School of Electronic Engineering and Computer Science

Final Report

Programme of study:
BSc Computer Science

Project Title: ToneBook: Online E-book maker

Final Year
Undergraduate Project 2021/22

Supervisor:
Dr Joseph Doyle

Queen Mary
University of London

Student Name:
Mohammad Ibrahim Salla

Date: 10th/05/2022

Abstract

Due to the increase of people using electronic devices, there has become a wider audience to reach when authors decide to publish e-books instead of hard copy papers. This shift in the market has opened opportunities whereby authors can also be able to create e-books online using online e-book makers. This has increased the number of online e-book makers by a tenth fold since they are so easily accessible. However, the main issues with online e-book makers are that they currently have a lot of shortcomings, and there will be an in-depth analysis of these issues.

ToneBook, which is an online e-book maker, will be developed to counteract some of the significant shortcomings of existing applications. ToneBook will have a text editor in which authors can easily import images and lists. ToneBook will detect the spelling and grammar mistakes as well as identify some fundamental tonality changes and give suggestions on them.

The main objective is to create an online application that will have a simplistic design to enhance the usability of the application as well as give the authors all the tools necessary to achieve their highest quality of work.

This report will first identify the background and problem statement. This will be followed by a literature review of word processing applications as a whole. Similar existing systems will be analysed to derive some important features as well as the drawbacks of the existing systems. This will then be followed by Requirements analysis, implementation, testing, and evaluation of the system.

Contents

Abstract 2
Chapter 1: Introduction 4
1.1 Background 4
1.2 Problem Statement 4
1.3 Aim 5
1.4 Objectives 5
1.5 Research Questions 5
1.6 Report Structure 6
Chapter 2: Literature Review 7
2.1 Word Processing 7
2.2 Similar Systems 8
2.3 Conclusion 11
Chapter 3: Requirements Analysis & Design 12
3.1 Requirements Analysis 12
3.2 Design 13
Chapter 4: Implementation 17
4.1 Languages and Frameworks 17
4.2 Essential API's 17
4.3 Core functionalities 18
Chapter 5: Testing 21
5.1 Django Unit Testing 21
5.2 Browser Compatibility Testing 24
5.3 User acceptance testing 24
5.4 Summary 25
Chapter 6: Evaluation 26
6.1 Heuristic Evaluation 26
6.2 Limitations 27
6.3 Summary 27
Chapter 7: Legal, Social and Ethical issues 28
7.1 Legal Issues 28
7.2 Social Issues 28
7.3 Ethical Issues 28
7.4 Sustainability 28
Chapter 8: Conclusion 29
8.1 Achievements 29
8.2 Challenges Faced 29
8.3 Future Improvements 30
8.4 Summary 30
References 31
Appendix A – Risk assessment 32
Appendix B – Project Plan 33

Chapter 1: **Introduction**

1.1 Background

We are in the middle of a computer revolution, which has affected our lives more than we could have ever imagined. All aspects of our lives have been infiltrated by technology, from the way we communicate to the way we travel. One of the most significant changes occurred in the past few years due to the pandemic. We never expected to move work-life home, but that is exactly what happened. This shift in work style has opened a market for online applications that will aid employees in their work.

Since a lot of people have started working from home, there is an even wider audience available online (Maxim and Maxim, 2012). This increased audience online has resulted in the increase in popularity of e-books which in turn has meant authors have started shifting their focus into making e-books as they are more sensible than hard copy books (Lai and Chang, 2011). There is a multitude of reasons why it is advantageous for authors and readers alike to start shifting towards e-book creations. One of the main reasons for this is that e-books are more cost-effective long-term than hard copied books are (Maxim and Maxim, 2012). This is mainly due to the fact that it is expensive to have prints outs made of books, while that cost is negated entirely by e-books. In addition to this, hard copy books are also very costly to store in cool, dry places. Another important reason is that e-books are cleaner than hard copies because most papers either end up in landfills or are incinerated (Schmidt, Holm, Merrild, and Christensen, 2007). This has significant consequences for climate change, that is why it is essential to start the move to e-books.

After covering the importance of why we need e-books, we need to talk about how they are created. There are a few ways an e-book can be created, but we will focus on only one of these, which is an online e-book maker. This is one of the most efficient ways to create an e-book, as they are accessible on the fly, anytime, and anywhere. This ease of access is immense for authors as new ideas can flood the authors at any time.

There are a few features of online e-book applications which attract authors to use them. There is the standard grammar and spelling checker, which almost all existing systems have variations off. A new feature that is starting to gain importance is that of tone analysis. The tonality of e-books is essential for authors so that they can correctly express their creative or intellectual ideas. Another additional feature that is very highly rated by authors is having a secure application. Similar to most items published on the internet nowadays, security should be at the forefront of development.

1.2 Problem Statement

One of the main pitfalls with existing applications is the inability to analyse the tone and style of the text. Authors have stated that it takes a significant amount of time for them to go through their projects to try and see if the tone of the book matches their initial plan. This lack of tone analysis software elongates the time it will take for the authors to finalise their e-books which will have an adverse effect on the release dates as well as the overall quality of the books.

Moreover, another shortcoming of existing systems is that of security. Some applications have weak security, which can be easily bypassed. This puts the writers' work in danger of being stolen or, even worse, losing their data entirely.

1.3 Aim

The main aim of the online e-book maker is to provide authors with the capabilities to create e-books with the crucial assistance of tone analysis and suggestions. This will overcome the shortcomings of many existing applications which do not cater to this functionality. The author will be able to write a project and have real-time feedback from the tone analysis software which they are using as well as suggestions on how to improve it. In addition to this, the web application must have a secure login system to prevent unauthorised access.

1.4 Objectives

The objectives of the project include:

- Analyse existing systems and elicit key functionalities and pitfalls of such systems.
- Requirements will be elicited from authors through existing systems as well as other online resources.
- To be able to identify the challenges related to using tone analysis software and how to go about implementing it effectively.
- To implement an API that will allow authors to be able to write text as well as have some other editing features like paragraphing and list creation.
- To perform both Django unit testing as well as user acceptance testing to deduce the effectiveness of the entire website.
- Use the information elicited from the user testing to be able to perform some heuristic evaluation on the system.

.

1.5 Research Questions

These are some of the questions that this project will answer:

- How much can the overall experience of using an online e-book maker be improved?
- How will the implementation of tone analysis and suggestion work?
- How secured can an online e-book maker be?
- What impacts does the usability of features have on the duration an author takes on a project?
- How important is it for authors to be able to export and import documents into the system?

1.6 Report Structure

Chapter 2: This contains the literature review of word processing applications, and this will also consist of the domain analysis of existing systems and then comparing their features in a table. This will then be concluded by a short summary of what main features will be added to the word processing application.

Chapter 3: This chapter details the functional and non-functional requirements for the system. This chapter will also include the design used to create the application in the form of a use case diagram.

Chapter 4: This chapter details the implementation process of Tonebook, including the different technologies used. All the main features of the online website will also be explained.

Chapter 5: This chapter details how Tonebook was tested. This was done in two parts, one of which is Django unit testing followed by the user acceptance testing. This chapter will be concluded with a summary of the overall results of the testing.

Chapter 6: This chapter details the overall outcome of the project. This will contain the heuristic evaluation, which will determine how effective the system is. The limitations of the project will then follow this. This will then be concluded with a summary of the evaluation results.

Chapter 7: This chapter details the ethical, social, and legal issues. This will then be concluded by the sustainability of the project.

Chapter 8: This chapter is about the conclusion of the project. The conclusion will talk about the achievements of the project as well as all the challenges faced when creating the word processing application. This will be followed by the possible ways that the project can be improved in the future. This will then be followed by a summary of the conclusion and a statement on whether the application was a success in its entirety.

Appendix A: Discusses the severity, likelihood, and impacts of some risks occurring during the development and implementation phase of the project.

Appendix B: This diagram has the project roadmap with the dates for each of the sections of the system life cycle, from the first stage, which was analysis and planning to the last section, which is the testing and evaluation of the system.
.

Chapter 2: **Literature Review**

This chapter has been structured in such a way that the first aspect of the literature review, which is the background of word processing, is discussed. After a thorough analysis has been established, existing systems will be described. After which, a table that has the existing systems and their identified features and limitations is displayed. A conclusion is then conversed.

2.1 Word Processing

Due to the rapid rise of technology, almost every aspect of our lives have been transformed by it. One of these is a word processing application that has been developed to assist in the development of creative or intellectual written ideas. There are a few other issues that can be solved by the creation of word processors. One of these is that libraries do not have enough space to keep a copy of all the books (Moore, 2015). In this section, we will discuss the differences between word processing applications and handwritten methods of writing.

Firstly, word processing applications have massively improved the quality of writing of individuals as they consist of grammar and spelling checkers which can detect errors as well as correct them. There was a study done that looked at the effects of a year-long word processing program by comparing two groups. One had a word processing application, while the other group had to use handwriting to complete assignments (Dalton and Hannafin, 1987). After the completion of the study, an analysis was done on the writing samples of both groups, which determined that word processing applications alone made minor changes to the able learners. However, they made massive improvements in the quality of writing for the low achieving students (Bangert-Drowns, 1993).

However, this ability for the word processing applications to correct mistakes could potentially make writers lazy and less concentrated on the grammar and spelling of their work. In fact, a study has claimed that word processing applications had only modest results compared to handwritten methods (McCarthy et al., 2022). It has been claimed that writing can only be improved by feedback that has targeted strategies rather than automatic corrections.

Another study has claimed that writing via word processing applications or handwriting can have influences on creativity (Yang and Liu, 2021). Another study has claimed that the way literature and philosophy are written is starting to be transformed because of the use of word processing applications (Heim and Gelernter, 1999).

Acknowledgment must be made of some of the downsides of using word processing applications, like making writers lazy and careless with their grammar and spelling. However, I believe overall, it has contributed to the growth of written ideas as well as the development of other sectors like work life.

2.2 Similar Systems

2.2.1 Google docs

This is a very well-known online word processing application as it is part of the Google applications. It has all the standard word processor features, including editing and formatting documents.

The system does not consist of any tone analysis and suggestion. There is a grammar checker, even though it is not always practical, and the suggestions might change the full context of the sentence. The spelling checker is one of the best available online. As it is a google product, it has two-factor authentication making it very secure.

The user interface is very straightforward and easy to use, and it also has robust editing options. The ease of collaboration is exceptional because a user can easily add and see the history of changes the document has been through.

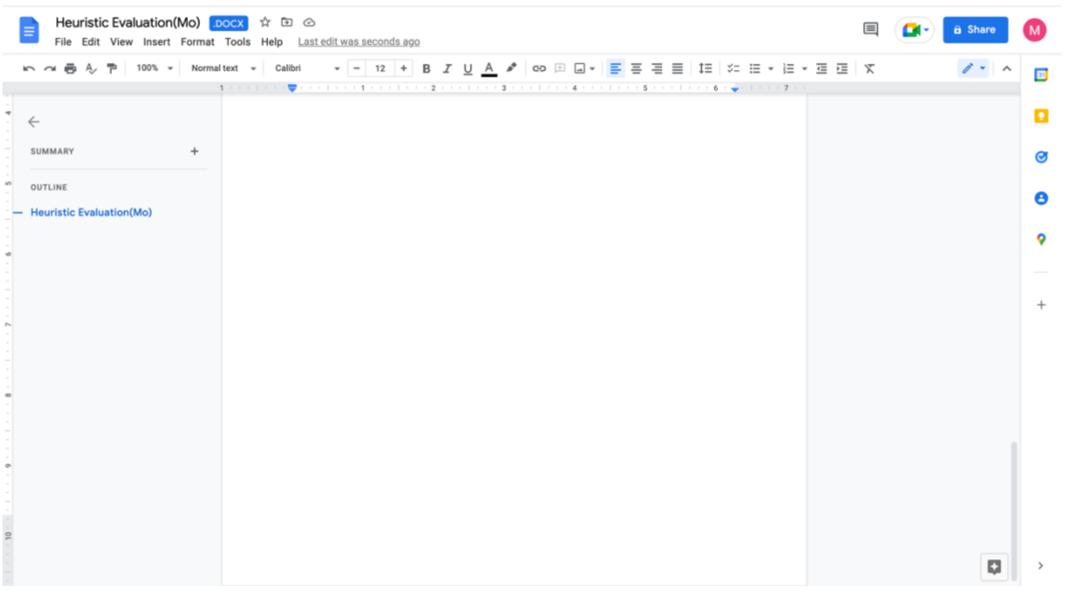


Figure 1: Google Docs

2.2.2 Zoho Writer

The following existing system we will discuss is Zoho writer. It has the same traditional word processor features like editing and formatting documents.

The system does have tone analysis/suggestion software, as well as monitoring the writing quality by detecting cliches and wordy sentences (Zoho Writer, 2022). There is a proficient grammar and spelling checker. There is also an autocorrect feature that the authors can customize to their liking. It also has a secured login authentication system keeping the authors' projects safe.

The software has complete compatibility with Microsoft word, which means that Microsoft Word files can be uploaded. Zoho writer documents can also be saved on the users' computers (Zoho Compatibility, 2022).

Zoho has a modern yet simplistic interface with all the tools laid out clearly on the screen. Zoho can allow authors to work on their projects straight from a click on their explorer or finder (Zoho Writer, 2022). The collaboration on Zoho is what makes it stand out from other online word editing applications. It allows users to invite other users to the project as well as gives the users the ability to communicate with one another using their collaboration text chat.

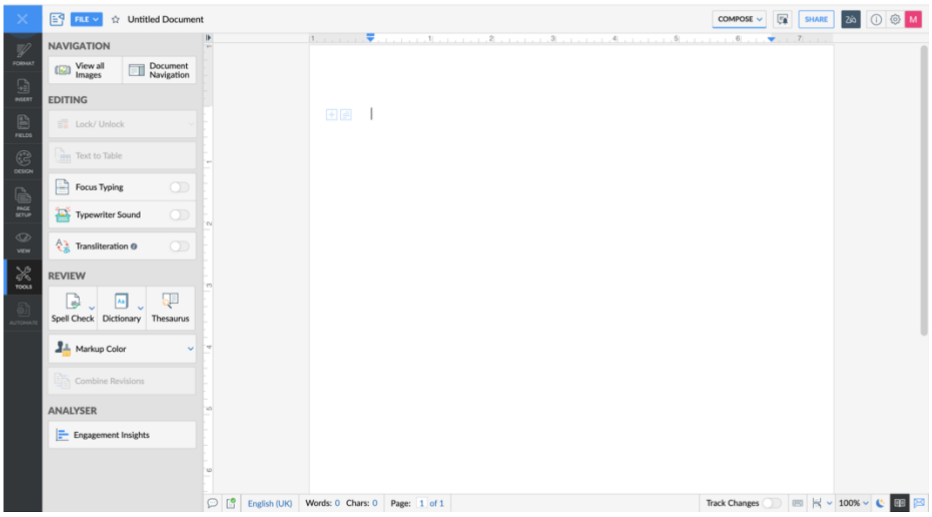


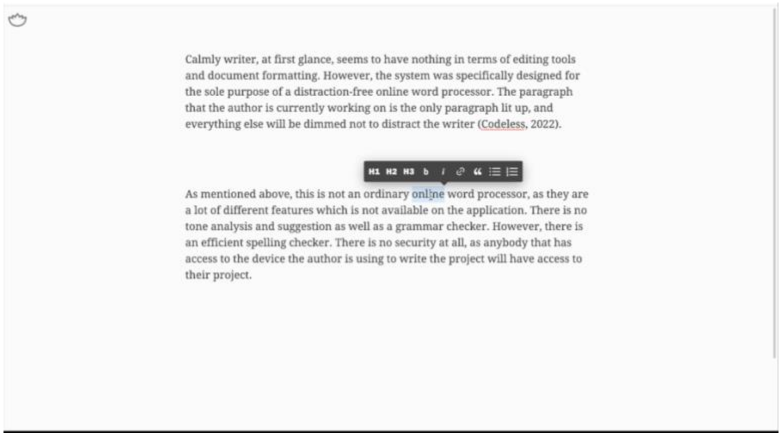
Figure 2: Zoho Writer

2.2.3Calmly Writer

Calmly writer, at first glance, seems to have nothing in terms of editing tools and document formatting. However, the system was specifically designed for the sole purpose of a distraction-free online word processor. The paragraph that the author is currently working on is the only paragraph lit up, and everything else will be dimmed not to distract the writer (Codeless, 2022).

As mentioned above, this is not an ordinary online word processor, as they are a lot of different features which is not available on the application. There is no tone analysis and suggestion as well as a grammar checker. However, there is an efficient spelling checker. There is no security at all, as anybody that has access to the device the author is using to write the project will have access to their project.

There is a complex design because all the editing options available are hidden under the preferences option in the already hidden menu bar. It does have the capability of adding an extension like Grammarly to the system.


Calmly writer, at first glance, seems to have nothing in terms of editing tools and document formatting. However, the system was specifically designed for the sole purpose of a distraction-free online word processor. The paragraph that the author is currently working on is the only paragraph lit up, and everything else will be dimmed not to distract the writer (Codeless, 2022).

As mentioned above, this is not an ordinary online word processor, as they are a lot of different features which is not available on the application. There is no tone analysis and suggestion as well as a grammar checker. However, there is an efficient spelling checker. There is no security at all, as anybody that has access to the device the author is using to write the project will have access to their project.

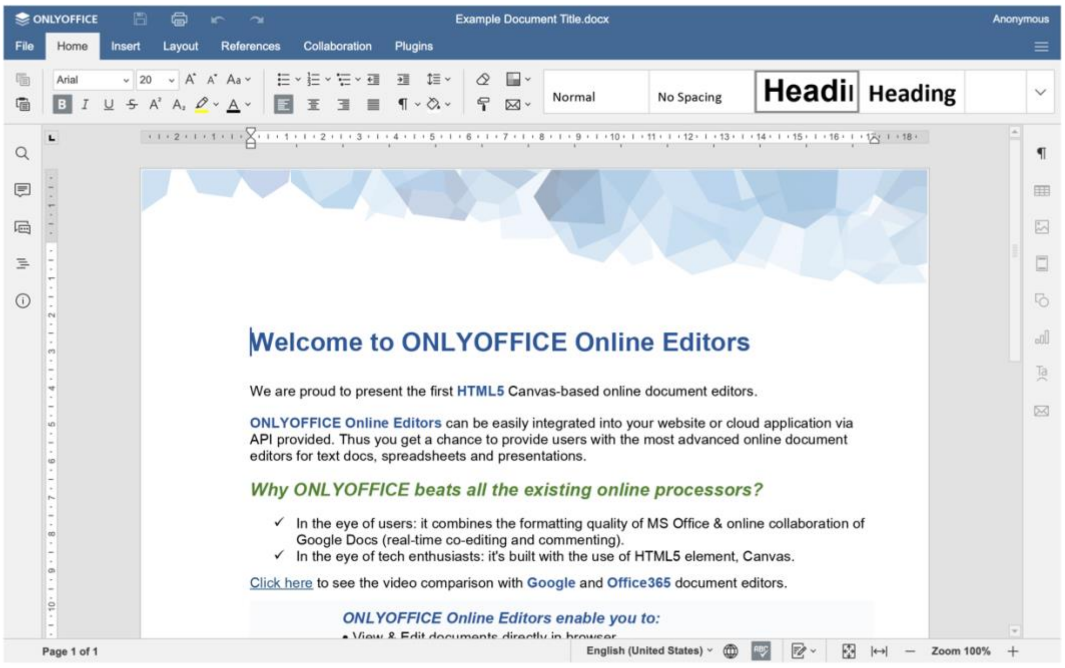
Figure 3: Calmly Writer

2.2.4OnlyOffice

This application has both a desktop and a mobile version, but we will be focusing on its online version. This is another online web application that has the main traditional editing and formatting features associated with a word processing editor.

OnlyOffice has no tone analysis or suggestion, but it does have a grammar and spelling checker. The OnlyOffice application has strong login authentication, and this allows the user to log in via any email address and resume where they left their project.

The user interface is very similar to the Microsoft word application, as it has almost the same layout. The interface is easy to use, and it has a lot of different editing features as well as the ability to insert images. The collaboration on OnlyOffice is excellent as it allows for the simple addition of users to the project. Users can also highlight a section of the project and assign a text to it stating their criticism of said section. All the other users will be able to see the critique and respond to it. Once a paragraph has been agreed upon, the paragraph can be locked to prevent any further editing of the paragraph.



Welcome to ONLYOFFICE Online Editors

We are proud to present the first HTML5 Canvas-based online document editors.

ONLYOFFICE Online Editors can be easily integrated into your website or cloud application via API provided. Thus you get a chance to provide users with the most advanced online document editors for text docs, spreadsheets and presentations.

Why ONLYOFFICE beats all the existing online processors?

- ✓ In the eye of users: it combines the formatting quality of MS Office & online collaboration of Google Docs (real-time co-editing and commenting).
- ✓ In the eye of tech enthusiasts: it's built with the use of HTML5 element, Canvas.

Click here to see the video comparison with Google and Office365 document editors.

ONLYOFFICE Online Editors enable you to:

• View & Edit documents directly in browser

Figure 4: OnlyOffice

Application	Features	Drawbacks
Google docs	-Two factor authentication -Grammar and spelling checker -Document collaboration	-No tone analysis and suggestion -Requires a google account
Zoho writer	-Tone analysis and suggestion -Machine learning grammar checker -Document collaboration with text chat	-Some features require payment to be used.

Calmly writer	-Spelling checker	-No tone analysis/suggestion -No form of security -No grammar Checker -Lacks a lot of editing features
OnlyOffice	-Grammar and spelling checker -Login Authentication - Document collaboration with text chat	-No tone analysis and suggestion -Some features require payment to be used.

Figure 5: Similar existing applications

2.3 Conclusion

After analysing several similar existing systems, we can see that there are some issues with the currently existing systems which can be improved upon. One of the main features that would be beneficial to the authors is rarely implemented in any of the existing systems. This is the tone analysis and suggestion API. Some of the systems have strong login authentication, which could be a welcome addition to the ToneBook. Most systems allow for collaboration which will further be analysed to see if it suits the authors' vision of ToneBook.

Chapter 3: **Requirements Analysis & Design**

The requirements for this project were vital in determining the design and implementation of the Tonebook word processing application. The requirement analysis was derived from the literature review, existing systems, and some online resources. Below we will state the Functional and Non-functional requirements involved in the system. This will be followed by the hardware and software requirements for the system. The use case used for the design of the application will also be displayed below. This will be concluded by the development environment.

3.1 Requirements Analysis

We will discuss the functional and non-functional requirements of the project collated from sources like previous existing systems, the literature review as well some outcomes of surveys and questionnaires gathered from online sources. Previous existing systems will help us get a clear idea of what the shape of the project will look like, while backend features will also be determined from online sources.

3.1.1 Functional Requirements

Some of the Functional requirements are:

1. Authors will be able to sign up for the application using the signup page. Authors will enter their username and password to be able to sign in. The system will not allow for submission if the username already exists or if there is an empty field.

2. Authors should be able to log in to the online application using the username and password they created while signing up.

3. The system should allow the authors to be able to see the previously saved documents that they have worked on.

4. The system should be able to open a brand new document for the author to start working on.

5. The system should be able to save the documents the authors have worked on.

6. The system should allow authors to be able to make changes to the documents they are working on.

7. The system should be able to give the author spelling and grammar feedback when they make mistakes.

8. The system should also be able to read a text and analysis the tonality of the text, and give suggestions.

.

3.1.2 Non-functional Requirements

Some of the Non-functional requirements are:

1. The system should respond to user input in a reasonable amount of time without a considerable delay.

2. Authors' information should be stored so that even if they close the application and come back to it, their data should still be available.

3. The system should check all user input and validate that it meets the criteria required for the input. The system should then be able to display an error message when one of these validation methods fails.

4. The system should be able to provide a simple and easy to use interface which requires minimal learning time of the system.

5. The system should provide a help page whereby an author can call or email an admin for assistance with the issue they require assistance with.

6. The system will use a web browser as its interface, and it will support Google chrome usage as well as other web browsers like Safari and Mozilla Firefox.

3.1.3 Hardware Requirements

These are the hardware requirements for the users of the system:

1. The authors will require a Desktop or Laptop with a good amount of CPU and memory as the online word application is a bit computational heavy. The better the CPU and memory of the computer, the better for the running of the application.

1. In order to access the online word processing application, the user requires a robust and stable internet connection. To be able to sign up, log in and perform other tasks like importing and exporting a document.

3.1.4 Software Requirements

These are some of the software requirements to be able to run the system:

Software	Version
Python	3.8.0
Django	4.0.4
JavaScript	ES5
Bootstrap	5.0.2
Grammarly SDK	N/A

Figure 6: Software Requirements

3.2 Design

3.2.1 Use Case Diagram

A use case diagram was used to describe the scope of the Tonebook application. We only have one actor from our system, and that is the author. As soon as the project is loaded, the first thing the author will see is the welcome page. From there, the author will click on the register button, which will redirect the author to the register page. The author can either register or go to the login page. After successful completion of the login and registration, the author will be redirected to the accounts page, which has all the documents the author has created. From there, the author can click on create a new document which will take them to the text editing page.

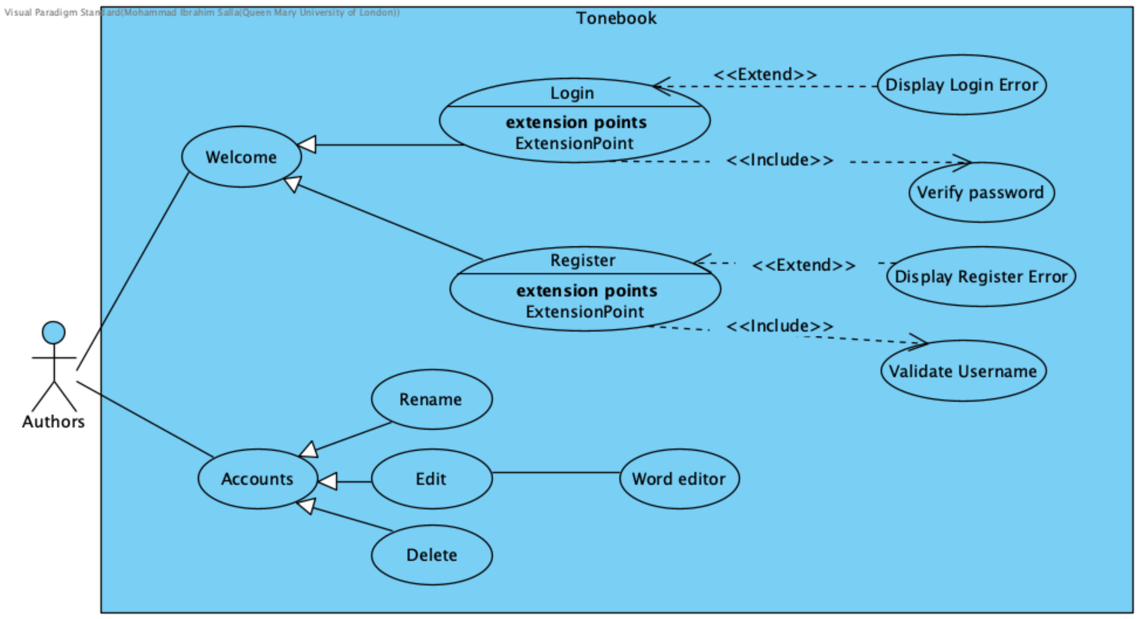
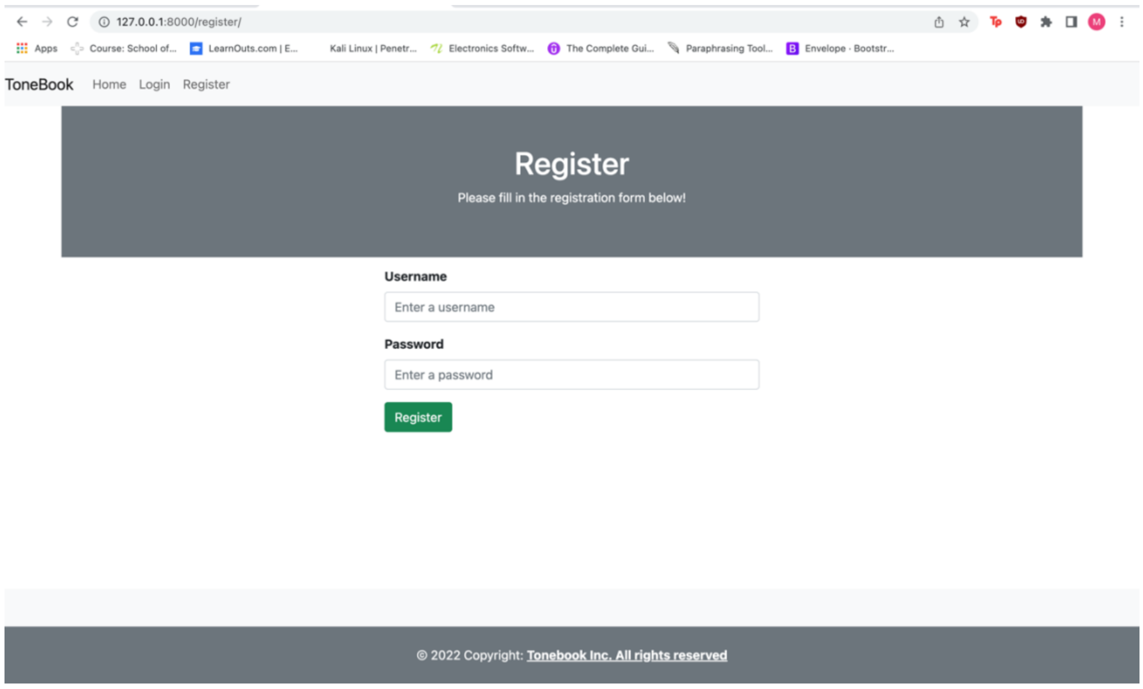


Figure 7: Use Case Diagram

3.2.2Use case examples

Registering a user: To register an author, the user first must go to the welcome page, which is the first page that will come up when the website is loaded. When the user clicks on register an account, it will forward them to the register HTML page whereby the user can create an account.



Register

Please fill in the registration form below!

Username

Enter a username

Password

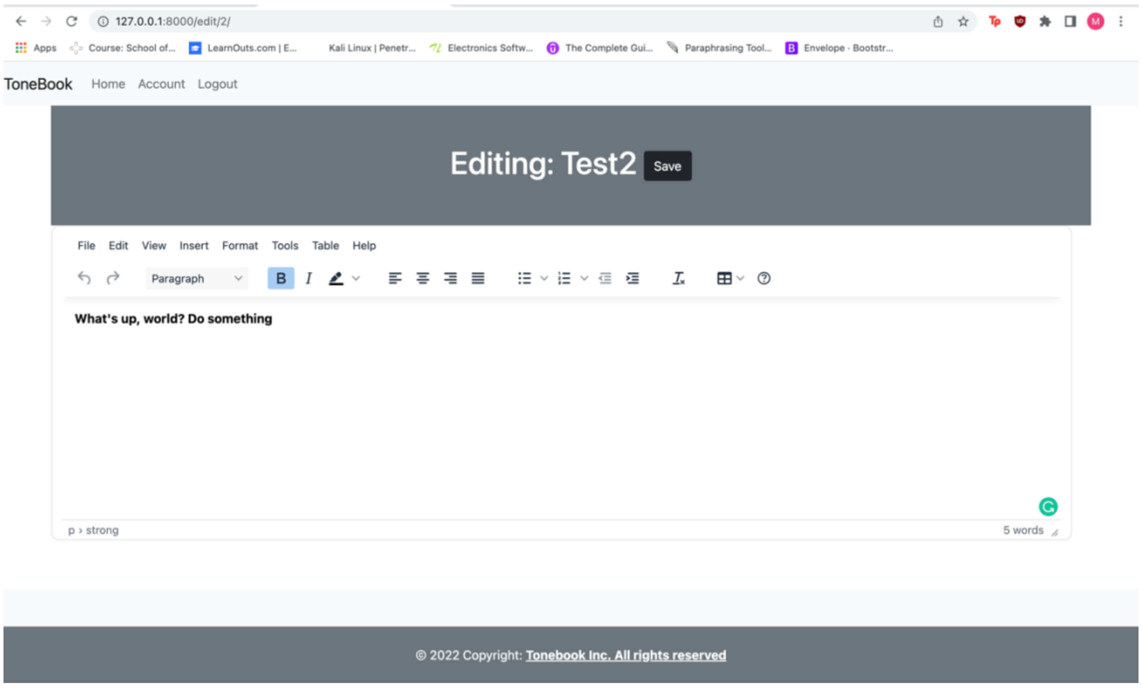
Enter a password

Register

© 2022 Copyright: Tonebook Inc. All rights reserved

Figure 8: Register Page

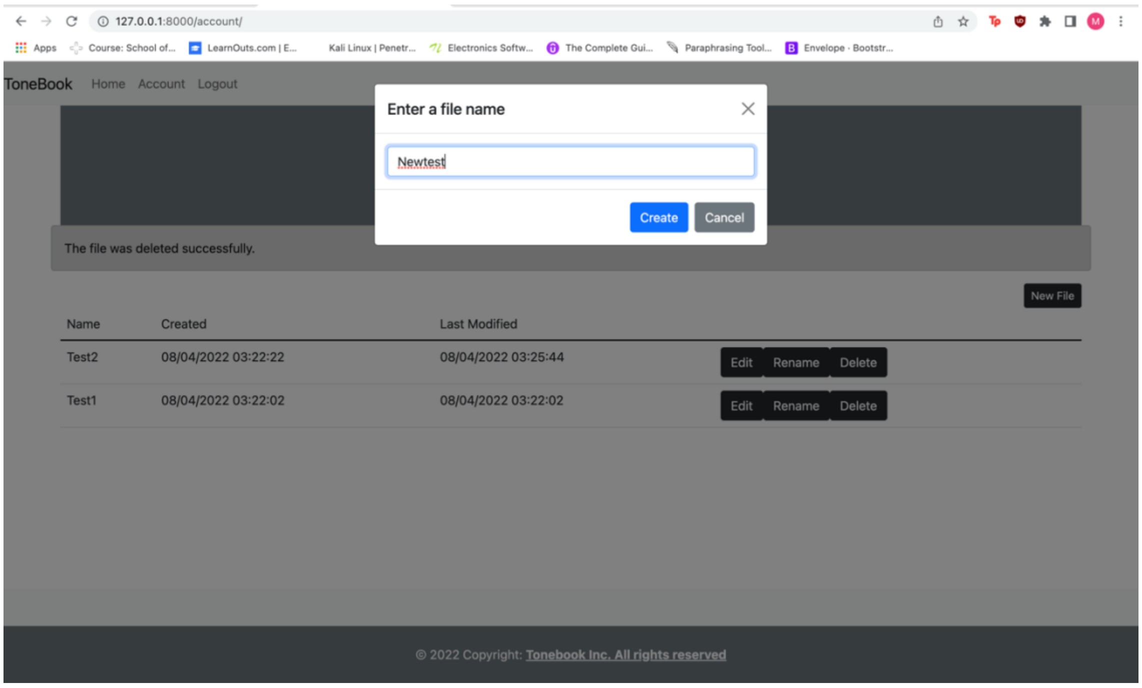
Editing a document: To edit a document, we have to first assume that the user has a document to edit. After the user logs into the system, the first thing they will see is the accounts page, whereby they can see an edit button that will forward them to another HTML document. In this document, the user has the actual text editor with what was previously saved to that file.



What's up, world? Do something

Figure 9: Editing a Document

Creating a document: To create a new document, the user should log in first. After the user has logged into the system, they will see the accounts page, whereby they will have an option to create a new document using the New button. From there, the user will have to fill in the name of the document.



The file was deleted successfully.

Name	Created	Last Modified	
Test2	08/04/2022 03:22:22	08/04/2022 03:25:44	Edit Rename Delete
Test1	08/04/2022 03:22:02	08/04/2022 03:22:02	Edit Rename Delete

Figure 10: Creating a Document

3.2.3Development Environment

This application was developed using the web framework called Django, which is based on the programming language python. The Tonebook application was implemented using several popular features under the Django web framework like login, authentication, and testing. All these features combine to simplify the development process of the application. To be able to run the application, minconda was installed to allow for the creation of a virtual environment. From the creation of a virtual environment, we were able to access the most critical Django features. After this virtual environment was created, a version of python had to be downloaded to be able to have access to some of the libraries available to python. Some other packages were downloaded to help enhance the quality of the Tonebook word processing application. Some of these features, which include bootstrap, were timesaving during the development of Tonebook.

Chapter 4: **Implementation**

This chapter will discuss all the technologies used to implement the application as well as describe the way in which the central core functionalities of the application were achieved. A step-by-step process of how successful deployment was achieved will also be described.

4.1 Languages and Frameworks

This will go through each of the frameworks and languages used to create the application, as well as some of the CSS frameworks used.

4.1.1 Python with Django

The language used in the development of this word processing application in python, which is a very high-level programming language. The main reason for using python for the development of this application is because of how versatile it can be as well as the massive number of extensive support libraries it has at its disposal. Before implementation, revision on the python module Algorithms and Data Structures had to be done to refresh my memory of the programming language. This barely helped because of the scale at which I needed to use python. Django is a web framework that I have recently become familiar with through some modules done in the second year.

4.1.2 Vue

This is an open-source front-end JavaScript framework for the primary purpose of building user interfaces. It was used to display the editor, which the authors can use to write text in. Once the text is written and stored, Vue will be used to render the document object model. Vue is also responsible for the effects that occur when a document is deleted, edited, renamed, or created.

4.1.3 Bootstrap and CSS

This is an open-source CSS framework that is used for the creation of websites and web applications. Bootstrap was built on JavaScript and HTML to be used for the easy development of web applications. Due to how easy it is to use it is as well as the responsiveness of the framework, made it a must for ToneBooks development. However, it can be a bit rigid to use on its own, so it was partnered up with standard CSS, whereby bootstrap will do all the heavy lifting, and CSS can be used for some touch ups.

4.2 Essential API’s

Below are the essential APIs which have been used to implement some of the main features of the application:

4.2.1 TinyMCE

TinyMCE is an online text editor which is released as an open-source software where users can convert HTML text area fields into editor instances. This was used in conjunction with Vue to be able to display and create the documents that have been stored in a user’s object. TinyMCE is crucial to the running of the application as all of the formatting of the document will have to be done using the TinyMCE interface. Due to the TinyMCE software, the word count feature, as well as the main tone analysis API, were able to be implemented, as TinyMCE was used as a base for the Grammarly API.

4.2.2Grammarly

The Grammarly API is a software development kit that consists of a lot of tools inside one package. Once the plugin of the Grammarly API is installed and implemented into the code of the project, the user should be able to see grammar and spelling suggestions when the Grammarly app has been implemented. Another vital part of the Grammarly API is the ability to perform a tone analysis as well as give suggestions on possible ways to improve the tone of the text. The Grammarly API was used in conjunction with the TinyMCE so that anything written on the text editor would be checked for mistakes. The Grammarly API also gives the user the ability to set the audience, formality, and type of text the user will write. This information will be used to adjust the tone analysis and to base its suggestions on the information provided.

4.3 Core functionalities

This section will describe how the core functionalities were created. Code snippets were also used to explain these core functionalities:

4.3.1Register page

To be able to log into the application, the user has to be able to register a username and password into the database. The first check that is done is to check whether the username already exists in the database, and if this is true, the user will see an error message stating that the username already exists. From the code below, we can see how this was implemented:

```
def register(request):
    if request.user.is_authenticated:
        return redirect("index")

    if request.method == "POST":
        if
User.objects.filter(username=request.POST["username"]).exists():
            return render(request, "register.html", {
                "error": "The username you have used already exists."
            })

        user = User()
        user.username = request.POST['username']
        user.set_password(request.POST['password'])
        user.save()

        return render(request, "register.html", {
            "error": "You have successfully signed up"
        })

    return render(request, "register.html", {})
```

On the successful registering of the user, there will be a success message stating that the user has successfully registered.

4.3.2Login page

Before users are allowed to use the text editor or even view the files that they have saved previously, they need to log in. This is a security measure that will prevent unauthorised access to the system. The explanation of the login page is done with the assumption that the user has already registered in the system. To be able to login into the website, the user has to provide a username and password, which will then be cross referenced with the username and passwords in the database. If the username exists, then the passwords will be matched up, and if they are identical, then the user will be redirected to the accounts page.

This is the theoretical explanation of how the login works, but below, we can see what the coding will look like:

```
def login(request):
    if request.user.is_authenticated:
        return redirect("index")

    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']

        if auth.authenticate(request, username=username,
password=password):
            auth.login(request, User.objects.get(username=username))
            return redirect("account")

        # Not successful in the log in
        return render(request, "login.html", {
            "error": "Your credentials are incorrect, or your account
does not exist."
        })

    return render(request, "login.html", {
        "info": "Please, enter your credentials."
    })
```

The first part is about whether the user already has a session in the background. If there already is a session, then that means a user is already logged into the system. The second part is just about verifying that the username and password used are already in the system. If the username or password used to log in is incorrect, then the user will see an error message stating that their login credentials are incorrect.

4.3.3Editing a document

After the user has been logged into their account, they will be redirected to the accounts page, whereby they can create new documents, which they will also be able to edit. After the file has been created, there will be an edit button on the side of the filename, which, when clicked, will redirect the user to the editing page. This is where the user can type and add text and images to the system. After they have typed out what they wish, there will be a save button next to the name of the file, which, when clicked, will store all the information the user has typed in. The code below shows what the code will look like in the views.py file.

```
def edit(request, file_id):
    if not request.user.is_authenticated:
        return redirect("login")

    user = User.objects.get(username=request.user.username)
    file = user.files.filter(id=file_id)

    if not file.exists():
        return render(request, 'error.html', {
            "error": "Not authorised",
            "description": "This file either does not exist or you are
not the owner of the file"
        })
    file = file.get(id=file_id)

    content = json.dumps(file.full_obj())

    return render(request, 'edit.html', {
        "file": content,
        "name": file.name
    })
```

The first line is just to make sure that the user is on a session for them to be able to edit a document. If not authenticated, they will be redirected to a login page. As we can see from the code, the user's object will be taken from the database. This will then be cross-referenced if the file they want to edit is available in the objects of the user. If the file does not exist, there will be an error message, and if the file exists, the contents of the file will be rendered in the text editing document page, which is referenced as edit.html.

Chapter 5: **Testing**

Testing is a fundamental part of software development as it can lead to a better-quality application as well as more minor bugs and issues. This chapter will discuss the Django unit testing as well as the user acceptance testing with some sample test cases for the former. This chapter will then be concluded with an overall summary of what can be derived from the tests.

5.1 Django Unit Testing

Testing an application can sometimes be a tedious task as we have to divide the application into smaller modules which will make it easier to test and thus result in more accurate testing results. Django unit testing is a class of code that will run independently from each other to determine whether a module is running correctly or not. This will help identify bugs, therefore, finding out solutions on how to solve these bugs.

Below a few test cases will be looked at to determine whether the following necessary modules are working correctly or not.

5.1.1 Login testing

In this testing scenario, we will be looking at whether login is possible with a set username and password, as well as finding out if the users are authenticated before they are logged into the account. Another aspect of the login which is looked at is whether the login will take us to the account page, which means that we have successfully logged in. The code below has been used to determine whether all of the sections have passed the test.

```
def setUp(self):
    self.client = Client(enforce_csrf_checks=False)

    self.user_acc = {
        "username": "sala",
        "password": "1234",
    }

    self.client.post(reverse('register'), self.user_acc)

def test_authenticate(self):
    self.assertTrue(authenticate(
        username=self.user_acc["username"],
        password=self.user_acc["password"]
    ))

def test_view_login(self):
    response = self.client.get(reverse("login"), follow=True)

    self.assertEqual(response.status_code, 200)
    self.assertTemplateUsed(response, "login.html")

def test_attempt_login(self):
    response = self.client.post(reverse('login'), self.user_acc)
    self.assertRedirects(response, reverse("account"))
```

```
        user = User.objects.get(username=self.user_acc["username"])
        self.assertTrue(user.is_authenticated)

    def test_view_account(self):
        self.client.post(reverse('login'), self.user_acc)
        response = self.client.get(reverse('account'))

        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, "account.html")
```

From the code above, we can see that a dummy username and password had been set to test that the login page was working fine. Authentication is done to find out if the username and password set at the top will be able to be verified and thus log in. The final aspect which is looked at is whether upon successful login, it will lead to the user being redirected to the account page.

5.1.2Register testing

In this testing scenario, we will be looking at whether the register form is working well to determine whether an account can be created. From the code below, we can see the process used to check the quality of the registration of the system and whether a new object with these details will be created.

```
    def setUp(self):
        self.client = Client(enforce_csrf_checks=False)

        self.user_acc = {
            "username": "sala",
            "password": "1234",
        }

    def test_view(self):
        response = self.client.get(reverse("register"))

        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, "register.html")

    def test_register(self):
        response = self.client.post(reverse("register"),
self.user_acc)

        self.assertEqual(response.status_code, 200)
        self.assertEquals(User.objects.all().count(), 1)

        user = User.objects.filter(username=self.user_acc["username"])
        self.assertTrue(user.exists())
```

We can see that a setup account was made in the first line with a username and password. The first test is to check whether the register.html file will render. The second test is to attempt to register with the setup account above. If the registration was successful, then the amount of users in the application will change. The last two lines is to check whether the username set at the setup function currently resides in the database.

5.1.3Main page testing

In this testing scenario we will be looking at the main page of the system. This is a simple test as its only purpose is to find out if the index page will be rendered. From the code below we can see how this was tested.

```
class MainPageTest(TestCase):
    def setUp(self):
        self.client = Client(enforce_csrf_checks=False)

    def test_view(self):
        response = self.client.get(reverse("index"))

        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, "index.html")
```

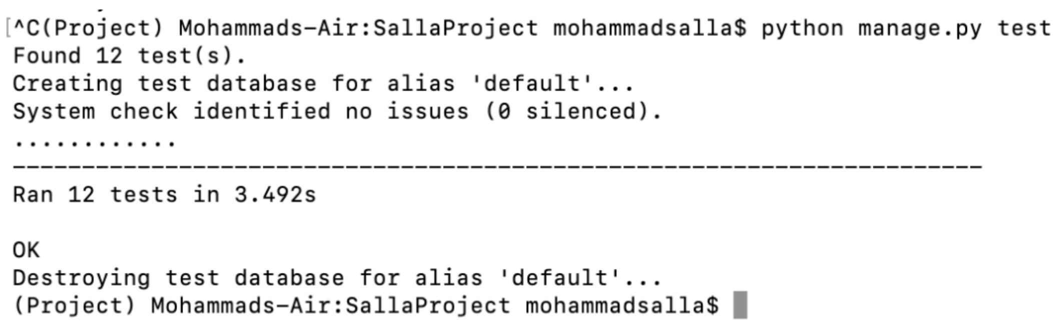
5.1.4API testing

In this testing scenario, we will look at the most crucial aspect of the system, which is the API. The API testing is broken into five main parts. The first part is whether the user will be able to create a new document. The second part is whether the user will be able to rename files they have saved on their accounts. The following part is if the user clicks on edit and starts making some changes to the document, will the save button effectively save all the changes they have made. The next part after this is whether the user will be able to delete a file they have previously worked on. The final part is to determine whether clicking the edit button will redirect the user to the text editing page. Some of the snippet code which was implemented in the Main API class like the setup of the class as well as creating new files and deleting files have been mentioned below.

```
    def setUp(self):
        self.client = Client(enforce_csrf_checks=False)

        self.user_acc = {
            "username": "sala",
            "password": "1234",
        }

        self.client.post(reverse('register'), self.user_acc)
        self.client.post(reverse('login'), self.user_acc)

    def test_create_file(self):

        response = self.client.post(reverse("create"),
data=json.dumps({
            "name": "File Name",
        }), content_type="application/json")

        self.assertEqual(response.status_code, 200)

        user = User.objects.get(username=self.user_acc["username"])

        file = user.files.filter(name="File Name")
        self.assertTrue(file.exists())
        file = file.get(name="File Name")
        self.assertEqual(file.name, "File Name")


    def test_delete_file(self):
        self.client.post(reverse("create"), data=json.dumps({
            "name": "File Name",
        }), content_type="application/json")

        user = User.objects.get(username=self.user_acc["username"])
        file = user.files.get(name="File Name")

        response = self.client.post(file.obj()["delete"])
        file = user.files.filter(id=file.id)

        self.assertEqual(response.status_code, 200)
        self.assertTrue(not file.exists())
```

5.1.5 Summary of the results of the different testing functionalities

All of the test cases mentioned above were put together in the tests.py file and run to determine the results of each of the tests. An error message will be displayed if there is an issue with a particular test case, and an okay sign will be displayed if there are no issues that arose from testing the different test cases. Below we will find the results of all of the tests.



```
[^C(Project) Mohammads-Air:SallaProject mohammadsalla$ python manage.py test
Found 12 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
............
----------------------------------------------------------------------
Ran 12 tests in 3.492s

OK
Destroying test database for alias 'default'...
(Project) Mohammads-Air:SallaProject mohammadsalla$ 
```

Figure 11: Test results

As we can see from the results, they were a total of 12 tests run and all of them have been run with no issues or bugs being found regarding their functionalities.

5.2 Browser Compatibility Testing

An essential aspect of the system is to be able to work on different web browser applications. This is essential because no matter what browser the user has installed on their computer, they need to be able to use the online word processing application. The three most popular web browsers around were used to test whether the web application was able to be run on them. The applications include Google Chrome, Safari, and Mozilla Firefox. Below we can find the results of what worked and what was unable to work in each of these applications:

Browsers	Features that passed	Features that failed
Google Chrome	All features	
Mozilla Firefox	All features	
Safari	Most Features	-Date modified and created -Issues with saving the application

Figure 12: Browser Testing

5.3 User acceptance testing

The purpose of user acceptance testing is to compare how the end-to-end flow of the online application was compared to the functional requirements. This is done to determine whether the application has met its specified criteria. This test was done by positioning two volunteers as end-user’s while taking into account the amount of user experiences a user would have.

All the major functionalities were thoroughly tested by the volunteers, and a few usability issues were determined. One of these usability issues is to display a confirmation message before a user can delete a file. Currently when a user deletes a file it only takes one click to delete the file. It would be more sensible to add a confirmation message like “Are you sure you want to delete this file?”.

Another usability issue which both volunteers referred to was that when they create a document the pop up window which they use to enter the filename does not automatically close when the user clicks on create.

These were the main issues brought forward by the volunteers of the user acceptance testing. Some other comments include wanting to be able to rename the file in the document editing page rather than the accounts page. Others include increasing the aesthetic of the webpages to make it look more appealing. Improving the overall website structure will make it easier to understand where all the pages are. This last issue resulted in a navbar at the top of each page so that users can clearly identify where they are and whether they want to move to another page.

5.4 Summary

The testing, which took place in the form of Django unit testing as well as a user acceptance testing, has shown that Tonebook Online e-book maker has successfully accomplished the main functionalities it was originally set out to complete. The results in their entirety have shown that Tonebook will be able to successfully give authors the ability to enhance the quality of their written work.

Chapter 6: **Evaluation**

In this chapter, the evaluation method used to gauge the effectiveness of the system will be described. This will be followed by a discussion of the limitations of the system. To conclude, a summary of the overall evaluation will be used to conclude this chapter.

6.1 Heuristic Evaluation

For the heuristic evaluation process, a trained volunteer managed to analyse the online word processing system. This was done so that an unbiased analysis of the usability issues and flaws could be analysed. The volunteer was briefed on the critical aspects of the system to be aware of, as well as what they should consider when giving a value to the heuristics.

After the briefing was completed, the volunteer used the online web application to rigorously test the system and try and elicit some of the flaws or issues with the system. The volunteer was given a set of 10 heuristics to be the main focus of their analysis of the system. The reason for this was to ensure that the volunteer could come up with an educated rating of the system with a structured approach.

A heuristic evaluation contains a severity rating of each of the usability principles. The ten heuristics analysed for Tonebook online word application have been stated below next to a severity rating.

Here is a key for the severity ratings:

0 = Not a usability problem at all
1 = Cosmetic problem only
2 = Minor usability problem
3 = Major usability problem
4 = Usability catastrophe

These are the results of the heuristic evaluation:

Usability Heuristic	Severity Rating (0-4)
Visibility of system status	1
Match between system and real world	0
User control and freedom	0
Consistency and standards	1
Error prevention	1
Recognition rather than recall	0
Flexibility and efficiency of use	2
Aesthetic and minimalist design	1
Help users recognise and recover from errors	3
Help and documentation	2

Figure 13: Heuristic Evaluation Table

6.2 Limitations

Overall, the implementation of Tonebook was successful in achieving the requirements defined in the requirements chapter, but it still has its limitations.

The first significant limitation was that the TinyMCE API has a premium version which requires funding to be able to implement. Some of the premium features available would have enhanced the quality of Tonebook by a large margin. If funding was available, the extra features that could have been used include but are not limited to a collaboration system. Another essential feature is that of exporting documents into a pdf format for easy electronic distribution. Advanced tables and extra image editing features would have also been available.

Tonebook relies on users copying data they have and pasting it into the application rather than having an import function that would have allowed users to select a document from the user's local file system and open it using the Tonebook application.

Another limitation was the sample size of users available to test the system, as this would have given much more feedback on the issues at hand. For the user acceptance testing, a single volunteer was used to determine the quality of the project, while a bigger sample size would have been of more benefit.

However, this does not take from the fact that the user testing was showing some encouraging results.

6.3 Summary

The aim of building an online-based application to help authors create projects of the highest quality has been achieved to some extent. This is based on the heuristic evaluation. However, this does come with a few downsides, like some of the limitations mentioned above.

Chapter 7: **Legal, Social and Ethical issues**

This chapter will describe some of the legal, social and ethical issues which were raised from the development of the Tonebook word processing application.

7.1 Legal Issues

The major concern that was brought up when it comes to the legal side of the project is that of storing the authors data. This is mainly because several laws and regulations have been implemented to protect the distribution and storage of personal data, of which the most prominent is the General Data Protection Act (GDPR, 2022). Since authors data can be very data sensitive it was imperative to create an application that could store the authors data in a secured database. To secure the database Django’s built-in database was used as it is more secure to store data from the user on the framework it is currently residing on. Another legal issue that of concern was the storing of personal data like name, email and date of birth. Due to how sensitive this personal information can be the system was changed to not require any of these details.

7.2 Social Issues

There are a couple of social issues when it came to the development of the online word processing application. A major issue is that the current technology in use is not accessible to people with disabilities. It will be almost impossible for a blind user to be able to use the application, and this is a major issue that needs to be rectified. The implementation of a voice over text system will solve this issue. There is also a lack of training to become proficient with the application. Another issue is that there is no social environment whereby authors can communicate with one another, and this can reduce the likelihood of using the application.

7.3 Ethical Issues

What was morally wrong about the development of the word processing application. Firstly, it does not enable the authors to be able to export their data. This is massive as the authors creation is rightfully theirs, and thus they should have full control on what they can do with their data. This includes being able to save or distribute their files however they wish. As mentioned above it is morally wrong to limit authors with disabilities from using the system. This is also an ethical issue that should be resolve in further updates on the system.

7.4 Sustainability

Tonebook was implemented in such a way that it could be updated with additional features. The separation of some of the main features by using different API’S can result in using better API’s which provide more functionality. The overall structure of the webpages also allows for the easy addition of extra pages with different features.

Chapter 8: **Conclusion**

This chapter will conclude all of my achievements, challenges as well as future improvements I would like to make to my word processing application Tonebook. This will then be concluded with the sustainability of the project.

8.1 Achievements

Overall, the project has managed to tick off many of the requirements that have been made in chapter 3, therefore I consider it a huge success. Since I was introduced to web programming, it has always been an ambition of mine to create a web application on my own with a specific end-user in mind. With the help of Django, I managed to progress the word processing application to a state I never thought I would have reached. This has given me immense joy as all the challenges faced were mostly overcome.

Furthermore, I started this project with very few technical skills, but over the course of the semester, I have managed to learn and develop myself as a programmer. I have gained an insight into system development as I had to deal with all the different stages to come out with the overall project. I have managed to apply all I learned in the Software Engineering module as well as parts of the Graphical User Interface module.

In addition to this, I have learned many vital lessons about how important planning is as well as what to do and what not to do when a stage of the system life cycle is past the originally intended deadline. All of these have combined to give me more confidence in both front and backend development, as well as improved my programming skills in Python, JavaScript, Html, and CSS.

Moreover, I managed to work with text editing technologies like TinyMCE, which gave the authors the ability to use numerous editing features which can be influential in creating an excellent quality project. It was very satisfying to have finally figured out a way to save the documents the user has developed into the built-in Django database.

8.2 Challenges Faced

I have faced many challenges in developing this online word processing application, and some of these include having issues deploying the application to the server. There were a whole bunch of errors that came up, but with proper debugging techniques as well as the use of stack overflow, I managed to fix all of the issues and deployed the application successfully. This occasionally happened because as the project was progressing, I was adding different libraries and APIs to achieve the requirements successfully.

However, the biggest challenge faced when developing the application was implementing the text editing API TinyMCE into the system. Managing to add text editing software into the website was critical to the success of the system. After using a few other text editing software, I realised that most of them had very little functionality and were very difficult to run alongside the built-in Django database. After so many failures and shortcomings of other text editing software, I eventually found out about TinyMCE. This had all the editing features that I needed for the system, as well as making it very easy to implement into the Django framework compared to other text editing APIs.

Another challenge was time management as balancing the project development with other school assignments, which included numerous group assignments. This coupled with the fact that I had part-time work as well as societies, made it very challenging for me to find time to develop the project. But with sheer dedication and unwavering support from my friends and family, I managed to create time and make all of this possible. This project would also not have been successfully completed without the assistance and guidance of my supervisor.

8.3 Future Improvements

There are a couple of future improvements I would like to make to further enhance the quality of the online word processing application. One of these is adding higher-level security to the project, like two-factor authentication or different forms of encryption to store the authors' data. This will be a massive boost to the system as a major concern from existing systems is the security of their projects. Authors' projects can be very data sensitive, so it is something to consider for the future improvements of the system.

Another substantial improvement is that of collaboration between authors. Many authors team up to produce very high levels of work, but the current system does not allow for this to happen. One possible way of accomplishing this would be to allow authors to befriend other authors. This way, when an author is working on a project, they can have access to a share button that will open their list of friends, and from there, they can share that document with other authors. This will allow both of them to work on a project at the same time. This could further be enhanced by creating a chat box on the right of the page, which will allow for the authors to communicate with one another.

The final improvement I would like to have, is to give the authors the ability to import and export documents. Currently, the authors cannot import a document into the system because there are no current features in the system that support for this to happen. This, coupled with the fact that authors cannot export documents is warrant for a huge new improvement to address this issue.

8.4 Summary

The successful completion of this project has proven how important it can be to have an online word processing application with the capabilities my project has. Furthermore, through extensive background research, I have become aware of a gap in the market for online word processing applications with functionalities like tone analysis, among other things. I have also learned how crucial it is to understand how the interface should be structured, as the ease of use of a system can be vital to its success.

To conclude, after dealing with so many challenges and coming from it to create a word processing application with the main functionalities aimed at the start, I consider this overall application a success. I still believe there is still room to enhance the quality of this project further.

References

BANGERT-DROWNS, R., 1993. THE WORD PROCESSOR AS AN INSTRUCTIONAL TOOL: A META-ANALYSIS OF WORD PROCESSING IN WRITING INSTRUCTION. REVIEW OF EDUCATIONAL RESEARCH, 63(1), PP.69-93.

CODELESS. 2022. CALMLY WRITER REVIEW 2019 - CODELESS. [ONLINE] AVAILABLE AT: <HTTPS://CODELESS.IO/BEST-WRITING-APPS/CALMLY-WRITER/#:~:TEXT=CALMLY%20WRITER%20IS%20A%20FREE,ON%20YOUR%20PHONE%20OR%20COMPUTER.> [ACCESSED 28 APRIL 2022].

DALTON, D. AND HANNAFIN, M., 1987. THE EFFECTS OF WORD PROCESSING ON WRITTEN COMPOSITION. THE JOURNAL OF EDUCATIONAL RESEARCH, 80(6), PP.338-342.

GENERAL DATA PROTECTION REGULATION (GDPR). 2022. GENERAL DATA PROTECTION REGULATION (GDPR) – OFFICIAL LEGAL TEXT. [ONLINE] AVAILABLE AT: <HTTPS://GDPR-INFO.EU/> [ACCESSED 10 MAY 2022].

HEIM, M. AND GELERNTER, D., 1999. ELECTRIC LANGUAGE. 2. ED. / WITH A FOREWORD BY DAVID GELERNTER - NEW HAVEN: YALE UNIVERSITY PRESS.

HELP.ZOHO.COM. 2022. ZOHO COMPATIBILITY. [ONLINE] AVAILABLE AT: <HTTPS://HELP.ZOHO.COM/PORTAL/EN/KB/WRITER/USER-GUIDE/GETTING-STARTED/COMPATIBILITY/ARTICLES/COMPATIBILITY> [ACCESSED 28 APRIL 2022].

LAI, J. AND CHANG, C., 2011. USER ATTITUDES TOWARD DEDICATED E-BOOK READERS FOR READING. ONLINE INFORMATION REVIEW, 35(4), PP.558-580.

MAXIM, A. AND MAXIM, A., 2012. THE ROLE OF E-BOOKS IN RESHAPING THE PUBLISHING INDUSTRY. PROCEDIA - SOCIAL AND BEHAVIORAL SCIENCES, 62, PP.1046-1050.

MCCARTHY, K., ROSCOE, R., ALLEN, L., LIKENS, A. AND MCNAMARA, D., 2022. AUTOMATED WRITING EVALUATION: DOES SPELLING AND GRAMMAR FEEDBACK SUPPORT HIGH-QUALITY WRITING AND REVISION?. ASSESSING WRITING, 52, P.100608.

MOORE, K., 2015. ARE WE THERE YET? MOVING TO AN E-ONLY COLLECTION DEVELOPMENT POLICY FOR BOOKS. THE SERIALS LIBRARIAN, 68(1-4), PP.127-136.

SCHMIDT, J., HOLM, P., MERRILD, A. AND CHRISTENSEN, P., 2007. LIFE CYCLE ASSESSMENT OF THE WASTE HIERARCHY – A DANISH CASE STUDY ON WASTE PAPER. WASTE MANAGEMENT, 27(11), PP.1519-1530.

YANG, H. AND LIU, T., 2021. THE INFLUENCE OF HANDWRITING AND WORD-PROCESSING ON CREATIVITY IN THE FICTION PRODUCTION: A CASE STUDY OF FAY WELDON'S FICTIONS. 2021 IEEE 21ST INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, RELIABILITY AND SECURITY COMPANION (QRS-C),

ZOHO. 2022. FREE WRITING ASSISTANT | ZOHO WRITER. [ONLINE] AVAILABLE AT: <HTTPS://WWW.ZOHO.COM/WRITER/FREE-WRITING-ASSISTANT.HTML> [ACCESSED 28 APRIL 2022].

Appendix A – Risk assessment

Risk	Risk impacts	Risk level	Likelihood	Prevention
Tone suggestion software is inaccurate.	The Tone suggestion gives inaccurate results.	High	Medium	Try out different methods of making a tone suggestion API.
The spell/grammar checker is inaccurate.	The spell/grammar checker gives wrong suggestions or doesn't pick up on typos.	High	Medium	Try out different spelling/grammar API's and select the most accurate one.
Security is not strong enough.	The lack of proper security will be a case for concern for the users.	High	Medium	Enable secure ways to login to the applications.
Poor user interface	The user interface will look visually unappealing to the user.	Medium	Low	Compare user interface to similar successful models and then arrange for regular feedback on it.
Poor data collection.	Poor quality of data collected.	Medium	Low	Plan out specific data collection techniques that give a better quality of data.
Loss of work	This could result in major complications in the project.	High	Low	Continuously save work and keep backups whenever possible.
Busy schedule	This could result in incomplete project.	High	Medium	Set aside enough time to work on project.
Illness	This could result in major delays in the project.	High	Medium	Communicate to supervisor of illness and if severe enough, send an EC form and resume when recovered.

Figure 14: Risk assessment

Appendix B – Project Plan

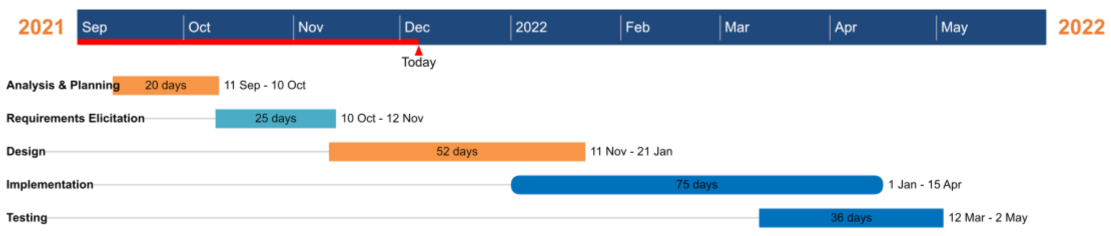


Figure 15: Project Plan