-OCR	self-collected	arXiv (EN) CC-MAIN-PDF (EN) ebooks (CN)	Digital Document Scanned Document	Page OCR	180k
					Text Spotting (w/ grounding)	172k
					Text Recognition (Text/Equation/Table)	157k
	OmniNote-HWOCR	synthesized (from public)	CASIA-HWDB (CN) IAM-line (EN)	Scanned Handwriting (Offline)	Text Recognition	3k
Visual Question Answering	OmniNote-VQA	public dataset	VQA v2	Natural Image	/	443k
			LRV_Chart	Natural Image		7k
			TallyQA	Natural Image		38k
			TextVQA	Digital Image Captured Image Scene Text Image		34k
			DocVQA	Scanned Document		39k

Figure 3.5: Dataset Statistics of OmniNote-1M

of each document, such as text blocks, images, and tables, in a high-dimensional space. Next, we use K-nearest Neighbors to group documents with similar layouts and structures. This clustering process enabled us to divide the original dataset of over 780k raw samples into distinct clusters based on layout patterns. As a result, 180k high-quality samples were selected that were both structurally diverse and representative of the broader dataset. This careful selection ensures that the data used for model training is clean, well-organized, and balanced across various document types.

Task	Format	Sample
General OCR	text	Here is some sample text.
OCR with format	markdown	#Title\n\n ##Section\n\n Some sample text.
Equation	LaTeX	\[\begin{array}{r}{n=64.} ...
Table	html	<html><body><table><tr><td>Compound</td>...

Figure 3.6: OCR data format for various tasks

Data Format and Prompt Design For standardization purposes, the dataset was formatted according to the specific requirements of each task as shown in fig. 3.6. For general OCR tasks, we utilize plain text. For OCR with formatting tasks include structured outputs such as markdown for documents, LaTeX for equations, and HTML for tables.

Task	Prompt	Label
General OCR	`OCR:`	`<text>OCR Result</text>`
Text Spotting (Paragraph Level)	`OCR with grounding:`	`<text>OCR Result</text><box>x1 y1 x2 y2</box>` (repeated for N page elements)
Text Recognition	`Read the text in the <ref>box</ref><box>x1 y1 x2 y2</box>.`	`<text>OCR Result</text>`
VQA	`Question. Answer:`	`Answer.`
VQA with Grounding	`Question. Provide the location coordinates of the answer when answering the question. Answer:`	`Answer. <box>x1 y1 x2 y2</box>`

Figure 3.7: OCR data format for various tasks

In addition, we followed and extended the prompt design methodology from a prior work, TextMonkey [9], while adopting special token settings from QWen Tokenizer [14]. This schema ensures compatibility with a wide range of current language models. For example, the VQA tasks utilize a clear question-answer format, and all grounding tasks provide positional coordinates normalized from 0 to 1000.

3.3 Application Development

The development of the application involves three major components: the system architecture, the implementation of frontend and backend systems, and the integration of core features such as notes management and advanced functionalities.

System Architecture Design The system is designed with a robust and efficient structure, aiming for a modular architecture in both backend and frontend implementations. The design prioritizes cross-platform compatibility, high performance, and maintainability. As illustrated in fig. 3.8, the system adopts an Electron-Flask structure, which facilitates support for multiple operating systems and devices.

The interaction between the different system components follows a structured workflow to ensure smooth communication and data flow. Users interact with the application through desktop or mobile interfaces. The frontend sends HTTP requests to the Flask backend via RESTful APIs, transmitting user input such as images, handwritten drafts, and reference documents for OCR processing or note querying. The backend processes the input data and returns the results to the frontend in JSON format, where the UI is subsequently updated to display the results.

Frontend Development The frontend is built as a cross-platform system to support desktop and mobile environments. It features an intuitive user interface (UI) for note input, organization, and model querying. The interface will also integrate real-time features such as handwriting recognition and sketch conversion. Additionally, the UI layout is designed for aesthetic appeal and user-friendliness.

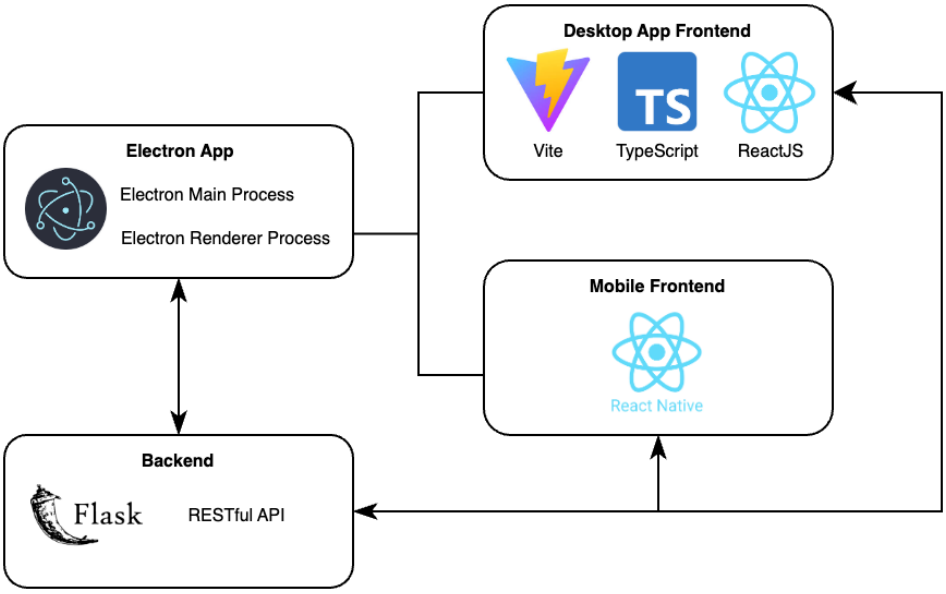


Figure 3.8: OmniNote System Architecture and Tech Stack Overview

The desktop application is implemented with Electron, which provides a modular architecture separating the main and renderer processes to enhance scalability and maintain stability across operating systems. The frontend UI within the Electron framework is developed using ReactJS and TypeScript, utilizing Vite for fast and responsive performance.

The system is planned to be further expanded to mobile platforms, as handwritten input is more practical on devices such as iPads or smartphones. The mobile frontend will be developed using React Native, a cross-platform mobile framework that enables the development of applications for both iOS and Android using a single codebase. React Native was selected to ensure efficient development by leveraging the reusability of React components and maintaining consistency across desktop and mobile platforms.

Backend Development The backend, developed with Flask, serves as the central framework for application logic and integration with external APIs. Flask was selected due to its simplicity and efficiency in building RESTful APIs, making it an ideal choice for serving machine learning models. Two primary APIs have been integrated: the OCR API processes handwritten notes and converts them into digital formats for better readability, while the VQA API facilitates contextual insights through visual question-answering. These APIs are optimized to work efficiently with the frontend, providing efficient and accurate analysis of user inputs.

Advanced Features Development Core features of the application include file upload functionality, question-answering functionality, model selection, text editing, drawing, and image insertion. These functionalities enable users to interact with their notes in a versatile manner. The notes management

system, a critical component of the application, uses a tree-like hierarchical structure to organize notes into folders and subfolders. This structure allows for intuitive management and retrieval of notes. Metadata tagging and search functionalities enhance the efficiency of this system, addressing the need for structured and scalable note storage.

Testing and Evaluation Rigorous testing and evaluation will be conducted, including comprehensive unit and integration tests for system modules, as well as end-to-end system testing. Metrics will be designed to evaluate system performance, including accuracy, response time, and user experience. User feedback will also be gathered during the testing stage to inform further improvements.

By adhering to this comprehensive methodology, we aim to develop a highly functional and user-centric note-taking assistant application. This systematic approach ensures that every aspect of the project—from model selection and fine-tuning to system architecture and user interface development—is meticulously planned and executed. The result will be a robust, efficient, and intuitive application that not only meets the defined requirements but also enhances the overall user experience in the realm of digital note-taking.

4 Preliminary Results

4.1 Experimental Result

Type	Model	Param	Text		Equation		Table	
			EN	CN	EN	CN	EN	CN
Pipeline Tools	Mathpix	/	0.101	0.358	0.306	0.454	0.322	0.416
Expert VLMs	GOT	0.5B	0.191	0.314	0.360	0.523	0.459	0.520
	Nougat	0.4B	0.365	-	0.488	-	0.622	-
General VLMs	GPT-4o	/	0.144	0.409	0.425	0.606	0.363	0.474
	Qwen2-VL	72B	0.252	0.251	0.468	0.572	0.591	0.587
	InternVL2-Llama3	76B	0.353	0.290	0.543	0.701	0.616	0.638
Ours	GOT*	0.5B	**0.118**	**0.297**	**0.294**	**0.629**	**0.369**	**0.312**

Table 4.1: OmniDocBench Results (Edit Distance)

Document Parsing Benchmark We benchmarked our models on OmniDocBench [**<empty citation>**], a public document parsing benchmark, which evaluates tasks such as text, equation, and table parsing. Our baseline model outperformed other expert VLMS, such as Nougat and the original GOT model, in most tasks, particularly in text extraction. Although larger generalist models like Qwen2VL and InternVL2 perform better across a broader range of tasks, we observed minimal performance differences for document parsing tasks. Our model strikes an optimal balance between efficiency and performance, making it more feasible for practical deployment.

Model	$\text{ChartQA}_{\text{Test}}$	$\text{OCRVQA}_{\text{Test}}$	$\text{TextVQA}_{\text{Val}}$	$\text{DocVQA}_{\text{Test}}$
Qwen2VL-2B (Baseline)	**71.48%**	**74.65%**	78.67%	**89.00%**
Qwen2VL-2B-1M-10epoch (ours)	59.84%	43.37%	69.41%	77.16%
Qwen2VL-2B-80k-10epoch (ours)	69.34%	56.88%	79.20%	85.32%
Qwen2VL-2B-80k-3epoch (ours)	70.54%	62.92%	**80.43%**	88.75%

VQA Benchmark The VQA benchmark results demonstrate that the performance on all models decreased compared to the baseline Qwen2VL-2B model. The ChartQA test accuracy dropped from 71.48% for the baseline to 59.84%, 69.34%, and 70.54% for the 1M-10epoch, 80k-10epoch, and 80k-3epoch models respectively. Similarly, the OCRVQA test accuracy fell from 74.65% to 43.37%, 56.88%, and 62.92% for those three models. The TextVQA validation accuracy also declined from the baseline 78.67% to 69.41%, 79.2%, and 80.43%. Finally, the DocVQA test accuracy decreased from 89.00% for the baseline to 77.16%, 85.32%, and 88.75% for the three variants.

Two reflections are provided - the need to potentially build a custom benchmark for their specific tasks, and the recognition that data quality is more important than quantity, necessitating data cleaning on their dataset. Scaling down the model and data size led to reduced VQA performance across multiple benchmarks, highlighting the importance of data quality and specificity of benchmarks to the target application. Targeted data curation and custom benchmarking could help optimize performance for their particular use case.

4.2 Application Development Progress

Significant progress has been achieved in the application's development. A functional prototype has been created, integrating multiple key modules. The file upload system allows users to import handwritten notes or diagrams, which are processed by the backend's OCR and VQA APIs. The text editing and drawing modes are operational, providing users with basic tools to create and modify content. The user interface also includes a dark and light theme option, ensuring adaptability to diverse user preferences.

The design and functionality of the prototype can be seen in in Figure 4.1, which illustrates features such as model selection, file upload, text editing, drawing, image insertion, and UI themes. These features collectively enhance the application's usability and provide a versatile platform for managing notes effectively.

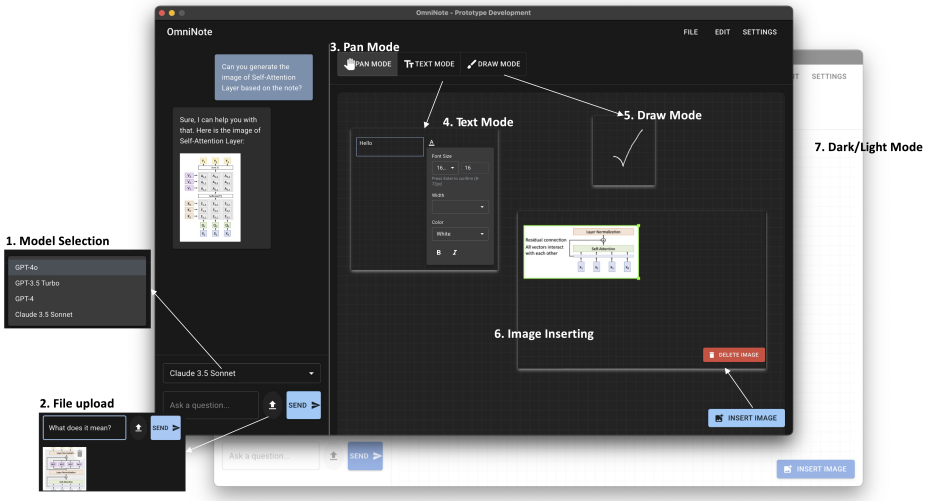


Figure 4.1: Design and functionality of the desktop application

The backend successfully connects to the frontend via APIs, enabling smooth communication between the user interface and the underlying processing models. Additionally, the application supports model selection, allowing users to choose specific LLMs tailored to their tasks. Despite these achievements, there are still limitations in the current implementation. The note management system lacks the ability to

efficiently store and organize multiple notes, which hinders its scalability for more extensive use cases. Additionally, the text editing mode does not yet support Markdown formatting or advanced customization options, limiting its flexibility. The handwriting recognition system in the drawing mode also requires improvement to better distinguish between handwritten text and graphical elements. These challenges will be addressed in the upcoming development phases to align the application with its overall objectives of providing a robust and user-friendly note management system.

5 Schedule and Milestones

The project spans four phases from August 2024 to April 2025, each targeting key milestones. It starts with research and planning, followed by data collection and model fine-tuning, application development, integration, and concluding with testing and evaluation. The project schedule and plan have been adjusted and refined based on the progress made during the first two phases. Below is a detailed breakdown of the timeline, key deliverables, and status updates for each phase as shown in Table 5.1.

Phase	Period	Deliverables & Milestones	Status
0	Aug - Sep 2024	Research & Detailed Project Plan **Phase 0 Deliverables:** Detailed Project Plan & Project Website Setup	Done
	Overall Status: Completed		
1	Oct - Nov 2024	Data Collection Fine-tune the Vision Language Model (VLM) for OCR	Done
	Dec 2024	Fine-tune the Large Language Model (LLM) for DocQA Task Desktop Application Development **Phase 1 Deliverables:** Interim Report & First Presentation	Done
	Overall Status: Completed		
2	Jan - Feb 2025	Increase Data Volume & Balance Dataset Advance VLM and LLM Model Development	Doing
	Mar - Apr 2025	Continue Application Development Finalize VLM and LLM Model Integration **Phase 2 Deliverables:** Complete Application	Todo
	Overall Status: In Progress		
3	Apr 2025	Conduct User Experience Survey Test and Refine the System **Phase 3 Deliverables:** Final Report & Final Presentation	Todo
	Overall Status: Pending		

Table 5.1: Project Phases and Milestones

Bibliography

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (2023).

[2] L. Blecher, G. Cucurull, T. Scialom, and R. Stojnic. “Nougat: Neural optical understanding for academic documents”. In: *arXiv preprint arXiv:2308.13418* (2023).

[3] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, et al. “Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 24185–24198.

[4] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (2024).

[5] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. “Making the v in vqa matter: Elevating the role of image understanding in visual question answering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6904–6913.

[6] G. Kim, T. Hong, M. Yim, J. Park, J. Yim, W. Hwang, S. Yun, D. Han, and S. Park. “Donut: Document understanding transformer without ocr”. In: *arXiv preprint arXiv:2111.15664* 7.15 (2021), p. 2.

[7] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang. “CASIA online and offline Chinese handwriting databases”. In: *2011 international conference on document analysis and recognition*. IEEE. 2011, pp. 37–41.

[8] F. Liu, X. Wang, W. Yao, J. Chen, K. Song, S. Cho, Y. Yacoob, and D. Yu. “MMC: Advancing Multimodal Chart Understanding with Large-scale Instruction Tuning”. In: *arXiv preprint arXiv:2311.10774* (2023).

[9] Y. Liu, B. Yang, Q. Liu, Z. Li, Z. Ma, S. Zhang, and X. Bai. “Textmonkey: An ocr-free large multimodal model for understanding document”. In: *arXiv preprint arXiv:2403.04473* (2024).

[10] U.-V. Marti and H. Bunke. “The IAM-database: an English sentence database for offline handwriting recognition”. In: *International journal on document analysis and recognition* 5 (2002), pp. 39–46.

[11] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, and F. Wei. “Kosmos-2: Grounding multimodal large language models to the world”. In: *arXiv preprint arXiv:2306.14824* (2023).

[12] A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach. “Towards vqa models that can read”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8317–8326.

[13] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. “Yolov10: Real-time end-to-end object detection”. In: *arXiv preprint arXiv:2405.14458* (2024).

[14] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, et al. “Qwen2-VL: Enhancing Vision-Language Model’s Perception of the World at Any Resolution”. In: *arXiv preprint arXiv:2409.12191* (2024).

[15] H. Wei, L. Kong, J. Chen, L. Zhao, Z. Ge, J. Yang, J. Sun, C. Han, and X. Zhang. “Vary: Scaling up the vision vocabulary for large vision-language models”. In: *arXiv preprint arXiv:2312.06109* (2023).

[16] H. Wei, C. Liu, J. Chen, J. Wang, L. Kong, Y. Xu, Z. Ge, L. Zhao, J. Sun, Y. Peng, et al. “General OCR Theory: Towards OCR-2.0 via a Unified End-to-end Model”. In: *arXiv preprint arXiv:2409.01704* (2024).