Note-AI

Team 1: Jason Mihalopoulos, Larue Linder, Justin Bravo, Taran Agarwal, Griffin Montalvo

Opening Remarks

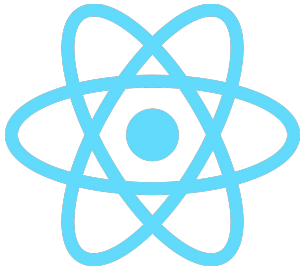
- Problem
  - Traditional note-taking is limited
  - Long, unstructured notes are hard to study
- Purpose
  - Enhance note-taking and studying experience by combining document editing with AI-powered features

Live Demo - App Features

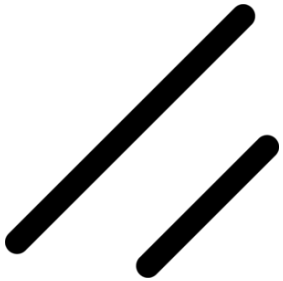
1. Autocomplete Generation (Copilot)
2. AI Chat with Notes
3. Mock Test Generator
4. Concept Map Generator
5. AI menu features (improve writing, longer, shorter, summarize, spelling and grammar)
6. Organize notes
7. Dashboard chat with all notes

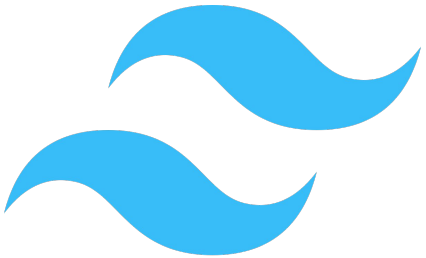
Technical Architecture

Frontend



— Plate





Backend

AI Integration

- Used OpenAI exclusively to integrate AI features into our app
- AI features include:
  - Organizing notes
  - Context map generation
  - AI menu (improve writing, longer, shorter, summarize, spelling and grammar)
  - Quiz generation
  - In-text autocomplete
  - AI chat in notes
  - Dashboard chat with all notes

Concept Map

- React Flow for interactive graph visualization
- Extract key concepts and relationships through prompting
- Data Flow
  - Note content → OpenAI API → Structured concept map → Database → UI rendering
  - stored as json objects
- Node positioning logic based on Canvas dimension 1200px x 600px
- Interactive node/edge editing with custom node and edge type

AI Menu

- Plate API - gets information about the editor state
  - cursor position
  - selected block
- Current block sent as context to the backend
  - selected block
  - cursor position
- OpenAI API
  - context
  - custom prompt based on user selection
- Frontend
  - plate ui components
- Custom hook orchestrates AI communication

Ask AI (Dashboard)

- Temporary chat modal-based interface on the dashboard
- Used to interact with and ask questions with all of the notes as context
- Ask AI Flow:
  - Uses Convex’s similarity search to find relevant embeddings to the prompt
  - Fetches relevant embedding’ notes to use as context
  - Uses GPT-4o to generate responses and outputs response as well as clickable note sources
  - Maintains separate chat contexts for individual notes and global queries

Embeddings

- Chunking Strategy
  - Chunk size of 500 characters with 100-character overlap between chunks to maintain context
  - Intelligent chunk boundaries at sentence endings or paragraph breaks
- Processing Pipeline
  - Processed when navigating away from a note (background processing)
  - Uses OpenAI's text-embedding-3-small model for vector generation
- Cleanup and Maintenance
  - Automatic deletion of embeddings when notes are deleted
  - Background processing to avoid UI blocking

In-text Autocomplete

- Uses Plate’s copilot plugin as reference
- Data Flow: User Input (Ctrl+Space) → Text Extraction → Convex Backend → OpenAI API → Ghost Text Generation → User Acceptance (Tab) → Text Insertion
- Uses note up to the point of the cursor as context for autocomplete processing
- State management to determine when the ghost text should be displayed or not

Test Generator

- Convex generateTest action:
  - OpenAI API
  - System prompt takes note content, question types, # of questions and difficulty level
  - Returned test is structured JSON object - question, options, answer, source
  - Source Tracking: take AI source and uses custom plugin to highlight in notes
- gradeShortAnswer action
- Database Schema
  - Store generated tests in tests table
  - Include metadata, questions, settings

```
{
  answer: "Privacy",
  options: [
    "Global warming",
    "Privacy",
    "Deforestation",
    "Space exploration",
  ],
  question:
    "According to the notes, what is a key issue in computer ethics?",
  source:
    "Privacy: The ability of computers to invade personal information through
    non-consensual surveillance and data collection.",
  type: "mcq",
},
```

Custom Highlight Plugin

- Originally created for search bar, applied to test generator and concept map generator source tracking
- DOM-based text processing
- Algorithm
  - Clear old highlights
  - Make a copy of the DOM container
  - Recursive node traversal to find matches
  - Swaps actual HTML with updated highlighted copy
- Search-bar
  - debouncing (300ms) before searching
  - Case-sensitive and case-insensitive search

Organize Notes

- Prompt Engineering:
  - A sophisticated system prompt guides gpt-4o.
  - Key Instructions to AI:
    - Identify logical sections and create clear headings (e.g., <h1>, <h2>).
    - Convert dense paragraphs into scannable bullet points and sub-bullets.
  - Critical Constraint: Preserve *all* original content, formatting (bold, italics, font sizes, colors), and unique element IDs. The AI must return a structurally identical JSON array, just reorganized and augmented.
  - Strict JSON output format is enforced.

Closing Remarks

Limitations

- No real-time collaborative editing
- No browser context integration
- No speech-to-text capabilities
- Organize note feature will likely not work on a note that exceeds 3000 words

-

Future Work

- Continue working on our app to resemble the framework of Google Docs
  - Real-time collaboration between users
  - Ability to create folders to put our notes in
  - Speech to text
  - Upload images/tables
  - Version history
- Agent Mode
  - give instructions to an AI model about additions to make to the notes and it makes edits on the user's document
- AI generated flashcards

Project Takeaways

- Dealing with Plate editor is quite challenging
  - Some plugins have to be customly made since Plate is not completely open-source
- Merge features earlier in the sprint instead of waiting to merge at the end
- Too many LLM calls can slow down performance and isn’t scalable
  - Process slow LLM calls in the background