KNOW Y – SMART NOTES APP

Ganesha Subramanya V
Nitte Meenakshi Institute of Technology
ganeshsubramanya8@gmail.com

***Abstract*—This paper presents KNOW Y, an AI-powered academic note-taking system that implements reverse-concept learning to deepen student understanding. The system extracts key concepts from lecture material and uses a large language model (via the Groq API) to explain each concept by tracing back to its foundational ideas. For example, rather than merely defining inheritance, the system first reviews basic object-oriented principles and builds up to the specific concept, thereby reinforcing prerequisite knowledge. The web application is built on a MERN stack (MongoDB, Express, React, Node) with a Python backend for AI tasks. I introduce an adaptive "Understanding Score" model that fuses Item Response Theory (Rasch) and Elo-rating approaches to dynamically assess each student's mastery of concepts based on their interactions, notes, and quiz results. The architecture and data flow of the system are described in detail. In lieu of real user trials, I discuss simulated benefits: richer conceptual learning, personalized feedback on weak areas, and enhanced metacognitive insight. Future work will involve real-world evaluation and expanding the concept database.**

***Index Terms*—Artificial Intelligence, Natural Language Processing, Note-Taking, Tiptap, Markdown, MongoDB, JWT Authentication, Groq LLM**

I. Introduction

Traditional note-taking relies on students recording information verbatim or in linear outlines, which often encourages rote memorization rather than true understanding. Recent AI tools for students (e.g., Notion AI) have begun to automate summarization and content generation, but they do not explicitly incorporate pedagogical strategies for deep learning. I propose KNOW Y, a concept-based note-taking assistant that uses AI to drive reverse-concept learning: each topic is taught by first reviewing its underlying principles.

My contribution is three-fold: First, I survey related work in AI-driven educational tools, concept-based pedagogy, note-taking applications, and learning analytics. Second, I detail the methodology of KNOW Y, including my technology stack (MERN + Python), the Groq LLM API for content generation, the logic for extracting and organizing concepts from unstructured notes, and the design of the adaptive Understanding Score model. Third, I present the system architecture and discuss expected results. In this model, each interaction (notes, AI dialogue, quiz answers) updates a personalized understanding score for relevant concepts. The score is computed using a hybrid of Rasch and Elo rating formulas to provide adaptive feedback. We conclude with a discussion of simulated benefits and directions for future work.

II. Literature Survey

AI-driven educational technologies have seen significant advancements in recent years. Adaptive learning systems increasingly utilize statistical models to tailor instruction based on individual learner profiles. One foundational approach is the Rasch model, which calibrates student ability and question difficulty on a unified scale. Lye and Lim [1] demonstrate how Rasch-based estimates enable mapping of learner competence to the zone of proximal development (ZPD), facilitating actionable feedback on mastery progression in mathematics.

Complementing Rasch, Elo-based algorithms, originally designed for chess rankings, have been adapted for educational settings. These models treat students and items as dynamic competitors, updating scores after each interaction based on performance. Abdi et al. [2] introduced a multivariate extension of the Elo model that captures temporal shifts in learner proficiency, showing its effectiveness for real-time updating and personalizing learning paths in intelligent tutoring systems.

In parallel, concept-based pedagogy has emerged as a transformative instructional design, emphasizing the understanding of cross-disciplinary big ideas over rote memorization. Main [3] asserts that this approach positions conceptual understanding as the primary driver of learning, encouraging learners to connect knowledge across contexts. KNOW Y aligns with this philosophy by structuring notes around interconnected concepts, supported by AI-generated explanations that reinforce these links.

Several commercial tools integrate AI into the note-taking process, though most are productivity-oriented rather than pedagogically grounded. Notion AI, for example, provides features for summarization, rewriting, and brainstorming within the user's workspace. As described in the official guide [4], it serves as an "assistant" to accelerate content generation but lacks a strong instructional framework. Similarly, Google's NotebookLM and other tools offer summarization from user-provided documents, but they remain limited in educational intent.

KNOW Y differentiates itself by embedding educational structure through concept extraction, prerequisite-based explanations, and quiz integration. These features are supported by Groq's LLM inference endpoint, which offers extremely fast natural language processing capabilities for real-time interaction [5].

Lastly, the field of learning analytics provides a methodological foundation for using educational data to enhance instruction. Wise [6] defines learning analytics as the computational analysis of student-generated data to improve learning outcomes. KNOW Y applies these principles by capturing note-taking patterns, quiz scores, and AI feedback loops.

This data supports both learner self-reflection and instructor insights, enabling a continuous cycle of improvement.

REFERENCES

III. METHODOLOGY

I implemented KNOW Y as a MERN-stack web application with a Python backend for AI processing. The front-end is a React app that students use to write and review notes. The back-end, built with Node.js/Express, interfaces with a MongoDB database to store user profiles, notes, extracted concepts, and scores. Heavy AI tasks (natural language understanding, summarization, and query generation) are handled by Python microservices. I connect to Groq's API to call a state-of-the-art LLM with a large context window.

A. Concept Extraction

When a student submits new notes (or imports existing lecture text), the system extracts concepts automatically. We define a "concept" as a domain term or idea (e.g. "inheritance," "linked list") that belongs to the course ontology. In practice, we use a prompt-based LLM pipeline: the notes text is sent to the Groq LLM with a system message instructing it to list key concepts and their relationships. For example, "Extract all the key concepts and prerequisites from this computer science note." The model returns a structured list or graph of concepts. This output is parsed by Python and stored. If needed, we apply post-processing like lemmatization or matching to a known concept list to improve consistency.

Once concepts are identified, we create a concept map or graph. Edges indicate prerequisite or "concept of" relations (e.g. "A is a part of B" or "A leads to B"). Our system uses this map to present personalized content: for a given concept, we can retrieve its parent concepts for review. In reverse-concept learning, when the student selects or queries a particular concept, the system first fetches its foundational nodes and generates explanations starting from those. This mechanism enforces backward chaining through the concept graph, aligning with concept-based pedagogy.

B. Groq AI API Integration

We utilize Groq's API endpoints to run language-model completions. The Python backend uses the `groq` client library. Each API call includes system and user messages. For example, to explain a concept, the prompt might instruct: *"Explain the concept 'inheritance' to a beginner, first reviewing necessary background on classes and objects."* The model then generates a pedagogically-structured explanation. For efficiency, we select a Mixtral-based model (e.g., `mixtral-8x7b`) that supports long contexts, so we can feed large note segments or concept lists. The analytics blog notes that Groq's LPU provides "ultra-low latency capabilities crucial for AI support technologies" and can outperform conventional GPU APIs. This allows users to get AI responses in a few seconds even on detailed prompts.

We also use the LLM to generate quiz questions and hints. After a concept is covered, the system can ask Groq to create a short quiz or a conceptual question related to that concept. These generated questions are saved and presented to the student. Answering them triggers updates to the understanding score (see next section). In summary, Groq provides flexible natural language capabilities: summarization, concept definition, question generation and answer explanation, all woven into the note-taking workflow.

C. Understanding Score Model

A key innovation of KNOW Y is the "Understanding Score" (US) assigned to each student-concept pair. This score estimates how well the student understands a specific concept. We combine ideas from the Rasch model and Elo ratings to update this score in real time.

Conceptually, each student has a latent ability θ_s and each concept has a difficulty b_c. Using the Rasch item response logistic function, the probability $P_{s,c}$ that student s answers a concept-related question correctly is:

$$P_{s,c} = \frac{1}{1 + \exp(-(\theta_s - b_c))}.$$

Initially we set $\theta_s = 0$ and $b_c = 0$ for all. When a student attempts a quiz question on concept c, we observe a binary outcome $R_{s,c}$ (1 = correct, 0 = incorrect). We compute the expected score $E_{s,c} = P_{s,c}$. We then apply an Elo-style update to both the student ability and concept difficulty. For example, using a fixed learning rate K, we update:

$$\theta_s \leftarrow \theta_s + K(R_{s,c} - E_{s,c}),$$

$$b_c \leftarrow b_c - K(R_{s,c} - E_{s,c}).$$

These updates mean that if a student performs better than expected ($R > E$), their ability increases and the concept's effective difficulty decreases slightly (making it easier). Conversely, if they do worse, θ_s goes down and b_c goes up. This formulation ensures the ratings self-correct over time.

We compute the student's understanding score for concept c simply as a function of their updated ability minus difficulty:

$$US_{s,c} = \theta_s - b_c.$$

A higher US indicates stronger mastery. We also normalize these scores to a 0–100 scale for display. After each quiz or interaction, we recalculate $US_{s,c}$ for each involved concept. We use $K = 10$ (for example) as the update factor; this parameter can be tuned. This combined Rasch/Elo approach gives a probabilistic model of performance and a simple linear update rule.

In addition to quiz results, we allow other signals to affect the score. For instance, if a student actively uses AI prompts to elaborate on a concept (e.g. asking clarifying questions in the chat), we interpret this as engagement and give a small bonus to θ_s. Conversely, skipping important foundational review (e.g. never reviewing a prerequisite) could reduce the score, though we handle that conservatively. These heuristic adjustments aim to encourage the user to explore concepts thoroughly. The detailed logic combines:

- **Quiz Updates**: As above, correct/incorrect updates ability and difficulty.
- **Note Interaction**: When the student reads or queries notes on concept c, we slightly increase their $US_{s,c}$ (reflecting exposure).
- **Neglect Penalty**: If a student repeatedly fails to answer questions on c, we gradually decay $US_{s,c}$ to reflect forgetting.

Mathematically, the core update uses the logistic error term and linear Elo adjustment. Over time, $US_{s,c}$ converges to represent how often the student answers correctly relative to difficulty. This score drives our adaptive feedback: for example, if $US_{s,c}$ falls below a threshold, the system will recommend a concept review or additional examples for c.

IV. System Architecture

The system is built in a standard client-server configuration, described here in text:

- **Front-End (React):** The user interface provides a note editor, a list of concepts extracted, and quiz questions. It also shows the current Understanding Scores for each concept in the course. Students log in and select a course, then write notes or upload text (lecture transcripts). The UI components use state to track user inputs and display AI-generated content in real time.
- **Back-End (Node/Express):** The React app communicates with a Node.js server via RESTful APIs. When the user submits notes, the text is sent to the server. The server stores the raw notes in MongoDB and then invokes the Python AI module.
- **Python AI Service:** A Python service listens for tasks (concept extraction, explanation generation, quiz generation). It uses the Groq client: for concept extraction it sends the notes to Groq with a prompt to list key concepts. For each concept query, it sends prompts to generate explanations or questions. The service receives the LLM response, formats it, and returns it to Node.
- **Database (MongoDB):** The database stores user profiles, raw notes, extracted concepts and relations, generated explanations, quiz questions, and the current values of θ_s and b_c for all students and concepts. Each student-document contains a vector of Understanding Scores.
- **Workflow:** After processing, the server sends the AI-generated content (explanations, concept list, etc.) back to the front-end. The front-end presents these to the user. When the student answers a quiz question or re-reads a note, the front-end sends that outcome back to the server, which updates $US_{s,c}$ and stores the new θ, b values. The updated scores are then displayed in the UI, allowing the student to see their progress.

No external graphics or figures are included, so the entire architecture is described in text. In summary, the flow is: **User Action → React UI → Node Server API → Python/Groq LLM → Response → Database → Back to UI**. Security features include authenticated API tokens and encrypted database connections.

V. Results

As this is a design paper, we did not conduct live experiments. Instead, we reason about expected outcomes. KNOW Y aims to achieve:

- **Deeper Learning:** By forcing students to explain concepts in terms of their prerequisites, the system encourages elaborative learning. Educational research shows concept-based instruction promotes "deeper understanding" over rote memorization. We expect students to form stronger mental models, since reviewing foundational topics (with AI help) reduces knowledge gaps.
- **Personalized Feedback:** The Understanding Score provides tailored insights. For example, if a student struggles with the concept of "pointers," the system might notice low $US_{s,c}$ and highlight this weakness, offering extra examples or review questions. This mirrors Rasch/Elo adaptive feedback, where items in each student's ZPD are identified:contentReference[oaicite:21]index=21. Such feedback guides the student to focus on concepts that need remediation.
- **Metacognitive Awareness:** Students can view their score charts for each concept, reflecting on which topics are mastered and which are not. This aligns with learning analytics goals of making data-driven insights actionable. Knowing one's own understanding score can prompt self-reflection and motivate review. We anticipate that visualizing $US_{s,c}$ over time will help students regulate their study strategies.
- **Workflow Efficiency:** The integration of note-taking and AI means students spend less time on mechanical tasks. As Notion AI documentation notes, an embedded assistant "makes you more efficient" by automating summarization and formatting. We expect KNOW Y to similarly save time; for instance, meeting notes or textbook excerpts can be quickly transformed into structured study material with embedded conceptual explanations.

Consider a simulated example: a student takes notes on binary trees and answers a generated quiz on recursion. Each correct answer raises their US for "recursion" and "binary tree." If they repeatedly answer incorrectly on "recursion," the system might detect that their current US for recursion is low, and recommend reviewing a simpler prerequisite like "function calls." Over several sessions, the student's pattern of scores guides the AI to tailor support.

We do not have quantitative data, but we reason from analogous systems (adaptive tutors, learning analytics dashboards) that the combination of concept mapping and real-time adaptivity should yield positive effects on engagement and retention. Future user studies will be needed to confirm these expectations.

VI. Conclusion and Future Work

I have introduced KNOW Y, an AI-driven note-taking app that emphasizes reverse-concept learning. The system uses

modern web and AI tools (MERN stack with Groq LLM) to create a responsive learning assistant. Key innovations include automated concept extraction and the hybrid Understanding Score model that adapts via Rasch/Elo. These components work together to give students personalized feedback and deeper conceptual understanding.

For future work, I plan to conduct user studies to evaluate learning gains. I also aim to refine the scoring model, perhaps by incorporating richer student data (e.g., time spent on each concept, eye tracking) and using the Rasch model's full capabilities for partial-credit items. Additional enhancements could include multi-language support, mobile accessibility, and integration with learning management systems. By continuously expanding the domain ontology of concepts and improving the AI prompts, KNOW Y can evolve into a comprehensive academic learning tool that bridges note-taking and active learning.

REFERENCES

[1] C. Y. Lye and L. Lim, "Interpretating Rasch Ability and Difficulty Estimates to Inform Mathematics Learning through an Adaptive Learning System," in *Proc. Pacific-Rim Objective Measurement Symp. (PROMS)*, 2023, pp. 502–509.

[2] S. Abdi, H. Khosravi, S. Sadiq, and D. Gasevic, "A Multivariate ELO-based Learner Model for Adaptive Educational Systems," in *Proc. Int. Conf. Educational Data Mining (EDM)*, 2019, pp. 228–233.

[3] P. Main, "Concept-Based Learning," *Structural Learning*, Apr. 26, 2024. [Online]. Available: structural-learning.com/post/concept-based-learning.

[4] Notion Labs, "Use Notion AI to write better, more efficient notes and docs," Notion Help Center, 2023. [Online]. Available: notion.com/help/guides/notion-ai-for-docs.

[5] A. Ajay, "Getting Started with Groq API: The Fastest Ever Inference Endpoint," *Analytics Vidhya Blog*, Jan. 7, 2025. [Online]. Available: analyticsvidhya.com/blog/2024/03/getting-started-with-groq-api.

[6] A. Wise, "What is Learning Analytics?," NYU LEARN Network (LEARN), 2019. [Online]. Available: steinhardt.nyu.edu/learn/learning-analytics-101.