AI Study Assistant: An AI-powered Learning Companion for Secondary Students

Feng Xinyue
202364870652
& $RenHongyu$
202364871032 & $DengJingru$
202364870602

Abstract
We present *AI Study Assistant*, an integrated AI-powered system designed to support secondary school students through note transcription, error tracking, mind-map generation, and learning analytics. The system combines speech-to-text (iFLYTEK ASR and Whisper), image/PDF OCR (Qwen-VL), large language models (DeepSeek LLM) for structured note generation, and an interactive web-based frontend for mind maps and dashboards. The system implements multi-account management (student and parent roles), automatic note generation from audio/images/PDFs, error book functionality with similar problem generation, interactive mind map visualization using Mermaid.js, and comprehensive learning analytics. We describe the problem motivation, related work, system architecture and implementation, AI components integration, and present case studies from the deployed demonstration. We conclude with limitations and a roadmap for future improvements. The project is open-source and deployed at `https://ai-study-assistant-2ozw.onrender.com`.

1 Introduction & Problem Definition

Secondary school students face significant challenges in organizing learning materials and tracking their progress effectively. Traditional methods of note-taking, error tracking, and study planning are time-consuming and often inefficient. Students need tools that can:

- **Transform raw classroom materials**: Convert audio recordings from lectures, photos of notes and exercises, and PDF documents into structured, searchable, and revisable learning material.
- **Track and analyze mistakes**: Automatically recognize student errors (from handwritten or typed work), classify them by subject and knowledge point, and generate targeted practice problems.
- **Visualize knowledge structure**: Create interactive mind maps that help students understand relationships between concepts and organize their learning.
- **Monitor learning progress**: Provide analytics and insights for both students and parents to track study time, mastery of topics, and areas needing improvement.

Goal and Contribution This project designs and implements an end-to-end web-based system that addresses these challenges through AI integration. The main contributions are:

1. A modular architecture integrating multiple AI services (ASR, OCR, LLM) into a cohesive learning platform
2. Multi-account system supporting both student and parent roles with appropriate permission management

3. Automatic structured note generation combining transcript/OCR extraction with LLM-based summarization

4. Error tracking system with visual recognition, automatic classification, and similar problem generation

5. Interactive mind map generation and visualization supporting multiple graph layouts

6. Comprehensive learning analytics dashboard with progress tracking and AI-powered suggestions

7. Deployed working prototype with open-source implementation available on GitHub

The system is implemented as a Flask backend with vanilla JavaScript frontend, deployed on Render cloud platform, and provides a practical demonstration of integrating modern AI capabilities into an educational tool.

2 Related Work

Our work integrates several research strands: **(1) Multimodal document understanding**: OCR and visual-language models (Qwen-VL [1]) extract text and formulas from images, including handwritten content. **(2) Speech recognition**: Whisper [2] and iFLYTEK ASR provide robust multilingual transcription. **(3) Large language models**: GPT-4, DeepSeek, and similar models [3, 4] enable structured content generation and summarization. **(4) Retrieval-Augmented Generation** (RAG) [5] grounds LLM outputs in source documents to reduce hallucinations. **(5) Intelligent tutoring** [6] and **learning analytics** [7] provide personalized learning and progress tracking. **(6) Mind mapping** [8] improves learning through knowledge visualization. Our system uniquely combines these technologies in a lightweight, modular web platform with multi-account support and multiple AI service integrations.

3 System Design

We designed a modular web system with clear separation between backend AI services, data persistence, and frontend user interface. The implementation prioritizes simplicity and maintainability while integrating multiple AI capabilities.

3.1 Architecture Overview

Figure 1 shows the high-level system architecture. The system consists of:

- **Frontend**: Vanilla JavaScript, HTML5, CSS3 with libraries for math rendering (MathJax) and graph visualization (Mermaid.js)
- **Backend**: Flask 3.0.0 web framework with modular Blueprint-based organization
- **AI Services**: Integration layer (`ai_service.py`) connecting to external APIs (DeepSeek, Qwen-VL, iFLYTEK ASR, Whisper)
- **Data Storage**: SQLite3 database with file storage for uploads
- **Deployment**: Gunicorn WSGI server on Render cloud platform

Figure 1: High-level system architecture showing the flow from user through frontend, backend, AI services, and data storage.

3.2 Module Organization

The backend (`backend/modules/`) implements: `auth.py` (authentication, account switching), `note_assistant_db.py` (note generation/management), `error_book.py` (problem OCR, classification, practice generation), `map_generation.py` (mind map generation/export), `learning_dashboard.py` (statistics, progress tracking), `settings.py` (user preferences), and `track.py` (study session tracking). Each module is a Flask Blueprint with RESTful JSON APIs.

3.3 Key Workflows

Note Generation: User uploads audio/image/PDF → ASR/OCR extracts text → LLM generates structured note (title, summary, key points, examples in Markdown with LaTeX) → saved to database → frontend renders with MathJax.

Error Book: Upload problem photo + handwritten solution → Qwen-VL recognizes text + handwriting → LLM classifies by subject/knowledge point → stores error record → generates similar practice problems.

Mind Map: Select note or input text → choose style (hierarchical/radial) → LLM generates Mermaid syntax → frontend renders → user edits code/whiteboard → export as PNG.

4 Methods (AI Components)

4.1 Speech Recognition (ASR)

iFLYTEK ASR: Commercial Chinese ASR with auto language detection. Converts audio to PCM (16kHz), splits into 55s segments, sends via WebSocket, receives real-time transcription with timestamps.

Whisper: Open-source multilingual ASR via OpenAI API. Supports various formats (MP3, WAV, M4A), provides timestamps and language detection, handles noisy audio robustly.

Implementation in `note_assistant_db.py` includes audio preprocessing (denoising, format conversion) before ASR.

4.2 OCR and Vision-Language Models

Qwen-VL (Alibaba DashScope) provides: general OCR, handwriting recognition, math formula detection with LaTeX conversion, and multi-image understanding. Implementation uses MultiModalConversation API. For error book, OpenCV preprocesses images (thresholding, contour detection, cropping) before sending to Qwen-VL with task-specific prompts. System parses structured JSON responses.

4.3 Large Language Model

DeepSeek LLM (OpenAI-compatible API) serves three main tasks:

Note Generation: Prompt converts transcripts/OCR to structured notes (title, 2-4 sentence summary, 5-7 key points, worked examples) in Markdown with LaTeX math (temperature 0.5-0.7).

Problem Classification: Analyzes problems and outputs JSON with subject, knowledge points, difficulty, and type.

Similar Problem Generation: Given a problem, generates 3-5 variations with same concept but different numbers/context and progressive difficulty (temperature 0.8-1.0).

4.4 Mind Map Generation

`generate_mindmap_mermaid` in `ai_service.py` converts text to Mermaid syntax for three styles: radial (native mindmap), hierarchical top-down (TD graph), and hierarchical left-right (LR graph). Prompt engineering ensures valid syntax: strict indentation (2 spaces/level for mindmap), proper node IDs, avoiding special characters in labels, and balanced tree structure. Client-side rendering with Mermaid.js enables interactive visualization.

4.5 Future: Embeddings and RAG

Architecture supports future RAG enhancement: embed document chunks (OpenAI text-embedding-3), store in vector database (FAISS/Milvus), retrieve top-k chunks by cosine similarity at query time, and inject context into LLM prompts to reduce hallucinations.

5 Implementation Details

Database: SQLite stores users (email, password, role, parent_id), notes (title, content, subject, source), error_book (problem, answer, classification), practice_problems, mindmaps, study_sessions, and learning_goals. Operations centralized in `db_sqlite.py`.

API: RESTful JSON endpoints via Flask Blueprints. Auth: `/api/auth/*` (register, login, switch account, session). Notes: `/api/note/*` (generate, list, get, update, delete). Error Book: `/api/error/*` (upload, list, generate-similar, practice). Mind Maps: `/api/mindmap/*` (generate, list, update, export). Dashboard: `/api/dashboard/*` (stats, chart-data, parent-report).

Frontend: Vanilla JS with modular design (main.js, config.js, page-specific JS). Static files served by Flask. Libraries: Font Awesome 6 (icons), MathJax 3 (LaTeX), Mermaid.js 10 (mind maps). Session-based auth via `fetch`.

6 Experiments & Case Studies

The system is deployed at `https://ai-study-assistant-2ozw.onrender.com`. We present qualitative case studies and performance measurements.

6.1 Case Study 1: Note Generation

Input: 2-min audio lecture on quadratic equations + textbook photo with formula $x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$.

Process: ASR (3s) $\rightarrow$ OCR (2s) $\rightarrow$ LLM (3s) generates title, summary, 6 key points, worked example.

Result: Accurate note with correct LaTeX formulas, minor formatting edits needed. Total: 8s.

6.2 Case Study 2: Error Book

Input: Photo with printed problem "Solve 2x + 5 = 13" + handwritten work with error.

Process: OpenCV preprocessing $\rightarrow$ Qwen-VL recognizes text + handwriting $\rightarrow$ LLM identifies mistake and classifies (Math, Linear Equations, Easy) $\rightarrow$ generates 3 similar problems.

Result: Good accuracy on neat handwriting, appropriate similar problems. Total: 12s.

6.3 Case Study 3: Mind Map

Input: Note on "Photosynthesis" with definition, light/dark reactions, molecules, factors.

Process: Select radial style $\rightarrow$ LLM generates Mermaid mindmap with 4 branches (Light Reactions, Dark Reactions, Requirements, Factors), 2-3 sub-items each.

Result: Valid syntax, accurate structure, user edited and exported PNG. Occasional syntax errors require regeneration. Total: 4s.

6.4 Performance

Table 1 shows representative latencies (not rigorous benchmarks):

Operation	**Latency (s)**	**Notes**
ASR (1-min audio)	6–25	iFLYTEK faster, Whisper more accurate
OCR (single image)	0.5–4	Handwriting slower
LLM note generation	1–6	Depends on length, RAG
Mind map generation	2–8	Complex structures longer
End-to-end note	3–30	Network + model dependent

Table 1: Representative latencies from prototype deployment.

User Feedback (5 students): Positive—fast generation, helpful structure, good math rendering. Issues—OCR errors on messy handwriting, occasional LLM hallucinations, mind map syntax errors. Suggestions—manual correction tools, better handwriting support, more languages.

7 Limitations & Future Work

7.1 Current Limitations

OCR/Handwriting: Biggest challenge—struggles with poor lighting, blurry photos, cursive handwriting, complex math notation, and mixed languages. Primary error source.

LLM: Occasional hallucinations (adding non-source info, factual errors in examples, misinterpreting ambiguous transcripts, invalid Mermaid syntax). Lightweight RAG doesn't fully ground outputs.

Scalability: API costs and rate limits; SQLite unsuitable for production; no vector database; single-server deployment has latency/concurrency limits.

Evaluation: No quantitative benchmarks, large-scale user study, A/B testing, or diverse subject/population testing.

Features: Incomplete RAG (no vector DB), AI chat placeholder, partial notifications, web-only (no mobile app), limited exports.

7.2 Future Work

Recognition: Fine-tune VLM, add image enhancement preprocessing, confidence scoring + manual review, diagram recognition.

RAG & Verification: Vector database (FAISS/Milvus), chunk-based indexing, provenance tracking, symbolic math verification (SymPy), fact-checking layer.

Scale: PostgreSQL, Redis caching, Celery job queue, Kubernetes deployment, batch processing, local inference.

UX: Real-time collaboration, mobile app, offline mode, rich text editor, more exports (PDF/Word/Markdown), themes, accessibility.

Education: Spaced repetition, adaptive difficulty, gamification, peer learning, teacher dashboard, LMS integration.

Research: Controlled user study (pre/post learning outcomes), model benchmarking, public dataset (with consent), long-term retention study.

Privacy: End-to-end encryption, on-device inference, regulatory compliance (COPPA/FERPA/GDPR), permissions, audit logging.

8 Conclusion

We presented AI Study Assistant, an integrated learning platform for secondary students that combines speech recognition, OCR, LLMs, and graph generation into a practical educational tool. Key achievements: end-to-end workflow from raw artifacts to structured materials, multi-account student-parent system, automatic note generation with LaTeX, error tracking with handwriting recognition, interactive mind maps, and analytics dashboard. The deployed prototype demonstrates value in real scenarios, though limitations remain in OCR accuracy, LLM reliability, and scalability. The modular architecture provides a foundation for future enhancements. This open-source project serves as both a practical tool and research platform for investigating AI in education.

Broader Impacts and Ethics

Privacy: System processes sensitive student data (materials, handwriting, voice, performance). Production deployment requires strong access controls, encryption, opt-in consent, regulatory compliance (COPPA, FERPA, GDPR), clear retention policies, and transparency about data use.

Equity: AI tools risk exacerbating inequalities—students without devices/internet cannot access system; API costs limit availability in low-income schools; current focus on Chinese/English excludes other languages; OCR issues affect students with less neat handwriting. Future work should address through offline modes, low-resource language support, and partnerships for equitable access.

Learning Impact: Over-reliance on AI-generated content could harm learning—skipping active note-taking reduces retention; auto-generated practice may miss individual gaps; AI errors mislead students. System should augment, not replace, traditional learning and teacher guidance.

Environment: Large-scale AI inference has carbon footprint. Future work should investigate efficient models and local inference.

AI Tools Statement

AI in System: The AI Study Assistant integrates: *ASR* (iFLYTEK ASR, OpenAI Whisper for transcription), *Vision/OCR* (Alibaba Qwen-VL via DashScope for text extraction and handwriting recognition), *LLM* (DeepSeek via OpenAI-compatible API for note generation, problem classification, similar problem generation, mind map generation), and *Embeddings* (architecture support for future RAG). Details in Section 4 and open-source repository at `https://github.com/kellyfeng0807/ai-study-assistant`.

AI in Report Writing: Authors used AI assistance (GPT-4/GPT-5 via ChatGPT) for: content organization (structuring to NeurIPS format), technical writing (converting README/code comments to academic prose), LaTeX formatting (syntax, tables, bibliography, style compliance), language refinement (clarity, conciseness, academic tone), and citation generation. All factual content based on actual codebase. Authors reviewed and verified all AI-generated content. Any errors remain authors' responsibility.

Acknowledgments

We thank users who tested the prototype and provided feedback. This course project benefited from educational resources and computational support from our institution.

References

[1] Bai, J., et al. (2024). Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. arXiv preprint arXiv:2308.12966.

[2] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust Speech Recognition via Large-Scale Weak Supervision. In *Proceedings of ICML 2023*.

[3] Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*.

[4] Touvron, H., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. arXiv preprint arXiv:2302.13971.

[5] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*.

[6] Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent Tutoring Systems. *Science*, 228(4698), 456–462.

[7] Siemens, G., & Long, P. (2013). Penetrating the Fog: Analytics in Learning and Education. *EDUCAUSE Review*, 46(5), 30–40.

[8] Buzan, T., & Buzan, B. (2006). *The Mind Map Book*. BBC Active.

[9] Feng, K., Ren, H., & Deng, J. (2025). AI Study Assistant: An AI-powered Learning Companion for Secondary Students. GitHub repository: `https://github.com/kellyfeng0807/ai-study-assistant`.