Design Project Proposal

NoteSum: Summarize Application

Authors
Michael Mulder
Kevin Singpurwala
Yoeri Otten
Barış İmre

Supervisor
Marieke Huisman

Client
Joel Tyhaar

September 23, 2020

Contents

1 Introduction

1.1 Motivation

Many students spend hours summarizing large texts. Especially in behavioral and medicinal studies here in Twente one needs to learn a lot of content from books of hundreds of pages. Summarizing these can become tedious and often involves much copy pasting from one document to the other. An act which does not improve the study process and makes the important process of summarizing less interesting to students. In this project we want to improve this process by making an application which improves this process and will hopefully support students in their study.

1.2 Project

NoteSum is a web application where one can seamlessly create notes from a book, web page or other supported source without the hassle of copy-pasting.

NoteSum extracts highlighted information from a PDF. The file that you want to summarize is uploaded and is viewable on one half of the application window. The output is extracted to the other half of the application window. Here you can view and edit your summary. NoteSum will have a number of features to make formatting easier, including the option for headings, sub-headings, and paragraphs, next to the options for bold, italic, underlining text, and the copying of pictures should be supported.

1.3 Setup and supervision

The project team consists of four people, the authors of this document. We as a team are primarily responsible for meeting the wishes of the client and the application he envisions.

Next to that the project also has a supervisor which is ultimately responsible for assessing our work.

2 Requirements

The requirements of the design are categorized based on their importance and feasibility. We have discussed these requirements with the client on multiple occasions and finalized as follows.

2.1 Stakeholders

Currently there are two stakeholders: the owner of the application (currently our client), and the end user.

The owner of the application is the primary supporter of the application and makes it available to end users. Currently the design does not include any functionality for the owner with the exception of hosting capabilities.

In the initial design the only real user is the end user. Currently this is primarily focused on students which have to read and summarize large amounts of text.

2.2 Functional Requirements

These requirements are categorized based on the MoSCoW model [1].

Must

- Be able to open a PDF file.
- Be able to select text from the PDF.
- Be able to copy the selected text to the summary file.
- Be able to export the summary file as a word file.
- Be able to edit the summary file in the application.
- Be able to take a picture from the book and save that in the summary file.
- Be able to style text.
  - Be able to put text in a heading.
  - Be able to put text in a subheading.
  - Be able to put text in a paragraph.
  - Be able to make text bold.
  - Be able to make text italic.
  - Be able to underline text.
  - Be able to create itemized lists.

Should

- Be able to export the summary file as PDF.
- Be able to save intermittent state locally, remember where in the pdf the user was and save the summary-file.
- Have a table of contents to easily navigate in PDF.
- Have a page number selection for the PDF file.
- Have hotkeys to style the text in the summary.
- Keep selected text in the PDF highlighted.

Could

- Be able to search in the PDF.
- Be able to deselect previous selected parts in the PDF.
- Be able to organize files into projects.
- Be able to make multiple summaries around one PDF.
  - A project can have multiple files.
  - Each file can have their own summary.
  - There can be a summary over multiple files or the entire project.
- Be able to link to another summary.
- Be able to open word files
- Be able to open websites
- Be able to highlight something as extra important.
  - The highlighted text will have a different color in the PDF.
  - The extracted text will have extra emphasis in the summary.
- Have a dark mode.
- Be able to copy a quote in APA style.
  - A guess for the needed information should be made by the program.
  - The information should come up in a popup.
  - The information should be editable by the user.
- Be able to link the exported text in the summary file back to where it was exported from the PDF.
- Have registration for users and only be available to logged in users.
- Have support to share a summary file with another user.
- Have support to have users collaborate on a summary.
- Have support to have users collaborate in real-time.
- Be able to organize summaries in folders.
- Be able to search the web for selected text.
  - Extra: Have the results depend on user's status, e.g. high school or Researcher.
- Have support for a table with important key terms.

Won't

- Have multi-screen support.
- Be able to make automatic quiz-questions based on the summary.
- Be able to automatically show connections between summaries.
- Have auto-summarization, be able to automatically select important text.

2.3 User Stories

To show how the above requirements could be used by our users we have made a small list of user stories. Since there are no other stakeholders which currently use the application we've only written user stories for the main user.

- As an end user, I want to be able to upload a PDF copy of a book, so that I can start making notes on it.
- As an end user, I want to be able to extract key information from a text book to make notes, so that I can study more efficiently.
- As an end user, I want to be able to edit my notes, so that I can understand the material better.
- As an end user, I want to be able to add pictures to my notes, to enhance my visual learning ability.
- As an end user, I want to be able to perform different actions on text such as changing text size, text colour, text style, text highlighting, so that my notes are more organized.
- As an end user, I want to be able to import and export PDF's within six seconds, so that I can save time.
- As an end user, I want to be able to make a bullet point summary out of text extracted from the book, in order to better structure my notes.
- As an end user, I want to be able to enable dark-mode so that I can read my notes with ease at night time.
- As an end user, I want to be able to export the notes I made to PDF format, so that I can easily read and share notes.

3 Design

The visual design of the web application was discussed with the client in the first meeting we held. From that point minor adjustments have been made for implementation and general aesthetic reasons, however the overall look is still as the client wanted.

The user design of the web application will be intuitive and straightforward. The outline will be a split panel with a PDF Viewer on the left of the screen and the summary being created on the right of the screen. There should be clear buttons for formatting and viewing functionalities on the top of the screen. Loading new files and saving work should be easy and accessible. Figure 1 shows a very initial version of what the design looks like.

Figure 1: NoteSum application in early stage of development

4 Design Choices

4.1 Programming Language

Since we are primarily making a front facing application, we have chosen to use web technologies in our project. This guarantees that our application can be easily made to work on a multitude of devices and formats which is beneficial for the project. With this in mind we chose to primarily make our application using TypeScript [2]. TypeScript is a superset of JavaScript [3] - the scripting language all browsers run - developed by Microsoft to add type checking to the web and to make a more stable language which can be compatible with older browsers.

These functions are expected to be beneficial to us during development since it would mean many issues which are encountered when using older browser technologies can be avoided. This way we can be more confident in the correct working of our application. Next to that the backing of a big software company like Microsoft gives us the confidence that TypeScript will be supported far into the future.

4.2 Frameworks

During this project we do not want to reinvent the wheel in any way, and be able to focus on the requirements we set out to achieve. Therefore we will be using several known frameworks. In this section we will talk about which frameworks we are planning to use and our reasons to use them.

4.2.1 React

React [4] is a framework and methodology to have a more concise way of creating front-end facing applications. Developed by Facebook it currently is one the most popular front-end framework next to Vue.js.

We have chosen this framework since it provides us with the capabilities to make a modern 'web app'. Next to that we noticed that the limited experience with web technologies within our group React was the most known.

4.2.2 Draft.js

Draft.js [5] is the library we chose for the summary editor on the right panel of the application. Its modular and React based design allows for easy functionality and reliability. Draft.js was created by Facebook as a tool to easily create WYSIWYG[1] editors for the web since the way this functionality is implemented is known for their edge cases.

4.2.3 PDF.js

PDF is a popular file format used for static documents which has a long history in its design. It therefore would not be feasible to create our own reader and renderer for PDF documents.

[1] WYSIWYG: What You See Is What You Get, a smart editor which hides the underlying structure to the user.

Instead we have chosen to use the PDF.js library [6] created by the Mozilla Foundation originally to provide a PDF reader in their browser: Firefox. This library provides and easy interface for us to display PDF files in our application.

4.2.4 Other frameworks

Throughout the project we might encounter other frameworks to use in our application. Since it is hard to predict what we will need we have a set of guidelines on when and how to use a framework.

- The framework should solve some task or add functionality which would not be feasible within the project without the framework.
- The framework should have active maintainers to assure that vulnerabilities and compatibility will be guaranteed in the future.
- New frameworks should only be used when no currently used framework is able to solve that task.

Next to these requirements all development is reviewed as explained in section 5, Methodology.

5 Methodology

This section of the report outlines how we will work on a daily bases and what are the standards we will use while working together as a group.

5.1 Task Management

We will use an online task board (kanban board) to track individual tasks in all contexts of work. With columns for tasks to be done, tasks in progress, and finished tasks. This will keep all the members of the team informed of what other members are doing and keep progress organized.

5.2 Collaborating

We will use GitHub to store and version control our implementation. The mentioned task board is also located on the GitHub interface. On GitHub we will use revisions by team members for other team members' contributions to keep the code clean and under check by the whole team.

5.3 Continues integration

To make sure that our code quality is up to par we use continues integration in our project for things like testing and linting new code. This helps us to early on figure out problems in the project and helps a reviser with their task to check other team members' contributions.

5.4 Testing

5.4.1 System testing

To make sure our application works as intended we we'll add automated system tests. Since our application is interface focused these test will make sure that the functions presented to the user work in the expected manner, e.g. that documents are rendered correctly and the formatting works correctly.

5.4.2 User testing

Once we have created our minimum viable product, testing will be performed throughout the remaining weeks as our application progresses. Our plan is to start testing by week 5. We will perform user testing with university students. This will allow us to gain an understanding in to what the end user wants. Their feedback will enable us to enhance our application.

6 Planning and Time Frame

The duration of the project is roughly 10 weeks since it is a quarter module. Time is divided between designing elements, meetings with stakeholders, and the actual implementation of the project.

6.1 Design

The first 3 weeks are mainly reserved for designing the application. However, the design process and the improvement of the design will be a recurring theme for the whole project. At the end of the 3 weeks a document explaining the design choices will be handed in.

6.2 Meetings with Stakeholders

Weekly meetings will be held with the stakeholders, namely the supervisor and the client. The process of the project as well as the progress of the design and implementations elements will be discussed in these meetings.

6.3 Implementation

During the design phase of the project, implementation will be a slow but steady process. However, once the design phase has ended, the must be done requirements are aimed to be finished by week 5 of the project. After the testing of such requirements the should be done requirements will be finished by the end of week 7. It is not clear how much can be achieved after this point in terms of the "could" requirements.

6.4 Planning outline

Our planning outline can be found in table 1.

6.5 Division of tasks

We did not make a division of tasks as of yet. Currently we have an initial base version of our project running and have put all of our requirements in our task management system. This way every member can choose which task to work on in collaboration with the other members. Dividing up every task between our members would cause problems in the planning. Since we're working with new technologies it is hard to make a fair division since the road blocks we'll encounter are unknown. Therefore not making it possible to make a fair division.

Of course this does mean we will need to check up weekly on what everyone has achieved and adjust where necessary, to make sure we meet the deadlines which are set.

week	
1	Start work on initial design requirements list Setup processes to work in Setup initial development environment
2	Revise design requirements and work on design proposal Start work on initial basic version of project
3	Feedback on design proposal Revise design proposal based on peer feedback Deadline design proposal Continue work on initial application
4	Work on requirements of the project Finished minimum viable product Start basis of final project report
5	Deadline must requirements Creation of volunteer testing plan Volunteer aggregation for testing Work on project report
6	Test product with volunteers Work on requirements of the project Work on project report
7	Finish testing with volunteers Deadline should requirements Work on project report
8	Revise work based on testing Work on other features and features which were delayed
9	Finish project report Revise work based on testing
10	Poster presentation

Table 1: Outline of planning

References

[1] D. Clegg and R. Barker, *CASE method fast-track: a RAD approach.* Addison-Wesley, 1994.

[2] Microsoft, "Typescript." [Online]. Available: https://www.typescriptlang.org/

[3] B. Eich, "Javascript." [Online]. Available: https://www.javascript.com/

[4] Facebook, "React." [Online]. Available: https://reactjs.org/

[5] Facebook, "Draft.js." [Online]. Available: https://draftjs.org/

[6] Mozilla, "Pdf.js." [Online]. Available: https://github.com/mozilla/pdf.js/