Proceedings of the 2nd International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/132/2024.20528

ArthurNote: An AI-Driven Note-taking Platform Eliminating Repetitive Tasks

Chuxuan Gao[1,3,*], **Zhan Shi**[2,4]

[1]School of Computer Science & Technology, Soochow University, Suzhou, 215000, China
[2]International School of Information Science & Engineering, Dalian University of Technology, Dalian, 116024, China

[3]arthurfish8051@gmail.com
[4]no39chaoshope@gmail.com
*corresponding author

Abstract. The note-taking procedure performs a significant role in education and business, offering a simple methodology to schedule routines, emphasize important parts, and review previous events. However, it is tedious and superfluous for students to record common knowledge that already exists in the public domain. Existing online noting platforms can not satisfy our requirements. This paper introduces an innovative approach to enhancing the note-taking experience by integrating large language models (LLM) and cutting-edge development technology within our AI-aided note-taking platform. The development of the new platform is based on a methodology of front-end and back-end separation, and it is empowered by the NoSQL cloud database and online LLM service to increase its scaling ability. With LLM's help in completing, the redundant efforts of users have been eliminated to a large degree.

Keywords: note-taking LLM AI education cloud database efficiency

1. Introduction

Taking notes is an essential way to outline newly studied content and review the mastery of knowledge. However, redundant efforts to write down information already available in the public domain are both boring and disappointing.

Consider a situation in a calculus class: students may write down the differential law on their laptop or tablet while listening to the lecture. However, this note-taking behavior will not only distract the student but also be a waste of time: it is easy to find the mentioned information on the Internet. Additionally, it can be easily inferred that the student may later note down the chain rule or the integral law.

There are already several choices for note-taking. However, none of these satisfy the requirement of eliminating redundant work. Notion is an integrated note-taking platform that offers the ability to collaborate in "Workspaces" with different modules like databases, task managers, and calendars. Despite its popularity and high productivity, it is a closed-source commercial software, which means users cannot modify and conduct secondary development on it to get personalized and sufficiently efficient editing experiences. Moreover, its AI integration is unnatural and sometimes counter-intuitive.

Affine is a promising open-source alternative to Notion and provides a better experience for mixing notes with drawings and text. It offers the ability to connect to AI during the drafting process. Although it seems a better choice for note placement, it is still under development and unstable to use.

GitHub Copilot is a smart code completion tool for programming and development. Empowered by cutting-edge large language models, it can generate accurate code snippets from inline comments and offer improvement suggestions based on its analysis of the codebase. It effectively completes tasks as expected for coding work. However, it cannot provide completions for plain text.

To summarize, existing platforms cannot satisfy the demand to reduce redundant work in the note-taking process. We believe learners should not be burdened with writing down knowledge that can be easily obtained from public sources or doing repetitive work. Note-taking should not shoulder the duty of copying and pasting but focus more on outlining and reflection. To achieve this vision, we propose a next-generation AI-based online note-taking platform: ArthurNote. It eases users' unnecessary burden through offering completion function by LLM invocation. The platform contributes to integrating ordinary note-taking application's functions and auto-completing under LLM's assistance. Nevertheless, advanced developing technology is applied to leverage the user experience.

2. User Interface Design

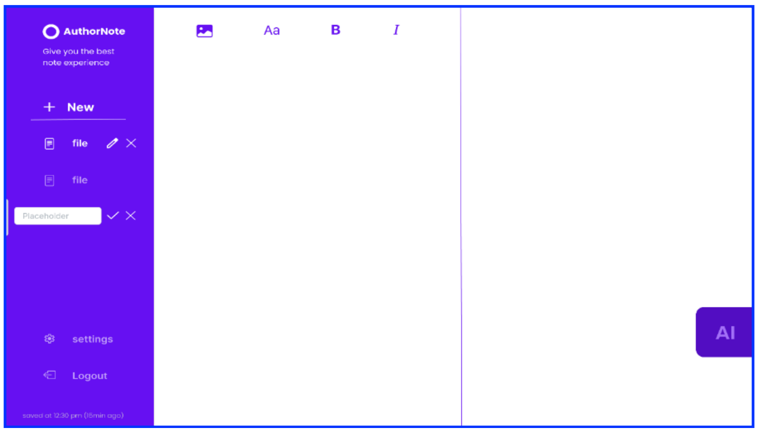
AuthorNote
Give you the best note experience
New
file
file
Placeholder
settings
Logout
saved at 12:30 pm (15min ago)
Aa
B
I
AI

Figure 1. Main page user interface design draft

2.1. Color and Style

Theme Color: The dominant color, Deep Purple (#6610F2), imparts a modern and technological ambiance.

Background and Text: The background is maintained in a lighter hue to accentuate the deep purple sidebar and buttons, while the text is presented in white or light gray to enhance readability. The dominant color is a deep purple (#6610F2), which lends a contemporary and technological ambiance to the overall theme.

The background and text are presented in the following manner: To enhance the visual contrast, the background is maintained in a lighter hue, thus accentuating the deep purple sidebar and buttons. To facilitate readability, the text is presented in white or light gray.

2.2. Layout Design

Sidebar: This is located on the left side of the interface and is used for navigation purposes such as file management and settings options. The sidebar can utilize Bootstrap's off-canvas component for responsive hiding on smaller screens.

Text Editing Area: This area is located in the center and offers a large space for users to input and edit text. It can be implemented using a text or content editable element with borders that match the deep purple theme.

AI Text Generation Window: The text displayed on the right-hand side has been generated by AI. This area may be designed using a card layout, which could potentially include tool buttons at the top, such as "save" and "clear."

2.3. Interactive Elements Design

AI Button: Located in the bottom right corner, styled as a Floating Action Button (FAB) in deep purple, with noticeable visual feedback upon hovering or clicking. The functionality of the AI button may be utilized in two distinct ways, contingent on the specific usage scenario. Firstly, users may click the AI button directly to generate ChatGPT's answer to the entire note content. Secondly, users may employ the right mouse button to cross out keywords and then press the AI button to realize precise generation.

User Guidance: Combining text prompts and visual icons to guide users on how to use various areas. For instance, providing brief explanations or icons at the top of the text editing area and the AI text generation window.

Functions: The software offers a variety of functionalities, including the ability to alter the appearance of notes through modifications such as font, font depth, and color changes. Additionally, it provides the option to utilize notebook ground storage, which can enhance work efficiency. Furthermore, users can customize other note settings, such as incorporating personalized fonts and third-party plug-ins, to create a distinctive note-taking experience. The software also allows for the direct integration of AI-generated content into notes through a drag-and-drop interface, which can streamline the note-taking process.

3. Architecture Selection

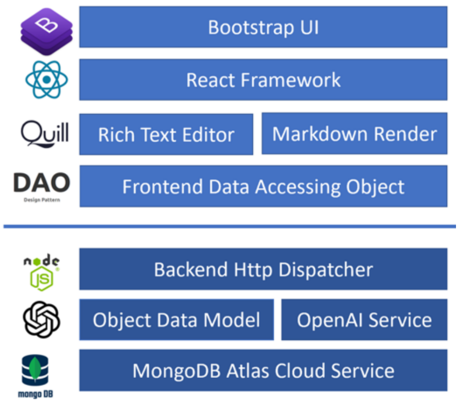

Figure 2. System Hierarchy

We want the application to run on a wide range of devices, from laptops and tablets to phones, so we chose a Browser/Server architecture. We decided to build a web app.

We selected Software-as-a-Service (SaaS) as the delivery method for this application to reduce development time. David Norton[1], a research director at Gartner, noted that the SaaS model is well-suited for iterative system development and has the potential to shorten the overall development cycle.

Moreover, the SaaS approach aligns well with our commitment to continuous improvement and user-centric design. It allows us to gather real-time usage data and implement improvements based on actual user behavior, rather than relying solely on pre-launch projections. This data-driven approach, facilitated by the SaaS model, promises to yield a more refined and user-friendly final product.

Taivalsaari[2] explored the feasibility of using web browsers as a platform for applications. While there is still room for improvement, web browsers effectively support progressive development, which aligns well with our development constraints. Modern browsers now offer powerful APIs, enhanced performance, and improved security features, making them increasingly suitable for hosting sophisticated software. This evolution has blurred the lines between traditional desktop applications and web-based solutions, opening up new possibilities for developers.

The architecture of the web app follows typical Frontend and Backend separation practices. Given the close project deadline and the shortage of developers, the system had to focus primarily on dealing with complexity. The logic should be structured, the framework should be flexible enough to be extended, and the codebase should be easy to maintain.

Traditional backend web template rendering solutions cannot satisfy the demand of reducing complexity. If we adopted this pattern in the process, the backend development effort would far exceed the frontend's. Frontend programmers would just need to transform the design draft to HTML & CSS files, but backend programmers would have to consider more aspects such as page routing and note page traversal logic and would be trapped in language switching within a single file.

According to the analysis, we finally chose this architecture for this project. The architecture is simply demonstrated in Fig.2. At Fig.2, frontend components sit on the top of the figure, in contrast, backend components lie on the bottom.

The front’s architecture is shown in the upper part of Fig 2. React the framework plays a glue role among other frontend components for its outstanding performance among other frameworks[3]. Bootstrap UI accounting is the top-most position in the Fig.2 for offering a pretty and user-friendly interface. Quill is a rich text editor taking the responsibility of note editing. A key factor in choosing Quill.js is its compatibility with Y.js, which provides peer-to-peer, real-time collaborative editing capabilities[4]. This integration satisfies the requirements for effective teamwork and collaboration. Markdown renders Remark.js render AI responses in a well-organized way. Frontend’s data access object plays a throttle and encapsulating role and bridges the user interface and backend controllers.

The backend’s hierarchy is demonstrated in the lower part of Fig.2. Express Node.js The server conducts the work of dispatching HTTP requests sent from the front end to the object data model and the OpenAI Service. The object data model reduces boilerplate codes in the database connection. OpenAI Service forwards the frontend’s completing request with a hand-crafted prompt and sends back LLM’s response through Server Sent Event. The bottom component in Fig.2 is MongoDB Atlas cloud service, for the purpose of reducing complexity[5] when deploying locally and getting NoSQL’s flexibility when data schema changes.

Fig.2 shows the separation of frontend and backend provides a balanced and simple architecture of the application. With the separation, the work of frontend and backend could be dealt with parallelly and integrated easily.

4. Application Architecture Description

A rough look at data flow is described in Fig.3. The Frontend / Backend division can be easy to view in the figure. Cloud service is separated from the backend just for the purpose of clarity.

Frontend takes responsibility for rendering the AI-assisted note-taking interface and reacting to user editing and operating actions immediately. Control logic and display styles are encapsulated into individual React components, with messages passed through React's unidirectional data flow using props (properties) and callback functions.

Frontend components are wrapped in a yellow dotted frame in Fig.3 and consist of three important components: note selector, rich text editor, and AI assistant. These three components were painted yellow in Fig.3 and interacted with the App component. Briefly speaking, the note selector displays the titles of notes, enabling users to select notes they want to focus on, and providing a note creation option. The rich text editor allows users to unleash their creativity during composition, making it easy to emphasize sentences and list points worth noting, and even provides the ability to upload and save photos. The AI assistant manages the connection to fetch AI responses from the server and renders the Markdown-formatted responses attractively. These three components work together asynchronously, not only rendering content but also handling data fetching and persistence requests.

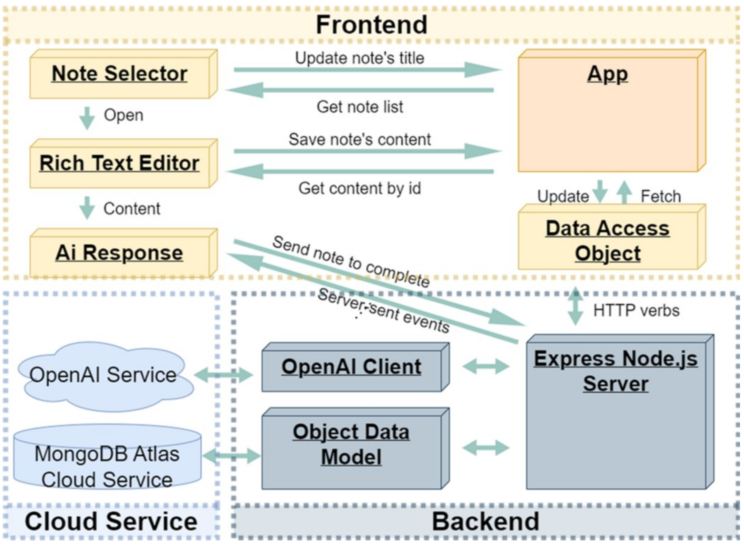

Figure 3. Application Architecture Diagram

The backend is painted grey in Fig.3, which can be logically divided into three parts: a Node.js Express server, a Data Access Object with MongoDB cloud service Atlas, and an OpenAI client to connect to OpenAI online LLMs. To avoid dealing with the complexity of deploying and managing a database locally, we used a cloud service instead of local installation.

During the development process, we noticed that cloud service databases have many advantages compared to local deployments, such as the convenience of authorization, payload monitoring, and manipulating data with a user-friendly Graphical User Interface instead of a Command Line Interface. Fixed connection URLs make it easier to deploy the website on public cloud platforms.

The Express Node.js server handles requests sent from the frontend React app and dispatches them simply and asynchronously during the event loop. The module is responsible for responding to data-fetching requests, AI assistance requests, and data-updating requests. This module runs on the Express.js framework and is based on Node.js runtime. Empowered by Node.js's non-blocking I/O and single-thread event-driven programming model, it runs with efficient performance.

5. System Interaction Sequence

The whole logic can be divided into two parts: the note modification part (on the left-hand of Fig.4), and the AI response fetching part (On the right-hand of Fig.4). The note modification parts are constructed by three participants: UI, Data Access Object (DAO), and the backend server. The UI and the DAO are wrapped in the frontend boundary, which is shown on the left half of Fig.4. When the user enters the platform, the UI will get all the notes from the backend server through the frontend’s DAO.

DAO bridges the data transmission through function encapsulation and data catching and plays a throttle role between the UI and the server. In Fig.4, it is posited between the UI and the server to demonstrate this relation. When the user begins working on the note’s editing, the modification sent from the UI component to the DAO will be cached to reduce unnecessary data transmission through the Internet. Until some condition is satisfied like a time interval arrived, then the modification will be synchronized to the backend. It is easy to view the process in Fig.4: Although the user performed so many operations in a short time, there is just one synchronization workload to the server.

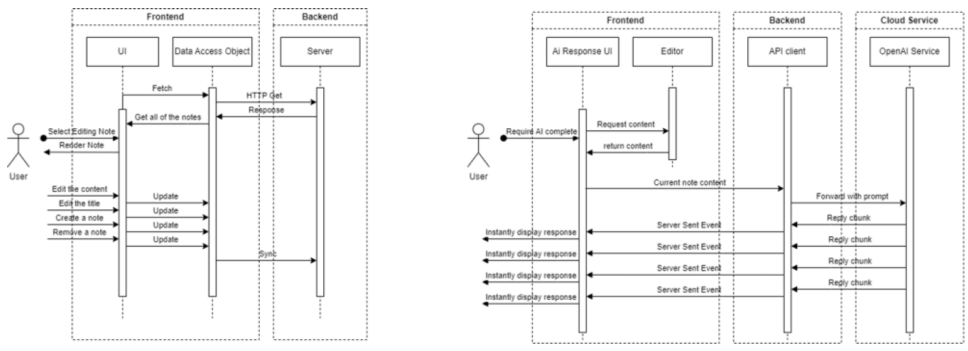

Figure 4. The Sequence Diagram of the application

The right half of Fig.4 shows the sequence of the AI response part. The focus point of the part is the Server-Sent Event (SSE) forwarded from the backend's API client. This technology-enhanced user experience by reducing the time to first respond and decreasing wait anxiety. Fig.4 visualizes this by painting one user request closely followed by so many continuous returned chains of content. Every chain is constructed by the cloud service's reply chunk, backend sent Server Sent Event, and UI displayed instant response.

6. Development Process

The development process can be divided into two prototyping phases, called the Linear Block Phase and the Document Phase.

During the Linear Block Phase, the notes were separated into several blocks, each of which was connected to the next in a linear sequence. Each text editing box was coupled with an AI response block, which provided information. However, this approach proved ineffective due to the complexity of addressing each paragraph individually at the user interface level.

The rationale for the termination of this methodology was the exorbitant and impractical modifications that would have been necessary to implement it. The rendering of different blocks with disparate editor components necessitated substantial modifications to the third-party editor component. The proposed modifications were highly intricate and complex, and the compressed project timeline did not allow for the necessary time to implement them. Consequently, we elected to abandon this development path and pursue an alternative approach, leveraging the experience gained in previous endeavors.

Subsequently, we proceeded to the phase of the process designated as the "Document Phase." The designation "Note Document" was chosen to reflect the decision to consider the text as a unified entity rather than as a collection of discrete paragraphs. The use of the Delta file format enabled more concise modeling of the text's relationships than would have been possible with a linear paragraph-linked list. The AI component was unified rather than fragmented. Subsequent research demonstrated that this schema modification significantly enhanced the overall system.

The schema adjustment resulted in a reduction in the complexity of the backend data transfer and frontend content rendering processes. From the perspective of the backend, the processing and transfer of a single document is a significantly more straightforward undertaking than the separation of paragraphs while still maintaining the ability to manage the relationships between paragraphs. From the perspective of the front end, the rendering of a single editor is evidently a more efficient process than the rendering of multiple editors. The data became more unified, and the user interface became more straightforward to arrange.

7. Discussion

The issue of public knowledge entry becoming a source of distraction has been resolved in a straightforward manner. However, subsequent to the development of this iteration of the platform, it became evident that there is still scope for enhancement with respect to user interaction. Although the prototyping stage has demonstrated that clicking a button to obtain the LLM's answer as completion is an effective method, it is evident that this approach may be perceived as an interruption when users are engaged in note-taking activities. Drawing inspiration from integrated development environments like Visual Studio, we could implement a system that offers context-aware suggestions as the user types. This method could provide real-time assistance without requiring explicit user action. Furthermore, Implementing a system that learns from user behavior and tailors its suggestions accordingly could significantly enhance the relevance and usefulness of completions over time.

In subsequent work, the objective is to develop a more sophisticated and less obtrusive user interface. One potential avenue for exploration is the provision of completions in a manner akin to Visual Studio's IntelliSense. An alternative approach could be the active presentation of completions via a floating menu. Regardless of the selected approach, it is essential to anticipate and address the challenges associated with increased complexity and work volume as the project evolves. Maintaining a modular architecture to allow for easier updates and maintenance may be helpful in achieving this goal. To address these challenges basically, we might need to adopt an agile development approach with frequent iterations and user feedback

8. Conclusion

The act of taking notes is an effective method for acquiring and processing information. However, the act of entering common knowledge may prove to be a distraction, potentially leading users to neglect the critical process of applying their own reasoning and analytical abilities. Despite the plethora of note-taking platforms with diverse functionalities, none of them can assist users in circumventing this pitfall. We put forth a proposal for an AI-aided note-taking platform with the objective of preventing users from spending time on the repetitive entry of commonly known knowledge. The platform addresses this issue by invoking LLM to facilitate the auto-completion of users' notes. Moreover, the platform has been developed using cutting-edge technology with the objective of reducing hazardous complexity, ensuring robust data storage, and providing a user-friendly interaction experience throughout the note-taking process.

Acknowledgement

Chuxuan Gao and Zhan Shi contributed equally to this work and should be considered co-first authors.

References

[1] G. Goth, “Software-as-a-Service: The Spark That Will Change Software Engineering?,” 2008. [Online]. Available: http://download.sonicsoftware.com/open/

[2] A. Taivalsaari, T. Mikkonen, D. Ingalls, and K. Palacz, “Web browser as an application platform,” in *EUROMICRO 2008 - Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2008*, 2008, pp. 293–302. doi: 10.1109/SEAA.2008.17.

[3] O. C. Novac, M. C. Novac, M. Oproescu, A. C. Mlinarcic, C. E. Gordan, and C. M. Dindelegan, “Employing Comparative Study between Frontend Frameworks. React Vs Ember Vs Svelte,” in *Proceedings of the 16th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ECAI61503.2024.10607455.

[4] P. Nicolaescu, K. Jahns, M. Derntl, and R. Klamma, “Near real-time peer-to-peer shared editing on extensible data types,” in *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, Association for Computing Machinery, Nov. 2016, pp. 39–49. doi: 10.1145/2957276.2957310.

[5] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, “SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review,” Jun. 01, 2023, *MDPI.* doi: 10.3390/bdcc7020097.