UNIVERSITY OF TWENTE.

NoteSum

DESIGN REPORT

NoteSum: Summarize Application

Authors
BARIŞ İMRE
YOERI OTTEN
MICHAEL MULDER
KEVIN SINGPURWALA

Supervisor
MARIEKE HUISMAN

Client
JOEL TYHAAR

B.Sc. Design Project Group 1

Faculty of Electrical Engineering, Mathematics and Computer Science

November 6, 2020

Contents

1 Introduction

1.1 Motivation

There were many interesting design projects to choose from. Our team felt most passionate about NoteSum. We saw that it was build for students by a student startup. This made us want to be involved in such entrepreneurship and activism by students. Many students spend hours summarizing large texts. Especially in behavioral and medicinal studies. Here at Twente, one needs to learn a lot of content from books consisting of hundreds of pages. Summarizing these can become tedious and often involves a lot of copy-pasting from one document to another. An act which does not improve the study process and makes the important process of summarizing less interesting and more time consuming for students. In this project, we wanted to make an application which improves this process. Our goal was to support students to study and create notes more effectively. This motivated us to work hard as we had the ability to greatly impact the way students create notes.

1.2 Research

After choosing to make NoteSum our design project, we firstly had to decide what kind of technologies and what kind of application we would create. After researching what the possibilities were it was decided that a web based application would be the best choice. The advantages of a web based application over a native app are extensive in our situation. Firstly, if we made a native application for a certain operating system, then a lot of work would have to be done to make that app also work on other operating systems. One can make native applications with for example React Native, but other native platform capabilities would still need to be implemented. Some down sides are that there may be a bug on one platform but not the other and testing would have to take place on multiple operating systems. Testing would have to be more extensive and would be more time consuming. On top of that it may be hard to keep the application cross-platform consistent. When creating a web application one does not need to worry about different operating systems. Unlike operating systems, all common web browsers perform more less the same. Thus, our web application can run on any machine with an internet connection.

1.3 Project Overview

NoteSum is a web-based application that creates summaries of notes. Its main feature is the ability to have a split screen of a PDF file on the left and a rich text editor on the right. One can quickly and seamlessly generate summaries by highlighting and screen cropping, without the hassle of copy-pasting. 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 has a number of features to make formatting text and adding images easier. These include the option for headings, sub-headings, and paragraphs alongside the options to format text as bold, italic or underlined. For images one can quickly take a screenshot and the image will automatically be added to your note taking editor. NoteSum runs mainly on the client side using React.[1]

1.4 Setup and Supervision

The Team The project team consisted of four people, the authors of this document. To build team moral and to get to know each other we organised team building activities. We also set up some brief introductory meetings to get to know everyone's background in terms of software development such as what languages and frameworks we had each worked with before. This played an important role in maintaining team cohesion and created the foundation upon which we created our initial design plan.

The Client We as a team were primarily responsible for fulfilling the wishes of the client and the application he envisions. We regularly met with the client to see if our implementation was on the right track, to see if any new feature requests were conceived by the client, to clarify what requirements had highest priority, to schedule when and how testing would take place and to gain a better understanding of what features the client wanted.

The Supervisor Next to that, the project also had a supervisor who was ultimately responsible for assessing our work. Our client was mainly concerned with our implementation and progress in making the end product. Our supervisor's main responsibility was to assess our work overall, from how well we worked together as a team and how we distributed the tasks at hand, to how well we organised meetings. Our supervisor also assessed how complete our reports were and how our implementation was going. We met weekly with our supervisor and updated our supervisor on our progress by presenting demos of our app and writing reports.

2 Requirements

The requirements of the design are categorized based on their importance and feasibility. We discussed these requirements with the client on multiple occasions and finalized them 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 who have to read and summarize large amounts of text.

2.2 Functional Requirements

These requirements are categorized based on the MoSCoW model [2].

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 have only written user stories for the main user.

1. 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.
2. As an end user, I want to be able to upload multiple PDF copies of books, so that I can make a summary from multiple texts.
3. As an end user, I want to be able to upload a PDF within six seconds, so that I can save time.
4. As an end user, I want to be able to render part of a PDF within 4 seconds, so that I can start creating notes quickly.
5. 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.
6. As an end user, I want to be able to edit my notes, so that I can understand the material better.
7. As an end user, I want to be able to add images from the book to my notes, to enhance my visual learning ability.
8. 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.
9. As an end user, I want to be able to resize the editor and PDF, so that I can focus on one aspect more.
10. As an end user, I want to be able to toggle the highlight button, so that I can read through text without accidentally adding it to my notes.
11. As an end user, I want to be able to enter full screen mode, so that I can review my notes in the editor.
12. As an end user, I want to be able to use shortcut keys, so that I can format text faster.
13. 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.
14. As an end user, I want to be able to navigate through PDF sections and go to a specific page in a PDF, so that I can read the PDF with ease and so that I do not spend extra time scrolling to the right page.
15. As an end user, I want to be able to export the notes I made to .docx and PDF format, so that I can easily read and share notes.
16. As an end user, I want to be able to create multiple projects, so that I can work on multiple subjects at the same time.
17. As an end user, I want to be able to register and log in, so that I can use the application.
18. As an end user, I want the app to save the state of all my current projects, so that I can revisit and look at work done in the past.
19. As an end user, I want to be able to save the state of my summaries. If I reload the page, lose internet connection or login from another device the state of my summary

is saved.

20. As an end user, I want to be able to navigate between different projects, so that I can work on multiple projects consecutively.
21. As an end user, I want to be able to navigate between different PDFs within the same project, so that I can work on multiple PDFs consecutively.

3 Planning

The duration of the project was roughly 10 weeks since it is one quartile of the academic year. Time was divided between designing elements, meeting with stakeholders, and the actual implementation of the project.

3.1 Design

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

3.2 Meetings with Stakeholders

Weekly meetings were held with the stakeholders, namely the supervisor and the client. The progress of the project as well as the progress of the design and implementations elements were discussed in those meetings.

3.3 Implementation

During the design phase of the project, implementation was a slow but steady process. However, once the design phase ended, the "must" requirements were aimed to be finished by week 5 of the project. After testing these requirements, the "should" requirements were finished by the end of week 7. It was not clear how much could have been achieved after this point in terms of the "could" requirements. We decided to implement some of the "could" requirements based on two factors, how easy they were to implement and how useful those features would be for users.

3.4 User Testing Plan

3.4.1 Testing Bounds

Scope The novelty of the web-application was not in the scope of this particular testing period, since it had already been tested well enough. What was tested is every interaction the user made with the system. We tested the functional requirements of the application, how intuitive the layout of the application was and the non-functional requirements of the application.

We did not test the back-end server side of the application, such as how much server space was required to support a given number of clients nor did we test any potential security vulnerabilities the system may have had. Security can be tested at a later date, but this was not within scope for this project. However it should be mentioned that all back-end security is handled by the Laravel library itself.

Test Environment The test took place in an open environment. There were not any control groups or restrictions put in place. We wanted to see how users naturally interact with the application. All one needed to participate was a laptop or computer with an internet connection and a URL to the website. This worked out quite well for us as in our final product people used our application in a similar fashion. We also maintained distance and stayed in line with corona measures and guidelines.

3.4.2 Testing Strategy

During alpha testing, after each new feature added, we tested the application in house to see if everything functioned properly. After which we commenced phase 1 of beta testing. A sizeable number of students from the University of Twente used our application in a production environment. We broke up testing in to three phases. At the end of each phase we asked users to fill in a feedback form and gather google analytics data. This further helped us analyse each user in tandem with their feedback form. In this way, we got a more objective view of our application as we had a wider range of questions. This reduced the possibility of not receiving feedback on core features. Alongside that, we also had some online interviews where we planned to discuss our application more thoroughly with users.

At this stage our minimum viable product had been created [3]. Thus, we started testing our application to see if our functional and non-functional requirements were up to par.

Phase 1 Testing commenced on the 9^{th} of October. For the initial testing phase we planned to focus more on the overall usability and feel of the application. This was to help see if all of the functional requirements were met.

Phase 2 In the second phase of testing, there was planned to have a greater emphasis placed on the non-functional requirements. In this phase we tested the performance of the system. From how quickly a PDF can be uploaded to how long you have to wait to export a summary. We also planned on testing any functional requirements which were altered or created after phase 1 of testing.

Phase 3 For our final phase we tested our near finished product. Improvements from phase 2 were tested. With an iterative testing plan we were able to reduce the likelihood of bugs slipping passed us [4]. The focus of this phase was on feedback related to user experience. During phase three it was planned to gather in depth feedback by interviewing users who tested our product. This gave us insights in to the product we developed and allowed us to ask follow up questions, which was something a feedback form would not have allowed. At the end of phase 3 we planned on having a polished and finished product.

3.5 Planning Outline

Our planning outline can be found in table 1.

3.6 Planning Reflection

In hindsight, our planning was mostly spot on for what we have actually done. All of the implementation followed the planning almost all of the time, with some minor bugs

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

taking slightly longer than anticipated to patch. The deadlines for the "must" and "should" requirements were reached within the time frame and testing started on the right time. It is worth mentioning that testing did take longer than anticipated, but since there were no implementation delays this did not create a problem.

3.7 Division of Tasks

We started with an initial base version of our project running and put all of our requirements in our task management system. This way every member chose which task they wanted to work on in collaboration with the other members. Dividing up every task between our members would have caused problems in the planning. Since we were working with new technologies it was hard to fairly divide up the tasks as it was hard to predict what road blocks we would encounter. Therefore, it was not possible to assign tasks equally amongst members.

Of course this does mean we needed to check up weekly on what everyone has achieved. This was done via GitHub issues. We adjusted where necessary to make sure each member met the deadlines which were set.

4 Methodology

This section of the report outlines how we worked on a daily bases and what the standards we used were, whilst working together as a group.

4.1 Task Management

In a project of this scale, keeping accurate track of issues and changes has a vital role. For this reason we used GitHub issues alongside an online task board (kanban board) to track individual tasks in all contexts of work. This board had columns for tasks to be done, tasks in progress, and finished tasks. This kept all the members of the team informed of what other members are doing and kept progress organized. At the end of the project, we have made and resolved more than fifty issues ranging from bugs to new features.

4.2 Collaborating

We used Git on GitHub to store and version control our implementation. The mentioned task board is also located on the GitHub interface. On GitHub we used revisions by team members for other team members' contributions to keep the code clean and under check by the whole team. We also worked on a new branch for new feature basis on our project, meaning every new addition was developed on a new branch on our repository, reviewed by other team members and then merged to the stable version of NoteSum.

4.3 Continuous Integration

To make sure that our code quality is up to par we used continuous integration in our project for things like testing and linting new code. This helped us figure out problems early on in the project and helped a reviser with their task of checking other team members' contributions. The importance of linting cannot be overstated especially for a language like TypeScript which has the potential to be extremely messy. After the deployment of the first live version of NoteSum, we also build a GitHub Actions workflow to automatically deploy a new version of the stable branch to the web, updating the front end regularly for better performance and user interaction.

4.4 COVID-19

Much like anything in these times, we cannot not mention COVID-19 for our project. During the implementation, there were times where most of the team members had to self isolate. This made working together harder than usual. We coped with this situation using online meeting both internally and with other stakeholders. We also benefited from the issues and project boards on GitHub.

5 Design Choices

5.1 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. This will be a commercial product and thus all the technologies availed of will be free to use for commercial purposes. With this in mind we chose to primarily make our application using TypeScript [5]. TypeScript is a super-set of JavaScript [6] - 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 through cross compilation.

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.

5.2 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.

5.2.1 React

React [1] 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. React works with “components” which build up the frontend in a tree like structure.

React can be used alongside many other libraries such as Redux. Redux is a small open-source JavaScript library that can perform state management.[7] In our project, implementing state management allows users to save their work. If a user reloads the application or even if a user closes the web application, any notes made will be saved. In other words, any changes to the state of the application are saved.

We have chosen this framework since it provides us with the capabilities to make a modern “web app”. Aside from these reasons, we noticed that there was a lack of web-based technologies knowledge in our team, and React was the most well known framework to us.

5.2.2 Draft.js

Draft.js [8] 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 many edge cases. Draft.js is very bare-bones and only supplies us with an interface to interact with the internal structure of the editor's contents.

5.2.3 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. Instead we have chosen to use the PDF.js library [9] 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.

5.2.4 Material-UI

We used Material-UI [10] for all the styling components, and most of the visual functionality, such as pop-ups, menus, and icons. Material-UI is a user interface framework written in React and uses the material design [11] guidelines. It was really helpful to realize the design elements with ease and inspired us to implement features in user friendly ways by browsing the catalogue for components.

5.2.5 Laravel

The backend has been written in PHP, specifically in the Laravel framework [12]. We chose to use Laravel because it enabled us to quickly set up a complete backend. It allows for easy database interaction and user handling. Besides the standard Laravel setup we use Laravel Passport for authentication [13], CORS middleware to handle Cross Origin Resource Sharing [14], and Scribe for documentation to keep a clear overview of our API [15].

5.2.6 Other Frameworks

Throughout the project we encountered other minor frameworks which we used in our application. Since it was hard to predict what we needed we had 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 was reviewed as explained in chapter 4, Methodology.

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

6 Design Process

We designed this application with our target market in mind. These are individuals who want to seamlessly create summaries from PDF documents using a clean desktop interface. We do not focus on mobile users as their screen is too small to provide a good user experience, however NoteSum is a web based product so one can use NoteSum from any device with an internet connection. The design process is split into sprints which are all two weeks of different types of activities. The following sections briefly explain each sprint and its contents.

6.1 Sprint One

The visual design of the web application was discussed with the client in the first meeting we held. We also discussed the requirements and performed requirements prioritization. This allowed us to rank requirements from high to low priority. This is viewable in section 2.2. After finalizing the requirements we created a wireframe diagram. Figure 1 depicts this. From that point on, minor adjustments were made to the implementation for general aesthetic reasons. However, the overall look and feel of the application was still as the client envisioned.

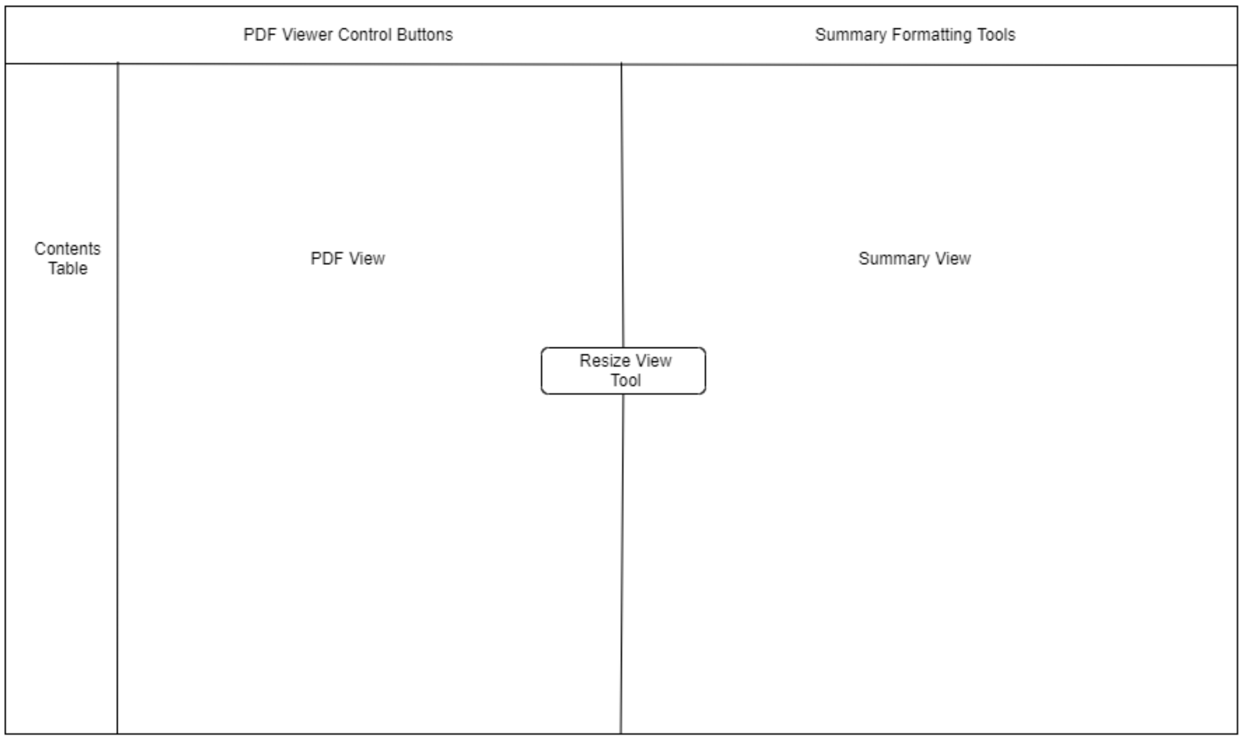
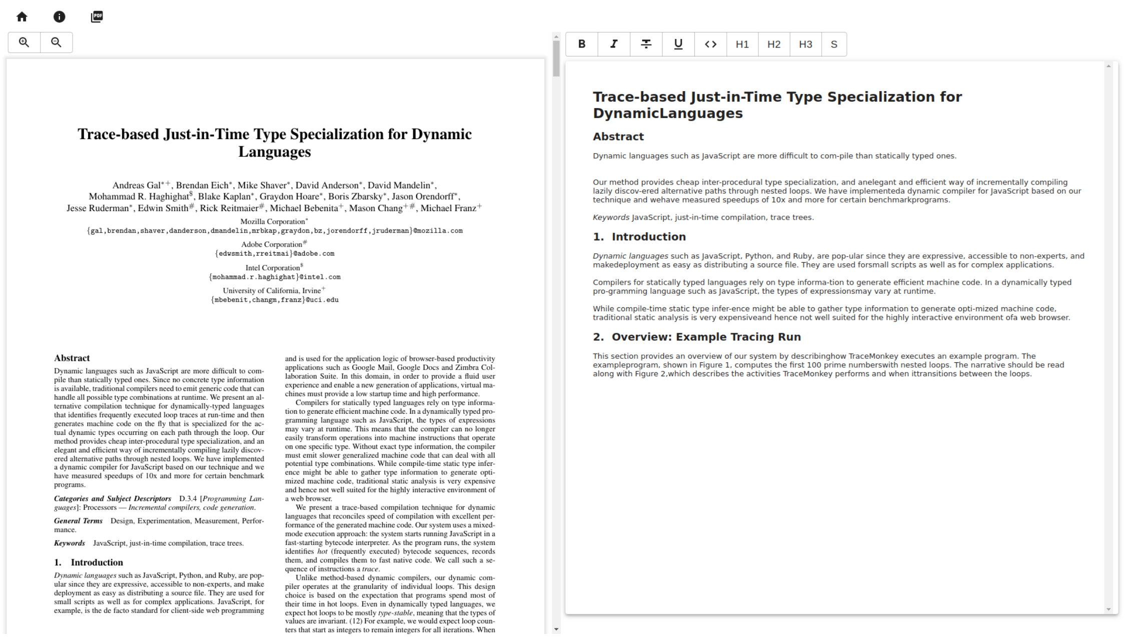


Figure 1: Wire-frame for NoteSum

The user design of the web application is intuitive and straightforward. The outline is a split panel with a PDF Viewer on the left of the screen and a rich text editor seen as a summary being created on the right of the screen. There are clear buttons for formatting and viewing functionalities on the top of the screen. Loading new files and saving work are also intuitive, easy to use and accessible. Figure 2 shows a very initial version of what the design looks like.



Trace-based Just-in-Time Type Specialization for Dynamic Languages

Andreas Gal*+, Brendan Eich*, Mike Shaver*, David Anderson*, David Mandelin*, Mohammad R. Haghighat$, Blake Kaplan*, Graydon Hoare*, Boris Zbarsky*, Jason Orendorff*, Jesse Ruderman*, Edwin Smith#, Rick Reitmaier#, Michael Bebenita+, Mason Chang+#, Michael Franz+

Mozilla Corporation*
{gal,brendan,shaver,danderson,dmandelin,mrbkap,graydon,bz,jorendorff,jruderman}@mozilla.com

Adobe Corporation#
{edwsmith,rreitmai}@adobe.com

Intel Corporation$
{mohammad.r.haghighat}@intel.com

University of California, Irvine+
{mbebenit,changm,franz}@uci.edu

Abstract

Dynamic languages such as JavaScript are more difficult to compile than statically typed ones. Since no concrete type information is available, traditional compilers need to emit generic code that can handle all possible type combinations at runtime. We present an alternative compilation technique for dynamically-typed languages that identifies frequently executed loop traces at run-time and then generates machine code on the fly that is specialized for the actual dynamic types occurring on each path through the loop. Our method provides cheap inter-procedural type specialization, and an elegant and efficient way of incrementally compiling lazily discovered alternative paths through nested loops. We have implemented a dynamic compiler for JavaScript based on our technique and we have measured speedups of 10x and more for certain benchmark programs.

Categories and Subject Descriptors D.3.4 [*Programming Languages*]: Processors — *Incremental compilers, code generation*.

General Terms Design, Experimentation, Measurement, Performance.

Keywords JavaScript, just-in-time compilation, trace trees.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are popular since they are expressive, accessible to non-experts, and make deployment as easy as distributing a source file. They are used for small scripts as well as for complex applications. JavaScript, for example, is the de facto standard for client-side web programming and is used for the application logic of browser-based productivity applications such as Google Mail, Google Docs and Zimbra Collaboration Suite. In this domain, in order to provide a fluid user experience and enable a new generation of applications, virtual machines must provide a low startup time and high performance.

Compilers for statically typed languages rely on type information to generate efficient machine code. In a dynamically typed programming language such as JavaScript, the types of expressions may vary at runtime. This means that the compiler can no longer easily transform operations into machine instructions that operate on one specific type. Without exact type information, the compiler must emit slower generalized machine code that can deal with all potential type combinations. While compile-time static type inference might be able to gather type information to generate optimized machine code, traditional static analysis is very expensive and hence not well suited for the highly interactive environment of a web browser.

We present a trace-based compilation technique for dynamic languages that reconciles speed of compilation with excellent performance of the generated machine code. Our system uses a mixed-mode execution approach: the system starts running JavaScript in a fast-starting bytecode interpreter. As the program runs, the system identifies *hot* (frequently executed) bytecode sequences, records them, and compiles them to fast native code. We call such a sequence of instructions a *trace*.

Unlike method-based dynamic compilers, our dynamic compiler operates at the granularity of individual loops. This design choice is based on the expectation that programs spend most of their time in hot loops. Even in dynamically typed languages, we expect hot loops to be mostly *type-stable*, meaning that the types of values are invariant. (12) For example, we would expect loop counters that start as integers to remain integers for all iterations. When

Trace-based Just-in-Time Type Specialization for DynamicLanguages

Abstract

Dynamic languages such as JavaScript are more difficult to com-pile than statically typed ones.

Our method provides cheap inter-procedural type specialization, and anelegant and efficient way of incrementally compiling lazily discov-ered alternative paths through nested loops. We have implementeda dynamic compiler for JavaScript based on our technique and wehave measured speedups of 10x and more for certain benchmarkprograms.

Keywords JavaScript, just-in-time compilation, trace trees.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are pop-ular since they are expressive, accessible to non-experts, and makedeployment as easy as distributing a source file. They are used forsmall scripts as well as for complex applications.

Compilers for statically typed languages rely on type informa-tion to generate efficient machine code. In a dynamically typed pro-gramming language such as JavaScript, the types of expressionsmay vary at runtime.

While compile-time static type infer-ence might be able to gather type information to generate opti-mized machine code, traditional static analysis is very expensiveand hence not well suited for the highly interactive environment ofa web browser.

2. Overview: Example Tracing Run

This section provides an overview of our system by describinghow TraceMonkey executes an example program. The exampleprogram, shown in Figure 1, computes the first 100 prime numberswith nested loops. The narrative should be read along with Figure 2,which describes the activities TraceMonkey performs and when ittransitions between the loops.

Figure 2: NoteSum application in the early stage of development

6.2 Sprint Two

In this sprint many function, and non-functional requirements were implemented. The PDF-viewer was enhanced. It now rendered much faster and could support large size 600 page PDFs. A page navigation bar was also implemented. With this feature a user can enter the page number on the PDF they wish to visit. A user can also navigate to a specified section, if the uploaded PDF contains such information.

The Editor was also substantially improved. More features such as full screen editor mode were added. Selecting text from the PDF to the summary view was also much smoother. There were also more options on how one can format text such as creating bullet points. One could also export summaries as word (.docx), PDF, text (.txt), and html formats. Later on we found out that this implementation only worked locally and thus some more work was required to allow users to export files on the web app. A screenshot-feature was added. This allows users to add images from the PDF to their summary.

Before this point in time we thought we may be able to make the whole application without a back-end by saving states locally on a user's computer or other electronic device. However, we encountered some difficulties as storing states locally is volatile. Ultimately we thought that it would be far better to have a back-end. Users' data is stored on a server instead of only locally on their machine as this is more reliable. The back-end was implemented using Laravel.[12]

State-management was done with Redux-store.[7] It saves the whole state of the application. Before, users would lose all of their summary if they reloaded the application.

Now the state of a user's summary is saved. Thus, notes in the summary view will remain as the state of the editor was saved. If the user is working on a summary and accidentally closes the tab or refreshing the application, the state of the application would have already been saved and the user will still have their summary available in accordance with the most recently saved state.

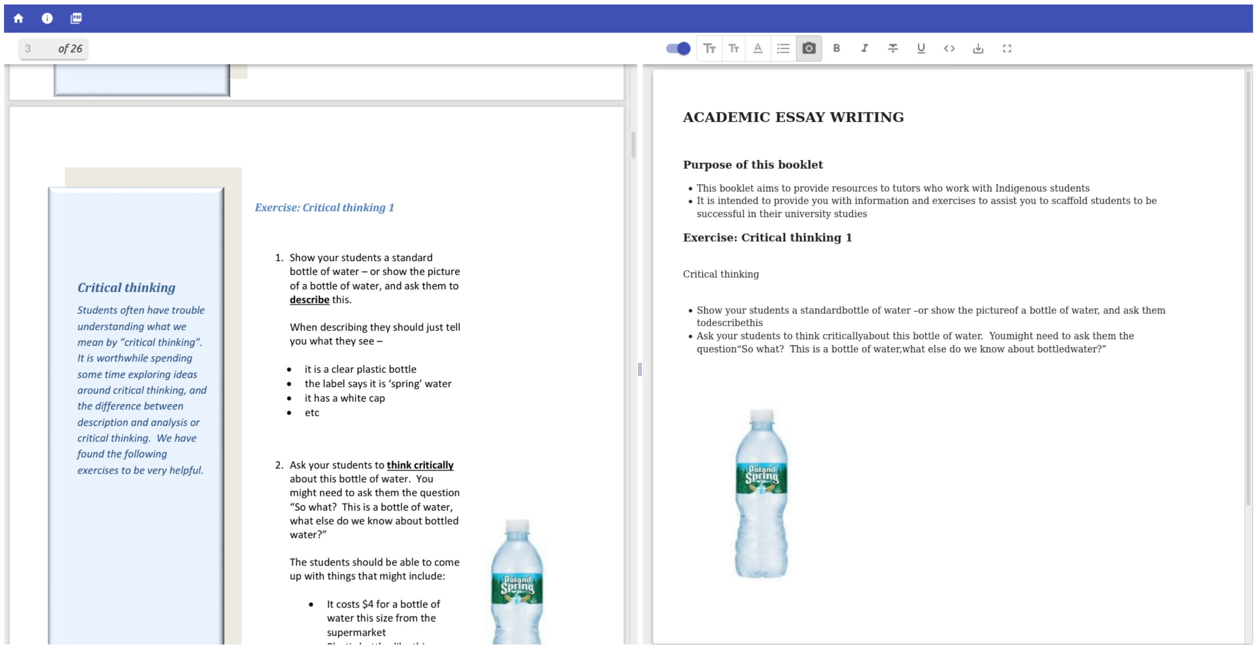
6.3 Sprint Three

Prior to this time we were working locally on our machines. In this sprint we set up a server, merged with the existing NoteSum website and went online. Most of the functional requirements responded the same as they did locally. However, the export feature did not work. This was then patched and a user was then able to export their summary from the web application to word(.docx) ,PDF , text(.txt) and html formats. Furthermore, the state of the summary and the location of the PDF (the part of the PDF that is currently viewable on screen) are saved locally. This further reduces the possibility of losing summaries that are currently being created.

A screenshot functionality was also added. This gives the user the ability to insert images in to their summaries by selecting areas of the PDF as a screenshot. The selected area will be added to the summary view as a base 64 image. Hotkeys were added to the application which makes it easier for users to navigate around the application using shortcuts. This was done by availing of key bindings in the editor. A nice feature that allows the user to resize the PDF and summary views was added. This gives the user the ability to focus on one view more so than the other. The ability for a user to upload multiple PDFs was also being explored in this sprint.

6.4 Sprint Four

Since we now integrated the NoteSum website with our application and had the application running online we were almost done with the creation of our minimum viable product. The last major task was to set up a login and registration page. In this way a user can sign up and have an account. Once a user has registered they can sign in and use the application. The advantage of having a registration and login system for users is that the system can keep track of each user's projects and uploaded PDFs. This type of system also enables the possibility for users to login from different devices which would not be possible if we only stored data locally on the user's machine. Our client wants this application to be a commercial product. With a registration and login based system it is easy to keep track of users who are entitled to use the product. A user now had the ability to upload multiple PDFs to the PDF view. In this way a user can use multiple texts to create one summary. This feature makes our application much more versatile.



Critical thinking

Students often have trouble understanding what we mean by "critical thinking". It is worthwhile spending some time exploring ideas around critical thinking, and the difference between description and analysis or critical thinking. We have found the following exercises to be very helpful.

Exercise: Critical thinking 1

1. Show your students a standard bottle of water – or show the picture of a bottle of water, and ask them to **describe** this.

   When describing they should just tell you what they see –

   - it is a clear plastic bottle
   - the label says it is 'spring' water
   - it has a white cap
   - etc

2. Ask your students to **think critically** about this bottle of water. You might need to ask them the question "So what? This is a bottle of water, what else do we know about bottled water?"

   The students should be able to come up with things that might include:

   - It costs $4 for a bottle of water this size from the supermarket

ACADEMIC ESSAY WRITING

Purpose of this booklet

- This booklet aims to provide resources to tutors who work with Indigenous students
- It is intended to provide you with information and exercises to assist you to scaffold students to be successful in their university studies

Exercise: Critical thinking 1

Critical thinking

- Show your students a standardbottle of water –or show the pictureof a bottle of water, and ask them todescribethis
- Ask your students to think criticallyabout this bottle of water. Youmight need to ask them the question"So what? This is a bottle of water,what else do we know about bottledwater?"

Figure 3: NoteSum Application - Project Window

Apart from the implementation, we started user testing in this sprint. Section 8.3 outlines this.

6.5 Sprint Five

In this sprint all remaining loose ends were tied up. Further documentation for the source code was created, bug fixing was undertaken, interviews were performed with users and a design project poster was created. Alongside this, the design report was finalised,we prepared to exhibit our final design presentation and we wrote about the ethical implications of our design choices and implementations. This sprint ended after the final presentation where we showed off all the work we did during the course of the design project, making sure to explain why we chose certain implementations over others with an academic undertone.

7 Technical Implementation

This section discusses the technical aspects of the implementation and explains in detail how individual components work.

7.1 Overall View

The project contains two sub-projects: the backend and frontend. These two entities are separate from each other and connect through an Application Programming Interface (API) hosted in the backend.

In our initial design proposal we had not foreseen implementing a backend since there were no requirements which required this. However during development we encountered browser limitations which we had not foreseen before. One of these limitations was the storage available to us in the browser. This was severely limited, making us only able to save a maximum of one PDF locally. Next to that, the browser gave no promises about the volatility of this storage, meaning anything saved locally could randomly disappear. To solve these issues we opted to make a simple backend which stores the PDFs and containes a simple mechanism to save a copy of user's state as well. As we had not foreseen to create a backend we opted to use the Laravel framework[12] since it provided us with already implemented security and stability features and was already known within the team.

The frontend is written in TypeScript with React. This framework allowed us to make NoteSum a modern, interactive website which benefits further development, usability and code quality. Next to React we used Redux for our state management and other dependencies for specific components.

7.2 Project Structure

7.2.1 Backend

The backend consists of a database, and a storage containing files. All interactions with either of these has to go through the API. See table 2 for all API endpoints, for further explanation of what each endpoint does and what frameworks are used is explained in section 7.7. See figure 4 for an overview of all relevant database tables.

7.2.2 Frontend

The frontend is divided in two parts: the components and Redux.

The Redux part is responsible for state management within the application, and is devided in:

- **Types**: containing type definitions for all possible actions on the state and the type of the state itself.
- **Actions**: functions to convert the component actions into Redux actions which can be reduced by the reducer.
- **Reducers**: applies Redux actions on the current state.
- **Migrations**: Update the local saved state to the newest type of state.
- **Store**: The main Redux state storage.

Action	Kind	URL	Required fields	Auth
Register	POST	/register	name, email, password	No
Login	POST	/login	email, password, client_name	No
Logout	POST	/logout	client_name	Yes
Info	GET	/user		Yes
Index	GET	/projects		Yes
Show	GET	/projects/{project}		Yes
Store	POST	/projects	name	Yes
Update	PUT/PATCH	/projects/{project}	name	Yes
Destroy	DELETE	/projects/{project}		Yes
Index	GET	/files		Yes
Show	GET	/files/{file}		Yes
Store	POST	/files	project_id, file	Yes
Update	PUT/PATCH	/files/{file}		Yes
Destroy	DELETE	/files/{file}		Yes
PDF	GET	/files/{file}/pdf		Yes

Table 2: Overview of all API methods

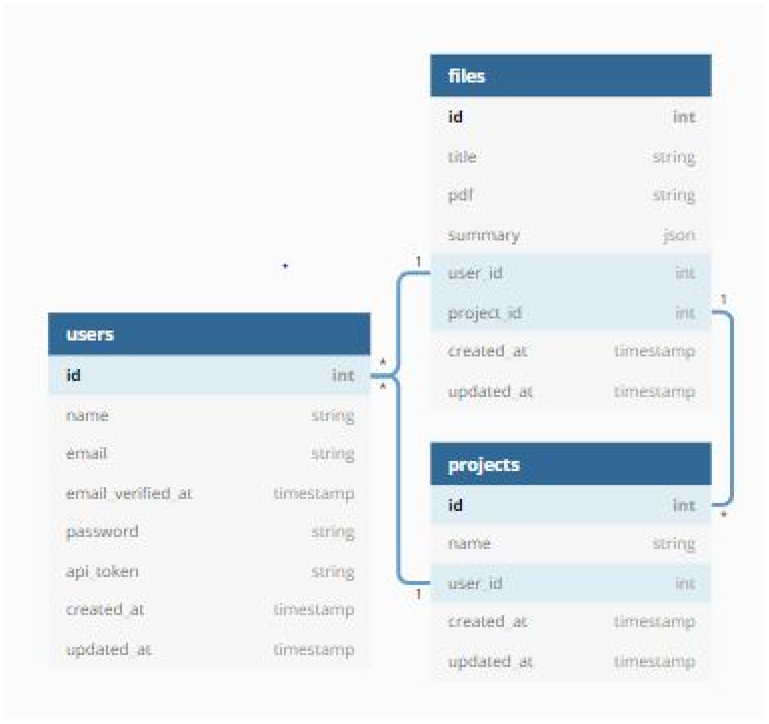


Figure 4: Database Diagram

The Redux state is stored locally and is synchronized with the backend when necessary.

The components are the items rendered by React in a tree like structure. To make future usability as easy as possible we have tried to split up as many components as possible. This has made it possible to easily reuse components on other parts of the website. The primary components are discussed in the next sections.

7.3 PDF Viewer

The PDF viewer is a simple viewer which can load and display PDF files, and can be created through a React component.

Because of the requirements set we could not use any readily available PDF viewer. For instance the screenshot capability is directly interacting with the underlying components which show the page. Therefore we opted on making our own components for this purpose with the associated rendering logic.

The PDF viewer is split into several sub-components to improve the concurrent rendering capabilities. An overview of these components are shown in figure 5 below.

The actual rendering is controlled from the page component but done by the PDF.js library. We felt it out of scope to create our own PDF renderer since the implementation of this would not be doable in the project. Next to that we benefited from the accuracy and speed this library provided.

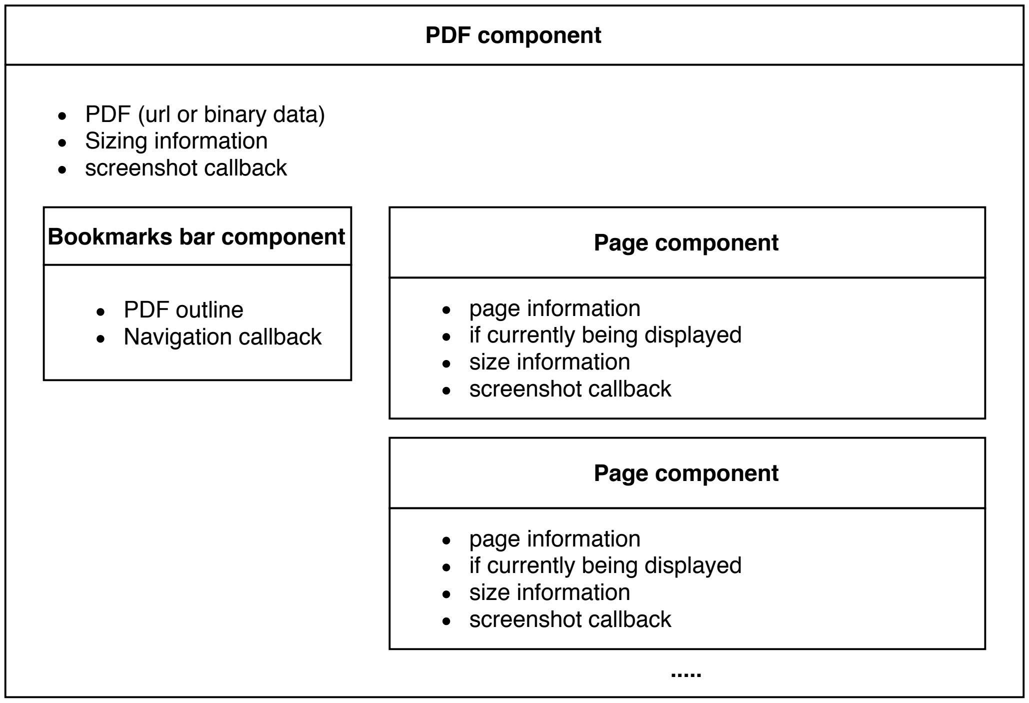


Figure 5: Simplified overview of the PDF component and its associated attributes.

7.3.1 PDF Component

The PDF component is primarily responsible for retrieving and interacting with the PDF. It downloads and loads the PDF and its associated information. However it is not responsible for the actual rendering of the PDF.

The PDF component also contains all controls for the PDF (except for the bookmarks button) with which the underlying pages can be controlled.

To render the PDF a separate page component is made for each page of the PDF. Through these page components the PDF component is able to retrieve the location of each page that is rendered to the user which is in turn used for the navigation.

7.3.2 Bookmarks Bar Component

The bookmarks bar component is a simple component which renders a tree like bookmarks menu. It is separated from the PDF component since in this way we are able to recursively define the tree structure of the bookmarks.

7.3.3 Page Component

The page component is primarily responsible for rendering the PDF. Internally this rendering consists of two elements: a draw layer and a text layer. Through the PDF.js library both these elements are rendered.

To make sure the newest version is always rendered the page component uses a small finite state machine internally. An overview of this state machine is given in figure 6. This FSM was necessary to make sure that only one rendering task was executed at the same time, if this does not happen, unusual artifacts can be rendered and the multiple rendering tasks would conflict with each other causing artifacts to be displayed.

To make sure no concurrency issues occur within the page, the state machine is implemented synchronously within the asynchronous rendering elements of the component.

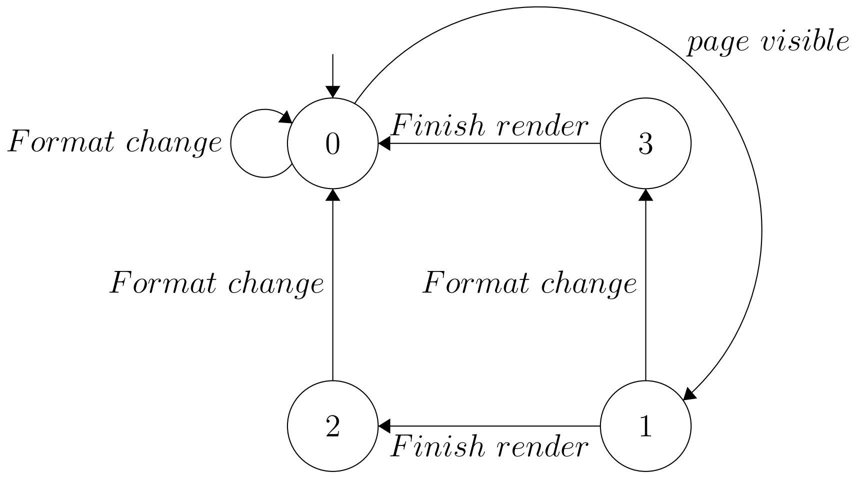


Figure 6: Final State Machine of PDF rendering process.
State 0 = Needs rendering, 1 = Rendering, 2 = Finished, 3 = Outdated during render

Next to the rendering, the page component also contains the functionality to create screenshots of the page. This is implemented by taking a copy of the selected area in the draw element and converting it to an image. Through the callbacks it receives, these screenshots are then propagated towards the PDF component and further to allow the editor to read out the contents.

7.4 Summary Editor

As mentioned in section 5.2, the framework choice for the summary editor is Draft.js. Since this is a bare bones library made to be extended and implemented based on needs of the developer, there are some important notes and usages relating to how this library is used. The following subsections try to give an explanation on how this project uses Draft.js and interacts with it from other components.

7.4.1 Main data structures

Draft.js has the convention of using maps (key value pairs) for all the important data structures. These are mostly wrapped in objects that also have some helper functions and properties. The entire contents of the editor with all the states is wrapped in an object called the

`EditorState`. This object contains the

`ContentState` which is where all the text is stored. There are also some other functions and data structures that are discussed in the next sub sections.

7.4.2 Content in the Editor

As mentioned content in the editor is kept in a

`ContentState`. This object contains the map of all the text with the inline styles or the block type (both will be explained) within a map called the

`blockMap`. This is what is rendered on the client side, and this is also the map that all the other functionalities in the application interact with. This map is used when converting to different formats or when serializing to store in the Redux store. The block map is keyed with random strings and the values are

`ContentBlock` objects, which are chunks of text that have the same block style. The block style is for example a list or a title where there can be various inline styles within a block.

7.4.3 Styling

Different from general content in the editor, Draft.js also provides functionality for inline styling of text. There is a utility class called

`RichUtils` that automatically toggles different styles to selected text based on the specs of the developer. For this implementation, using buttons, the user can toggle bold, italic, strike-through, underlined, and code styles. All of these work out of the box from Draft.js since the selected text is also a part of the

`EditorState`.

7.4.4 Highlighting from PDF

When the summary editor is rendered, an event listener is added to the web application to listen for mouse-up events. If after a mouse-up event there is highlighted text on the browser, with the parent element the PDF viewer the system understands that the user has highlighted something on the PDF and the text is copied over to the summary.

In more detail, if there is some text to be added to the summary, a new
`ContentBlock` object is added to the
`blockMap` with the selected style from the toolbar. This new
`ContentBlock` holds the text that the user highlighted. The editor recognizes this change end re-renders the summary. For nicer text, text blocks are stripped of the white space around them and are wrapped with new lines on top and on the bottom of the text. For the possible style choices of a
`ContentBlock` there are some custom made options enabled.

7.4.5 Images

The general idea of the images or screenshots is the same as highlighting from PDF. The minor differences are in the way they are detected and the type of block used for the editor data structures.

If the toggle for the screenshot is enabled from the toolbar, a callback is triggered in the PDF component to start selecting an area. Once this selection is over, again a callback is triggered from the PDF to the editor signaling an image. Images are sent in the base64 format and can be natively interpreted by Draft.js. A new
`ContentBlock` is added with the type
`atomic` representing an image.

7.4.6 Exporting to Another Format

Of course it is very important that users are able to take their work and use other files format for it. For this reason two different file formats are implemented in the current product. The summary can be converted to a Microsoft Word Document and a PDF, keeping the overall style of the document, the title and bullet point ranges and of course the images. In both of the cases, a library is used after the summary data is converted to an intermediate state. For Word Documents the library is called
`docx` and for PDF documents the library is called
`pdfmake`. As explained they both work in a very similar fashion. First, the summary so the
`ContentState` is converted to JSON format by iterating the
`blockMap` and converting every
`ContentBlock` to a corresponding field in the JSON. One iteration of the
`blockMap` in this fashion already yields a basically universal object for libraries to use. Then the individual library is called to convert the JSON to their respective internal formats. Then both of these internal objects are downloaded based on the library documentation.

7.5 State Management

We use Redux [7] for front-end state-management with Redux Persist[16]. Redux allows us to store and access information from anywhere in the application. Redux Persist saves this information in local storage so that it is not lost when the page gets reloaded or closed and opened again later.

Next to this we also save the files and summaries on the back-end, see section 7.7. This allows us to get back any files or summaries from a user if they log in on a different browser or cleared their local storage.

7.5.1 Redux Store

We have defined 4 reducers in our Redux store. Each reducer handles a different subsection of the information that we store in the Redux Store.

auth This part stores everything related to authentication. It contains the following information:

- **loading**, a boolean to indicate whether we are waiting for a server response. Will be set to true when a server request for login or signup gets send and set to false when a response is recieved. While this is true a loading icon will be displayed.
- **isLoggedIn**, a boolean to indicate whether the user is logged in.
- **token**, the authentication token used to authenticate the user on the server.
- **errors**, any errors that get returned from the server if something goes wrong with the authentication.
- **user**, the user information, consisting of:
  - **name**, the user's name
  - **email**, the user's email

projects This part store all the information related to the different projects a user can have. It contains an array containing all the projects of the user. It uses the id of the project as a key. Each project has the following information:

- **id**, the id the server assigned to this project.
- **name**, the name the user gave to this project.
- **files**, an array of the id's of the files in this project.
- **lastOpenFile**, the id of the file that is currently open, or was opened last.
- **createdAt**, a timestamp indicating when this project got created.
- **updatedAt**, a timestamp indicating when this project last got updated.

files This part stores all the information on the files in the currently open project. It again uses an array with the file id's as keys. Each entry has the following information:

- **id**, the id the server assigned to this file.
- **title**, the title of the file.
- **summary**, a representation of the current summary as discussed in subsection 7.4.
- **currentPage**, the currently open page, or last opened page of the PDF.

- **pdf**, an url pointing to where the PDF can be fetched from the server.
- **needsSave**, a boolean indicating whether the current state has been saved to the server.
- **lastSavedSummary**, the instance of the summary that is currently saved on the server.
- **createdAt**, a timestamp indicating when this file-representation got created.
- **updatedAt**, a timestamp indicating when this file-representation last got updated.

redirect This part is to keep track of redirection. It contains only one value, an URL. It gets set on the start of the redirect and reset once the redirect has succeeded.

7.6 Authentication

As mentioned in the previous section, authentication status is kept in the Redux store in our application. However, the user still needs to interact with NoteSum and be able log in and out. For this, the frontend displays a couple different pop-ups for authentication. Also for general use, creating and account and using that account is mandatory per user since we have to store user specific data. For usability reasons, the website will prompt the log in pop-up immediately on load if the user is not already logged in, which the frontend again uses the Redux store to check. The users can also choose to make a new account from this prompt. Throughout the entire website on the top bar there is an authentication icon that will display the user's name. This is to tell the users they are signed in and also to sign out, since a logged in user can sign out by pressing the top bar icon and then confirming log out.

All of these features are built using regular HTML forms and some styling for consistency. They connect to the backend by using the respective requests described in the following section. If at any time, the backend sends an error message, a panel on top of the authentication pop-up will appear describing the error message.

7.7 Backend

In this section we will take a detailed look at all the functionalities of the backend and the expected requests and responses of the server. We have divided this chapter in three subsections, each taking a look at a different aspect of the API. Each section start with an overview of that aspect, followed by a detailed description.

The base URL for the back-end is `https://api.notesum.org/api`. If there is something wrong with the request the server will respond with:

```json
{
    "errors": [
        "Message describing what is wrong.",
        "Maybe even a second thing that is wrong"
    ]
}
```

7.7.1 Authenticating requests

To authenticate requests, include an Authorization header with the value
`Bearer ‘your-token’`.

7.7.2 Authentication

Overview See table 3.

Action	Kind	URL	Required fields	Authenticated
Register	POST	/register	name, email, password	No
Login	POST	/login	email, password, client_name	No
Logout	POST	/logout	client_name	Yes
Info	GET	/user		Yes

Table 3: Overview of authentication methods

Register A new user can register by sending a POST request to `https://api.notesum.org/api/register`. This request must have the following fields in its body:

- **name** : A string with the name of the new user
- **email** : A string with the email of the new user, cannot be an email that has been used before.
- **password** : A string with the password of the new user, must be at least 6 characters.

On success the server will respond with the following body:

```
{
    "access_token": "your-token"
}
```

Login A user can login to the backend by sending a POST request to `https://api.notesum.org/api/login`. This request must have the following fields in its body:

- **email** : A string with the email of the user.
- **password** : A string with the password of the user.
- **client_name** : A string with the name of the client doing the request.

On success the server will respond with the following body:

```
{
    "token_type": "Bearer",
    "expires_in": 31535999,
    "access_token": "some-token",
    "refresh_token": "some-token"
}
```

Here the expires_in field gives the time that the access token will be valid. After the access token has expired it would normally be possible to get a new access token using the refresh token. This has however not been implemented, but it is included so that that can be added in the future.

Logout A logged in user can logout by sending a POST request to `https://api.notesum.org/api/logout`. This request must be authenticated as specified in section 7.7.1. This request must also have the following fields in its body:

- **client_name** : A string with the name of the client doing the request.

On success the server will respond with the following body:

```
{
    "message": "You have been successfully logged out"
}
```

Info A logged in user can get its own account details by sending a GET request to `https://api.notesum.org/api/user`. This request must be authenticated as specified in section 7.7.1.

On success the server will respond with the following body, here example values are used:

```
{
    "data": {
        "name": "John Doe",
        "email": "test@test.com",
        "created_at": "2020-09-19T16:11:58.000000Z",
        "updated_at": "2020-09-19T16:11:58.000000Z",
    }
}
```

7.7.3 Projects

Overview See table 4. Projects are linked to a user, so all requests must be authenticated as specified in section 7.7.1.

Action	Kind	URL	Required fields	Authenticate
Index	GET	/projects		Yes
Show	GET	/projects/{project}		Yes
Store	POST	/projects	name	Yes
Update	PUT/PATCH	/projects/{project}	name	Yes
Destroy	DELETE	/projects/{project}		Yes

Table 4: Overview of project related methods

Index A logged in user can get all its projects by sending a GET request to `https://api.notesum.org/api/projects`. This request must be authenticated as specified in section 7.7.1.

On success the server will respond with the following body, here example values are used:

```
{
    "data": [
        {
            "id": 2,
            "name": "project",
            "files": [
                {
                    "id": 9,
                    "title": "cc-take-home-1",
                    "pdf": "/files/9/pdf",
                    "summary": null,
                    "project_id": 2,
                    "created_at": "2020-10-09 19:02:49",
                    "updated_at": "2020-10-09 19:02:49"
                }
            ],
            "created_at": "2020-10-09 19:02:49",
            "updated_at": "2020-10-09 19:02:49"
        }
    ]
}
```

Show A logged in user can get the details of a specific project by sending a GET request to `https://api.notesum.org/api/projects/project`, where project is the id of the project the user wishes to get the information from. This request must be authenticated as specified in section 7.7.1.

On success the server will respond with the following body, here example values are used:

```
{
    "data": {
        "id": 2,
        "name": "project",
        "files": [
            {
                "id": 9,
                "title": "cc-take-home-1",
                "pdf": "/files/9/pdf",
                "summary": null,
                "project_id": 2,
                "created_at": "2020-10-09 19:02:49",
                "updated_at": "2020-10-09 19:02:49"
            }
        ],
        "created_at": "2020-10-09 19:02:49",
        "updated_at": "2020-10-09 19:02:49"
    }

}
```

Store A logged in user can create a new project by sending a POST request to `https://api.notesum.org/api/projects`. This request must be authenticated as specified in section 7.7.1. Furthermore the request should have the following field in its body:

- **name** : The name of the project to create

On success the server will respond with the following body, here example values are used:

```
{
    "data": [
        {
            "id": 2,
            "name": "project",
            "files": [],
            "created_at": "2020-10-09 19:02:49",
            "updated_at": "2020-10-09 19:02:49"
        }
    ]
}
```

Update A logged in user can edit the name of a project by sending a PUT or PATCH request to `https://api.notesum.org/api/projects/project`, where project is the id of the project the user wishes to update. This request must be authenticated as specified in section 7.7.1. Furthermore the request should have the following field in its body:

- **name** : The new name of this project

On success the server will respond with the following body, here example values are used:

```
{
    "data": [
        {
            "id": 2,
            "name": "New Project Name",
            "files": [],
            "created_at": "2020-10-09 19:02:49",
            "updated_at": "2020-10-09 19:02:49"
        }
    ]
}
```

Destroy A logged in user can delete a project by sending a DELETE request to `https://api.notesum.org/api/projects/project`, where project is the id of the project the user wishes to delete. This request must be authenticated as specified in section 7.7.1. This removes the project from the database.

On success the server will respond with the following body:

```
{
    "message":"Successfully deleted project"
}
```

7.7.4 Files

Overview See table 5. Projects are linked to a user, so all requests must be authenticated as specified in section 7.7.1.

Action	Kind	URL	Required fields	Optional fields	Auth
Index	GET	/files			Yes
Show	GET	/files/{file}			Yes
Store	POST	/files	project_id, file		Yes
Update	PUT/PATCH	/files/{file}		title, summary	Yes
Destroy	DELETE	/files/{file}			Yes
PDF	GET	/files/{file}/pdf		access_token	Yes

Table 5: Overview of file related methods

Index A logged in user can get all its files by sending a GET request to `https://api.notesum.org/api/files`. This request must be authenticated as specified in section 7.7.1.

On success the server will respond with the following body, here example values are used:

```
{
    "data": [
        {
            "id": 1,
            "file_name": "FileName",
            "summary": null,
            "user_id": 1,
            "project_id": 1,
            "created_at": "2020-10-09 19:02:49",
            "updated_at": "2020-10-09 19:02:49"
        }
    ]
}
```

Show A logged in user can get the details of a specific file by sending a GET request to `https://api.notesum.org/api/files/file`, where file is the id of the file the user wishes to get the information from. This request must be authenticated as specified in section 7.7.1.

On success the server will respond with the following body, here example values are used:

```
{
    "data": {
        "id": 9,
        "title": "cc-take-home-1",
        "pdf": "/files/9/pdf",
        "summary": null,
        "project_id": 2,
        "created_at": "2020-10-09 19:02:49",
        "updated_at": "2020-10-09 19:02:49"
    }
}
```

Store A logged in user can upload a new file by sending a POST request to `https://api.notesum.org/api/files`. This request must be authenticated as specified in section 7.7.1. Furthermore the request should have the following fields as form data:

- **project_id** : The id of the project this file should belong to. Has to be a valid project owned by the logged in user.
- **file** : The file in PDF format

On success the server will respond with the following body, here example values are used:

```
{
    "data": {
        "id": 9,
        "title": "cc-take-home-1",
        "pdf": "/files/9/pdf",
        "summary": null,
        "project_id": 2,
        "created_at": "2020-10-09 19:02:49",
        "updated_at": "2020-10-09 19:02:49"
    }
}
```

Update A logged in user can edit the name of a file, or save a summary belonging to a file by sending a PUT or PATCH request to `https://api.notesum.org/api/files/file`, where file is the id of the file the user wishes to update. This request must be authenticated as specified in section 7.7.1. Furthermore the request should have one of the following field in its body:

- **title** : The new title for this file.
- **summary** : The summary state in JSON format.

On success the server will respond with the following body, here example values are used:

```
{
    "data": {
        "id": 9,
        "title": "cc-take-home-1",
        "pdf": "/files/9/pdf",
        "summary": /* JSON summary state */,
        "project_id": 2,
        "created_at": "2020-10-09 19:02:49",
        "updated_at": "2020-10-09 19:02:49"
    }
}
```

Destroy A logged in user can delete a file by sending a DELETE request to `https://api.notesum.org/api/files/file`, where file is the id of the file the user wishes to delete. This request must be authenticated as specified in section 7.7.1. This removes the file information from the database and deletes the PDF from the server storage.

On success the server will respond with the following body:

```
{
    "message":"Successfully deleted file"
}
```

PDF A logged in user can get the PDF of a file by sending a GET request to `https://api.notesum.org/api/files/file/pdf`, where file is the id of the file the user wishes to get the PDF from, this URL is also specified in the response when getting a file. This request can be authenticated using the authentication header, as specified in section 7.7.1, or it can be authenticated by passing the token in the URL. In this case the URL would be

`https://api.notesum.org/api/files/{file}/pdf?token={token}` where file is the id of the file you want to get the PDF from and token is the access_token of the user.

On success the server will respons with the requested PDF.

7.8 Continuous Integration and Deployment

For our codebase to stay healthy we have opted to use Continuous Integration (CI) to check that the code quality was up to par.

Since our codebase is hosted on GitHub we have opted to use their GitHub Actions service. Github Actions is a format to specify actions to be taken on a remote machine when certain events are triggered. One such action is our linter. When pushing code to our repository a linter is started which automatically checks the code quality.

Because we also wanted our project to be intensely tested by users we needed a stable deployment strategy. Thus for deployment we have opted to use Docker containers. Docker creates a static environment for us to build and release our application in. This guarantees us that all builds and deployment happen under the same circumstances. Through GitHub Actions a new Docker container is built every time a new release is created.

On our hosting side we have setup a Docker platform which runs the Docker containers. Through Watchtower [17] these are automatically checked every five minutes and updated when necessary. This way our deployment happens fully automatically.

8 Testing

8.1 Code Peer Review

Each time a team member implemented a new feature, they pushed it to a branch on remote and created a pull request. This branch was then reviewed by another team member. Lint actions were used to show any linting errors that occurred when pushing to GitHub. Once all the changes in a branch were reviewed and when all errors were corrected, this branch was merged to master. This process helped to keep the quality of code high and prevented the likelihood of bugs appearing on the master branch.

8.2 Integration and System Testing

Multiple functional components were grouped together in the application. These components were tested using selenium. Selenium provides automated web software testing. To ensure that our application worked as intended, automated system tests were added. This tested the integrated component groups as a whole. Often times, components work on their own but may have some bugs when integrated together. Since our application was interface focused, these tests made sure that the functions presented to the user work in the expected manner. E.g. that documents are rendered correctly and that formatting works correctly. These tests were split in to three main areas. Login, Registration and the application.

Login This was a basic functionality test. It was tested that the right page loads after logging in and that the login page itself was accessible so one can input their login details.

Registration This included a basic test, a test to make sure that registering with an already registered email address was not possible and a test to make sure that the right error is thrown if your password does not match during registration.

The application Application testing was broken up in to two parts: the editor and the PDF reader. This was a system test as firstly one needed to log in, then open a project and then start testing the application. In this way multiple components were consecutively tested. The PDF tests included checking if one can navigate to a specified page number and section. The rich text editor tests consisted of checking if text could be correctly selected and copied to the editor, if text could be formatted correctly, if images could be added successfully and if the file could be saved as a `.docx` file. An integration test called `addImageTest.side` tests to see if an image was successfully added from the PDF reader to the text editor.

8.3 User Testing

Our application is heavily UI focused. Thus, usability testing is the most important and complete test for us as we get to see how real users interact with our app. Once we created our minimum viable product, testing was performed throughout the remaining weeks as our application progressed. Our plan was to start testing on October 9th. Thanks to good team communication and well planned sprints, the functionality deadline was met, however release was one day later due to stability issues of our release platform at that moment. We performed user testing with university students as they fall within our target group. This allowed us to gain an understanding in to what the end user wants. Their feedback and (anonymous) Google Analytics statistics enabled us to enhance our application. Section 3.4 outlines how user testing was planned to take place.

8.3.1 Product Risk Analysis

Due to restrictions such as time, we had to make tough decisions on what to test and how thoroughly to test each feature. We decided to make testing the functionality of the system our top priority. This was done in order to make sure that the functional requirements were not solely met but fully realized.

8.3.2 Overall Feedback

There has been overwhelming positive feedback from most of the users that have tested NoteSum in the last weeks of development. Around 80 individual users have made an account and tested to some extend the product. A great deal of the feedback is relayed to us by the client, since the client played a critical role in getting more users to try out NoteSum. On the first day of user testing a couple of critical oversights by the development team were discovered and patched within the day. Mainly making new accounts and logging in were not designed in a way that users found intuitive. These are now rebuilt with that feedback.

8.3.3 Shortcomings of User Testing

Even though the web application includes a clear feedback button, we have seen from almost every user that they simply ignore this. Of course, we cannot force users to submit feedback through a form they do not feel like filling out, but this leaves us with very little official feedback. As mentioned most of the feedback we got was through our client and as useful as that is, as developers we value direct feedback from users with us being able to ask them further questions. This is why we decided to personally interview some users to get feedback. The next section discusses this in detail.

8.4 User Interviews

Thanks to our clients support in finding testers for the application, we have interviewed two users in depth about their opinion after using the web site for some time. Due to the confidentiality of the testers they are not named but this section discusses the feedback received from those two interviews and the answers they provided to crucial questions for the application.

8.4.1 Setup

The two users interviewed are both university students with different studies and genders. They both have used the application before the interview to a level where they were comfortable giving feedback about it.

8.4.2 Other Formats Used

To both of the interviewees we asked which file formats they use for most of their studies other than PDF. The answer in both cases was PowerPoint presentations and rarely other book formats like e-pub. Since both of those formats are relatively easy to convert to PDF, our application seems to target almost all the file formats these testers are using.

8.4.3 Positive Feedback

The web application was praised by both the interviewees, with them expressing the novelty of the idea and saying it helped a lot with productivity. The project structure and different projects came up on both the interview as a strong point and the simplicity of the application was also mentioned multiple times. Both of the students repeatedly mentioned that they see this as a revision tool more than just a reading tool for themselves, meaning they would want to read the files they need to study first, and then revise using NoteSum. This gave us some insight on how the actual users fit this application into their work flows. In the future, this feedback can play a crucial role in the decision of which new features to implement.

At the end of both interviews, we asked the testers if they would use NoteSum consistently in their studies, assuming the service would be permanent. In both the cases they answered again with a very positive attitude, which furthered our belief in the novelty of the idea and the implementation.

8.4.4 What Would the Testers Add to the Implementation

In both of the cases the testers emphasized the need for a collaborative feature in NoteSum. This is obviously out of the scope of our project but a very good idea indeed. They also mentioned the usability concerns with the interaction of highlighting since it is not the most polished feature. Another offline feature that came up was the ability to use multiple PDF documents per summary, which by design is not possible right now. This should be investigated more if it actually is a beneficial feature to add. The overall quality of the summary editor was also discussed, both testers claiming it is acceptable for a minimal viable product but should be improved. All of these wishes are discussed in depth in section 10.3 with the reflection and future discussion.

9 Ethics, Data Protection and Privacy Law

9.1 Privacy

The privacy of our users is of the utmost importance to us. We ask a user for explicit consent before making use of their personal data, consisting of their username, email, (hashed) password, files, summaries, and usage statistics. The users have to consent in order to use the application. This is in fact already a universal law under GDPR (General Data Protection Regulation) and a rule that a reasonable and rational individual would follow [18]. If both parties agree then it is morally just. From a utilitarian perspective both parties benefit from this transaction. The user gets a service and we get their data and summaries. As a Kantian one would also be okay with this transaction. The categorical imperative is adhered to as one would accept the fair use of user data after consent is granted [19].

NoteSum is also responsible for the secure storage of our users information in the cloud. This ranges from the summaries they write to their email addresses. We have a duty to diligently hold this data. We do this by maintaining the integrity and confidentiality of the data. The ethical issue arises when we do not store this data securely enough and a data leak occurs, or if we would somehow sell users' data. We respect the privacy of our users by explicitly informing them of how we use their data. This can be seen in the terms and conditions which is readily available on our website. To use the NoteSum application one needs to agree to the aforementioned terms and conditions.

9.2 Copyright of Books

Another major ethical issue that arises in our project is that of pirated content. We expect our application to be mostly used to make summaries of study books. PDFs which are legally bought have, however, a different format and are not compatible with our application. Users can currently only upload either PDFs which are copyright-free or pirated. Therefore you could say that we are promoting the usage of pirated PDFs. From a utilitarian standpoint this seems acceptable. By pirating copyrighted PDFs the overall happiness or pleasure of society increases. However, in accordance with Kant this is immoral, as we would not like it if someone stole a service or product we provided. *"Act according to the maxim that you would wish all other rational people to follow, as if it were a universal law" - Kant.*
In accordance with the humanity principle we must treat the authors of these works as ends and not solely as means. This means that we should recognise them as individual beings with thoughts, feelings and desires. In this case the means would be acquiring a copy of the book. We can not solely think about getting a copy of the book , we must also make sure that we acquire that book in a just manner which compensates the authors and thus, we treat the authors as ends.

A solution, to make sure that we are not promoting the pirating of books, would be to make it possible to use our application with legally bought books. To do this we would have to build support for all major, legal, book formats. Unfortunately this is not doable in the time span and budget we have. This is also not in the scope of our minimal viable product. We have informed our client about this ethical consideration. If we look at care ethics, we can theoretically prevent users uploading pirated content. Performing this action would limit the users autonomy, however we, as the developers, could say that this is in their best interest. In reality what we do is inform the users to only upload non copyrighted material and then it is the users responsibility to uphold the terms and conditions they agreed to.

9.3 Plagiarism

Another ethical issue that arises, which is very hard to deal with, is that of users who create summaries that are very similar to the original, copyrighted text. Even though you bought a copy of the original creative work, this does not mean that you own the text. You only own the right to read the text of the copy that you bought in the context of the book. It would be unethical for one to redistribute a text as it infringes on the rights of the author under copyright law. If we look at this from a Kantian perspective we can see that in accordance with the humanity principle, we must treat the authors as ends. A user of our application is allowed to make summaries of a copyrighted text, however if the summary is too similar to the original work it can be considered plagiarism and/or copyright infringement. This is a difficult ethical issue to deal with in the context of our application, there are a number difficulties which arise. Firstly, it is hard to pinpoint at what point a summary of a given text is too similar to the original and would therefore be considered copyright infringement if redistributed. From a care ethics perspective, we as the developers could implement plagiarism detection and inform the user if there summary breaks copyright law if redistributed. This limits the user's autonomy. When one looks at this from the perspective of liberalism, one ponders to what extent is it worth infringing on the freedoms and liberties of our users. Is it a breach of privacy to analyze each users' summary? To uphold the agency, autonomy and liberty of our users one should adopt a more lax policy. In our project to deal with the issue of copyright infringement we inform the users in our terms of service that they should not perform any unlawful activity whilst using the NoteSum application. The application is a tool that increases the user's autonomy and agency. The user is then responsible to act in accordance with the law of the country they are in and their own moral principles.

9.4 Responsibility

The ethical dilemma of responsibility also arises in our design project. Who ought to be responsible when something goes wrong? This can often be a difficult dilemma to resolve. Many individuals partook in this project. Firstly, the project had to be analysed by the university to see if it was suitable, then we, the developers, created an application under the supervision of our supervisor whilst listening to the client's demands. All these individuals had a part to play in the development of the application. Thus, it is of vital importance that we clarify who is responsible and to what extent. The client is the one who is ultimately responsible for the NoteSum application. However, we as the developers still have an ex ante moral responsibility to not develop something with malicious intent. We have no ex post responsibility as that will lie with the client. Whether the client is responsible or whether the client creates a public entity which will be liable for the NoteSum application is both unpredictable and not within the scope of our ethical concern. It was made clear who was responsible for the NoteSum application by us signing the deed of copy rights assignment form. The full deed can been seen in the first appendix. It states "CLIENT indemnifies Developers against any third party claim or alleged third party claim". The client holds all rights to the source code and application. We as the developers are secure against legal liability for what we created as the NoteSum application is the property and responsibility of the client. We dealt with our ex ante responsibility by informing the client of any ethical issues we uncovered such as that of encouraging pirated content to be uploaded. We made sure to stay in accordance with GDPR by getting users explicit consent [20]. As a group we would discuss ethical concerns before implementing a feature. Thus we reduced the possibility of being ignorant to the ethical implications of our system. This helped us to act in a just way. Our motives were to help individuals study more effectively whilst upholding their right to privacy. With respect to users personal information, we do our best to store this data in a secure way. For example the usernames and passwords are hashed and salted with a cryptographic algorithm. This measure prevents one from getting direct access to user information in plain text. Unfortunately, these measures are not foolproof. We as the developers have the responsibility to implement as secure a system as possible. After we finish developing the product and hand over the source code, NoteSum then has the obligation to maintain this level of security and find the right balance between profit and our users security. Furthermore, we have a duty to inform the user that such a breach of security is possible. This gives the user enough information to make a proper decision as to whether they wish to use our service, overall increasing the autonomy and agency of our users. We did this in our Terms and Conditions.

10 Closing Remarks

10.1 Maintaining the Code

The technologies we used were highly versatile and the NoteSum application is highly customizable as each feature was built and can be modified. Frameworks such as React.js and Draft.js have large communities and are unlikely to become deprecated or obsolete any time soon. Our in-depth documentation, use of well known libraries and implementation of useful test cases are a great aid to any future developers who may part take in the future development and maintenance of the NoteSum application.

10.2 Transferring the Source Code and Migrating Server

After the end of this design project, we as the developers will no longer be working on the NoteSum application. It is important that we hand over the project to the client in an elegant and clear manner. All source code was documented and transferred to the client. Assistance was also provided to migrate servers so that the client can continue providing the NoteSum application online after the design project concludes.

10.3 Future Implementations that Could Enhance the NoteSum Application

There are many features that can enhance the usability and the stability of NoteSum. Many of these features have been discussed by us during the development of the application or in the interviews conducted with the testers. These can be roughly categorized in two parts.

10.3.1 Extensions to Existing Features

One of the major point in all user testing was the ease of highlighting. Natively, PDF documents are not made to copy paste from since spacing between characters depends more on the visual aspect of the document rather than the grammar. This means not every new line copies over a space between the trailing words if they are copied over, or if there are special characters such as dashes, the spacing is never consistent. There are many more cases of user unfriendly interaction, and these are directly transferred to NoteSum since the novelty of the application depends on copying over from PDF documents. A feature that can ease this trouble could play an intermediate role between the highlighting of the PDF and the text appearing on the summary editor. If many of the known cases are manually programmed the user interaction could be drastically improved.

Another more general improvement would be the re-design of the editor itself for a more user friendly interface and feature set. There can be small improvements such as changing the font or the font size, pagination of the summary, more styling options and some more. A good suggestion is to look at what other online text editor offer in terms of this and mimic their interface. Since users are already used to these application it could help with user retention.

10.3.2 New Features

Many of the users who have tested the application have said that they would really appreciate a sharing functionality or even a multiple user contribution functionality. This feature obviously comes with a lot of reworking of the backend and addition of new features to the front end. The reliability and the scalability concerns also increase, since simultaneous editing of an online document is a challenge on its own.

10.4 Reflection on Work Distribution

We will discuss what everyone did per person in alphabetical order. Besides these lists we shared responsibility over the different reports we had to hand in, including the design report, ethics report, and this report.

Overall work distribution was fairly easy within our group. We have the feeling that when a task arose like the peer feedback presentations that someone was quickly willing to step up and do the work. Where we could have some improvement is on how we divided tasks within our project itself. We have used the tools that GitHub offers for task creation and tracking and have divided tasks in those systems. However, reflection on how tasks went and what the current progress om those tasks was was not often done well. We feel that this could have been improved by having some fixed moments to go through all in progress tasks instead of waiting to hear from the assignee. Overall we feel like the work was distributed equally and that everyone was able to work within their strengths.

Barış İmre

- Design and implementation of the summary editor
    - Implementation of the actual editor component
    - Highlight functionality
    - Style and other functionalities of the editor
    - Downloading to other file formats
- Design and implementation of the auxiliary pages such as the landing page and other web related content
- Authentication front end
- Presenting the team's work on various activities

Yoeri Otten

- Frontend project setup.
- PDF reader.
- Project views.
- Project Redux integration.
- Various work on the frontend.
- Continues integration and deployment.
    - Setting up GitHub Actions workflows.
    - Creating Docker containers (build system).
    - Setting up server infrastructure of the project.

- Presenting the teams' work on various activities.
- Testing interviews.

Michael Mulder

- Design and implement the backend. This consists mainly of:
  - ∗ Backend setup, creating project and its Docker container.
  - ∗ Designing the database.
  - ∗ Implementing authentication through an API.
  - ∗ Implementing CORS middleware.
  - ∗ Designing how files and projects can best be stored.
  - ∗ Deciding on API endpoints and their variables.
  - ∗ Implementing all these API endpoints.
- Research and start implementation of state management through Redux.
- Research making asynchronous server calls from Redux.
- Make agenda's for our meetings.

Kevin Singpurwala

- Editor features such as hotkeys and Image addition.
- Automated testing with Selenium.
- Made presentation slides.
- Designing poster.

10.5 Reflection on What Could Have Been Improved

During the course of our project, many things went very well, however there was room for improvement. Whilst performing user testing, we anticipated to get a lot of feedback from forms. We spent time deciding on the right questions to add in the feedback form. Unfortunately none of our testers completed the form. They all opted to directly inform the client with any feedback they had. To improve on this we should have provided the feedback form in a more user friendly way so that one is coaxed in to filling out the form. This can be done by methods such as A/B testing. This would have allowed us to split our users in to two groups and we can see which layout performs best. If implemented throughout the three phases this may have increased the quality and amount of feedback we received.

During our initial implementation we though we could get away with saving states locally on users' devices and not need a backend. If we made a decision to include a backend from the beginning it would have been more efficient and would have caused less bugs. While working on the implementation our group thought about the ethical implications a lot later. For example, encouraging users to upload pirated PDFs. If we had taken these ethical concerns in to account from the beginning we may have made a better requirements list.

10.6 Personal Reflection

10.6.1 Baris Imre

Personally, I have learned a lot through this project. I had some web development experience before but making a complete application from scratch has taught me so much about the best practices in web development. I enjoyed working with my project group for the most part but I must point out the negative effects of the quarantine to our work flow. In the beginning while we sat together and worked, progress seemed much smoother and the communication between the group members was higher. In the last weeks we mostly had to self isolate which did not help with my motivation or the motivation of the whole group. Working with an actual client was something that I had never done before to this extend which also taught me a great deal of lessons for the future.

10.6.2 Yoeri Otten

This project was certainly something I needed in my professional development. Modern front-end development is a skill I have never been able to practice since I have never had the projects to work on it. I've learned a lot from the challenges which we faced during the project, skills I will certainly be able to apply in the other projects I work on and in general my professional live. Overall I enjoyed our group dynamic and the communication with the client. While I have certainly had my doubts about the project, the enthusiasm from the client and the feedback from the users certainly gave me the motivation and fulfillment which I needed. The biggest problem I've had in the last quarter was once again the uptick of the Coronavirus. I've noticed my motivation in these last weeks working from home having dropped tremendously, however this also showed me the watchful eye my project members had for our internal and external deadlines and the work they put into this project for which I'm very grateful.

10.6.3 Michael Mulder

Looking back on the past 10 weeks they have been quite interesting. I had never worked with front-end web development before this project, so there was quite a lot to learn in a short time. Besides that I also has little experience with building an API, so completely designing and building that was also a nice challenge. It is too bad that we had to deal with quarantine during this project. I think that our productivity and communication had gone a lot better if we had been able to have more regular face-to-face meetings. In the end I think it did teach us about effective online communication. There are definitely a lot of things that went less than optimal, but in the end we figured everything out together and I am quite happy with what we managed to accomplish.

10.6.4 Kevin Singpurwala

Reflecting back on the project, it was an overall success. Our project group maintained high morale and communicated well with the client. This helped us understand early on what kind of application we needed to create and what features to implement. We implemented all of the must and should requirements and provided the client with an application as intended and fit for purpose. The fact that I may be able to improve how students study and create summaries motivated me. I believe my work ethic was consistent throughout the module and that the group as a whole worked diligently on the application. Good communication was the bedrock to a successful design project and this readily manifested itself in the NoteSum application.

10.7 Conclusion

In the past 10 week we as group one designed, revised, implemented, tested, and documented NoteSum. The product we have made is far from complete or polished but we have achieved what we set out to do. We have created a minimal viable product. We have made our client happy and we have enabled him to take his startup to the next level. He now has a product which he can use to attract sponsors, partners and future employees.

Finally we thank our client and supervisor for their help, wisdom and feedback during the past ten weeks. Without whom this project would not have been possible.

References

[1] J. Walke, "React." [Online]. Available: https://reactjs.org/

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

[3] M. Mulder, K. Singpurwala, Y. Otten, and B. Imre, "Notesum summarize application," 2020. [Online]. Available: https://notesum.org/

[4] B.-Y. Tsai, S. Stobart, N. Parrington, and B. Thompson, "Iterative design and testing within the software development life cycle," *Software Quality Journal*, vol. 6, no. 4, pp. 295–310, 1997.

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

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

[7] D. Abramov and A. Clark, "Redux.js," 2015. [Online]. Available: https://redux.js.org/

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

[9] A. Gal, "Pdf.js," 2011. [Online]. Available: https://github.com/mozilla/pdf.js/

[10] "Material-ui." [Online]. Available: https://material-ui.com

[11] Google, "Material design." [Online]. Available: https://material.io

[12] T. Otwell, "Laravel," 2011. [Online]. Available: https://laravel.com/

[13] "Laravel passport." [Online]. Available: https://github.com/laravel/passport

[14] "Cors middleware for laravel." [Online]. Available: https://github.com/fruitcake/laravel-cors

[15] "Scribe." [Online]. Available: https://github.com/knuckleswtf/scribe

[16] "Redux persist." [Online]. Available: https://github.com/rt2zz/redux-persist

[17] "Docker watchtower." [Online]. Available: https://containrrr.dev/watchtower/

[18] European Parliament and Council of the European Union, "General data protection regulation," https://gdpr-info.eu/, Brussels, 2016.

[19] H. J. Paton, *The categorical imperative: A study in Kant's moral philosophy.* University of Pennsylvania Press, 1971, vol. 1023.

[20] P. Voigt and A. v. d. Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed. Springer Publishing Company, Incorporated, 2017.

Deed of copy rights assignment

The parties,

1. Name, residing at __ , born ……………. 20..,

2. Name, residing at __ , born ……………. 20..,

3. Name, residing at __ , born ……………. 20..,

4. Name, residing at __ , born ……………. 20..,

5. Name, residing at __ , born ……………. 20..,

and

6. NAME CLIENT (or his/her company possibly name b.v.), established at ………………… , hereinafter referred to as "CLIENT" ,

parties 1 to 5 inclusive, hereinafter referred to as "Developer(s)",

whereas

- Developers in assignment of CLIENT developed a software program within the framework of their study at University of Twente, hereinafter referred to as "Software program", such Software program further globally described in the annex to this deed, being an integral part thereof;

- CLIENT is interested in the Software program and wishes to acquire the copy rights on the source- and object code thereof from Developers;

- Developers are willing to assign such copy rights,

have agreed as follows:

- Developers assign free of charge all copy rights on the Software program to CLIENT;

- This assignment concerns both the source code and the object code of the Software program;

- CLIENT thus obtains the unencumbered right to use the Software program for any desired purpose;

- Developers assign the Software program "as-is", which means that any use of the Software program is for CLIENT's risk and account integrally;

- CLIENT indemnifies Developers against any third party claim or alleged third party claim, who may have suffered or suppose to have sufferered any damage based on any use of the Software program or any derivative thereof, if and in so far CLIENT has made available the Software program to such third party(ies).

Drawn up in …. fold and agreed at

(Place)…………... , ………..20xx

(name Developer)………………………….

(Place)…………... , ………..20xx

(name Developer)………………………….

(Place)…………... , ………..20xx

(name Developer)………………………….

(Place)…………... , ………..20xx

(name Developer)………………………….

(Place)…………... , ………..20xx

(name Developer)………………………….

(Place)…………... , ………..20xx

(name CLIENT)………………………….