# CHAPTER ONE INTRODUCTION

## **Background of Study**

Document collaboration is the process of using software to collaborate on projects, records, and tasks in real-time. Team members can work together from the same office or from anywhere in the world, making these tools vital to the success of distributed or remote teams (Hooker, 2022).

Document collaboration occurs within a file that multiple users can edit simultaneously. This teamwork will lead to one final document instead of multiple versions. The file can be a word document, spreadsheet, presentation, PDF, report, video, or graphic design. Document creation software is often used when teams and departments are looking to customize, edit, and share their work (Hooker, 2022).

Few documents, presentations, reports, spreadsheets, or projects are developed by just one person. At its core, document collaboration involves designated shared access to files for editing by groups of any size – be it two or 2,000. Usually, a document initiator or owner manages revisions, but today's collaboration tools can simultaneously incorporate all contributors’ edits and still maintain the integrity of prior revisions. Many platforms act as a central repository or holding area for the work of a group and can incorporate elements such as task and assignment lists, calendars, agendas, and feedback (Hooker, 2022).

Document collaboration goals will vary for each organization, so before you choose a tool, define your needs. For example, a group building a social media campaign may require video, graphs, and pictures as part of brainstorming or mind mapping. By contrast, a technical build for software values document integrity through controlled revisions, so security and access may be more important. Consider your company’s functions and needs when adopting a document collaboration solution (Hooker, 2022).

Document collaboration software allows teams to collaborate remotely on the same document in real-time. This document could be a text file, spreadsheet or PDF (Hooker, 2022).

Multiple team members can simultaneously edit, update and comment on a document, making the document a true team effort. The right document collaboration software prevents version conflicts and is able to handle multiple, real-time edits (Hooker, 2022).

Document collaboration software eliminates the need to save multiple versions of the same file, as all version history is stored in the editor. Users can revert back to a previous version at any time (Hooker, 2022).

## **Statement of the Problem**

Students find it hard to share documents during collaborative work & assignments or for lecturers to share assessment documents for same purpose. Resulting in fragmented work during the assignment and project which makes the lecturers work hard when supervising the project for assessment.

## **Aim and Objectives of the Study**

The goal is to develop an app for keeping the users logged in, creating new documents, viewing the list of documents created by user, updating the title of the document, sharing link of the document, adding a rich text editor, collaboratively editing with as many users as you want and creating responsive design.

The objectives are:

1. An engaging and easy to use UI with good UX using Flutter.
2. The logic will be handled using Node.js and the data will be stored using Firebase.
3. Evaluating and verifying the app.

## **Scope of the Project**

The area covered in the project focus on developing an editing, cloud collaborative document.

## **Limitation of the Study**

The platform is limited to document only based on google doc design.

## **Significance of the Study**

The platform or web app stands to benefit the students to share and edit documents during group assignments and projects.

## **Project Organization**

The project is divided into five chapters. The outlines are presented below:

**Chapter One: Introduction**

Chapter one introduces this project work, the background of the study, the statement of the problem, the aim and objectives, the scope of the study, limitations of the study, the significance of the study, project organization, and the definition of terms.

**Chapter Two: Literature review**

This chapter focuses on the literature review, and the contributions of other scholars on the subject matter being discussed.

**Chapter Three: Methodology and Design**

This chapter is concerned with the presentation of the results of system analysis and design. It presents the research methodology used in the development of the system to facilitate an understanding and effective future implementation of the system.

**Chapter Four: System Implementation Evaluation**

This chapter describes the system implementation and documentation, analysis of modules, and system requirements for implementation.

**Chapter Five: Summary, Conclusion, and** **Recommendation**

The chapter provides a summary of major findings, conclusions, and recommendations based on the study conducted.

# CHAPTER TWO LITERATURE REVIEW



## **Introduction**

A look at what the programming language and database used the project are and a review of related literature.

## **What Is Document Collaboration?**

One of the first real-time collaborative editor tools can be found in “The Mother of All Demos,” presented by Douglas Engelbart in 1968. The next major milestone for document collaboration appeared in 1991 with the release of Instant Update for the Mac OS and Microsoft Windows.

Since then, document collaboration has grown to cover tools that enable real-time access and updates for concurrent work across the room or around the world. This practice (of multiple people working simultaneously on a single document) is called co-authoring. Modern tools mitigate the challenges of non-real-time and unautomated collaboration, such as paper trails and red-lining, unwieldy email chains, or overwriting original material.

## **What Is a Collaborative Document?**

A collaborative document is a file that multiple people edit or contribute to, with the goal of creating single final version through collaboration. Collaborative documents may be word processing or text documents, spreadsheets, presentations, images, PDFs, videos, or maps.

### **Common Uses of Document Collaboration**

Few documents, presentations, reports, spreadsheets, or projects are developed by just one person. At its core, document collaboration involves designated shared access to files for editing by groups of any size be it two or 2,000. Usually, a document initiator or owner manages revisions, but today's collaboration tools can simultaneously incorporate all contributors’ edits and still maintain the integrity of prior revisions. Many platforms act as a central repository or holding area for the work of a group and can incorporate elements such as task and assignment lists, calendars, agendas, and feedback.

Document collaboration goals will vary for each organization, so before you choose a tool, define your needs. For example, a group building a social media campaign may require video, graphs, and pictures as part of brainstorming or mind mapping. By contrast, a technical build for software values document integrity through controlled revisions, so security and access may be more important. Consider your company’s functions and needs when adopting a document collaboration solution.

### **Top Document Collaboration Tools**

Document collaboration tools, like any other widely adopted software platform, are always growing and improving. Here are the most popular solutions:

1. Google
2. Microsoft Office 365
3. Quip
4. Dropbox Paper

### **Document Collaboration for Developers**

Software developers depend on accurate version control, and they know both the benefits and the perils of document collaboration. When choosing a version control platform, development teams should value accessibility and sharing. At the same time, look for systems that enable simultaneous editing but deter overwriting. Tools and solutions for development collaboration are available through IBM, Microsoft, Subversion, and Helix.

Another popular platform choice is a distributed version-control (DVC) system, which allows numerous working copies to be reconciled with the main document. Tools from Git and Mercurial provide some of the best open-source solutions for DVC.

### **Benefits of Document Collaboration**

Many organizations have adapted to an activity-based working (ABW) environment, which takes the focus off individual effort and instead values shared ownership of work. This involves effective collaboration, including the right choice of document collaboration tools.

Teams increase engagement through instant communication and resolution. Because everyone works in the same space on the same project or document with minimum barriers, cooperation improves.

But one of the most elemental benefits of focused collaboration is the ability to increase productivity. With the appropriate tools and administration, time-consuming efforts such as manually reconciling long email chains and complex revision control are automated and streamlined to synchronize, identify, and resolve changes in real time. This allows a team to immediately know they are viewing the latest version of a document, which saves time and limits confusion.

Most collaboration tools have instant communication methods to alert the team when work is taking place and invites them to review it. The tools can be used for both structured activities and brainstorming; they also provide a central hub and historical record of everything done to date and the project’s progress in accordance with a goal or timeline. And since digital document sharing can be done anywhere and at any time, teams have more flexibility to contribute.

## **The Evolution of Collaborative Document Editing**

Professionals have been editing text documents collaboratively since companies have existed. The prevalent examples of such documents are terms and conditions, company policy, and marketing messages. Before the internet, collaborative document editing happened in person.

### **In-person collaborative editing**

Various members of the documentation team gathered in one room with a whiteboard to discuss strategies to create and refine content. Although many content teams still follow this process, it was the norm when documents were created on paper.

This method of collaborative real-time document editing offered an advantage that none of the other methods described below can in-person interactions during content creation. This kind of collaboration leads to the accurate exchange of information without any ambiguity.

The disadvantages of in-person document editing and collaboration are that you can only work with the ones around you and for a limited amount of time as everyone needs to be available simultaneously for it to happen.

The review cycles with in-person collaborative editing of documents are usually longer because setting meetings, gathering everyone together, and clarifying issues in a room are tedious. Teams across all domains have to spend a lot of their resources to keep this process running.

### **Collaborative Editing Through Traditional Tools: Email and Video Calls**

As better communication tools became available for content teams, collaborative document editing became both asynchronous and distributed. Emails and video calls changed the way team members communicate and are still used among external teams to share information.

Emails allow multiple collaborators, even those from external teams, to share ideas, suggestions, and feedback while creating documents. Collaborators can tend to the requests on their own time, facilitating asynchronous collaborative document editing.

Video calls allow synchronous document editing and collaboration which has the advantages of in-person communication. This also allows content team members across the globe to work on documents at the same time.

The efficiency of this method, however, depends on the availability of a collaborative real-time editor (such as Google Docs) for document production. Without such a tool, the collaborative document editing version control will be complicated which will make content management difficult.

Even with a collaborative real-time editor, it can be difficult for content teams to find relevant information quickly in long email chains. Also, scheduling and sitting through video calls can be frustrating while working with a distributed team.

The final disadvantage that collaborators face here is dealing with unintended information. No one wants to be a part of an email chain or a video call where they don’t need to be.

### Third-Party Content Editing Tools: Google Docs, Microsoft Word, Etc.

Real-time collaborative editing software such as Google Docs and Microsoft Word have transformed content production workflows. Now, team members don’t need to exchange emails or sit through long video calls to collaborate on documents.

Tagging, commenting, suggesting, etc., are some of the most used functionalities of these online document editing collaboration tools. All the disadvantages of in-person collaboration, email chains, and video calls are taken care of by these tools.

However, there are still some challenges that affect the document production process of the teams that rely on these tools. WordPress content creators, for instance, have to move content from these tools to the Gutenberg Editor during the editorial process. The back and forth of content usually continues until the content is approved for publication.

Another difficulty is that these third-party collaborative document editors cannot be used to review certain types of content like videos, image carousels, CTA blocks (dynamic content), etc. This forces content creators to rely on emails and video calls to review them. Although collaborating with external teams is simple with tools like Google Docs, it also means giving those members unrestricted access to your organizational data. This can cause issues if the permissions are not managed properly through each stage of document creation.

## **Two Types of Collaborative Editing**

There are two types of collaborative editing in project management:

1. Task
2. Document

Task collaborative editing is when project tasks are assigned to multiple team members at once, allowing them to work together collectively.

For example, imagine you have ad campaign copy that’s ready for review. In a larger organization, you may have one reviewer checking the content, another approving the visuals, and a third ensuring there are no legal concerns.

Thanks to task collaborative editing, they can all be assigned to the work at the same time. This means if one of them requests a change, the others can see it and determine if that will impact their review and approval.

Document collaborative editing is when multiple people are editing the same document simultaneously. This is similar to task editing except it’s solely in relation to documents and files attached to project tasks and excludes any other task work.

Document collaborative editing can be useful for several different scenarios:

1. If you have a long document such as an eBook and you want the editor to be working on older chapters or pages while the writer is still creating more content
2. If you believe there will be multiple revisions and back and forth required between the writer and editor
3. If you have multiple editors who need to review the content at the same time

## **Benefits of Collaborative Editing Within Project Management Software**

When you have collaborative editing functionality within your project management software, it provides many benefits for you, your team, and other stakeholders.

You can solve conflicting feedback in real-time, work toward the best ideas quickly, and save time consolidating feedback across email, spreadsheets, etc.

Instead of keeping multiple versions of a document in different places, you get a single master document or task that everybody can access. Everyone on the team can view, edit, and update information easily at any time. Your team members can discuss requirements, elaborate on product design, create a conference agenda, and find solutions to problems.

It minimizes the necessity of meetings and calls to update a piece of work. It also saves team members from sending emails with revised documents back and forth or comparing versions.

Collaborative editing, revision history, email notifications about changes, and links between pages all work together to ensure everyone who needs to be kept in the loop is updated about changes to tasks and documents on your project.

Any changes are instantly visible to all of the team members online and via email and RSS. This allows everyone to stay up-to-date and quickly react to the changes when they need to.

While collaborative editing lets many people access and change a document or task at once, your project software provides version control, so that no original thought is lost or overwritten in error.

Compared to traditional project management tools that place the project manager in the center of communications, systems with collaborative editing allow everyone on the team to immediately see the latest updates and contribute to the project plan in real-time. This streamlines communications and increases the productivity of your team.

Meanwhile, the ability to restore tasks to their previous version at any time allows the project manager to keep control. Your system retains the information about what changes were made, when they were made, and by whom. Plus, it enables you to review, compare pairs of revisions, and restore tasks to previous versions if needed.

Change-tracking features allow you to see exactly what was added, edited, or removed by each contributor.

Teams can take advantage of document collaboration tools to share ideas, develop plans, and pass along essential data. This approach makes it simple to provide immediate access to drafts and updated information. Conducting real-time collaboration online allows each team member to develop concepts and proposals individually, instantly share with the group, and immediately receive feedback.

Collaborative writing in a document or spreadsheet can become one of several means of communication for team members, along with phone, email, and instant messaging.

Team collaboration tools can drive expanded engagement in several ways:

1. Collaborative editing promotes transparency as employees in different departments can see each other's drafts.
2. Project management software with collaboration features can create new communication opportunities, breaking down silos that damage efficiency.
3. Employees have more opportunities to share their expertise and build trust rather than competing with one another.

## **Advantages of collaborative editing**

Collaborative editing tools provide a lot of benefits, including:

1. **Improved efficiency:** These tools streamline your content creation process by allowing everyone to work on the same version of a document at the same time.
2. **Better collaboration:** The shared space provided by these real-time collaboration tools provides a shared space for communication, feedback, and the sharing of ideas, even asynchronously.
3. **Better quality control:** No need to worry about teammates working on an outdated version of a document. Collaborative editing ensures everyone is working on the most updated version, which reduces errors.
4. **Increased flexibility:** Because collaborative editing tools tend to use the cloud, they’re available anywhere. This is especially ideal for remote workers who may need to access to a document from a different location or time zone.

## **Challenges with collaborative editing**

Of course, no solution is perfect. Collaborative editing may also pose some challenges for your team, and it’s best to be aware of those ahead of time. Some potential drawbacks include:

1. **Security concerns:** Ensure your collaborative editing tool uses two-factor authentication (2FA) for logins and encrypts your documents, especially if your team works with sensitive information. Some tools may also allow you to set permissions and limit document access to specific people.
2. **Training:** New tools require new training. Thankfully, most collaborative editing tools mimic the word processing tools most of us are used to working with, like Microsoft Word and Google Docs. So the learning curve is likely not very steep.
3. **Organization:** While collaborative editing allows everyone to work on the same version of a document, it’s still possible for older versions to get saved. Make it a habit to clean up your document library to keep outdated content in the archives. Better yet, train your team on how to help you keep the content hub organized.

## **Programming Languages Used**

### **Flutter & Dart**

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for iOS and Android) (Thomas, 2021).

Flutter consists of two important parts (Thomas, 2021):

1. An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
2. A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

To develop with Flutter, you will use a programming language called Dart. The language was created by Google in October 2011, but it has improved a lot over these past years (Thomas, 2021).

Dart focuses on front-end development, and you can use it to create mobile and web applications (Thomas, 2021). If you know a bit of programming, Dart is a typed object programming language. You can compare Dart's syntax to JavaScript (Thomas, 2021).

#### **Simple to learn and use**

Flutter is a modern framework, and you can feel it! It’s way simpler to create mobile applications with it. If you have used Java, Swift, or React Native, you'll notice how Flutter is different (Thomas, 2021).

#### **Quick compilation: maximum productivity**

Thanks to Flutter, you can change your code and see the results in real-time. It’s called Hot-Reload. It only takes a short amount of time after you save to update the application itself.

Significant modifications force you to reload the app. But if you do work like design, for example, and change the size of an element, it’s in real-time! (Thomas, 2021).

#### **Ideal for start-up MVPs**

If you want to show your product to investors as soon as possible, Flutter is a good choice (Thomas, 2021).

Here are some of the top 4 reasons to use it for MVP (Thomas, 2021):

1. It’s cheaper to develop a mobile application with Flutter because you don’t need to create and maintain two mobile apps (one for iOS and one for Android).
2. One developer is all you need to create your MVP.
3. It’s performant – you won't notice the difference between a native application and a Flutter app.
4. It’s beautiful – you can easily use widgets provided by Flutter and personalize it to create a valuable UI for your customers (you can find examples of applications made with Flutter below).

### **Node.js**

Node.js is an open-source, cross-platform JavaScript runtime environment and library for running web applications outside the client's browser. Ryan Dahl developed it in 2009, and its latest iteration, version 15.14, was released in April 2021. Developers use Node.js to create server-side web applications, and it is perfect for data-intensive applications since it uses an asynchronous, event-driven model (Sufiyan, 2022).

#### **Why Do We Use NodeJs?**

There are many reasons for which we prefer using NodeJs for the server side of our application, some of them are discussed in the following (Sufiyan, 2022):

1. NodeJs is built on Google Chrome’s V8 engine, and for this reason its execution time is very fast and it runs very quickly.
2. There are more than 50,000 bundles available in the Node Package Manager and for that reason developers can import any of the packages any time according to their needed functionality for which a lot of time is saved.
3. As NodeJs do not need to wait for an API to return data , so for building real time and data intensive web applications, it is very useful. It is totally asynchronous in nature that means it is totally non-blocking.
4. The loading time for an audio or video is reduced by NodeJs because there is better synchronization of the code between the client and server for having the same code base.
5. As NodeJs is open-source and it is nothing but a JavaScript framework, so for the developers who are already used to JavaScript, for them starting developing their projects with NodeJs is very easy.

#### **Features of NodeJs**

1. Asynchronous in Nature and Event driven: The servers made with the NodeJs never waits for the from an API. Without waiting for the data from the API, it directly moves to the next API. So all the APIs of NodeJS are totally non-blocking in nature. In order to receive and track all the responses of the previous API requests, it follows an event driven mechanism. Hence we can say that all the NodeJs API are non-blocking in nature (Sufiyan, 2022).
2. Single Threaded Architecture: With event looping, a single threaded architecture is followed by NodeJs and for this architecture makes NodeJs more scalable. In contrast to other servers, limited threads are created by them for processing the requests. Whereas for the event driven mechanism, the NodeJS servers reply in a non-blocking or an asynchronous manner and for this reason NodeJS becomes more scalable. If we compare NodeJs with other traditional servers like Apache HTTP servers, then we can say NodeJs handles a larger number of requests. A single threaded program is followed by NodeJS and this allows NodeJs to process a huge amount of requests (Sufiyan, 2022).
3. Scalable: Nowadays, scalable software is demanded by most of the companies. One of the most pressing concerns in Software Development is addressed by NodeJs and that is scalability. Concurrent requests can be handled very efficiently using NodeJs. A cluster module is used by NodeJs for managing the load balancing for all the active CPU cores. The most appealing feature of NodeJs is that it can partition the applications horizontally and this partition procedure is mainly achieved by it due to the use of child processes. Using this feature, the distinct app versions are provided to the different target audiences and also for customization it allows them for catering to the client preferences (Sufiyan, 2022).
4. Quick Execution time for code: V8 JavaScript runtime motor is used by NodeJs and this is also used by Google chrome. A wrapper is provided for the JavaScript by the hub and for that reason the runtime motor becomes faster and for this reason inside NodeJs, the preposition process of the requests also become faster (Sufiyan, 2022).
5. Compatibility on the cross platforms: Different types of systems like Windows, UNIX, LINUX, MacOS and other mobile devices can use NodeJs. For generating a self-sufficient execution, it can be paired with any appropriate package (Sufiyan, 2022).
6. Uses JavaScript: From an engineer's perspective, it is a very important aspect of NodeJs that this framework uses JavaScript Most of the developers are familiar with JavaScript, so for them it becomes very easier to grab NodeJs (Sufiyan, 2022).
7. Fast Data Streaming: The processing time of the data that have been transmitted to different streams takes a long time. Whereas for processing the data, NodeJs takes a very short amount of time and it does it at a very fast rate. NodeJs saves a lot of time because the files are processed and uploaded simultaneously by NodeJs. So as a result, the overall speed of data and video streaming is improved by NodeJs (Sufiyan, 2022).
8. No Buffering: The data is never buffered in NodeJs application (Sufiyan, 2022).

#### **Node.js Architecture**

Now that we established what is Node, let’s dig into its architecture. Node.js operates on a single-thread, allowing it to handle thousands of simultaneous event loops. Here’s a diagram, provided by Sinform.com, that best illustrates Node.js architecture (Sufiyan, 2022).

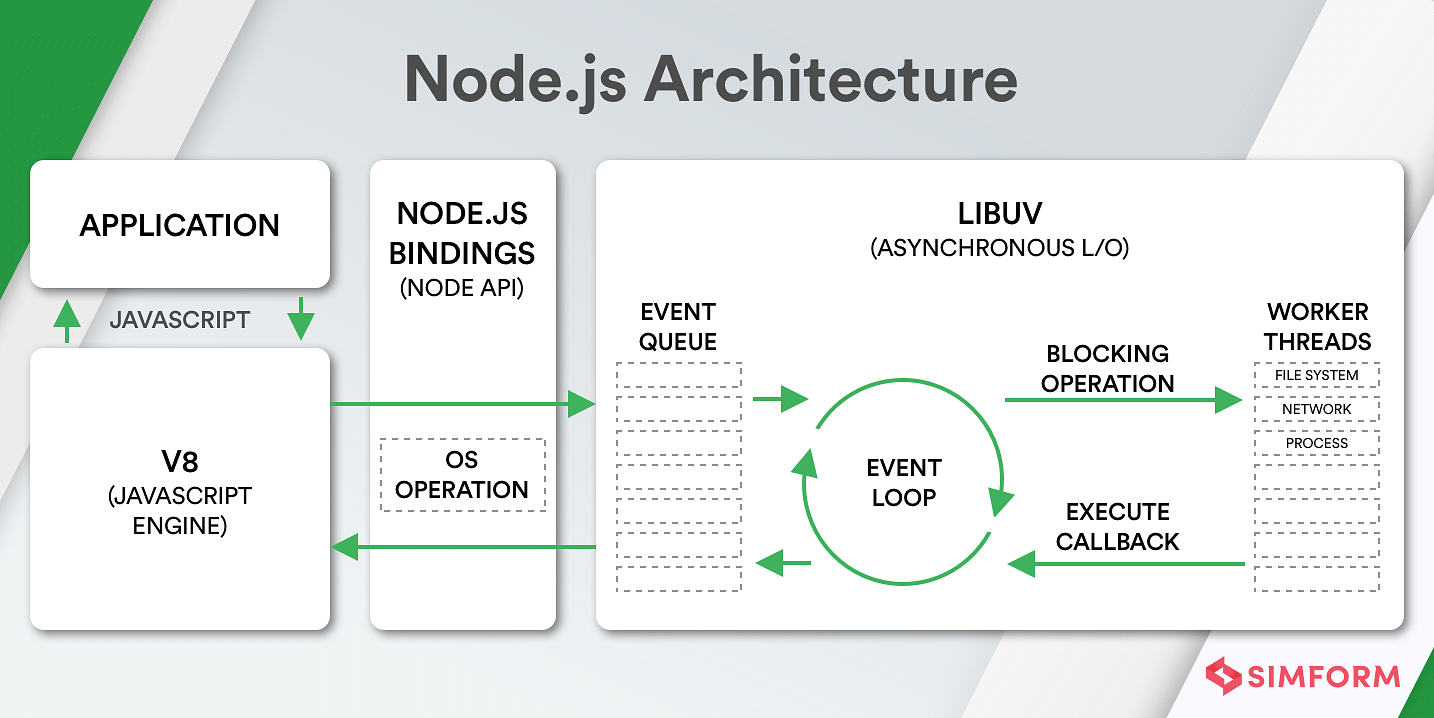


Figure 2.1: Architecture of Node.js (Sufiyan, 2022)

#### **Parts of Node.js**

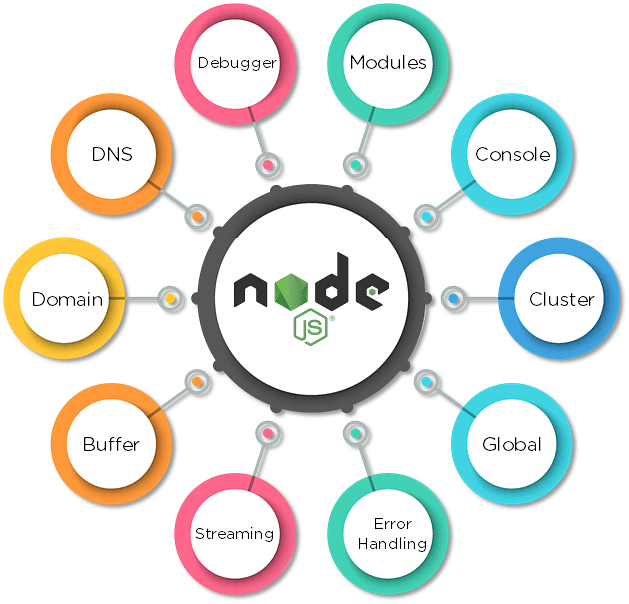


Figure 2.2: Parts of Node.js (Sufiyan, 2022)

### **Firebase**

What is Google Firebase? It is a mobile application development platform from Google with powerful features for developing, handling, and enhancing applications. Firebase is a backend platform for building web and mobile applications (Batschinski, 2022).

Firebase is fundamentally a collection of tools developers can rely on, creating applications and expanding them based on demand (Batschinski, 2022).

Firebase aims to solve three main problems for developers (Batschinski, 2022):

1. Build an app, fast
2. Release and monitor an app with confidence
3. Engage users,

Developers relying on this platform get access to services that they would have to develop themselves, and it enables them to lay focus on delivering robust application experiences (Batschinski, 2022).

Some of the Google Firebase platform’s standout features include databases, authentication, push messages, analytics, file storage, and much more (Batschinski, 2022).

Since the services are cloud-hosted, developers can smoothly perform on-demand scaling without any hassle. Firebase is currently among the top app development platforms relied upon by developers across the globe (Batschinski, 2022).

## **Related Literature**

### **Significance of Document Collaboration in Business**

#### **Track your team’s workflow**

When you use a document collaboration tool, it makes it easy to keep track of what everyone in your team is working on. You can view all the files that are currently being edited and ensure that deadlines are being met (Hooker, 2022).

#### **Improve productivity**

When employees have the right software to share their files and collaboratively edit, their productivity is improved. Less time is wasted hunting around for the right document and all files are stored right there in the cloud. Since more than one person can edit a file at a time, this speeds up the process of collaborating on a document and prevents version conflicts (Hooker, 2022).

#### **Empower employees with information**

Employees have access to the right information with a document collaboration tool at their fingertips. Your important documents are all stored in one place and a quick search is all it takes to find the right file (Hooker, 2022).

#### **Provide faster feedback and quick decision making**

When information is readily available to employees, they can gain feedback more quickly and make more rapid-fire decisions. All employees need to do is share a link to a document and they can collect feedback in real-time. There’s no more messing around with emails and multiple versions of a document (Hooker, 2022).

#### **Open communication with teams**

Document collaboration tools facilitate open communication within teams. It can be easier to share your thoughts in a document rather than wasting time in long meetings, and documents can be shared with the rest of the team to encourage their input (Hooker, 2022).

#### **Encourage employee involvement**

Employees will get more involved in their teams if they have access to the right document collaboration tool. The best tools on the market don’t have any barrier to entry and employees can quickly get up to speed with how the software works. It’s easy for anyone to share feedback on a document and make updates (Hooker, 2022).

#### **Supports remote working easily**

With a document collaboration tool, employees don’t need to be in the same physical location to collaborate on a project. The software can usually be accessed on any device and all you need is an internet connection to collaborate on files (Hooker, 2022).

#### **Eliminate organizational silos**

When you store your documents in a document collaboration tool, you’re making your company knowledge explicit. Teams will work better together as they have insight into what other teams are working on (Hooker, 2022).

### **Features To Look for In Any Document Collaboration Tool**

#### **Real-time editor**

You need an editor that allows multiple users to work on a document in real-time. The document is “live” and users can make edits and comments that are instantly reflected in the document (Hooker, 2022).

#### **Autosave**

Your work should automatically save as you are making changes to the document, making sure that you don’t lose important information. There is no need to manually save the document (Hooker, 2022).

#### **Versioning**

You need the ability to save different versions of a document and to revert back to a previous version if necessary. This ensures that you don’t lose access to your existing work even if new changes are made to the document (Hooker, 2022).

#### **Commenting and feedback options**

Comments are a way to leave feedback, ask questions and assign tasks without actually changing the document. It’s an important collaboration feature (Hooker, 2022).

#### **Cloud Storage and sharing**

Document collaboration requires a large number of people and documents to work together so they need a large cloud storage space in which to save all their work (Hooker, 2022).

#### **Tag a team member**

You need the ability to tag a team member and assign them a task in the document collaboration tool. The tag should send them a notification and get them to check the software to see what they need to do (Hooker, 2022).

#### **Ability to organize files and add tags**

Your document collaboration software should enable you to organize your files within a workspace and add tags to enhance discoverability of content (Hooker, 2022).

#### **Tracking changes and notification**

You should be able to track the changes that have been made to a document and receive a notification to alert you to the change (Hooker, 2022).

#### **Integrations and extensions**

Your collaboration tool is more powerful when it can integrate with other apps your team uses such as project management apps or team communication tools (Hooker, 2022).

In order to solve the issue of distance as a trade off to a project’s development and also the issue of effective file management, (Hooker, 2022) has design and developed a cloud-based collaboration and file management system targeted at mobile device end users.

Al-Samarraie & Saeed (2018) a review was conducted to identify studies on cloud computing tools for collaborative learning in a blended classroom. The review of the literature led to the inclusion of 29 relevant studies categorized as synchronized tools, Learning Management System (LMS) tools, and social networking tools. The review results revealed a set of evidences supporting the use of certain cloud computing tools for certain collaborative learning activities categorized under sharing, editing, communication and discussion.

Harjanto & Sumarni (2019) study reveals the teacher’s perception on the use of Google Classroom as a learning media. The qualitative research design which applies interview method is carried out. Seven high school teachers in Tangerang Selatan, Indonesia, who have actively integrated Google Classroom in their teaching for at least 1 year became the participants. Findings showed that teachers use it as a facilitation tool for managing students’ tasks, organizing classroom and accommodating students’ interaction. In general, teachers perceived the use of Google Classroom was very helpful to conduct their virtual classroom.

Ko et al. (2022) presented a framework, development, and application processes of a “cloud-based collaborative MES System” to support the value chain of “order-design-production-delivery” for the manufacture of personalized sportswear products in the fashion industry in Korea. To this end, first, nine future MES deployment directions and frameworks are presented. Second, it presents the UML modeling, conceptual framework, and functional framework for MES system development, considering six future MES establishment directions such as cloud and collaboration. Third, the application and effect of the designed and developed cloud-based collaborative MES system are analyzed for design, fabric, printing, and sewing companies that play a role in each stage of the sportswear value chain.

González‐Lloret (2020) article advocates that, through collaborative technology-mediated tasks, we can promote productive language output (spoken and written) and the type of interaction that facilitates language learning and motivates students to continue improving their language skills.

In a study by (Lee et al., 2022) with 11 blind participants, demonstrated that CollabAlly provides improved access to collaboration awareness by centralizing scattered information, sonifying visual information, and simplifying complex operations.

Arora & Atrey (2021), explored the use of secret sharing to maintain the confidentiality of user data in a collaborative document. Established that using secret sharing provides an average increase of 56.01% in performance over AES with a single set of coefficients and an average performance increase of 30.37% with multiple sets of coefficients, while not requiring maintenance and distribution of symmetric keys as in the case of AES. Discussed the incorporation of keyword-based search with the proposed framework and present the operability and security analysis.

Rahman et al. (2018) formalism considers two main collaboration-related factors—affinity and upper critical mass—appropriately adapted from organizational science and social theories. First, they formalize the notion of collaboration among crowd workers and propose a comprehensive optimization model for task assignment in a collaborative crowdsourcing environment. While they study the hardness of the task assignment optimization problem and proposed a series of efficient exact and approximation algorithms with provable theoretical guarantees. Finally, they presented a detailed set of experimental results stemming from two real-world collaborative crowdsourcing application using Amazon Mechanical Turk.

# CHAPTER THREE METHODOLOGY AND DESIGN



## **Introduction**

This chapter covers the analysis models (data flow diagram, sequence, class and entity relationship diagram) of the project and the higher-level solution (programming language) approach used.

## **Analysis Models**

Analysis models is a technical representation of the system. It acts as a link between the system description and the design model. In Analysis Modelling, information, behavior, and functions of the system are defined and translated into the architecture, component, and interface level design in the design modelling.

### **Use Case Diagram**

Use-case diagrams **describe the high-level functions and scope of a system**. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

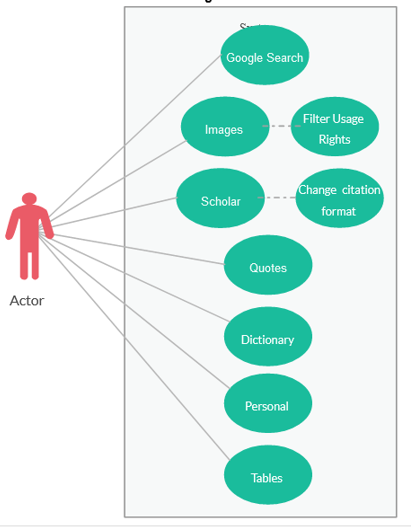


Figure 3.1: Use Case Diagram

### **Flow Chart Diagram**

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence. They can range from simple, hand-drawn charts to comprehensive computer-drawn diagrams depicting multiple steps and routes.

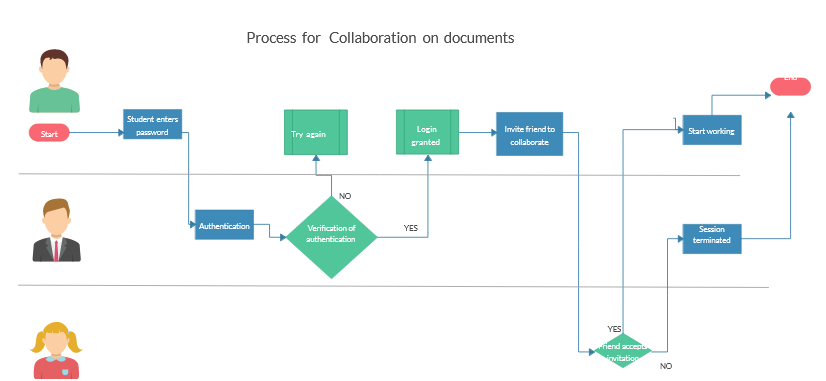


Figure 3.1: Flow Chart Diagram

### **Sequence Diagram**

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration.

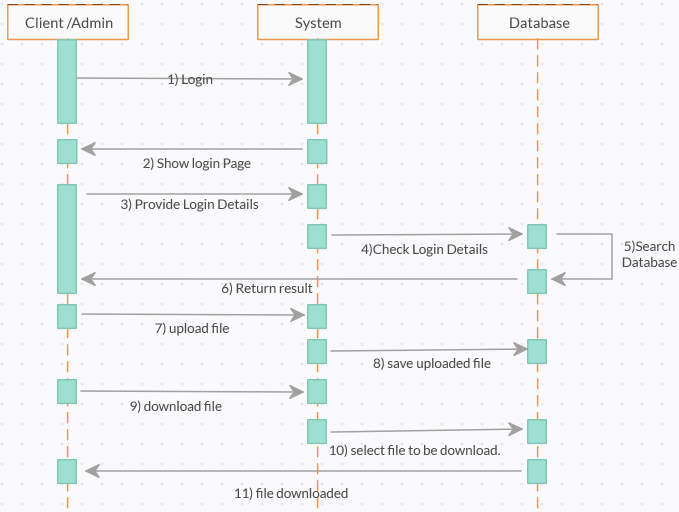


Figure 3.3: Sequence Diagram

### **Architecture Diagram**

Class diagram is basically **a graphical representation of the static view of the system and represents different aspects of the application**. A collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system.

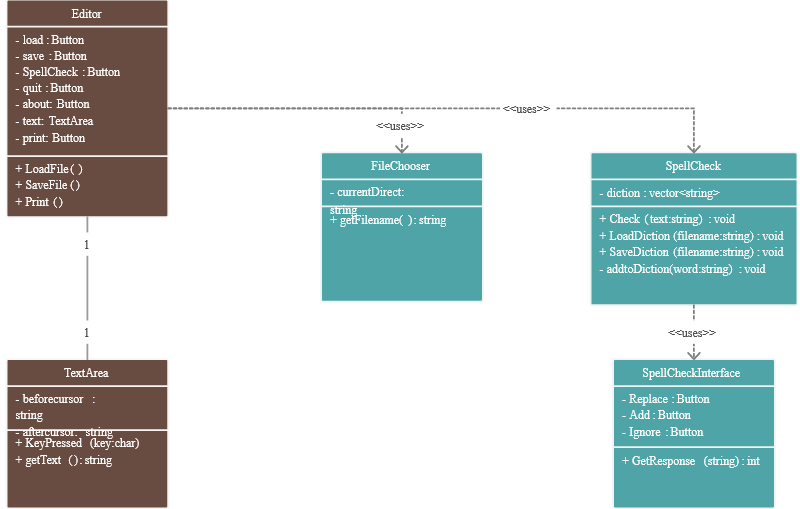


Figure 3.4: Class Diagram

## **Programming languages used**

1. Flutter
2. Node.js

**CHAPTER FOUR**

**SYSTEM IMPLEMENTATION EVALUATION**

**4.1 Introduction**

This section provides a comprehensive explanation of the implementation process for the new system, highlighting its efficiency and effectiveness. It presents practical instances of the functional aspects of the system and outlines the steps involved in its implementation.

* 1. **System Testing and Evaluation**

Testing the developed system is crucial for several reasons. One key purpose is to uncover any potential flaws within the system and devise appropriate solutions. In this project, a combination of unit and integration testing was employed to verify the effectiveness and efficiency of the design, ensuring that the new system fulfills its functional requirements without any errors.

**Unit Testing**

This part examines specific units or single components of the system individually to confirm that specific phases function properly and without problems.

**Integration Testing**

Integration testing was performed on the software, wherein all components were brought together and operated as a unified system. The objective of this testing was to validate the connectivity and proper integration of the various parts, ensuring seamless collaboration among the units.

**4.3 System Installation**

In order to use the proposed application on any computer system, the following steps need to be taken:

1. Make sure node.js, JDK, and chrome are installed on the system.
2. Copy your project folder to any location of your choice.
3. Open project folder in Visual Studio Code
4. run “flutter pub get” to get all the dependencies in the pubspec.yaml file
5. Select the Chrome browser as the device to be used”
6. Locate the main.dart file and run the file in debug mode

**4.4 Security Measures**

The application has a public scope, allowing all users to access the available information. However, certain functionalities are restricted to authenticated users, for example, the admin can create notifications, create and manage the lecturer’s account, etc. Access to these restricted functionalities is protected by passwords, ensuring that only authorized individuals can access the admin pages. Additionally, certain functionalities within the application may be restricted based on the specific user type, providing tailored access and permissions as needed.

**4.5 Sample Outputs**

These describe and give the pictorial representation of the program or software; it shows and gives a clear understanding of the design, and displays all the interfaces.

**Landing Page**

The image provided illustrates the landing page, which serves as the initial page and serves as a gateway to navigate and explore the various sections and functionalities of the pad

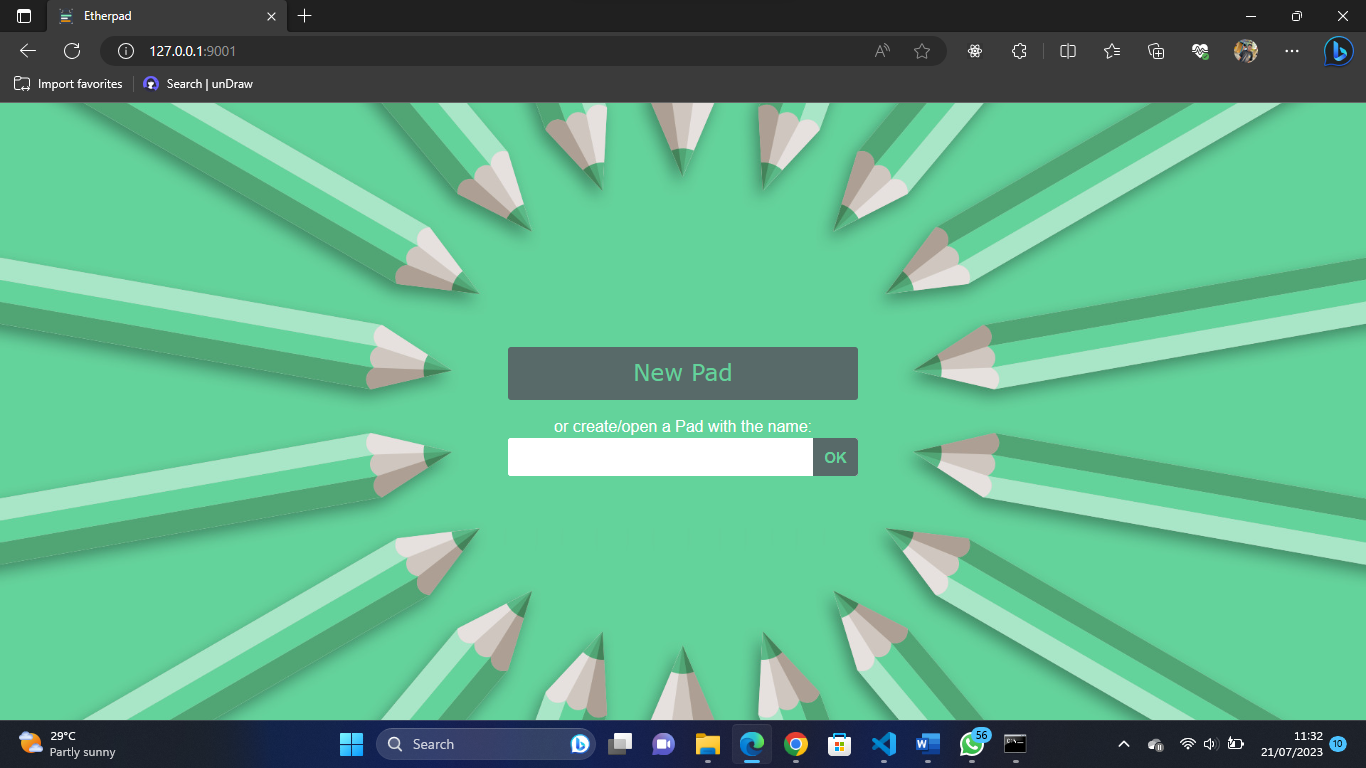


Fig 4.1 Landing Page

**Creating New Pad**

This page helps users to create a new pad for collaboration

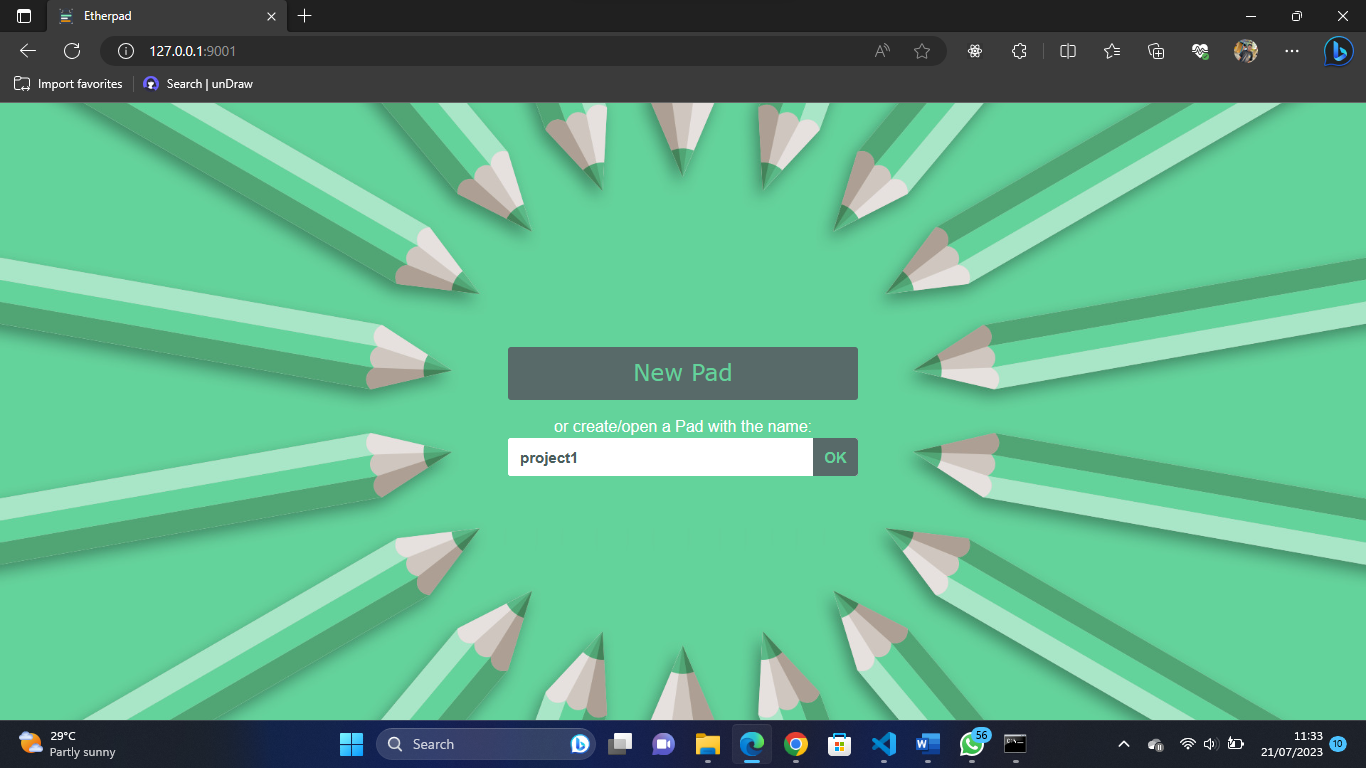


Fig 4.2 Creating New Pad

**Multiple User View**

Multiple users can collaborate on a single pad asynchronously

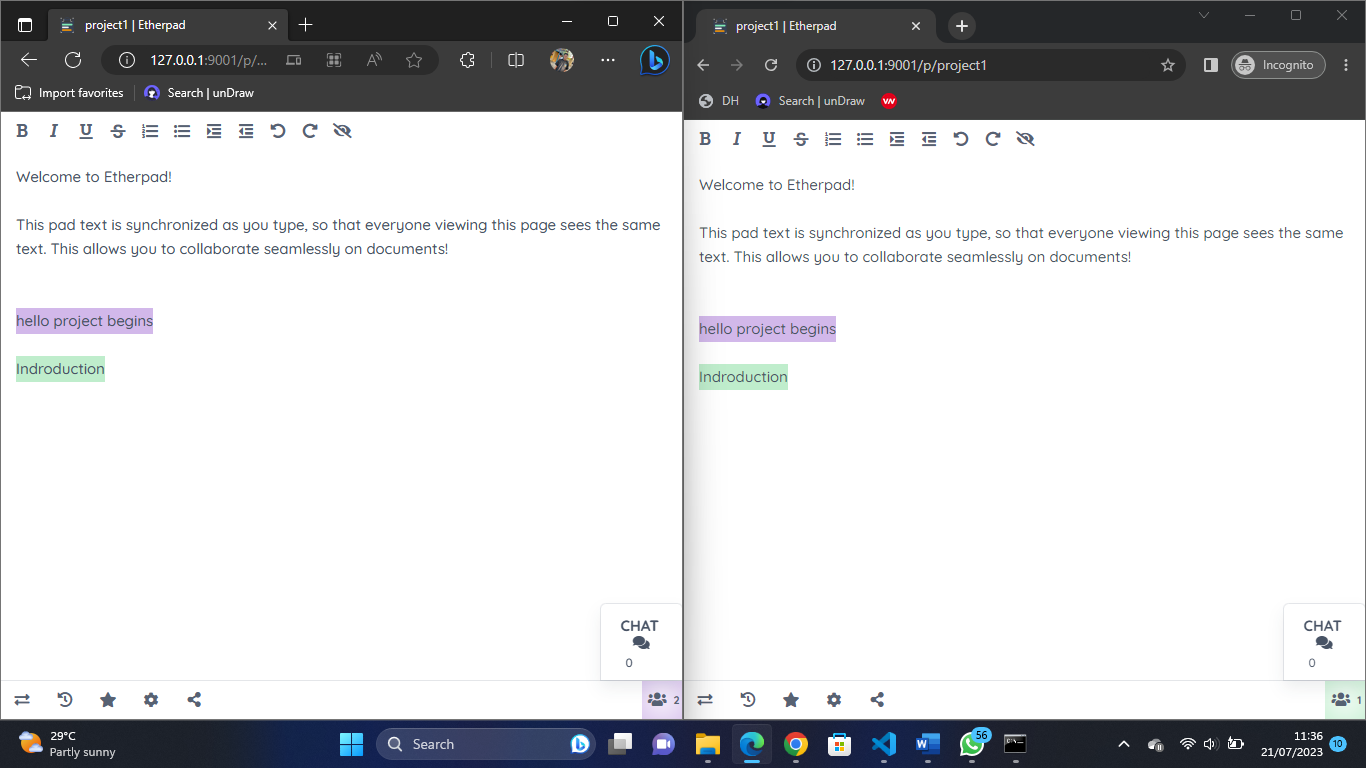
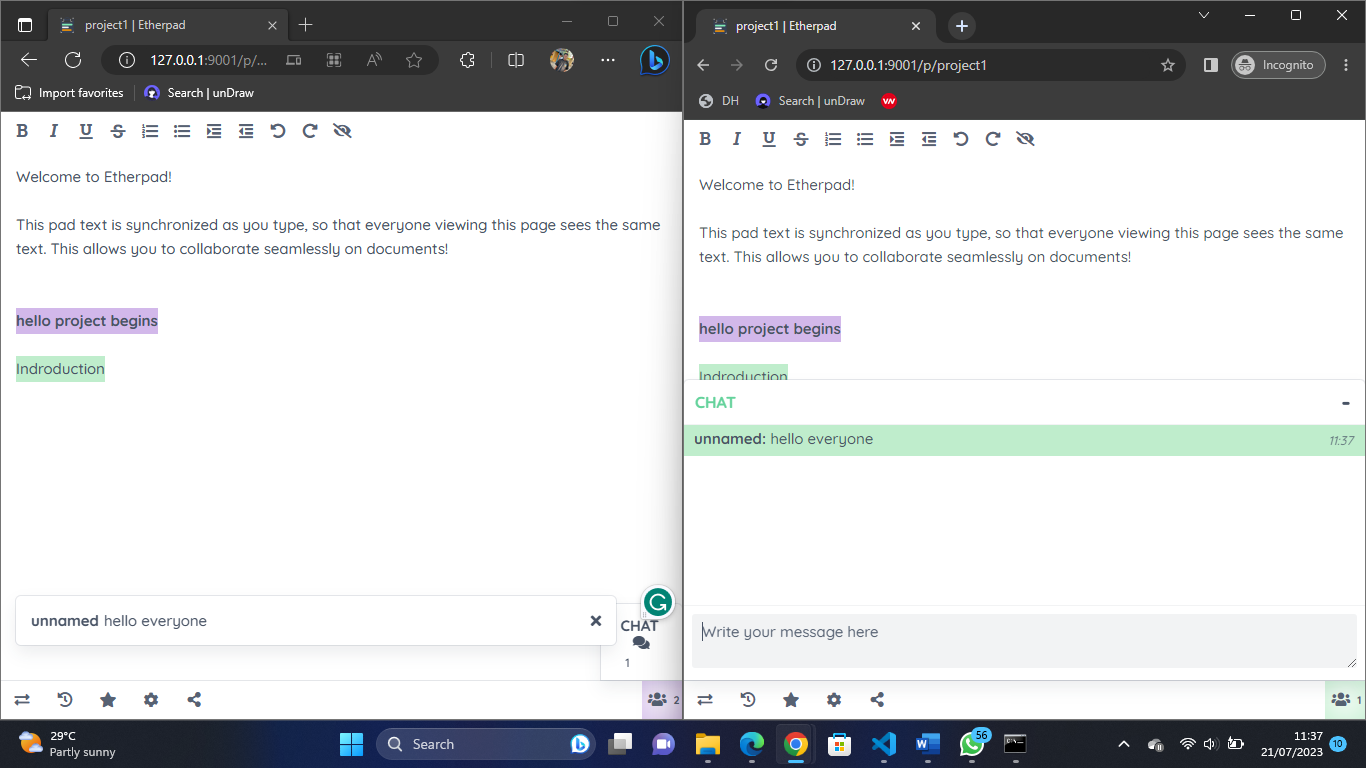


Fig 4.3 Multiple User View

**Chat Feature**

Multiple users can send messages to every other user on the pad

 Fig 4.3 Multiple User View

**Sharing Pad**

User can share the link of the current pad to any other user, and can also state the type of the view

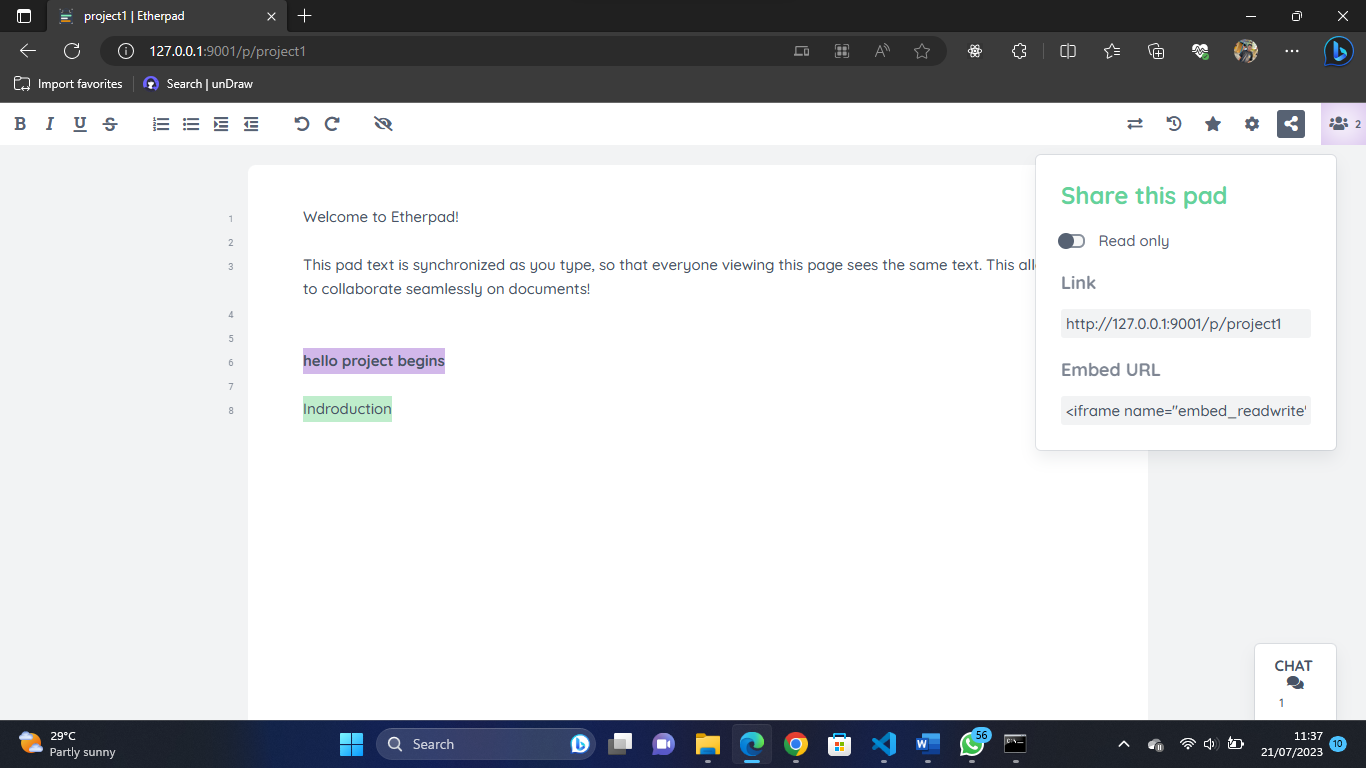


Fig 4.4 Sharing Pad

**Editing Pad**

Major text editing features are available on the pad, every user has a different color showing who made the changes

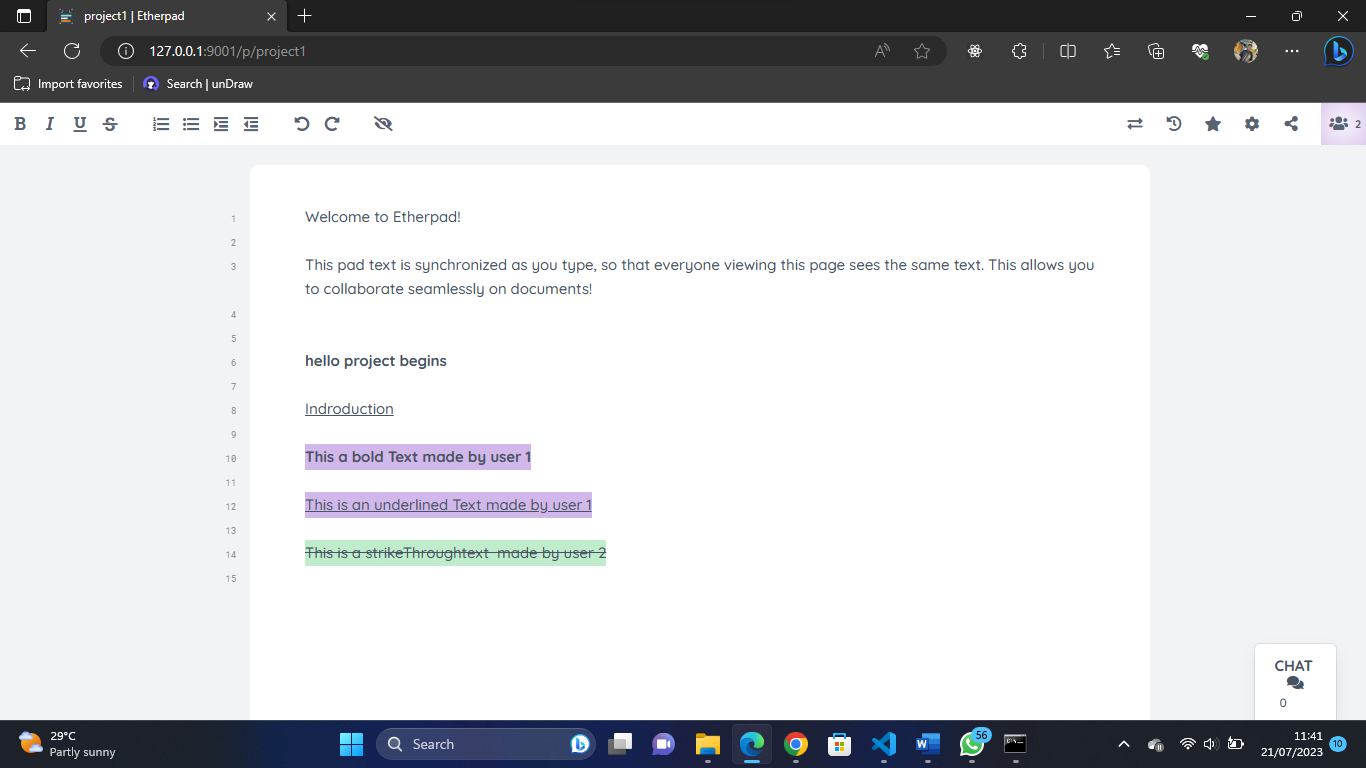


Fig 4.5 Editing Pad

**Running the Pad**

The application is powered by node.js and is launched on the terminal

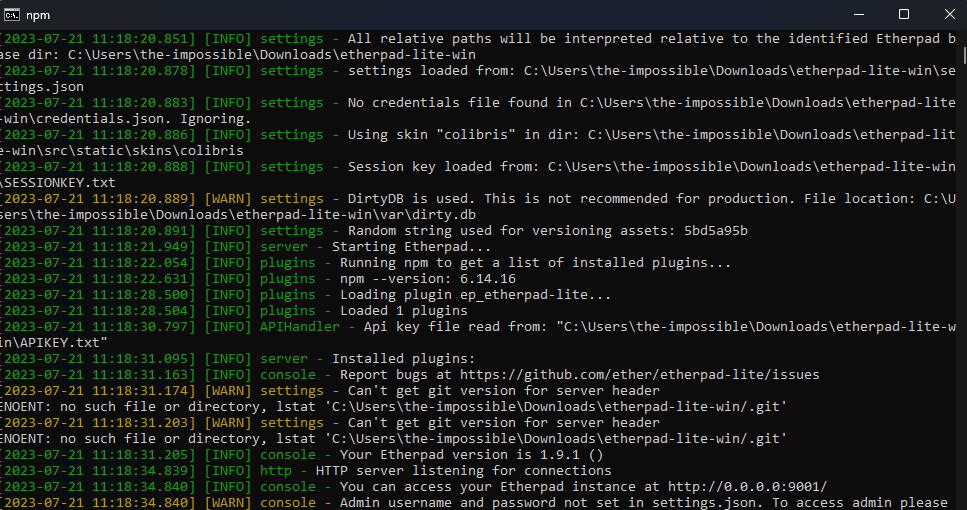


Fig 4.6 Running the Pad

**CHAPTER FIVE**

**SUMMARY CONCLUSION AND RECOMMENDATION**

**5.1 Summary**

Document collaboration is a vital aspect of modern teamwork, allowing multiple users to edit files simultaneously and creating a unified final document. Such collaboration tools are essential for distributed or remote teams, enabling seamless project management and file sharing. However, students often struggle to share documents effectively during collaborative assignments, leading to fragmented work and challenges for supervisors during assessment. To address this issue, the goal of this project is to develop an app that facilitates smooth document collaboration among users. The app will provide an engaging and user-friendly interface using Flutter, with logic handled by Node.js and data stored in Firebase. By creating a cloud-based collaborative document editing platform, this study aims to enhance collaboration and streamline group assignments and projects for students, optimizing their academic experience and promoting efficient teamwork.

**5.2 Conclusion**

In conclusion, document collaboration plays a crucial role in facilitating effective teamwork and project management in today's digital world. The ability to work together on the same document in real-time, without version conflicts, is a valuable asset for both students and professionals. However, students often face challenges in sharing and editing documents during group assignments and projects, leading to fragmented work and difficulties for supervisors during assessments. By providing an engaging and user-friendly platform, this app can empower students to collaborate more efficiently, enhance their academic experience, and foster productive teamwork. The significance of this study lies in promoting effective document collaboration, improving academic outcomes, and streamlining project work, ultimately benefiting both students and educators alike.

**5.3 Recommendation**

Based on the research and development of the collaborative document editing app, the following recommendations are proposed to further enhance its functionality and user experience:

1. Real-Time Collaboration Features: Implement real-time collaboration features that allow multiple users to edit a document simultaneously, providing instant updates to all collaborators.
2. User-Friendly Interface: Ensure the app's interface is intuitive and user-friendly, catering to both tech-savvy users and those less familiar with digital tools.
3. Notifications and Alerts: Implement notifications and alerts to keep users informed about document updates, comments, and collaboration activities.
4. Commenting and Feedback: Include a commenting and feedback system that allows users to provide input and suggestions on documents, fostering effective communication and collaboration.
5. Cross-Platform Compatibility: Ensure the app is compatible with multiple platforms and devices, such as desktops, laptops, tablets, and smartphones.

By implementing these recommendations, the collaborative document editing app can become a powerful tool for students and professionals alike, promoting seamless teamwork and enhancing productivity in collaborative projects and assignments. Moreover, the app's user-friendly design and robust features will contribute to a positive user experience and support efficient document collaboration in various educational and professional settings

**REFERENCES**

Al-Samarraie, H., & Saeed, N. (2018). A systematic review of Cloud Computing Tools for Collaborative Learning: Opportunities and challenges to the blended-learning environment. *Computers & Education*, *124*, 77–91. https://doi.org/10.1016/j.compedu.2018.05.016

Arora, S., & Atrey, P. K. (2021). Secure collaborative editing using Secret sharing. *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*. https://doi.org/10.1109/wifs53200.2021.9648395

Batschinski, G. (2022, August 10). *What is firebase? all secrets unlocked*. Back4App Blog. Retrieved December 22, 2022, from https://blog.back4app.com/firebase/

González‐Lloret, M. (2020). Collaborative tasks for online language teaching. *Foreign Language Annals*, *53*(2), 260–269. https://doi.org/10.1111/flan.12466

Harjanto, A. S., & Sumarni, S. (2019). TEACHERS’ EXPERIENCES ON THE USE OF GOOGLE CLASSROOM. *English Language and Literature International Conference (ELLiC)*, *3*, 172–178.

Hooker, T. (2022, November 7). *Top 9 document collaboration tools for growing teams*. Document360. Retrieved December 23, 2022, from https://document360.com/blog/document-collaboration-tools/

Ko, M., Lee, C., & Cho, Y. (2022). Design and implementation of cloud-based Collaborative Manufacturing Execution System in the Korean fashion industry. *Applied Sciences*, *12*(18), 9381. https://doi.org/10.3390/app12189381

Lee, C. Y., Zhang, Z., Herskovitz, J., Seo, J. Y., & Guo, A. (2022). Collabally: Accessible collaboration awareness in document editing. *CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3491102.3517635

Rahman, H., Roy, S. B., Thirumuruganathan, S., Amer-Yahia, S., & Das, G. (2018). Optimized group formation for solving collaborative tasks. *The VLDB Journal*, *28*(1), 1–23. https://doi.org/10.1007/s00778-018-0516-7

Sufiyan, T. (2022, November 24). *What is node.js: A comprehensive guide*. Simplilearn.com. Retrieved December 24, 2022, from https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs

Thomas, G. (2021, July 9). *What is flutter and why you should learn it in 2020*. freeCodeCamp.org. Retrieved December 22, 2022, from <https://www.freecodecamp.org> /news/what-is-flutter-and-why-you-should-learn-it-in-2020/

**APPENDIX**

**Index.html**

<%

var settings = require("ep\_etherpad-lite/node/utils/Settings");

%>

<!doctype html>

<html>

<title><%=settings.title%></title>

<meta charset="utf-8">

<meta name="referrer" content="no-referrer">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">

<link rel="shortcut icon" href="favicon.ico">

<link rel="localizations" type="application/l10n+json" href="locales.json">

<script type="text/javascript" src="static/js/vendors/html10n.js?v=<%=settings.randomVersionString%>"></script>

<script type="text/javascript" src="static/js/l10n.js?v=<%=settings.randomVersionString%>"></script>

<script src="static/js/vendors/jquery.js"></script>

<script src="static/js/index.js"></script>

<style>

html, body {

height: 100%;

}

body {

margin: 0;

color: #333;

font: 14px helvetica, sans-serif;

background: #ddd;

background: -webkit-radial-gradient(circle,#aaa,#eee 60%) center fixed;

background: -moz-radial-gradient(circle,#aaa,#eee 60%) center fixed;

background: -ms-radial-gradient(circle,#aaa,#eee 60%) center fixed;

background: -o-radial-gradient(circle,#aaa,#eee 60%) center fixed;

border-top: 8px solid rgba(51,51,51,.8);

}

#wrapper {

border-top: 1px solid #999;

margin-top: 160px;

padding: 15px;

background: #eee;

background: -webkit-linear-gradient(#fff,#ccc);

background: -moz-linear-gradient(#fff,#ccc);

background: -ms-linear-gradient(#fff,#ccc);

background: -o-linear-gradient(#fff,#ccc);

box-shadow: 0px 1px 8px rgba(0,0,0,0.3);

}

#inner {

position:relative;

max-width: 300px;

margin: 0 auto;

}

#button {

margin: 0 auto;

text-align: center;

font: 36px verdana,arial,sans-serif;

width:300px;

border:none;

color: white;

text-shadow: 0 -1px 0 rgba(0,0,0,.8);

height: 70px;

line-height: 70px;

background: #555;

background: -webkit-linear-gradient(#5F5F5F,#565656 50%,#4C4C4C 51%,#373737);

background: -moz-linear-gradient(#5F5F5F,#565656 50%,#4C4C4C 51%,#373737);

background: -ms-linear-gradient(#5F5F5F,#565656 50%,#4C4C4C 51%,#373737);

background: -o-linear-gradient(#5F5F5F,#565656 50%,#4C4C4C 51%,#373737);

box-shadow: inset 0 1px 3px rgba(0,0,0,0.9);

}

#button:hover {

cursor: pointer;

background: #666;

background: -webkit-linear-gradient(#707070,#666666 50%,#5B5B5B 51%,#474747);

background: -moz-linear-gradient(#707070,#666666 50%,#5B5B5B 51%,#474747);

background: -ms-linear-gradient(#707070,#666666 50%,#5B5B5B 51%,#474747);

background: -o-linear-gradient(#707070,#666666 50%,#5B5B5B 51%,#474747);

}

#button:active {

box-shadow: inset 0 1px 12px rgba(0,0,0,0.9);

background: #444;

}

#label {

text-align: left;

text-shadow: 0 1px 1px #fff;

margin: 16px auto 0;

display:block;

}

#padname{

height:38px;

max-width:280px;

}

form {

height: 38px;

background: #fff;

border: 1px solid #bbb;

border-radius: 3px;

position: relative;

}

button, input {

font-weight: bold;

font-size: 15px;

}

input[type="text"] {

border-radius: 3px;

box-sizing: border-box;

-moz-box-sizing: border-box;

line-height:36px; /\* IE8 hack \*/

padding: 0px 45px 0 10px;

\*padding: 0; /\* IE7 hack \*/

width: 100%;

height: 100%;

outline: none;

border: none;

position: absolute;

}

button[type="submit"] {

position: absolute;

left:253px;

width: 45px;

height: 38px;

}

@media only screen and (min-device-width: 320px) and (max-device-width: 800px) {

body {

background: #bbb;

background: -webkit-linear-gradient(#aaa,#eee 60%) center fixed;

background: -moz-linear-gradient(#aaa,#eee 60%) center fixed;

background: -ms-linear-gradient(#aaa,#eee 60%) center fixed;

}

#wrapper {

margin-top: 0;

}

#inner {

width: 95%;

}

#label {

text-align: center;

}

}

</style>

<% e.begin\_block("indexCustomStyles"); %>

<link href="static/skins/<%=encodeURI(settings.skinName)%>/index.css?v=<%=settings.randomVersionString%>" rel="stylesheet">

<% e.end\_block(); %>

<div id="wrapper">

<% e.begin\_block("indexWrapper"); %>

<div id="inner">

<% if (!settings.requireSession) { %>

<% if (settings.editOnly) { %>

<label id="label" for="padname" data-l10n-id="index.openPad"></label>

<% } else {%>

<button id="button" data-l10n-id="index.newPad"></button>

<label id="label" for="padname" data-l10n-id="index.createOpenPad"></label>

<% } %>

<form action="#" id="go2Name">

<input type="text" id="padname" maxlength="50" autofocus x-webkit-speech>

<button type="submit">OK</button>

</form>

<% } %>

</div>

<% e.end\_block(); %>

</div>

<% e.begin\_block("indexCustomScripts"); %>

<script src="static/skins/<%=encodeURI(settings.skinName)%>/index.js?v=<%=settings.randomVersionString%>"></script>

<% e.end\_block(); %>

<div style="display:none"><a href="/javascript" data-jslicense="1">JavaScript license information</a></div>

</html>

**Logic**

'use strict';

const spawn = require('child\_process').spawn;

const async = require('async');

const settings = require('./Settings');

const os = require('os');

// on windows we have to spawn a process for each convertion,

// cause the plugin abicommand doesn't exist on this platform

if (os.type().indexOf('Windows') > -1) {

exports.convertFile = async (srcFile, destFile, type) => {

const abiword = spawn(settings.abiword, [`--to=${destFile}`, srcFile]);

let stdoutBuffer = '';

abiword.stdout.on('data', (data) => { stdoutBuffer += data.toString(); });

abiword.stderr.on('data', (data) => { stdoutBuffer += data.toString(); });

await new Promise((resolve, reject) => {

abiword.on('exit', (code) => {

if (code !== 0) return reject(new Error(`Abiword died with exit code ${code}`));

if (stdoutBuffer !== '') {

console.log(stdoutBuffer);

}

resolve();

});

});

};

// on unix operating systems, we can start abiword with abicommand and

// communicate with it via stdin/stdout

// thats much faster, about factor 10

} else {

let abiword;

let stdoutCallback = null;

const spawnAbiword = () => {

abiword = spawn(settings.abiword, ['--plugin', 'AbiCommand']);

let stdoutBuffer = '';

let firstPrompt = true;

abiword.stderr.on('data', (data) => { stdoutBuffer += data.toString(); });

abiword.on('exit', (code) => {

spawnAbiword();

if (stdoutCallback != null) {

stdoutCallback(new Error(`Abiword died with exit code ${code}`));

stdoutCallback = null;

}

});

abiword.stdout.on('data', (data) => {

stdoutBuffer += data.toString();

// we're searching for the prompt, cause this means everything we need is in the buffer

if (stdoutBuffer.search('AbiWord:>') !== -1) {

const err = stdoutBuffer.search('OK') !== -1 ? null : new Error(stdoutBuffer);

stdoutBuffer = '';

if (stdoutCallback != null && !firstPrompt) {

stdoutCallback(err);

stdoutCallback = null;

}

firstPrompt = false;

}

});

};

spawnAbiword();

const queue = async.queue((task, callback) => {

abiword.stdin.write(`convert ${task.srcFile} ${task.destFile} ${task.type}\n`);

stdoutCallback = (err) => {

if (err != null) console.error('Abiword File failed to convert', err);

callback(err);

};

}, 1);

exports.convertFile = async (srcFile, destFile, type) => {

await queue.pushAsync({srcFile, destFile, type});