📝🚀 Coursepilot

4.0 Software Development Plan

4.0.1 Introduction:

With innovative tools and frameworks emerging surrounding LLMs we wanted to apply some new techniques such as RAG (Retrieval Augmented Generation) to provide value to students by making learning more effective and engaging. Our idea is to embed the semantic meaning of documents, which could be student-taken notes or slides uploaded by a professor, store the documents in a vector database, and use them to provide greater context to generate a better output when the LLM is used. Coursepilot's backend will be the RAG engine, that will parse documents, embed their semantic meaning, and store each document in a MongoDB Atlas database with an id, the text, and its vector embedding. This vector embedding allows us to retrieve relevant documents when querying later. We will also be making our Gemini API calls from the backend and connecting it to the frontend with Flask. The frontend will be mainly built with Next.js, and will handle user authentication with Clerk.

4.1 Project Deliverables

4.1.1 Requirements Overview

- Accelerate note taking using AI
- Provide study tools and jumpstart presentations/projects
- Help teach, without avoiding learning

4.1.2 Frontend Requirements

4.1.2.1 Notebook

- Oct 16, 2024 2.1.1 Classlist - a list view for organization displaying all classes added by the user
    - Emoji picker - an emoji picker to select icons for different classes
    - Title - a customizable title input for class name
    - Backend - selection updates note in storage
- Oct 23, 2024 2.1.2 Notelist - a list view for organization displaying all notes within the selected class
    - Emoji picker - an emoji picker to select icons for different notes
    - Title - a customizable title input for the note
    - Backend - selection updates note in storage
- Nov 13, 2024 2.1.3 Text Editor - a rich text editor for the selected note
    - Oct 16, 2024 Menubar - a menu of buttons to control text options
        - Connection to the text area and its selections
        - Bold - button to bold text
        - Italics - button to italicize text
        - Lists - button to create a numbered/bulleted list
        - Size - button to increase/decrease font size
        - Color - a picker for font color
    - Nov 13, 2024 Editor - the main text area
        - Connection to the menu bar and its selections
        - Formatting - markdown support for headings, lists, code blocks
        - AI - a debounced autocomplete
            - Preview - a gray text preview of autocompleted text that can be filled using tab
            - API - request made to Gemini using the context of the user's notes
        - Backend - modification updates note in storage

- Nov 27, 2024 2.1.4 Quizzes - a list view for generated flashcards from notes
    - Flashcard - a front/back Q&A
        - Front - a question to be answered by the back of the card
        - Back - the answer to the question on the front of the card

4.1.2.2 Authentication

- Oct 9, 2024 2.2.1 Supports sign up with email, Google, and optionally Microsoft
- Nov 27, 2024 2.2.2 Clerk - Library that provides authentication handling and component library
    - Sign up page - a page/component form for making sign in requests for new users
    - Sign in page - a page/component form for making account creation requests for existing users
    - Sign in button - a component for sign in/sign up page redirecting
    - Sign out button - a component for user sign out handling on the frontend
    - UserID - token used for database organization to store classes and notes per user
    - Access gating - rendering of the app can be conditional based on if a user is authenticated

4.1.3 Backend Requirements

4.1.3.1 RAG Engine

- Oct 9, 2024 3.1.1 Document chunker - currently implemented as a sentence based chunker but can be extended to perform semantic chunking
- Oct 16, 2024 3.1.2 Document embedder - utilizing the BERT (Bidirectional Encoder Representation for Transformers) language model to capture semantic meaning of text chunks in vector representations
- Oct 23, 2024 3.1.2 MongoDB Atlas Vector database - allows for storage of text data along with its vector representation, which allows for effective querying and retrieval using the KNN algorithm

4.1.3.2 Data Pipeline

- Oct 30, 2024 3.2.1 Flask server hosting an API will create a data pipeline from the RAG engine to the frontend
    - “/autocomplete”
        - Takes the context of the current note and returns the suggested autocomplete

- “/getContext”
    - (optional) Takes the current note and provides metadata for tooltips for related documents previously uploaded

4.2 Project Resources

4.2.1 Hardware Resources

Resource	**Dev**	**Execution**
Windows PC	X	X
AMD Ryzen 5 5600 3.8GHz 6 core	X	
32gb DDR5 RAM, Nvidia 3070 8gb VRAM	X	
Macbook Pro	X	X
Apple M1 Max	X	
32gb RAM	X	

4.2.2 Software Resources

Resource	**Dev**	**Execution**
Next.js	X	X
Flask	X	X
MongoDB	X	X
Heroku		X
Visual Studio Code	X	
Cursor	X	
Clerk Auth	X	X
Google Gemini LLM	X	X

TipTap	X	X

4.3 Project Organization

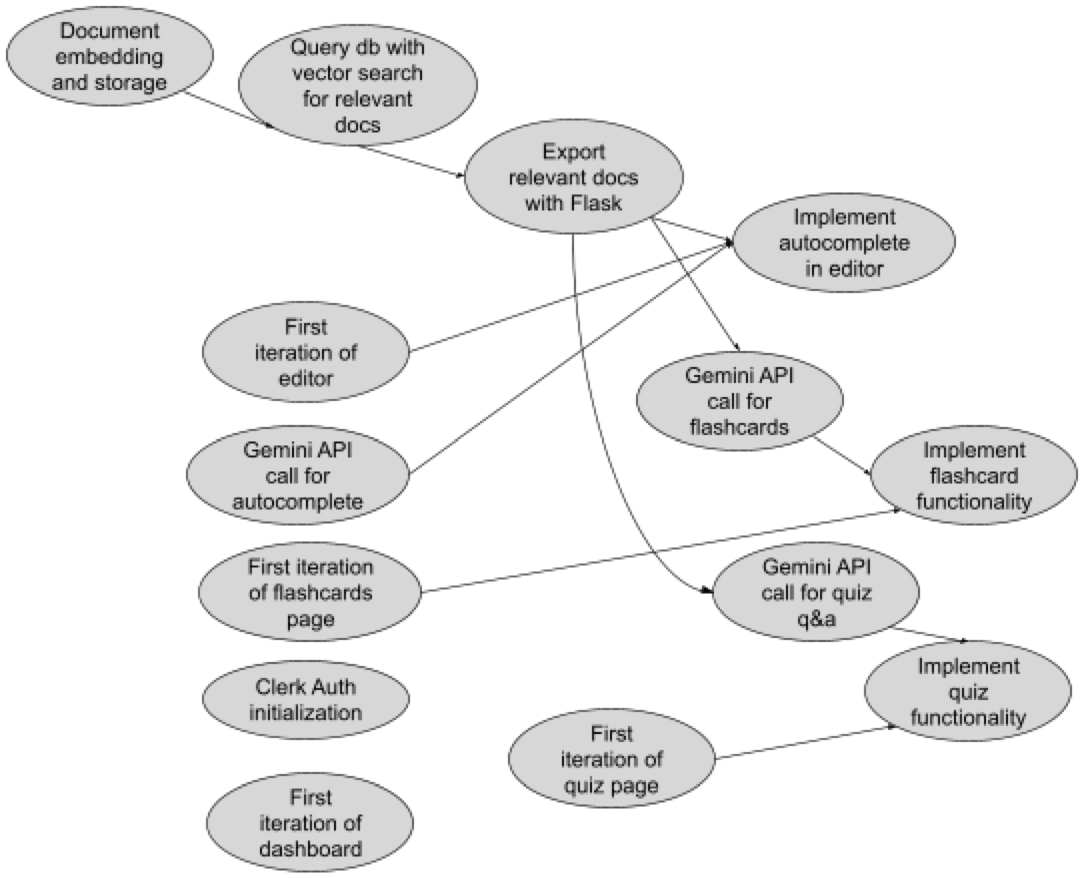
Davis Banks - Lead Engineer for Editor and Autocomplete
Lucian Prinz - Lead Engineer for Sidebar, Note Storage, and Authentication
David Yang - Lead Engineer for Retrieval Augmented Generation and Document Storage

4.4 Schedule

Week	Status	Assignee	Task
4	Launched	Lucian Prinz	Initialize Next.JS frontend
4	Launched	Davis Banks	Setup Clerk authentication
4	Launched	David Yang	Initialize MongoDB
5	Launched	Davis Banks	Initialize text editor
5	Launched	David Yang	Set up document chunking and embedding with BERT
5	Launched	David Yang	Implement document retrieval with kNN vector comparison
6	Launched	Davis Banks	Design autocomplete extension
6	Launched	Lucian Prinz	Implement sidebar
6	In progress	David Yang	Parse pdfs with LlamaParse with output .md
6	In progress	Lucian Prinz	Setup Note storage in MongoDB
7	In progress	Davis Banks	Modify autocomplete loading states/debounce times
7	In progress	David Yang	Chunk .md with semantic topic groupings
8	Not started	Davis Banks	Add code block, equation support to text editor

Week	Status	Assignee	Task
8	Not started	David Yang	Test autocomplete with retrieved docs and partially written text
9	Not started	Davis Banks	Design and implement quizzes/flashcards
9	Not started	Davis Banks	Connect text editor to backend API pipeline
9	Not started	David Yang	Compare and performance of Gemini without context vs context-aware prompt
9	Not started	David Yang	Generate flashcards and quiz questions with context
10	Not started	Davis Banks	Test autocomplete preview formatting
10	Not started	David Yang	Connect quiz and flashcard data to Flask
11	Not started	Lucian Prinz	Connect new sidebar with notes backend
11	Not started	All	Ship MVP (goal)
12	Not started	Davis Banks	Test/tweak editor UX
12	Not started	Lucian Prinz	Test/tweak editor/sidebar UX
12	Not started	David Yang	Test/tweak prompts and vector embedding queries
13	Not started	Lucian Prinz	Stripe billing
13	Not started	Lucian Prinz	Landing page design
14	Not started	All	Design landing page
15	Not started	All	Presentation

4.4.1 PERT Chart


Document embedding and storage
Query db with vector search for relevant docs
Export relevant docs with Flask
Implement autocomplete in editor
First iteration of editor
Gemini API call for flashcards
Gemini API call for autocomplete
Implement flashcard functionality
First iteration of flashcards page
Gemini API call for quiz q&a
Clerk Auth initialization
Implement quiz functionality
First iteration of quiz page
First iteration of dashboard

4.4.2 Task/Resource Table

ID	Task	Dependencies	Resources
001	Initialize Next.JS frontend	N/A	Next.js
002	Setup Clerk authentication	Task 001	Clerk
003	Initialize MongoDB	N/A	MongoDB Atlas
004	Initialize text editor	Task 001, 002	Next.js, Shadcn ui, TipTap
005	Set up document chunking and embedding with BERT	N/A	Nvidia 3070, Huggingface BERT Model

006	Implement document retrieval with kNN vector comparison	Task 005	MongoDB Atlas Vector Search
007	Design autocomplete extension	N/A	TipTap
008	Implement sidebar	Task 001, 002, 003	Next.js, Tailwind css, Shadcn ui
009	Parse pdfs with LlamaParse with output .md	N/A	LlamaParse
010	Setup Note storage in MongoDB	Task 003	MongoDB
011	Modify autocomplete loading states/debounce times	Task 007	NextJS Debouncing
012	Chunk .md with semantic topic groupings	Task 005	
013	Add code block, equation support to text editor	Task 004	NextJS, Tiptap, Shadcn, Tailwind
014	Test autocomplete with retrieved docs and partially written text	Task 005, 006, 009	Gemini, Flask, MongoDB Atlas Vector search
015	Design and implement quizzes/flashcards	Task 010	Gemini, Flask, MongoDB Atlas Vector search
016	Connect text editor to backend API pipeline	Task 004, 007, 014	Flask, Next.js serverside backend
017	Compare and performance of Gemini without context vs context-aware prompt	Task 005, 006, 014, 015	Gemini, Flask, MongoDB Atlas vector search
018	Generate flashcards and quiz questions with context	Task 010, 012	Gemini, NextJS, Flask Backend API
019	Test autocomplete preview formatting	Task 004, 0016	
020	Connect new sidebar with notes backend	Task 016	NextJS, Flask
021	Connect Stripe	All of the above	Stripe
022	Ship MVP	All of the above	

023	Test/tweak editor UX	All of the above	
024	Test/tweak editor/sidebar UX	All of the above	
025	Test/tweak prompts and vector embedding queries	Task 005, 006, 014, 017	Gemini, BERT, MongoDB Atlas Vector search
026	Landing page design	All of the above	Figma, Next.js, Shadcn ui