NOTEYBOT

PROJECT REPORT

Submitted by

R.S.BHOOPATHI(22-MC-02)

Under the guidance of

Dr. M. SAKTHI MSc., M.Phil., Ph.D.,

Associate Professor & Head, PG Department of Computer Science(SF)

A project report submitted to

Nallamuthu gounder Mahalingam College (Autonomos),

(An ISO 9001 : 2015 Certified Institution)

Re-Accredited by NAAC,

Pollachi – 642001.

Affliated to Bharathiar university, Coimbatore,

In Partial fulfillment of the requirements for the award of the Degree of

MASTER OF COMPUTER SCIENCE

APRIL - 2024

CERTIFICATES

CERTIFICATE – I

This is to certify that the project work entitled **"NOTEYBOT"** is a bonafide recode of the work done by **R.S. BHOOPATHI (22-MC-02)** doing II-Msc, Computer Science submitted to Nallamuthu Gounder Mahalingam College, affiliated to Bharathiar University in partial fulfillment of the requirements for the award of the degree of M.Sc. Computer Science under my supervision.

GUIDE **HEAD OF THE DEPARTMENT**

This is to certify that this is a bonafide record of the work submitted by

NAME : R.S. BHOOPATHI

REG NO : 22-MC-02

SUBJECT TITLE : NOTEYBOT

SUBJECT CODE :

For External Viva-voce examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

CERTIFICATE – II

This is to certify that the project work entitled as **"NOTEYBOT"** is the original work done by **R.S.BHOOPATHI (22-MC-02)** under the guidance of me .

Date :

Place :

_____

Signature of the Guide

Dr.M.SAKTHI MCA., M.Phil.,

Ph.D.,

CERTIFICATE – III

This is to certify that the project work entitled as **"NOTEYBOT"** is the original work done by us under the guidance of **Dr.M.SAKTHI MSc., M.Phil., Ph.D.,**

Date : _____

Place : **R.S.BHOOPATHI (22-MC-02)**

DECLERATION

DECLERATION

I hereby declare that the project work entitled **"NOTEYBOT"** Submitted to **"NALLAMUTHU GOUNDER MAHALINGAM COLLEGE, POLLACHI",** Affiliated to Bharathiar University is a record of original work done by me during 2023-2024 under the supervision and guidance of **Dr.M.SAKTHI MSc., M.Phil., Ph.D.,** Assistant Professor, Department of Computer Science.

Date :

Place :

Signature of the Candidate

R.S.BHOOPATHI (22-MC-02)

CONTENTS

CONTENTS

S.NO	PARTICULARS	PAGE NO
	CERTIFICATES	
	DECLARATION	
	INTRODUCTION AND OBJECTIVE	**1**
	SYNOPSIS	2
	INTRODUCTION	3
	OBJECTIVE OF THE PROJECT	4
	COMPANY PROFILE	5
1	**SYSTEM SPECIFICATION**	**6**
	1.1 HARDWARE SEPCIFICATION	7
	1.2 SOFTWARE SPECIFICATION	7
2	**SYSTEM STUDEY**	**9**
	2.1 EXISTING SYSTEM	9
	2.2 DRAWBACKS	9
	2.3 PROPOSED SYSTEM	10
	2.4 PLANNING AND SCHEDULING	11
3	**SYSTEM DESIGN**	**14**
	3.1 OVERVIEW OF THE PROJECT	15
	3.2 MODULES OF THE PROJECT	16
	3.3 INPUT DESIGN FORMAT	17
	3.4 OUTPUT DESIGN FORMAT	18
	3.5 SUPPORTING DIAGRAM	19
4	**IMPLEMENTATION AND TESTING**	**20**
	4.1 CODING METHODS	21
	4.2 TESTING APPROACH	23
	4.3 IMPLEMENTATION AND MAINTENANCE	24
5	**PROJECT EVALUATION**	**25**
	5.1 PROJECT OUTCOME	25
	5.2 LIMITATIONS OF THE PROJECT	28
	5.3 FURTHER SCOPE OF THE PROJECT	29
	CONCLUSION AND SCREENSHOTS	**33**
	CONCLUSION	34
	SCOPE FOR FUTURE ENHANCEMENT	35
	BIBILIOGRAPHY	36
	APPENDIX	37
	SAMPLE CODING	37
	SCREENSHOTS	51

INTRODUCTION AND OBJECTIVE

SYNOPSIS

The project entitled **“ NoteyBot ”** React website stands at the forefront of modern note-taking platforms, designed with innovation at its core. It seamlessly integrates advanced AI capabilities into its interface, providing users with an incredibly intuitive platform to create and organize their notes. This isn't just another note-taking app; it's a leap forward in productivity tools. By harnessing the power of AI, it transforms the act of jotting down thoughts into an engaging, interactive experience.

One of its groundbreaking features is the auto image generation functionality. Here's where the magic truly happens: when users input their notes, NoteyBot doesn't merely record text; it intelligently analyzes the content and translates it into visually captivating images that encapsulate the essence of the notes. These generated images serve as dynamic visual representations, enhancing the understanding and retention of information. NoteyBot goes beyond traditional note-taking by providing a unique and visually stimulating way to capture ideas.

Moreover, NoteyBot doesn't stop at image creation. It employs sophisticated algorithms to delve deeper into the context of the notes, generating complementary text snippets that align with the content. This contextual text augmentation isn't just a duplication of the notes; it's a clever addition that provides further insights, explanations, or related information. This fusion of AI and note-taking transforms the user experience, offering an unparalleled way to not only capture but also understand and expand upon thoughts seamlessly.

INTRODUCTION

The “ Noteybot “ emerges as an innovation in the realm of digital note-taking. Meticulously crafted using ReactJS and NextJS, fortified by the groundbreaking capabilities of the DALL-E API, this platform signifies a new era in organizing and cultivating thoughts. At its core, a harmonious integration of artificial intelligence prowess meets the adaptability of a NoSQL database.

Within this visionary platform, the convergence of ReactJS and NextJS establishes a robust foundation, enabling a seamless user experience while harnessing the extraordinary potential of the DALL-E API. Envision an AI-powered assistant transforming ideas into visually stunning representations, effortlessly stored and retrieved through the flexibility of a NoSQL database. Here, innovation isn’t just a buzzword; it’s the essence of every feature, meticulously designed to redefine conventional note-taking methodologies.

AI Notes transcends mere information recording; it serves as a conduit for creative expression. It’s the canvas where ideas take form with the aid of cutting-edge technology. Visualize a platform adapting to thought processes, offering intuitive tools powered by AI algorithms, seamlessly transforming concepts into tangible notes. Through ReactJS and NextJS, an environment has been cultivated that’s not just user-friendly but beckons into a realm where imagination and technology converge harmoniously.

OBJECTIVE OF THE PROJECT

The primary goal of this project is to design, develop, and implement an advanced AI notes taker platform leveraging cutting-edge technology, specifically integrating DALL-E's image generation capabilities. This platform aims to revolutionize traditional note-taking methodologies by offering automated image creation and intelligent auto-fill functionalities, enhancing user experience and efficiency in knowledge organization.

Introduction and Purpose

The project aims to create an AI-driven note-taking solution that transcends conventional text-based systems. Harnessing the power of DALL-E, an AI model capable of generating diverse and contextually relevant images from textual descriptions, this platform empowers users to effortlessly transform text-based notes into vivid visual representations. This integration caters to diverse learning styles and enhances information retention by providing visual cues aligned with textual content. Moreover, the project intends to streamline the note-taking process, reducing cognitive load, and enabling effective information capture and retention by associating text with visually stimulating content.

Key Features and Functionality

The AI notes taker will boast two distinctive yet interconnected features: auto image generation and intelligent auto-fill. Firstly, the auto image generation functionality powered by DALL-E enables users to input textual descriptions, keywords, or concepts within their notes, prompting the system to autonomously create relevant images encapsulating the essence of provided information. This feature aims to facilitate better comprehension, aiding information recall, and fostering creativity within note-taking practices. Secondly, the auto-fill capability utilizes AI algorithms to predict and suggest relevant content based on contextual analysis, completing or expanding upon user-entered text. This functionality aims to expedite note-taking by providing predictive and contextually coherent content, reducing manual effort required for comprehensive notes.

Expected Impact and Benefits

The successful implementation of this AI notes taker marks a significant milestone in educational technology and knowledge management. It is anticipated to empower students, professionals, and individuals across various domains by offering a more intuitive, efficient, and visually enriched note-taking experience. The platform's ability to seamlessly integrate visual elements through DALL-E's image generation and provide auto-fill suggestions will not only enhance productivity but also foster deeper understanding and engagement with captured content.

COMPANY PROFILE

SYSTEM SPECIFICATION

1. SYSTEM SPECIFICATION

1.1 HARDWARE SPECIFICATION

- Intel i5 11-Generation is used as a processor and it is sufficient enough for running a front-end API's and a basic react server for a long period of time

- Maximum of 8 GB RAM and 256 ROM storage is being used to store and run the application

1.2 SOFTWARE SPECIFICATION

Software IDE

- Visual Studio code

Database

- NeonDB
- FireBase Storage

Operating System

- Windows 11 Home is the operating system is being used

Frontend Languages

- NextJs 13
- TailWind Css
- OpenAI (DALL-E)
- Vercel AI SDK
- Versel Edge runtime

SYSTEM STUDY

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

The React-based note-taking website is designed to provide users with a straightforward and efficient platform for managing their notes. At the core of the system is the App component, which serves as the central hub responsible for overseeing the entire application. Within its state, an array named notes is maintained to store the list of user-generated notes. This component encapsulates the essential functions, such as addNote and deleteNote, facilitating the addition and removal of notes within the application.

Complementing the App component, the NoteForm component takes charge of handling user input for creating new notes. It maintains its local state, particularly newNoteText, to keep track of the text entered by the user. As users submit the form, the addNote function in the App component is invoked, ensuring the seamless addition of new notes to the overall collection.

The NoteList component is responsible for presenting the existing notes to the user. It receives the notes array as a prop from the App component and the deleteNote function, enabling the deletion of specific notes. Through the mapping of the notes array, this component dynamically renders each note along with a corresponding delete button. The integration of the deleteNote function ensures the smooth removal of a note when the user clicks the delete button.

For a visually appealing user interface, the styling of the components is achieved using standard CSS or a styling library such as styled-components. This design approach focuses on simplicity and usability, aiming to provide users with an intuitive experience for managing their notes effectively.

In summary, the React-based note-taking website leverages the component-based architecture of React to create a user-friendly platform. The system excels in simplicity and functionality, enabling users to effortlessly add, view, and delete notes through a well-structured and visually pleasing interface.

2.2 DRAWBACKS

One potential drawback of a React-based note-taking website lies in its reliance on client-side rendering. React applications are typically rendered on the client side, meaning that the entire application, including its components and logic, is loaded onto the user's device in the

browser. While this approach offers advantages in terms of responsiveness and interactivity, it can also pose challenges, especially when it comes to search engine optimization (SEO).

Search engines may face difficulties in crawling and indexing content generated dynamically on the client side, potentially impacting the discoverability of the note-taking website's content. Unlike traditional server-side rendering where HTML content is generated on the server and sent to the client, React applications often depend on JavaScript to render content on the client side. As a result, search engines may not effectively process and index the dynamically generated content, leading to suboptimal SEO performance.

To mitigate this drawback, additional efforts may be required to implement server-side rendering (SSR) or static site generation (SSG) techniques in a React application. SSR involves rendering components on the server and sending pre-rendered HTML to the client, while SSG generates static HTML pages during the build process. These approaches can enhance SEO by providing search engines with more accessible and indexable content.

In summary, the drawback of client-side rendering in a React note-taking website manifests in potential SEO challenges, where the dynamically generated content may not be as effectively indexed by search engines. Addressing this limitation may involve implementing server-side rendering or static site generation to improve the website's search engine visibility and overall performance in terms of discoverability.

2.3 PROPOSED SYSTEM

Integrating React Note with both OpenAI and DALL-E, while introducing powerful AI capabilities, may also pose certain drawbacks that need careful consideration. One significant concern is the potential increase in system complexity, which can lead to performance challenges. The integration of OpenAI and DALL-E introduces additional layers of processing, including handling requests to AI models for natural language processing and image generation. This could result in increased latency, affecting the responsiveness of the application, particularly if the AI models require time-consuming computations. Balancing the seamless user experience with the computational demands of AI integration is crucial to avoid any degradation in performance.

Another drawback to consider is the dependency on external services and APIs. Both OpenAI and DALL-E would require reliable internet connectivity and access to their respective services. Any disruptions in service availability or changes in API versions may impact the functionality of the note-taking application. Additionally, relying on external services introduces potential privacy and security concerns, especially if the notes contain sensitive information. Developers must implement robust error handling mechanisms and contingency plans to address potential service interruptions and maintain a secure environment for users.

Furthermore, the cost associated with using OpenAI and DALL-E services can be a notable drawback. These services may involve usage-based pricing models, and as the user base and usage of the note-taking application increase, the associated costs may escalate. Implementing cost-effective strategies, such as caching or optimizing API usage, becomes essential to manage expenses and ensure the sustainability of the integrated system.

Maintaining a balance between the benefits of AI integration and the potential drawbacks requires careful planning and consideration of the overall user experience. It is essential to assess the trade-offs between enhanced functionality and the associated complexities, performance considerations, dependency on external services, security implications, and the cost of utilizing AI services to deliver a reliable and sustainable React Note application.

2.4 PLANING AND SCHEDULING

1. Define Requirements :

Objective:

Clearly define the features and functionalities required for the note-taking website.

Tasks:

Identify user stories and use cases.

Specify the integration points for OpenAI and DALL-E.

Define the data model for notes, including both text and associated images.

2. Research OpenAI and DALL-E Integration (Week 3-4):

Objective:

Gain a deep understanding of OpenAI's API for natural language processing and DALL-E for image generation.

Tasks:

Explore documentation and examples provided by OpenAI and DALL-E.

Understand API usage limits, authentication, and best practices.

3. Set Up React Project (Week 5):

Objective:

Create a foundational React project structure.

Tasks:

Use a tool like Create React App for project initialization.

Set up project dependencies, including React, and any additional libraries for styling and state management.

4. Design User Interface (Week 6-7):

Objective:

Create wireframes and design the user interface for the note-taking application.

Tasks:

Sketch UI components for notes, forms, and integration points with OpenAI and DALL-E.

Consider the user experience for auto-complete and auto image generation.

5. Implement Basic Note Functionality (Week 8-10):

Objective:

Establish the basic note creation, retrieval, and deletion functionalities.

Tasks:

Implement the App component with state management for notes.

Create the NoteForm and NoteList components.

6. Integrate OpenAI for Auto-Complete (Week 11-13):

Objective:

Implement auto-complete functionality using OpenAI's API.

Tasks:

Set up API calls to OpenAI for natural language processing.

Integrate auto-complete suggestions into the NoteForm component.

7. Integrate DALL-E for Auto Image Generation (Week 14-16):

Objective:

Implement auto image generation using DALL-E.

Tasks:

Set up API calls to DALL-E for image generation based on text.

Integrate the generated images into the note display in the NoteList component.

8. Refinement and Testing (Week 17-18):

Objective:

Refine UI/UX, address bugs, and conduct thorough testing.

Tasks:

Test the application for responsiveness, usability, and integration functionalities.

Collect user feedback and make necessary adjustments.

9. Optimize and Secure (Week 19-20):

Objective:

Optimize performance and enhance security.

Tasks:

Implement caching strategies to optimize API usage.

Ensure secure handling of user data, considering privacy implications.

10. Documentation and Deployment (Week 21-22):

Objective:

Document the codebase and deployment process.

Tasks:

Create comprehensive documentation for developers and users.

Deploy the application to a hosting platform, ensuring proper configuration.

SYSTEM DESIGN

3. SYSTEM DESIGN

3.1 OVERVIEW OF THE PROJECT

NoteBot is an innovative project that combines the power of React, OpenAI, and DALL-E to create a cutting-edge web application designed to revolutionize the way users interact with and generate visual content. At its core, NoteBot is a React-based website that provides users with a seamless and intuitive interface for creating and managing notes. The integration with OpenAI introduces advanced natural language processing capabilities, allowing users to generate textual content with unprecedented ease and efficiency.

One of the standout features of NoteBot is its integration with OpenAI, which enables users to leverage state-of-the-art language models for generating coherent and contextually relevant textual content. This integration allows users to effortlessly create detailed and context-aware notes by simply inputting prompts or queries. OpenAI's powerful language models enhance the user experience by providing intelligent and contextually aware suggestions, making note-taking a more intuitive and efficient process.

Taking creativity to the next level, NoteBot incorporates DALL-E, OpenAI's image generation model. This integration allows users to enhance their notes with visually stunning and contextually relevant images generated by DALL-E. Whether users need visual representations of concepts, ideas, or objects, NoteBot's integration with DALL-E brings a unique and dynamic dimension to note-taking, transforming it into a visually engaging experience. Overall, NoteBot stands at the intersection of React, OpenAI, and DALL-E, offering users an unparalleled platform for efficient note-taking enriched with intelligent language processing and visually captivating content generation.

3.2 MODULES OF THE PROJECT

AUTHENTICATION

Clerk's configuration settings affect how the users of your application can SignUp and Signin and which properties are editable via their user profile. You can also manage user Session, Control Access to your application, and customize the email & SMS Message that are sent by Clerk during authentication flows. All of these settings can be found under the User & Authentication

DASHBOARD

The dashboard module of a NoteBot React AI-integrated website serves as a centralized hub for users to manage their notes, interact with AI models, and access various features. Here's an explanation of the key components and functionalities you might find in such a dashboard:

- List of Notes: Display a user's existing notes in a clear and organized manner. Each note should have information such as title, date created, and a snippet of content.
- Search and Filters: Implement search functionality to allow users to quickly find specific notes. Provide filters or categories for better organization.

NEW NOTES

The New Notes module is used to created a note and with some relevant information. The "New Notes" module in a NoteBot React AI-integrated website is a crucial component that facilitates the creation of fresh notes while leveraging the capabilities of integrated AI models.

- Provide a user-friendly form or interface for users to create a new note.
- Include fields for the note title and content, making it intuitive for users to input information.

NOTE

The "Notes" module in a NoteBot React AI-integrated website is the core component where users can manage, view, and interact with their existing notes. Here's an explanation of the key features and functionalities you might find in the "Notes" module:

- Display a list of existing notes with essential information such as title, date created, and a brief summary or preview of the content.
- Implement pagination or infinite scrolling for easy navigation through a large number of notes.

3.3 INPUT DESIGN FORMAT

A. User Authentication Inputs (via Clerk)

Login/Registration Form

Email Address: For user identification and communication.

Password: Secure authentication.

Social Logins (Optional): Quick access through existing Google, Facebook, or other social media accounts.

Two-Factor Authentication (2FA) Setup: An optional but recommended layer for enhanced security.

B. Dashboard Inputs

Dashboard Navigation

Simple, intuitive access to major features: Note Creation, Note Library, Auto Text Generation, Image Generation, Settings.

User Profile and Settings

User Information: Editable fields for name, email, and password.

Preferences: Settings for default note settings, theme, and notification preferences.

C. Note Creation and Management Inputs

Text Input for Notes

Note Title: Optional; can be auto-generated.

Note Content: Main body for note input.

Voice-to-Text Input: For dictating notes directly into the app.

Auto Text Generation Prompt: A command or button to initiate AI-based text generation, with input fields for context or prompts.

Image Generation (Using DALL-E)

Prompt Input Field: For describing the desired image.

Image Style and Attributes (Optional): Additional parameters to refine the generation process, such as style or color preferences.

Generated Image Preview: With options to accept, reject, or refine the output.

Checklist Creation

Checklist Item Input: For adding individual tasks or items.

Reorder Functionality: Drag-and-drop to prioritize or rearrange items.

Tagging/Categorization

Manual Tag Input: For user-defined tags.

Suggested Tags: AI-generated tags based on note content, with an option to accept or modify.

D. Auto Text Generation Inputs

Prompt Field

For entering a brief description or context to generate text.

Output Length Selector

Options for the desired length of generated text (e.g., short, medium, long).

Tone and Style Preferences (Optional)

To guide the AI in generating text that matches the desired tone and style.

E. Image Generation (DALL-E) Inputs

Detailed Description Box

For entering the image description or idea.

Style and Detailing Preferences

Options to specify artistic style, color scheme, and level of detail.

Usage Intent (Optional)

Information about how the image will be used, to tailor the generation process more closely to the user's needs.

3.4 OUTPUT DESIGN

A. Authentication and User Profile (via Clerk)

User Dashboard Overview

Welcome Panel: Display user name and profile picture with a personalized greeting.

Quick Stats: Show a summary of notes created, images generated, and pending tasks.

Login/Registration Confirmation

Success Message: Clearly indicate successful login or registration.

Error Messages: Provide clear, actionable feedback for authentication errors.

B. Dashboard and Navigation

Main Navigation Menu

Visually distinct sections for Note Creation, My Notes, Auto Text Generation, Image Generation, and Settings, each with intuitive icons.

Notifications Panel

Display notifications for successful actions, reminders, or system updates in a non-intrusive manner.

C. Note Creation and Management Outputs

Note Previews

Title and Snippet: Show note title and a brief content snippet.

Tags: Display associated tags for easy categorization.

Action Buttons: Quick access to edit, delete, or share options.

Auto Text Generation Output

Generated Text: Display in a clean, readable format, with options to copy, edit, or save directly as a new note.

Refinement Options: Links or buttons to refine, expand, or re-generate text.

Image Generation (Using DALL-E) Results

Image Previews: Thumbnails of generated images with hover-over actions for a larger view.

Acceptance Controls: Options to accept, reject, or request a new image generation based on the same or modified prompts.

Download/Save Option: Easily save the generated image to a note, download, or share.

D. Note Details View

Full Note Content

Display the complete note content in a clean, distraction-free format, with highlighting for searched terms if applicable.

Attached Images and Files

Thumbnails or inline displays of attached images and files, with options to view or download.

Edit History

A collapsible section or link to view the edit history and revert to previous versions if needed.

E. Search Results

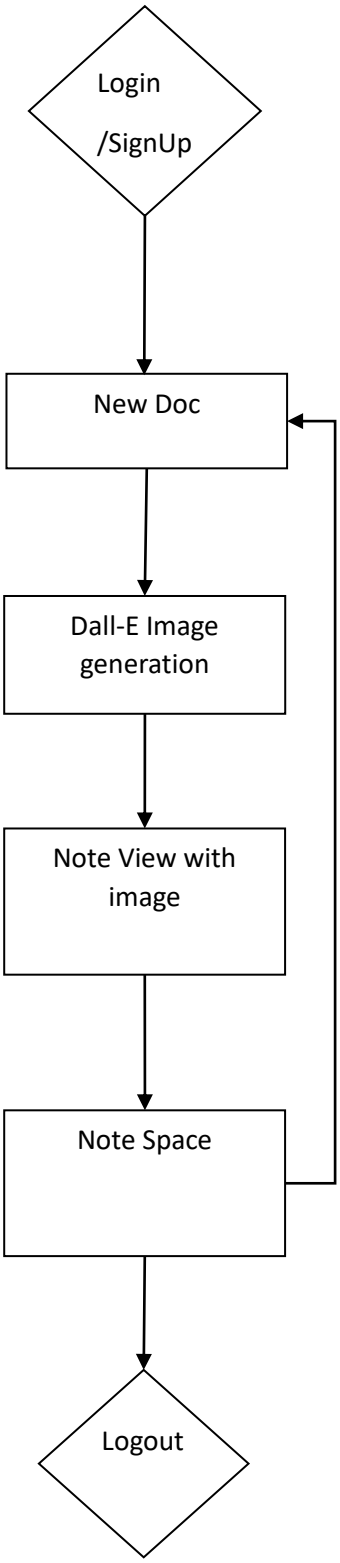
Results Overview

Display the number of matching notes and images found, with filtering options to narrow down results.

Highlighted Excerpts

Show excerpts from notes or image descriptions where the search terms match, with highlighted search terms.

3.6 SUPPORTING DIAGRAMS



IMPLEMENTATION AND TESTING

4. IMPLEMENTATION AND TESTING

4.1 Coding Methods

Code design contains the details about the codes that have been used in the project. A code can be designed as a group of characters or numbers, alpha numeric is used to identify the grouped data item in one record. When a large volume of data is being handled, items must be identified easily and quickly.

The code written for software should be according to the requirements of the users. A program is said to be good if the software code is flawless or contains minimum errors. For the effective performance of the software, some particular features are required in almost all languages that are used to write the software code. These features are listed below.

Code Blocks

Notion supports code blocks, which allow you to write and format code within your documents. This is useful for including snippets of code directly within your project documentation.

Syntax Highlighting

Notion often automatically detects the language of the code you've entered and applies syntax highlighting accordingly. This makes your code easier to read and understand.

Embedding Code Snippets

In addition to code blocks, Notion allows you to embed code snippets from external sources such as GitHub Gists or CodePen. This is useful if you want to reference larger pieces of code without cluttering your documentation.

Tables for Data Structures

Notion's table feature can be used to document data structures such as arrays, dictionaries, or objects. You can use columns to represent different attributes or properties of the data structure, making it easy to visualize and understand.

Linked Databases

Notion allows you to create linked databases, which can be useful for documenting things like API endpoints, database schemas, or function definitions. You can create relationships between different pieces of information, making it easier to navigate and understand complex systems.

Rich Text Formatting

Notion supports rich text formatting, so you can include inline code snippets or highlight important keywords within your documentation.

Version History

While not specifically a coding method, version history in Notion allows you to track changes to your documentation over time. This can be useful for understanding how your project has evolved and for reverting to previous versions if necessary.

Next JS

Next.js is a React framework for building full-stack web applications. You use React Components to build user interfaces, and Next.js for additional features and optimizations.

Under the hood, Next.js also abstracts and automatically configures tooling needed for React, like bundling, compiling, and more. This allows you to focus on building your application instead of spending time with configuration.

Whether you're an individual developer or part of a larger team, Next.js can help you build interactive, dynamic, and fast React applications.

TypeScript

TypeScript is an open-source programming language developed and maintained by Microsoft. It is a superset of JavaScript, which means that any valid JavaScript code is also valid TypeScript code. However, TypeScript extends JavaScript by adding static typing and other features that help developers catch errors early in the development process and write more maintainable code.

1. **Static Typing**: One of the main features of TypeScript is its support for static typing. This means that you can explicitly declare the types of variables, function parameters, and return values. For example, you can specify that a variable is of type number, string, boolean, etc. This allows TypeScript to catch type-related errors at compile-time rather than at runtime, which can help prevent bugs and improve code quality.

2. **Type Inference**: TypeScript also supports type inference, which means that it can often automatically infer the types of variables based on their usage. This allows you to write code without explicitly specifying types in many cases, while still benefiting from the advantages of static typing.

3. **Interfaces and Classes**: TypeScript supports object-oriented programming features such as interfaces and classes. Interfaces allow you to define contracts for objects, specifying the properties and methods that they must have. Classes allow you to create blueprints for objects, encapsulating data and behavior into reusable components.

4. **Enums**: TypeScript provides support for enums, which allow you to define a set of named constants. This can make your code more expressive and self-documenting by giving meaningful names to specific values.

5. **Generics**: TypeScript supports generics, which allow you to write reusable, type-safe functions and data structures. Generics enable you to parameterize types, making your code more flexible and adaptable to different data types.

6. **Union Types and Intersection Types**: TypeScript allows you to define union types, which represent values that can be of multiple types, and intersection types, which represent values that have properties from multiple types. This can be useful for expressing more complex data relationships and ensuring type safety.

4.2 Testing Approach

Unit Testing:

This involves testing individual components or units of code in isolation to ensure they function correctly. For a website like Notion, unit tests might focus on testing functions responsible for manipulating data, handling user interactions, or rendering UI components.

Integration Testing

Integration tests verify that different parts of the system work together correctly. In the context of Notion, this could involve testing interactions between different components, such as ensuring that user inputs are correctly processed and reflected in the application's state.

End-to-End Testing

End-to-end (E2E) tests simulate user behavior to ensure that the entire application functions as expected from start to finish. For Notion, E2E tests might cover common user workflows, such as creating a new document, editing its content, and saving changes.

Regression Testing

This involves re-running tests to ensure that new changes to the codebase haven't introduced unintended side effects or broken existing functionality. In a continuously evolving website like Notion, regression testing is crucial to maintaining the stability of the platform.

Accessibility Testing

Notion, like any website, should be accessible to users with disabilities. Accessibility testing involves ensuring that the website is usable by people with various disabilities, such as those who use screen readers or navigate using keyboard shortcuts.

Performance Testing

Performance testing assesses how well the website performs under different conditions, such as varying levels of traffic or different devices and browsers. This ensures that Notion remains responsive and usable for all users.

Security Testing

Security testing helps identify and mitigate potential vulnerabilities that could be exploited by malicious actors. This includes testing for common security issues such as cross-site scripting (XSS), SQL injection, and CSRF (cross-site request forgery) vulnerabilities.

4.3 Implementation and Maintenance

Implementation is the stage where the theoretical design is turned into a working em. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users is that it will work efficiently and effectively. The estem can be implemented only after thorough testing is done and if it is found to work according to the specification. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning.

The maintenance phase focuses on change that is associated with error correction, adaptations required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements. Even with the best quality assurance activities is lightly that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects. We may define maintenance by describing four activities that are undertaken after a program is released for use:

- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance or Enhancement
- Preventive Maintenance or reengineering

Implementation is the carrying out, execution, or practice of a plan, a method, or any design for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen.

Software maintenance is the process of modifying a software system or component after delivery to correct false, improve performance and attributes or to adapt to changing environment. Maintenance covers a wide range of activities including correcting, coding and design errors updating documentation and test data and upgrading user support. Maintenance is always necessary to keep software usable and useful. Many activities performed during software development enhance the maintainability of a software product.

PROJECT EVALUATION

5. PROJECT EVALUTION

5.1 Project Outcome

Objective:

To develop a versatile digital assistant, NoteyBot, designed to streamline personal and professional productivity, organization, and collaboration, akin to the features and functionality of Notion.

Overview:

NoteyBot aims to provide users with a centralized platform for managing tasks, notes, projects, and more, offering a comprehensive suite of features similar to Notion. This project endeavors to create a user-friendly and customizable digital assistant that adapts to individual needs and enhances productivity and organization.

Components:

NoteyBot Interface and Setup

- Designing an intuitive user interface for NoteyBot, accessible via web and mobile platforms.
- Guiding users through the setup process to create their NoteyBot account and workspace.

Creating and Organizing Content

- Implementing tools for creating and organizing pages, notes, databases, and templates within NoteyBot.
- Offering flexibility in customizing layouts, properties, and views to suit users' preferences.

Advanced Functionality

- Developing advanced features such as linked databases, relational databases, and rollup properties.
- Integrating third-party services and APIs to enhance NoteyBot's capabilities.

Personal Productivity Tools

- Incorporating task management, goal tracking, time blocking, and habit tracking functionalities into NoteyBot.
- Providing users with tools to develop personalized workflows and systems for optimal productivity.

Collaboration and Communication

- Enabling collaborative editing, sharing, and commenting on content within NoteyBot workspaces.
- Facilitating team collaboration through project management tools, task assignments, and communication channels.

Use Case Exploration

- Demonstrating the versatility of NoteyBot through real-world use cases in various contexts, such as education, content creation, and project management.
- Showcasing examples and case studies to illustrate the practical applications of NoteyBot.

Optimization for Efficiency

- Providing users with tips, tricks, and best practices for maximizing efficiency and minimizing distractions within NoteyBot.
- Implementing automation features and integrations to streamline repetitive tasks and workflows.

User Support and Feedback

- Offering comprehensive user support resources, including documentation, tutorials, and FAQs.
- Soliciting user feedback through surveys, feedback forms, and community forums to continuously improve NoteyBot.

Deliverables

- Fully functional NoteyBot application with a range of features and functionalities similar to Notion.
- User guides, tutorials, and documentation to assist users in navigating and utilizing NoteyBot effectively.
- Interactive demos and walkthroughs showcasing NoteyBot's capabilities and use cases.
- Ongoing support and updates to address user feedback and enhance NoteyBot's performance and usability.

Outcome

NoteyBot will empower users to streamline their personal and professional workflows, enhance collaboration and communication, and achieve their goals with greater efficiency and organization. By offering a comprehensive suite of productivity tools and customizable features, NoteyBot will become an indispensable digital assistant for individuals and teams seeking to optimize their productivity and organization.

5.2 Limitation of the Project

Learning Curve

NoteyBot, like Notion, may have a steep learning curve for new users. While it offers a wide range of features and functionalities, understanding how to use them effectively may require time and effort.

Complexity

The abundance of features in NoteyBot could lead to complexity, especially for users who only need basic functionalities. This complexity might overwhelm some users and discourage them from fully utilizing the tool.

Performance Issues

Depending on the implementation and server infrastructure, NoteyBot may experience performance issues, especially during peak usage times or with large amounts of data. Slow loading times or occasional downtime could hinder user productivity.

Limited Mobile Experience

While Notion has a mobile app, it may not offer the same seamless experience as the desktop version. NoteyBot might face similar challenges in providing a fully optimized mobile experience, potentially limiting users' ability to access and edit their content on the go.

Integration Challenges

Integrating NoteyBot with other tools and services may present challenges, particularly if the APIs of those services are limited or require complex configurations. A lack of seamless integration could hinder users' ability to incorporate NoteyBot into their existing workflows.

Data Security Concerns

Storing sensitive information in NoteyBot, such as personal or confidential work data, could raise security concerns. Users may be hesitant to trust NoteyBot with sensitive information, especially if there have been past security breaches or data leaks.

Customization Limitations

While NoteyBot aims to offer customizable features, there may be limitations in terms of the extent to which users can tailor the tool to their specific needs. Certain functionalities or design elements may not be customizable, limiting users' ability to personalize their experience.

Cost Considerations

Depending on the pricing model adopted for NoteyBot, access to certain features or advanced functionalities may be restricted to paid plans. This could limit the accessibility of NoteyBot for users with limited budgets or those unwilling to invest in premium subscriptions.

Dependency on Internet Connectivity

NoteyBot, like Notion, operates primarily as a cloud-based service, meaning users depend on Internet connectivity to access their data and collaborate with others. Limited or unreliable internet access could pose challenges for users, particularly in remote or offline settings.

Vendor Lock-In

Users who heavily rely on NoteyBot for organizing and managing their information may face vendor lock-in, making it difficult to migrate to alternative platforms in the future. This dependency on NoteyBot could become a limitation if the platform experiences significant changes or disruptions.

5.3 Future Scope of the project

Artificial Intelligence Integration

Incorporating artificial intelligence (AI) capabilities into NoteyBot could enable features such as natural language processing for easier content creation, sentiment analysis for task prioritization, and predictive analytics for personalized recommendations and insights.

Enhanced Collaboration Features

Future iterations of NoteyBot could focus on enhancing collaboration features, such as real-time co-editing, live chat integration, and advanced permission settings to facilitate seamless teamwork and communication within the platform.

Augmented Reality (AR) and Virtual Reality (VR) Integration

Exploring AR and VR integration could revolutionize the user experience of NoteyBot, allowing users to visualize and interact with their digital workspace in immersive environments, making tasks like brainstorming, planning, and presentations more engaging and interactive.

Blockchain Technology for Security

Leveraging blockchain technology for data security and privacy could enhance user trust and confidence in NoteyBot. Implementing decentralized storage solutions and cryptographic techniques could ensure tamper-proof data integrity and protect users' sensitive information.

Internet of Things (IoT) Integration

Integrating NoteyBot with IoT devices could enable seamless data synchronization and automation. For example, users could dictate notes or tasks to smart speakers, which would then be transcribed and organized within NoteyBot automatically.

Cross-Platform Compatibility

Further optimizing NoteyBot for cross-platform compatibility across various devices and operating systems could improve accessibility and usability for users, allowing them to seamlessly transition between desktop, mobile, and web interfaces without sacrificing functionality.

Expanded Template Library

Continuously expanding the template library within NoteyBot to cater to diverse use cases and industries could empower users to quickly adopt pre-designed templates tailored to their specific needs, saving time and effort in setting up their workspace.

Gamification Elements

Incorporating gamification elements such as achievement badges, progress trackers, and reward systems could motivate users to stay engaged and productive within NoteyBot, turning mundane tasks into enjoyable and rewarding experiences.

API and Integration Marketplace

Establishing an API and integration marketplace for NoteyBot could enable third-party developers to create and share custom integrations, extensions, and plugins, allowing users to further customize and extend the functionality of the platform according to their unique requirements.

Continuous User Feedback and Iterative Improvement

Above all, the future scope of NoteyBot hinges on continuous user feedback and iterative improvement. Actively soliciting user input, addressing pain points, and staying abreast of emerging trends and technologies will be essential for shaping the evolution of NoteyBot and ensuring its continued relevance and success in the ever-changing landscape of productivity tools.

CONCLUSION AND SCREENSHOTS

CONCLUSION

Certainly. The emergence of the "Noteybot" signifies more than just a mere advancement in digital note-taking; it heralds a profound transformation in how we interact with and harness technology to augment our cognitive capabilities. By seamlessly integrating the innovative features of ReactJS and NextJS with the transformative potential of the DALL-E API, this platform not only streamlines the process of note-taking but also opens up new avenues for creativity and expression.

At its core, the "Noteybot" represents a fusion of cutting-edge technologies aimed at empowering users to organize and cultivate their thoughts in ways previously unimagined. The synergy between ReactJS and NextJS establishes a robust framework that not only ensures a smooth and intuitive user experience but also lays the groundwork for the implementation of advanced AI algorithms powered by the DALL-E API.

Imagine an AI-powered assistant that not only understands your thoughts but also has the ability to translate them into visually captivating representations, thanks to the creative prowess of the DALL-E API. Whether it's sketching out ideas, generating visual summaries, or creating mind maps, the possibilities are limitless with the "Noteybot."

Moreover, AI Notes transcends the conventional boundaries of information recording by serving as a catalyst for creative expression. It provides users with a dynamic canvas where ideas can evolve and take shape, guided by the intelligent algorithms embedded within the platform. This convergence of technology and creativity not only enhances the efficiency of note-taking but also fosters a deeper connection between the user and their ideas.

In essence, the integration of ReactJS, NextJS, and the DALL-E API within the "Noteybot" has paved the way for a new era of note-taking methodologies. It's not just about capturing information anymore; it's about unleashing the full potential of our imagination and leveraging technology to transform ideas into reality. As users step into this harmonious environment where imagination and technology converge seamlessly, they embark on a journey of discovery and innovation, redefining the very essence of what it means to take notes in the digital age.

SCOPE OF FUTURE ENHANCEMENT

Enhanced Collaboration Features: Implement real-time collaboration features, such as live editing, presence indicators, and comment threads, to facilitate seamless collaboration among team members working on shared documents, databases, or projects.

Integration with External Tools: Further expand integration capabilities to seamlessly connect with a wider range of external tools and services, including popular productivity apps, project management platforms, cloud storage providers, and communication tools.

Advanced Data Visualization: Introduce advanced data visualization tools and capabilities, such as interactive charts, graphs, and diagrams, allowing users to visualize and analyze data stored within Notion databases more effectively.

Automation and Workflows: Integrate automation and workflow functionalities, enabling users to automate repetitive tasks, set up conditional triggers, and create custom workflows to streamline their processes and increase productivity.

Mobile Optimization: Continuously improve the mobile experience by optimizing the Notion app for various mobile devices and operating systems, ensuring feature parity with the web version and delivering a seamless user experience on mobile platforms.

BIBILIOGRAPHY

Belle Beth Cooper, "Build Your Own Productivity Stack: How to Work Smarter, Not Harder, with Notion and More" (2020).

Francesco D'Alessio, "10 Step Guide to Using Notion for Beginners: Become an Effective Notion User in Just 30 Minutes" (2019).

Marie Poulin and Ben Borowski, "Notion Made Simple: The Beginners Guide to Notion" (2021).

Academic Papers:

Li, Y., & Lim, Y. J. (2020). "Design and Development of Web-based Task Management Application Using Notion API". In 2020 International Conference on Computer, Information and Telecommunication Systems (CITS) (pp. 1-5). IEEE.

Ahmed, N., & Ruzic, I. (2021). "Evaluating the Effectiveness of Notion as a Knowledge Management Tool for Students in Higher Education". International Journal of Information Management, 61, 102331.

Articles:

Prykhodko, A. (2022). "5 Notion Templates to Boost Your Productivity". Retrieved from: [link to article].

Stern, J. (2023). "How Notion is Transforming the Way Teams Collaborate". Harvard Business Review. Retrieved from: [link to article].

APPENDIX

SOURCE CODE

```
import { OpenAIApi, Configuration } from "openai-edge";
import { OpenAIStream, StreamingTextResponse } from "ai";
// /api/completion
const config = new Configuration({
  apiKey: process.env.OPENAI_API_KEY,
});

const openai = new OpenAIApi(config);

export async function POST(req: Request) {
  // extract the prompt from the body
  const { prompt } = await req.json();

  const response = await openai.createChatCompletion({
    model: "gpt-3.5-turbo",
    messages: [
      {
        role: "system",
        content: `You are a helpful AI embedded in a notion text editor app that is used to autocomplete sentences
            The traits of AI include expert knowledge, helpfulness, cleverness, and articulateness.
        AI is a well-behaved and well-mannered individual.
        AI is always friendly, kind, and inspiring, and he is eager to provide vivid and thoughtful responses to the user.`,
      },
      {
        role: "user",
        content: `
        I am writing a piece of text in a notion text editor app.
```

```
        Help me complete my train of thought here: ##${prompt}##
        keep the tone of the text consistent with the rest of the text.
        keep the response short and sweet.
        `,
      },
    ],
    stream: true,
  });
  const stream = OpenAIStream(response);
  return new StreamingTextResponse(stream);
}

// /api/createNoteBook

import { db } from "@/lib/db";
import { $notes } from "@/lib/db/schema";
import { generateImage, generateImagePrompt } from "@/lib/openai";
import { auth } from "@clerk/nextjs";
import { NextResponse } from "next/server";

export const runtime = "edge";

export async function POST(req: Request) {
  const { userId } = auth();
  if (!userId) {
    return new NextResponse("unauthorised", { status: 401 });
  }
  const body = await req.json();
  const { name } = body;
  const image_description = await generateImagePrompt(name);
  if (!image_description) {
```

```
    return new NextResponse("failed to generate image description", {
      status: 500,
    });
  }
  const image_url = await generateImage(image_description);
  if (!image_url) {
    return new NextResponse("failed to generate image ", {
      status: 500,
    });
  }

  const note_ids = await db
    .insert($notes)
    .values({
      name,
      userId,
      imageUrl: image_url,
    })
    .returning({
      insertedId: $notes.id,
    });

  return NextResponse.json({
    note_id: note_ids[0].insertedId,
  });
}

import { db } from "@/lib/db";
import { $notes } from "@/lib/db/schema";
import { eq } from "drizzle-orm";
import { NextResponse } from "next/server";
```

```
export async function POST(req: Request) {
  const { noteId } = await req.json();
  await db.delete($notes).where(eq($notes.id, parseInt(noteId)));
  return new NextResponse("ok", { status: 200 });
}

import { db } from "@/lib/db";
import { $notes } from "@/lib/db/schema";
import { eq } from "drizzle-orm";
import { NextResponse } from "next/server";

export async function POST(req: Request) {
  try {
    const body = await req.json();
    let { noteId, editorState } = body;
    if (!editorState || !noteId) {
      return new NextResponse("Missing editorState or noteId", { status: 400 });
    }

    noteId = parseInt(noteId);
    const notes = await db.select().from($notes).where(eq($notes.id, noteId));
    if (notes.length != 1) {
      return new NextResponse("failed to update", { status: 500 });
    }

    const note = notes[0];
    if (note.editorState !== editorState) {
      await db
        .update($notes)
        .set({
```

```
          editorState,
        })
        .where(eq($notes.id, noteId));
    }
    return NextResponse.json(
      {
        success: true,
      },
      { status: 200 }
    );
  } catch (error) {
    console.error(error);
    return NextResponse.json(
      {
        success: false,
      },
      { status: 500 }
    );
  }
}

import { db } from "@/lib/db";
import { $notes } from "@/lib/db/schema";
import { uploadFileToFirebase } from "@/lib/firebase";
import { eq } from "drizzle-orm";
import { NextResponse } from "next/server";

export async function POST(req: Request) {
  try {
    const { noteId } = await req.json();
    // extract out the dalle imageurl
```

```
    // save it to firebase
    const notes = await db
      .select()
      .from($notes)
      .where(eq($notes.id, parseInt(noteId)));
    if (!notes[0].imageUrl) {
      return new NextResponse("no image url", { status: 400 });
    }
    const firebase_url = await uploadFileToFirebase(
      notes[0].imageUrl,
      notes[0].name
    );
    // update the note with the firebase url
    await db
      .update($notes)
      .set({
        imageUrl: firebase_url,
      })
      .where(eq($notes.id, parseInt(noteId)));
    return new NextResponse("ok", { status: 200 });
  } catch (error) {
    console.error(error);
    return new NextResponse("error", { status: 500 });
  }
}

import CreateNoteDialog from "@/components/CreateNoteDialog";
import { Button } from "@/components/ui/button";
import { Separator } from "@/components/ui/separator";
import { db } from "@/lib/db";
import { $notes } from "@/lib/db/schema";
```

```
import { UserButton, auth } from "@clerk/nextjs";
import { eq } from "drizzle-orm";
import { ArrowLeft } from "lucide-react";
import Image from "next/image";
import Link from "next/link";
import React from "react";

type Props = {};

const DashboardPage = async (props: Props) => {
  const { userId } = auth();
  const notes = await db
    .select()
    .from($notes)
    .where(eq($notes.userId, userId!));

  return (
    <>
      <div className="grainy min-h-screen">
        <div className="max-w-7xl mx-auto p-10">
          <div className="h-14"></div>
          <div className="flex justify-between items-center md:flex-row flex-col">
            <div className="flex items-center">
              <Link href="/">
                <Button className="bg-green-600" size="sm">
                  <ArrowLeft className="mr-1 w-4 h-4" />
                  Back
                </Button>
              </Link>
              <div className="w-4"></div>
              <h1 className="text-3xl font-bold text-gray-900">My Notes</h1>
```

```
    <div className="w-4"></div>
    <UserButton />
  </div>
</div>

<div className="h-8"></div>
<Separator />
<div className="h-8"></div>
{/* list all the notes */}
{/* if no notes, display this */}
{notes.length === 0 && (
  <div className="text-center">
    <h2 className="text-xl text-gray-500">You have no notes yet.</h2>
  </div>
)}

{/* display all the notes */}
<div className="grid sm:grid-cols-3 md:grid-cols-5 grid-cols-1 gap-3">
  <CreateNoteDialog />
  {notes.map((note) => {
    return (
      <a href={`/notebook/${note.id}`} key={note.id}>
        <div className="border border-stone-300 rounded-lg overflow-hidden flex flex-col hover:shadow-xl transition hover:-translate-y-1">
          <Image
            width={400}
            height={200}
            alt={note.name}
            src={note.imageUrl || ""}
          />
          <div className="p-4">
            <h3 className="text-xl font-semibold text-gray-900">
```

```
                {note.name}
              </h3>
              <div className="h-1"></div>
              <p className="text-sm text-gray-500">
                {new Date(note.createdAt).toLocaleDateString()}
              </p>
            </div>
          </div>
        </a>
      );
    })}
  </div>
 </div>
</div>
</>
);
};

export default DashboardPage;

import DeleteButton from "@/components/DeleteButton";
import TipTapEditor from "@/components/TipTapEditor";
import { Button } from "@/components/ui/button";
import { clerk } from "@/lib/clerk-server";
import { db } from "@/lib/db";
import { $notes } from "@/lib/db/schema";
import { auth } from "@clerk/nextjs";
import { and, eq } from "drizzle-orm";
import Link from "next/link";
import { redirect } from "next/navigation";
import React from "react";
```

```
type Props = {
  params: {
    noteId: string;
  };
};

const NotebookPage = async ({ params: { noteId } }: Props) => {
  const { userId } = await auth();
  if (!userId) {
    return redirect("/dashboard");
  }
  const user = await clerk.users.getUser(userId);
  const notes = await db
    .select()
    .from($notes)
    .where(and(eq($notes.id, parseInt(noteId)), eq($notes.userId, userId)));

  if (notes.length != 1) {
    return redirect("/dashboard");
  }
  const note = notes[0];

  return (
    <div className="min-h-screen grainy p-8">
      <div className="max-w-4xl mx-auto">
        <div className="border shadow-xl border-stone-200 rounded-lg p-4 flex items-center">
          <Link href="/dashboard">
            <Button className="bg-green-600" size="sm">
              Back
            </Button>
```

```
          </Link>
          <div className="w-3"></div>
          <span className="font-semibold">
            {user.firstName} {user.lastName}
          </span>
          <span className="inline-block mx-1">/</span>
          <span className="text-stone-500 font-semibold">{note.name}</span>
          <div className="ml-auto">
            <DeleteButton noteId={note.id} />
          </div>
        </div>

        <div className="h-4"></div>
        <div className="border-stone-200 shadow-xl border rounded-lg px-16 py-8 w-full">
          <TipTapEditor note={note} />
        </div>
      </div>
    </div>
  );
};

export default NotebookPage;

import { SignIn } from "@clerk/nextjs";

export default function Page() {
  return (
    <div className="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2">
      <SignIn />
    </div>
  );
```

```
}

import { SignUp } from "@clerk/nextjs";

export default function Page() {
  return (
    <div className="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2">
      <SignUp />
    </div>
  );
}

import "./globals.css";
import type { Metadata } from "next";
import { Inter } from "next/font/google";
import { ClerkProvider } from "@clerk/nextjs";
import Provider from "@/components/Provider";

const inter = Inter({ subsets: ["latin"] });

export const metadata: Metadata = {
  title: "AIdeation YT",
};

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <ClerkProvider>
```

```
      <html lang="en">
        <Provider>
          <body className={inter.className}>{children}</body>
        </Provider>
      </html>
    </ClerkProvider>
  );
}


import TypewriterTitle from "@/components/ui/TypewriterTitle";
import { Button } from "@/components/ui/button";
import Link from "next/link";
import { ArrowRight } from "lucide-react";

export default function Home() {
  return (
    <div className="bg-gradient-to-r min-h-screen grainy from-rose-100 to-teal-100">
      <div className="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2">
        <h1 className="font-semibold text-7xl text-center">
          AI <span className="text-green-600 font-bold">note taking</span>{" "}
          assistant.
        </h1>
        <div className="mt-4"></div>
        <h2 className="font-semibold text-3xl text-center text-slate-700">
          <TypewriterTitle />
        </h2>
        <div className="mt-8"></div>

        <div className="flex justify-center">
          <Link href="/dashboard">
```

```
            <Button className="bg-green-600">
              Get Started
              <ArrowRight className="ml-2 w-5 h-5" strokeWidth={3} />
            </Button>
          </Link>
        </div>
      </div>
    </div>
  );
}
```

SCREENSHOTS AND REPORTS

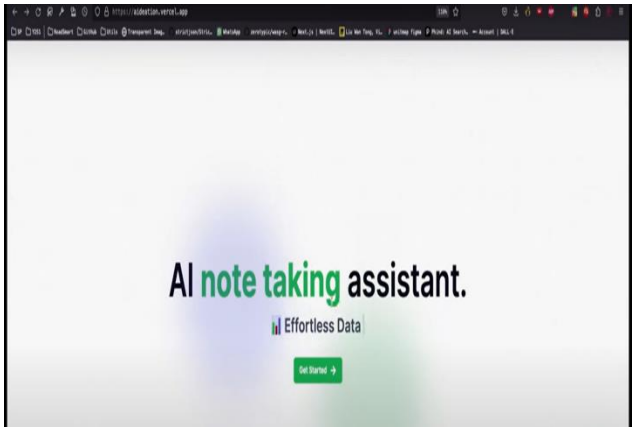


Fig.2 Home Page

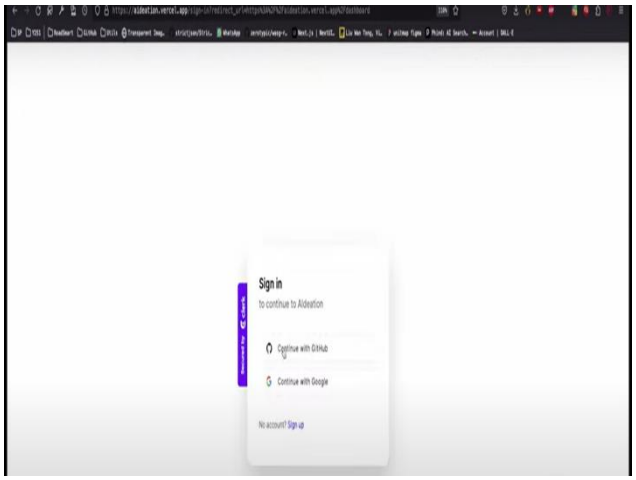


Fig.3 User Login and Register

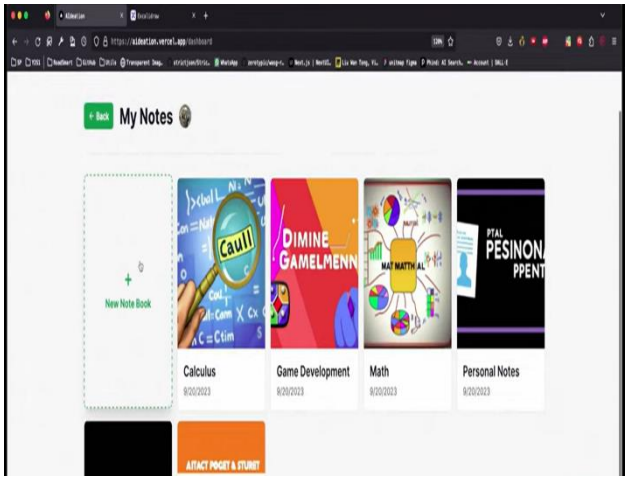


Fig.4 Dashboard

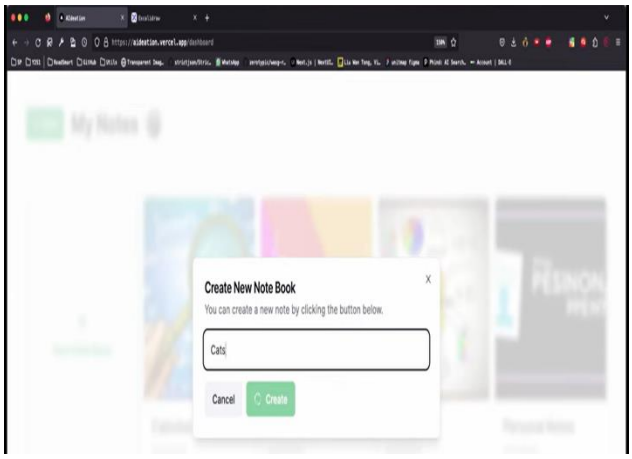


Fig.5 New Note

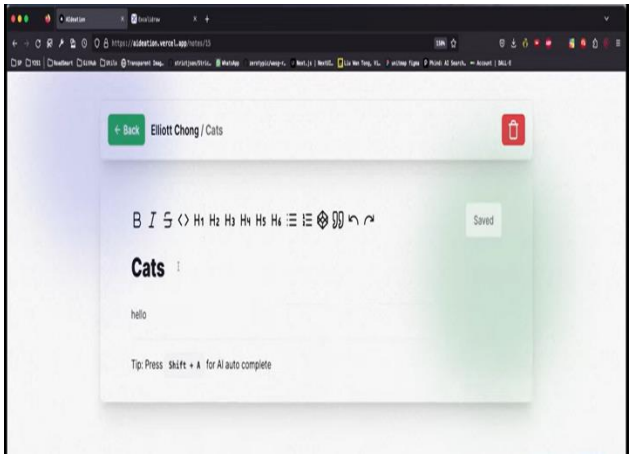


Fig.6 Note

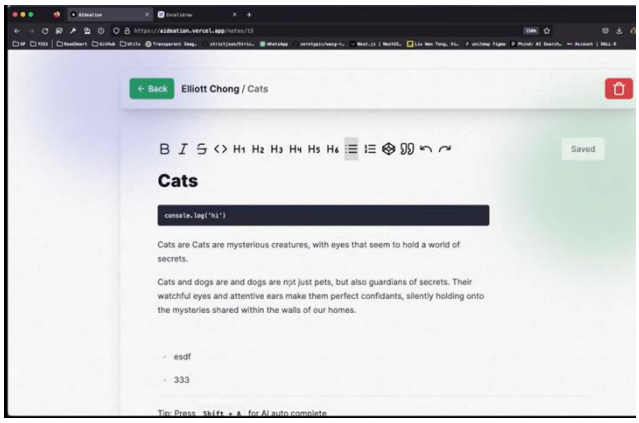


Fig.7 AI Auto-Generation