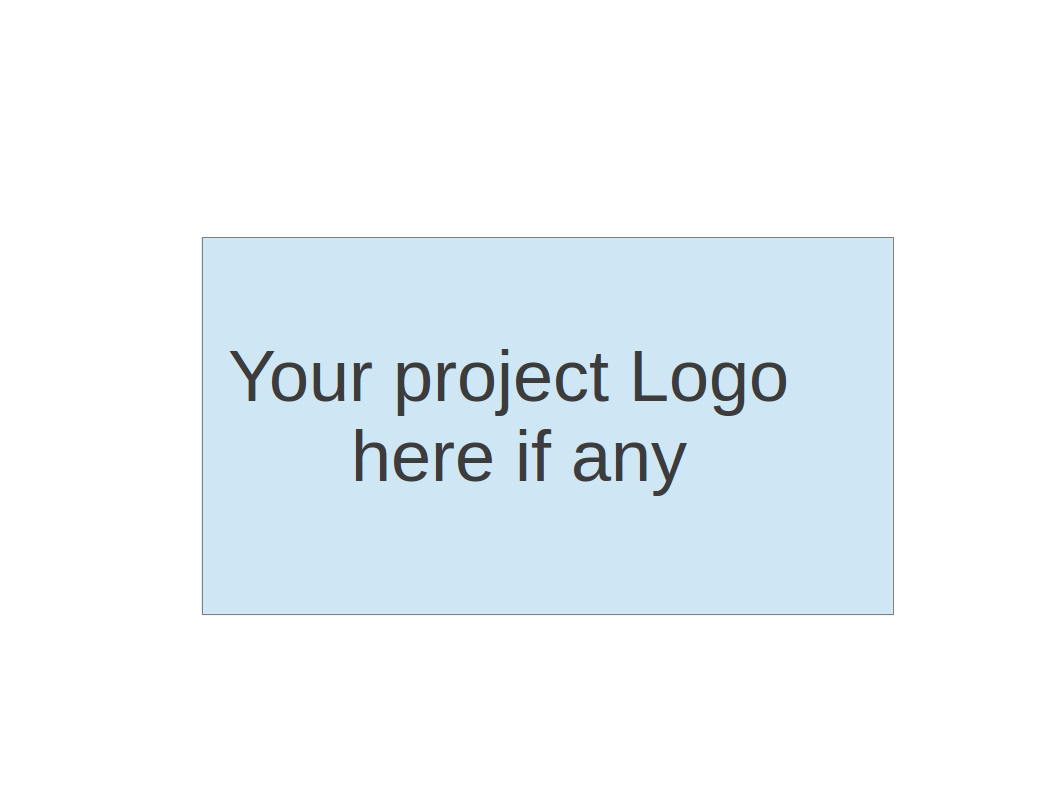
**CS673 Software Engineering** 

**Team5 - NoteAnt**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Frankie | Requirement leader | *Guancheng Huang* | Oct 19 |
| Chris | Design and Implementation leader | Wenhao Tian (Chris) | Oct 19 |
| Nicholas | Team Leader | *Nicholas Narmada* | Oct 19 |
| Wayne | Configuration leader | *Yichen Li* | Oct 19 |
| Siyuan Wan | Security Leader | *Siyuan Wan* | Oct 19 |
| Yibo(Wilbur) | QA Leader | *Yibo Wang* | Oct 19 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **iter2** | **Siyuan Wan，**  **Chris，**  **Wayne** | **19 Oct, 2023** | **First** |
| **Iter3** |  | **9 Dec, 2023** | **Second** |

# Introduction

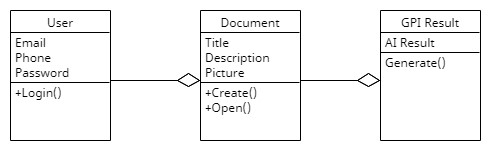
We all take notes at some point, either in class or in a meeting. One of the popular methods of note taking is typing down your thoughts while listening. A lot of times the notes we took can be hard to read or confusing.

To solve this problem, our project aims to develop an AI assisted note taking application that combines text editing and AI to provide users with a real-time, interactive content generation and editing experience. The application is designed to facilitate user interaction, data processing and integration with the OpenAI API.

# Software Architecture Client-Server Model

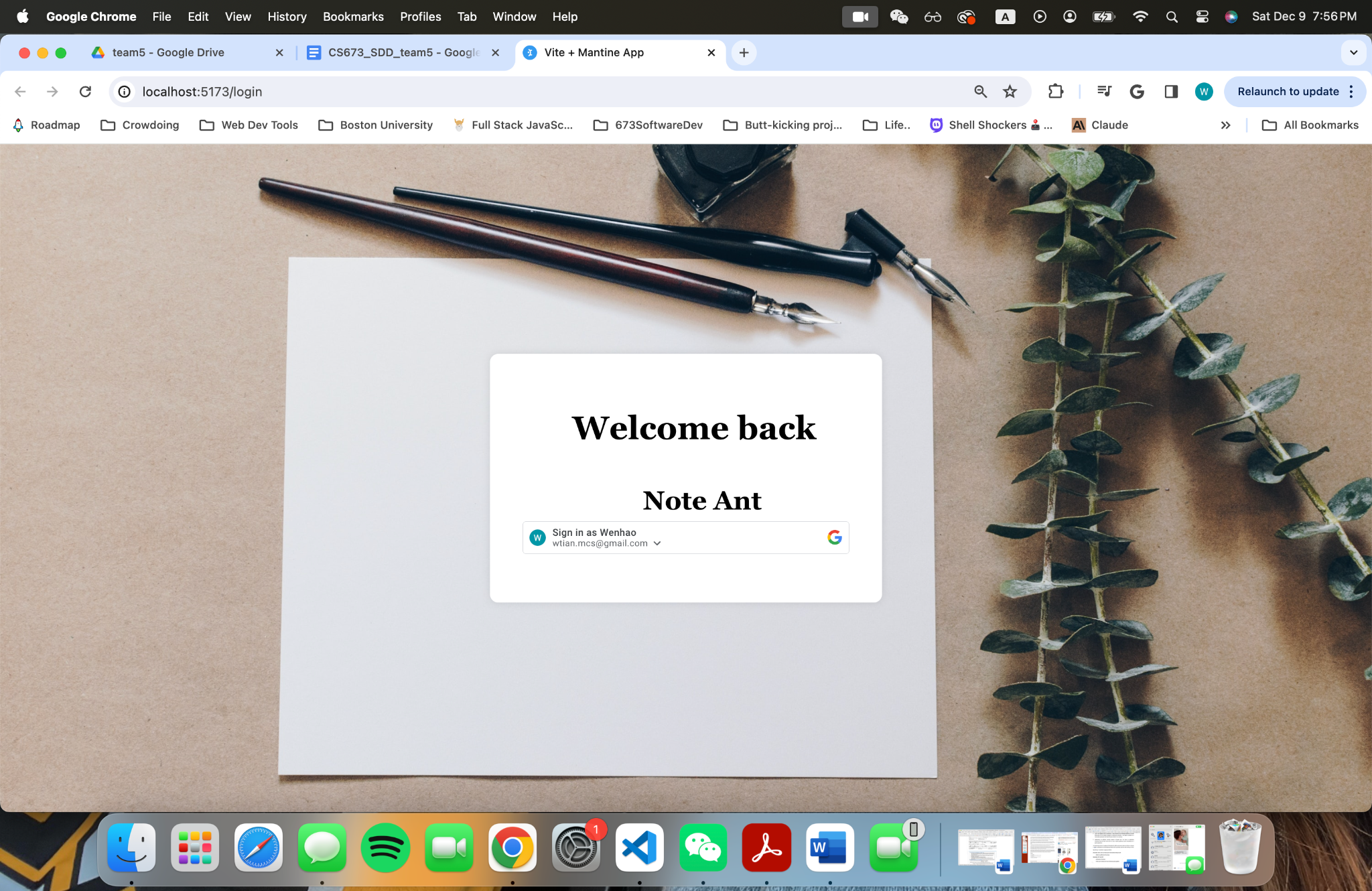
We first divide tasks into 3 pages: Login, user documents page, and document page. The login page handles user authentication and assigning tokens. User documents page displays all the documents belonging to this user. It also handles creating documents and delete documents requests. Then, when the user clicks onto one of the documents, the document id will be sent to the third page, document page. The document page will grab the document from MongoDB to present to the user. After the user finishes editing and clicks save, a post request will be sent to the document page for update. Another function on this page is generate. After the user clicks generate, document content will be collected and sent to OpenAI. OpenAI will return the summarized info back to the server.

# Class Diagram

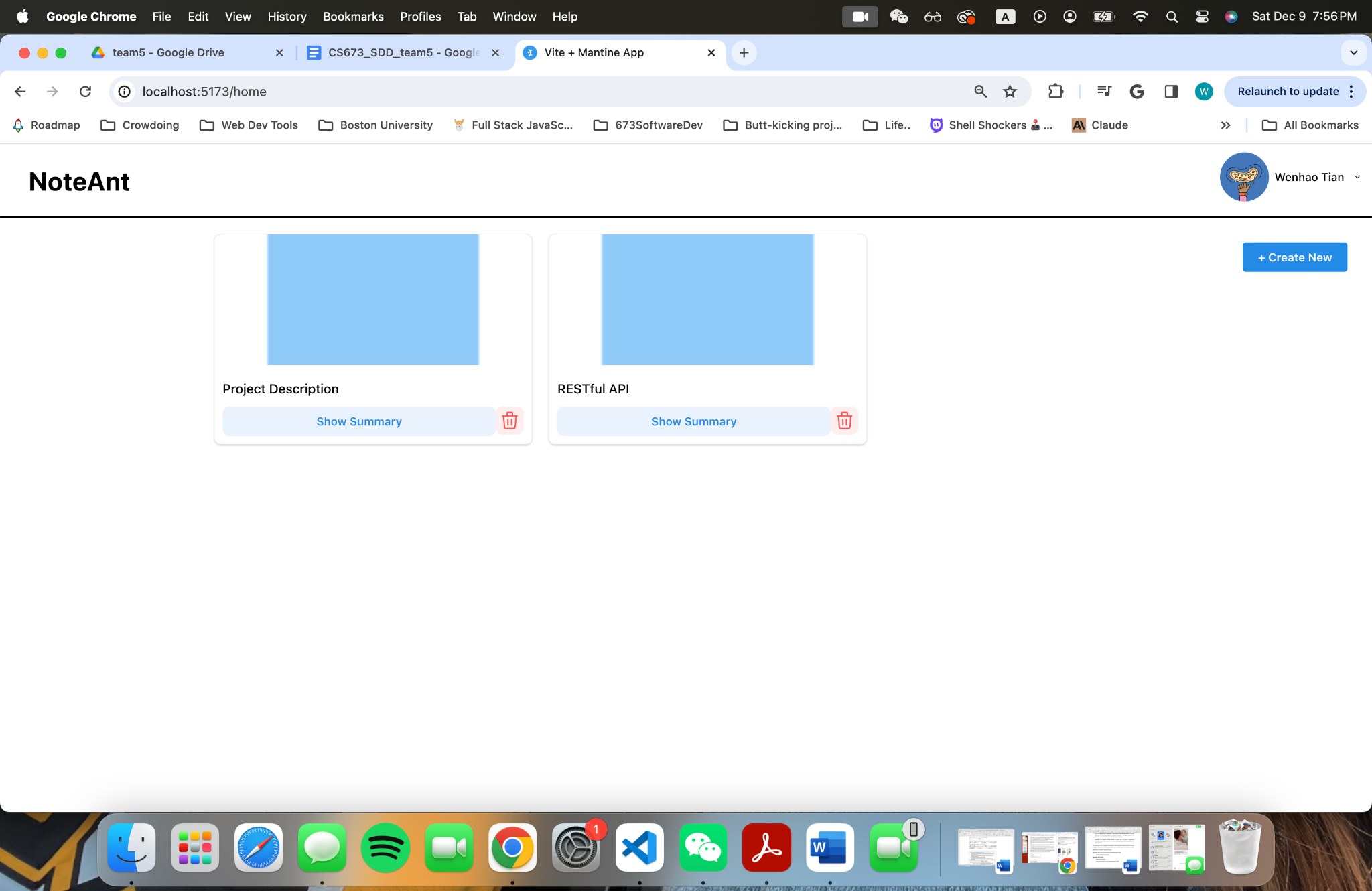


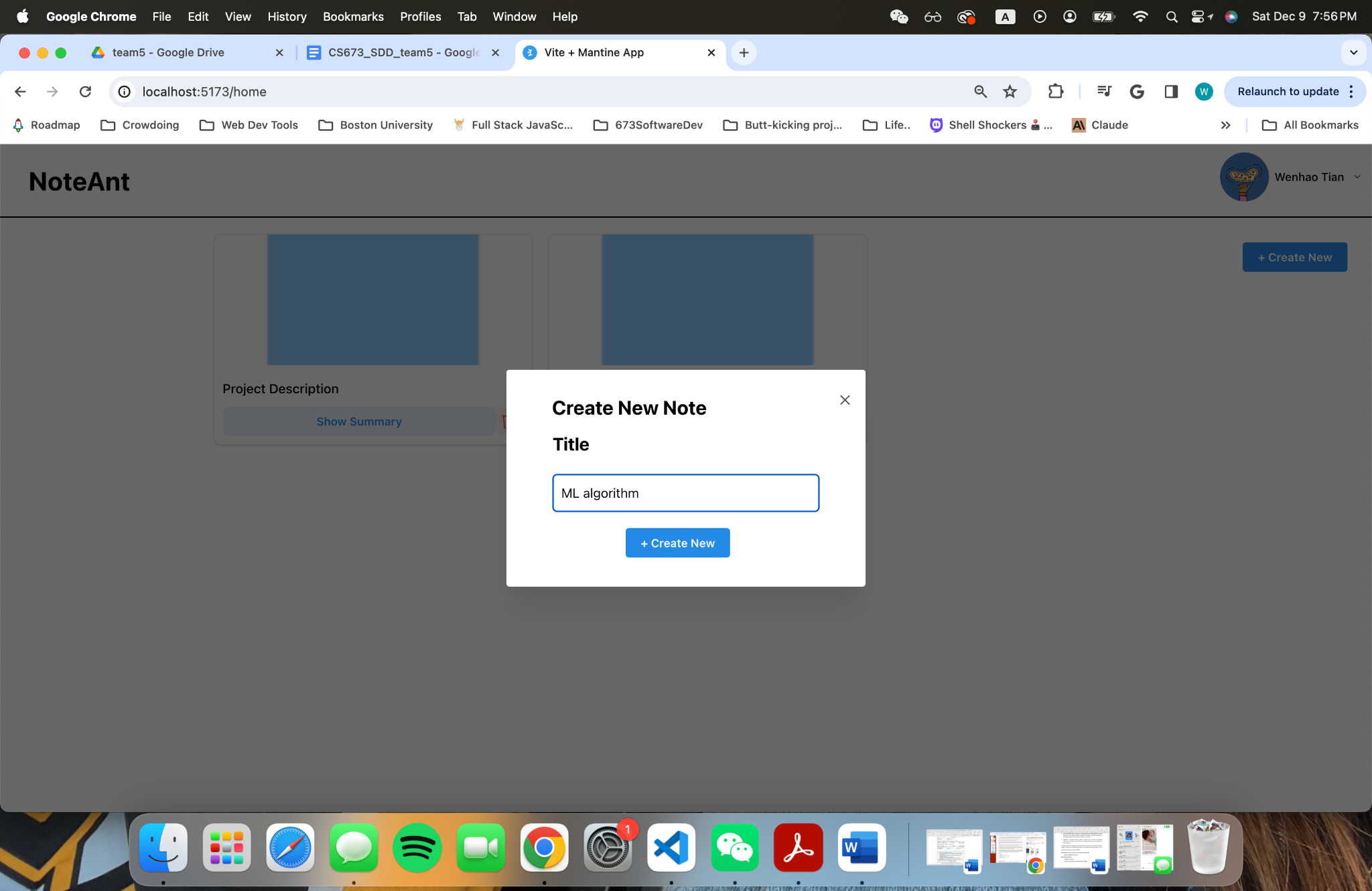
# UI Design (if applicable)

Our UI design part is divided into 3 pages.

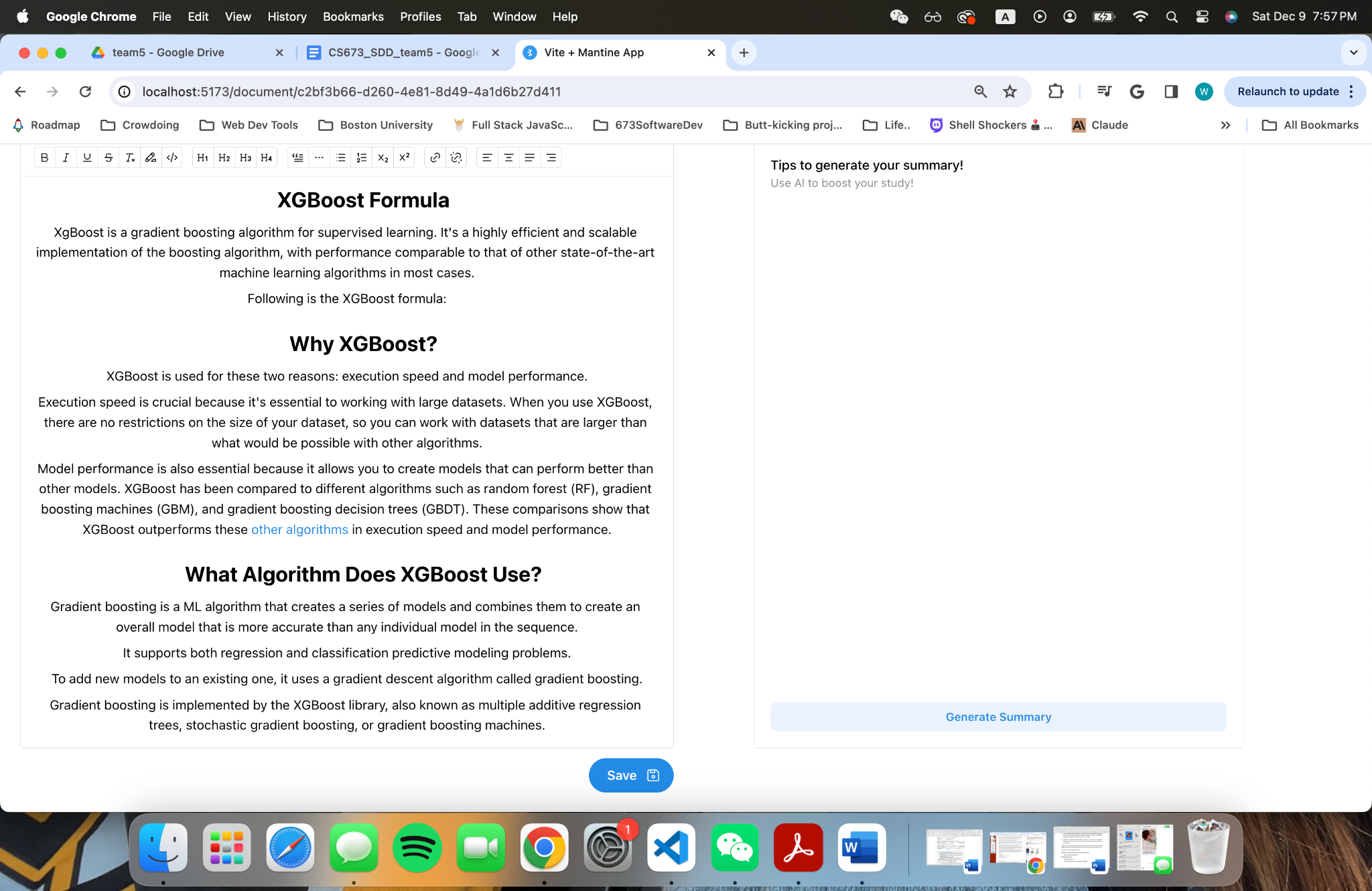
The first page is the login page. It uses GSI(Google Sign In), allowing users to login their account by their google account. 

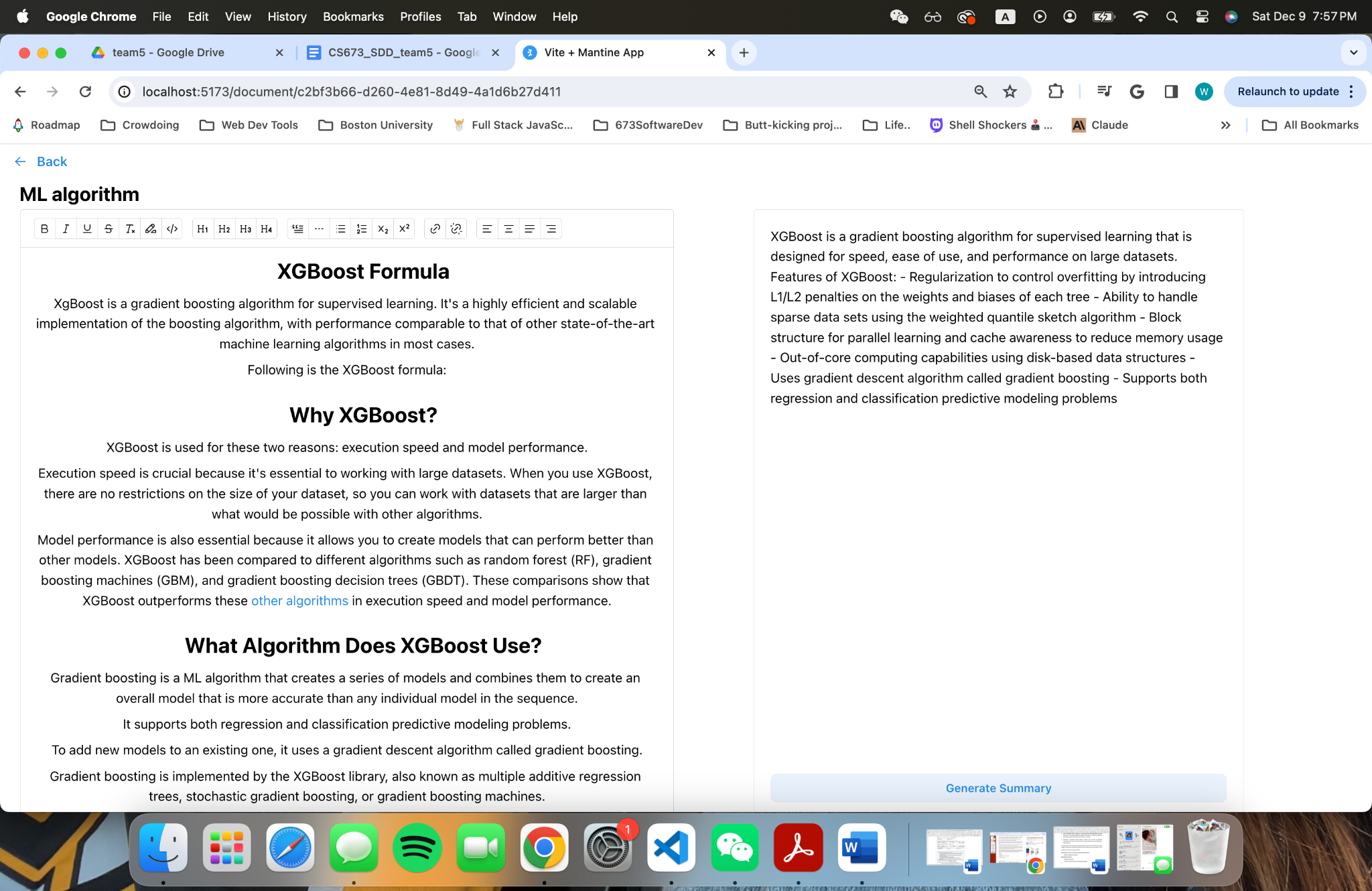
The second page is the user document page. Users can manage their personal info, their documents and create new notes (document) on this page. Our document card shows document pictures, document title and description. Then users double click one of their documents, they can be navigated to the third page. (GPI page)





The third GPI page contains the rich text editor on the left half and an AI assistant area on the right. Users can change their document style like font size, font alignment, bold text, italic text, etc. And users can save their note by clicking the save button and generate a ChatGPT assisted summary by clicking the generate button.The GPT will return the summarized content or something you want about your document.





# Database Design (if applicable)

For this project, we will use 2 data models: User and Documents.

The User model only contains : user\_id and username

The Document model contains : title, document.\_id, body, summary, last\_modified, and author, which reference to user.id.

# Security Design

Our software system prioritizes security by leveraging Google’s OAuth 2.0 protocol for authentication and authorization. This ensures that user’s passwords are never stored or seen by our application. Instead, Google authenticates the user and provides our application with a secure token. All communications are encrypted using HTTPS, and we only request necessary permissions from users, minimizing data access. Regular monitoring, logging and updates further bolster our commitment to safeguarding user data.

Also, on the server side, we used JWT to generate encrypted accessToken for managing client’s access to different routes. Once a user is signed in, an accessToken will be generated using the user's id, username, and the url for the user's avatar. Each token is only valid for 4 hours. Attackers won’t be able to access any of the user page and documents page methods without a valid accessToken and the secret key. When a user carries a valid accessToken in the header to access controller methods and database, the middleware, varifyToken, will verify the token with the secret key. After verification, middleware will decompose the token back to an user object and store it into the req field for all the next controllers to use.

By implementing Google’s OAuth2.0 and JWT safety measures, we are confident that the level of security would be sufficient for serving our clients and protecting our database.

# Business Logic and/or Key Algorithms

Json Web Token(JWT): JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way to securely transmit information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed with a secret (using the HMAC algorithm) or with a public/private key pair using RSA or ECDSA. It is used for our Authentication in our project.

# Design Patterns

We primarily used the MVC model. With front end paged being the view, controller methods being the control, and database

# References

# Glossary