**CS673 Software Engineering** 

**Team 5 - NoteAnt**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Frankie | Requirement leader | *Guancheng Huang* | 18 September |
| Chris | Design and Implementation leader | Wenhao Tian (Chris) | Sept 18 |
| Nicholas | Team Leader | *Nicholas Narmada* | 18 September |
| Wayne | Configuration leader | *Yichen Li* | Sept 18 |
| Siyuan Wan | Security Leader | *Siyuan Wan* | Sept 21 |
| Yibo(Wilbur) | QA Leader | *Yibo Wang* | Sep/21 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **Iter0** |  | **Sep/28/23** |  |
| **Iter1** |  | **Oct/19/23** |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_g6igqliy7rm)

[Related Work](#_bf21eadgjj29)

[Proposed High level Requirements](#_rgyo4hi9stmq)

[Management Plan](#_ts358bsdtbcv)

[Objectives and Priorities](#_nxeeppkjxgn4)

[Risk Management (need to be updated constantly)](#_tk7yixobah8p)

[Timeline (need to be updated at the end of each iteration)](#_iksrndohvx29)

[Configuration Management Plan](#_j5uvivmxqcsp)

[Tools](#_dzly5b9kz982)

[Deployment Plan if applicable](#_sd8zu6r3jisd)

[Quality Assurance Plan](#_vra5ptwu59qx)

[Metrics](#_vwjduhc9wuah)

[Code Review Process](#_hx3eaiwb8v3m)

[Testing](#_l9xnpmd6hh0y)

[Defect Management](#_5amsh8h9f0c7)

[References](#_pd9euov6m4du)

[Glossary](#_ty3i2nqffhtc)

# Overview

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, we will build a web application empowered by ChatGPT that helps users to complete and summarize notes.

# Related Work

Other note-taking apps include:

* Notion
* Google Docs

But all of these applications do not have an assistant to help us with studying. They can be powered with AI to help you take notes like autocomplete or grammar and spelling checking, but not AI geared towards helping students study.

# Proposed High level Requirements

* 1. Functional Requirements

Essential Features

* As a user, I want to create a new note, so that I can separate different notes from my different courses.
* As a user, I want to have a notes summary for every note I take, so that I can read the summary to help me memorize my notes better.
* As a user, I want to have my own account in this app, so that only I can see my own notes, and not share it with other users.

Desirable Features

* As a user, I want to have quizzes that are generated from my notes, so I can test myself to revise.
* As a user, I want to have other study tools or techniques such as flash cards, so that it helps me memorize my notes quicker.

Optional Features

* As a user, I want to collaborate with other users to work on the same notes, so that I can help and others can help me write a more complete note.  
  1. Nonfunctional Requirements
     1. Security requirements
        1. Make sure that student’s private information is not stolen
        2. Protect injection attack
        3. Sanitize html
     2. Invalid access
        1. An Error 404 catcher when a user put in a wrong url.
           1. /{userID}/{docsID}; these two fields could mismatch with our DB.
        2. Matching use sessions

# Management Plan

## Objectives and Priorities

Highest priority is to make sure that students can create notes, get their summaries, and have their own account.

The desirable features and optional features are of second and third priority respectively.

## Risk Management (need to be updated constantly)

**Risk Management Sheet Link:**

<https://docs.google.com/spreadsheets/d/1Mkqrwf78qoAUwuLOs1VyfQHNrrTR19mUvx5pOEzT0hg/edit#gid=0>

## Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration)

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | User sign in using Google | Implement GSI | 3 hrs |
| Locate specific user in DB | Extract unique user id from Google JWT | 2 hours |
| Display user’s documents | Design and setup page UI; using dummy data for listing document cards | 10 hours |
| Allow users to access and edit documents | Design and setup page UI; implement Mantine text editor component; Add save button and summary button | 10 hours |
| 2 | Once signed in, let user enter documents page | Remove dummy data; use Google JWT to locate specific user in DB; Retrieve all docs that belongs to this user | 5 hours |
| Let user summarize their note | Complete OpenAI and LangChain implementation; Display summary to the front end | 5 hours |
| Let user create and delete documents | Complete routes setup and methods, ensuring feature is completed. | 10 hours |
| 3 | Testing | Make sure everything works | As much as possible |
| Surveying | Add new features from surveys | TBD |

# Configuration Management Plan

## Tools

* Frontend:
  + Prettier & ESLint paired with Github Actions
  + Mantine UI
  + TanStack Query
  + React Router
  + Tabler
  + Google Cloud
* Backend:
  + ExpressJS: server-side framework
  + Joi: Server-side user authentication
  + Passport: username and password log in
  + LangChain: easy access to ChatGPT
  + Mongoose: build connection to MongoDB database and helps build data schema.
  + UUID: generate unique id for each user and documents.
  + Axios: data fetching; internet connection
  + MongoDB: store input data and quick data loading.
  + Cloudinary(tentative): store pdf, docs, jpg, and other files to the cloud. But why not MongoDB? MongoDB only supports textual information.
  + Jwt-decode
  + jsonwebtoken
  1. Code Commit Guideline and Git Branching Strategy

For code commits, we are planning to use conventional commits as a guideline.  
<https://www.conventionalcommits.org/en/v1.0.0/>

For git branching, we will first split up our tasks into issues, and use these issue numbers as branch names. Then, we will merge it all into a branch called iter{number}, depending on which iteration we are currently in.

## Deployment Plan if applicable

We are planning to deploy our web app with Vercel.

# Quality Assurance Plan

## Metrics

| Metric Name | Description |
| --- | --- |
| Lines of Code | Measures the total number of lines written in the software |
| Defect Rate | Number of defects per thousand lines of code |
| Test Case Pass Rate | Percentage of test cases that pass during testing |
| Number of user stories completed | Tracks the number of user stories or features that have been successfully implemented |
|  |  |

* 1. Coding Standard

We will use coding standards with the defaults provided by ESLint, which is a tool that helps make our code have a uniform rule. We can add plugins to ESLint that helps check our syntax in React and javascript.

## Code Review Process

Everyone should have their code submitted as a pull request. Design and Implementation Leader and Team Leader will both be responsible for reviewing the pull request.   
If it is a UI update, developers should also put screenshots of the changes made.

## Testing

* Tools/Framework: Use Jest for testing JavaScript applications.
* Personnel: The QA leader will handle integration testing. Each developer is responsible for unit testing their code.
* When and Types: Unit testing will be continuous, while integration testing will be done at the end of each sprint.
* Objectives: Ensure that all features work as expected and identify any potential bugs or issues.

## Defect Management

## Tool: GitHub Issues

## Types of Defects: Syntax errors, runtime errors, logical errors, and performance issues.

## Actions/Personnel: The team leader will prioritize defects. Developers are responsible for fixing defects in their code. Critical defects will be addressed immediately, while minor ones will be scheduled for future sprints.

# References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard. )

# Glossary