

1. Design Documentation 2

1.1 1. Introduction 3

1.2 2. Class Diagrams 4

1.3 3. User Interface Mockup 6

1.4 4. Summary 18

Design Documentation

Main

This is the Design documentation for COMPSCI 2212 - Fall 2023 - Group 1's project Spell Checker GUI. This document outlines a comprehensive plan for the software system's design, including specific goals and technical specifications to ensure successful implementation.

Tabular Revision History

Below is a table that records the changes made to the requirement documentation. It provides a summary of the modifications made, who made them, and when they were made.

Version	Date	Author(s)	Summary of Changes
1.0	October, 19th 2023	Alishba Farhan	Created the Document
2.0	October, 24th 2023	Fuad Ghareeb	Updates

Table of Contents

The following is an outline of the major sections and sub-sections of the document. It serves as a roadmap for the reader, providing a quick overview of the structure and content of the document

- [1. Introduction](#)
 - [1.1 Overview](#)
 - [1.2 Objectives](#)
 - [1.3 References](#)
- [2. Class Diagrams](#)
 - [2.1 Description](#)
 - [2.2 UML Diagram](#)
- [3. User Interface Mockup](#)
 - [3.1 Wireframe Description](#)
 - [3.2 Wireframe](#)
- [4. Summary](#)
 - [4.1 Brief Description](#)
 - [4.2 Table of Terms](#)

1. Introduction

1.1 Overview

Spell checkers are ubiquitous software tools that have become essential in modern society. They play a critical role in ensuring accurate spelling, which is crucial for effective communication and preventing misunderstandings. By automatically scanning documents for spelling errors, spell checkers improve the credibility of written text and enhance the overall quality of communication.

The purpose of this project is to develop a desktop application that can analyze entire documents by comparing them word-by-word to a dictionary. The application will provide users with a range of options when it encounters spelling errors, including automatic corrections and suggestions for further auto-correction. Once the user has selected all the necessary options, the application will have fulfilled its primary objective of producing an error-free document.

1.2 Objectives

- Applying effective software engineering principles as a team to solve a real-world issue
- Adhering to all the specifications presented
- Referring to the specifications to produce clear models of the design and requirements
- Ensuring good design implementation in Java and dealing with decisions made earlier in the process
- Creating graphical, user-facing content and user-friendly applications
- Writing code clearly and efficiently
- Documenting the Java code in a well-ordered fashion that adheres to best practices
- Constantly reflecting on the good/bad decisions made over the course of the project collectively

1.3 References

- CS2212 Group Project Specification - Fall session 2023
- CS2212 Group 1 Required Documentation

2. Class Diagrams

2.1 Description

The Spell Checker system's UML class diagram provides a comprehensive design for the spell checker application. At the core of the system is the **App** class, it is our main class acting as the gateway to the application's functionality. The App class initiates the user interface, allowing users to interact with the software's features. It holds instances of **Document**, **SpellChecker**, **Metrics**, **Settings**, and **FileHandler**, orchestrating their collaborative interactions. It also has a method to exit the application.

The **FileHandler** class is a pivotal component, streamlining the document's input and output operations. Whether users wish to load or save a document. The **Settings** class lets users tweak the application to their preferences, preserving their chosen configurations and enhancing their experience. It also has methods to get and set delimiters to then be used by the **WORDS** class to parse the words in the document.

The **Document** class represents the user's inputted text. It breaks down the content into individual words. It has methods that allow the user to access the document information like getter methods for character, word, and line count as well as setter methods to update the document to be saved after the user is done editing it. The **Document** class also uses a dependent class **Words** to parse the words based on the delimiters rules. The segmentation and error identification are then carried out by the **SpellChecker** class. This is done by creating a linked list of type **Error** which is another class that represents nodes of the linked list it has various information about the type of the error as well as its location. It also includes a method to display the error and provide the next error in the linked list. The **SpellChecker** also has a method that returns corrections to specific errors. **Correction** class is the class meant to apply changes to the document in case the user decides to this includes correction types like IGNORE, SUGGESTIONS, DELETION, MANUAL, and ADD_TO_DICTIONARY the latter being the user dictionary. The suggestions are provided through the **SpellChecker** when the provideCorrection method is called, it accesses the **DICTIONARY** and **USERDICTIONARY** for suggestions of corrections.

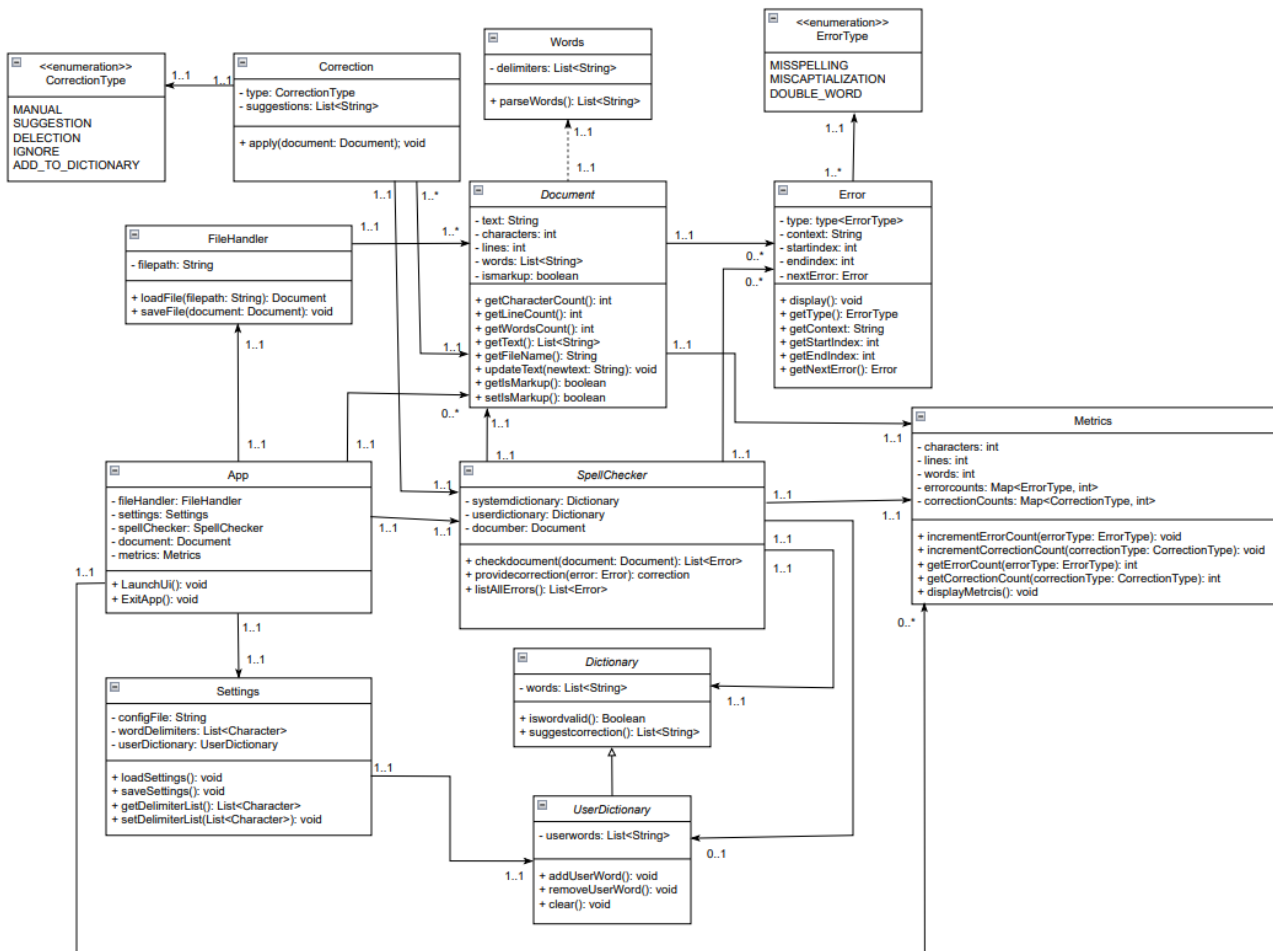
To ensure the SpellChecker operates with a wide and customizable lexicon, it works with the **Dictionary** and **UserDictionary** classes. The latter, designed to inherit attributes and methods from the primary Dictionary, allows for user-specific word entries, striking a balance between a standardized vocabulary and a user-personalized one. Together these two classes detect errors as well as suggest correction for them as requested by the **SpellChecker**.

The **Metrics** class further augments user experience by presenting a statistical breakdown of the identified errors. Such insights help users identify patterns in their mistakes, aiding their learning process. To organize errors and their subsequent corrections neatly, the system incorporates **ErrorType** and **CorrectionType** enumerations. The metrics are collected while the **SpellChecker** is running, it will be calling updating methods like `incrementErrorCount()` and `incrementCorrectCount()`. For displaying the results the **App** class can utilize getter methods in the **Metrics** to get the values that were calculated earlier.

The design approach that is taken when designing the UML diagram is centralization. The idea is to have central classes surrounded by helper classes that do specified functionality. The central classes like **App**, **Settings**, **Document**, **Metrics**, and **SpellChecker**, work with each other to give a seamless user experience.

This is a general view of how the app is going to be implemented, it is prone to updating and iteration as the developers are working on the program.

2.2 UML Diagram



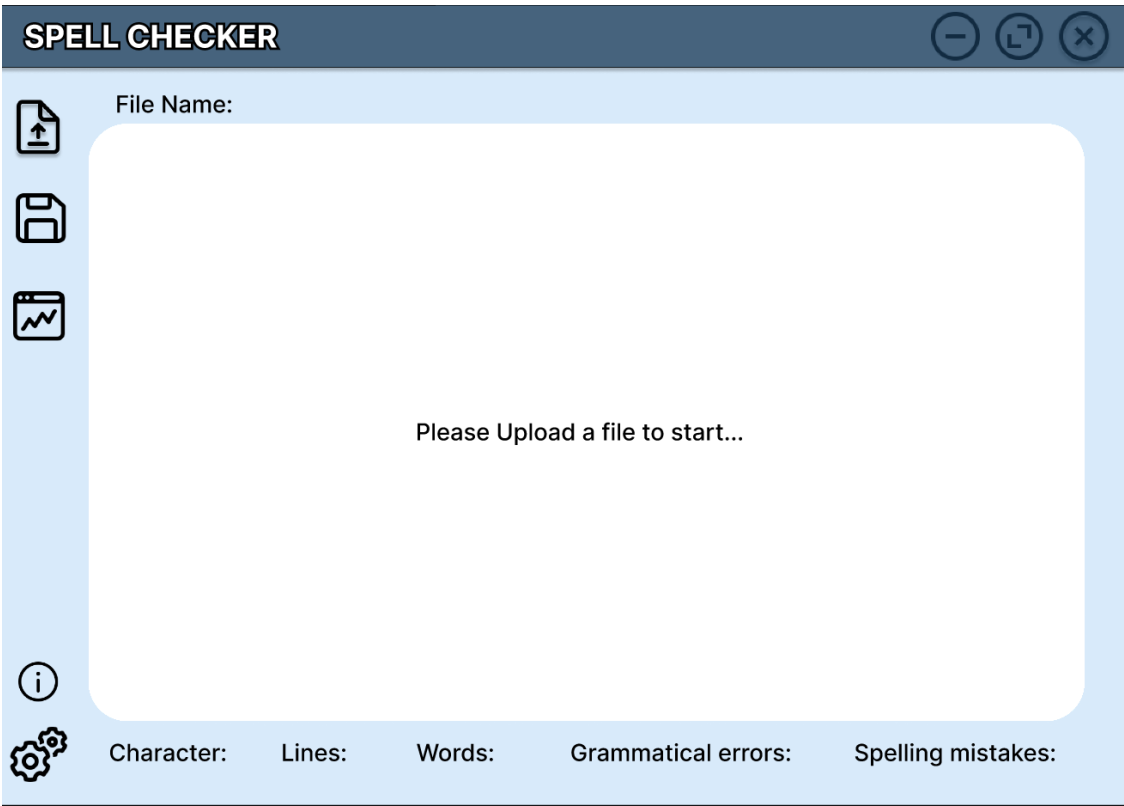
3. User Interface Mockup

3.1 Wireframe Description

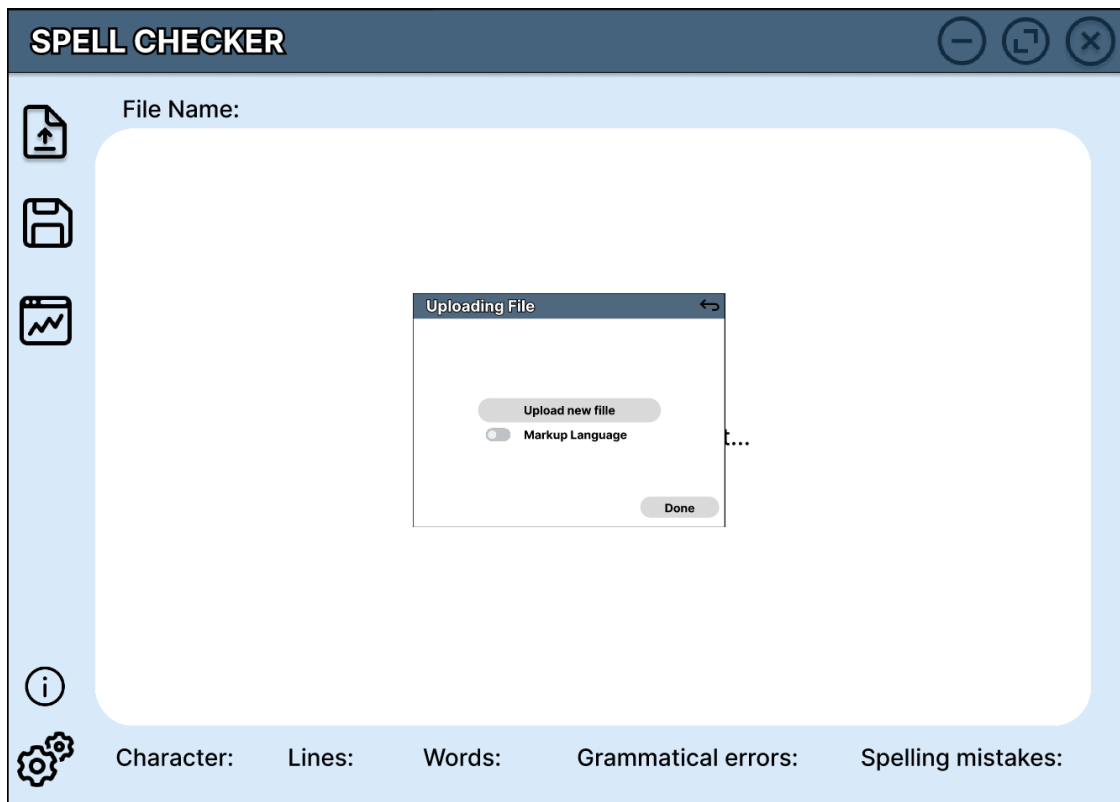
The spell-checker has been designed in such a way that makes it straightforward. Our main objective when developing this wireframe was to present the user with a nicely designed application that was easy to get the hang of. When the user first opens the spell checker, they are introduced to the main page, where all the application's features are displayed for the user to explore and use. When clicking the features, the user is instantly presented with a specific tab that allows you to use it. For example, one of the most important properties of a spell checker is the ability to upload a file. Therefore, if the user were to click the file upload feature found in the application, they would seamlessly be presented with a new tab allowing them to upload their file of choice. The design followed a simple structure of bars where there is a title bar presenting the application name and exit options. Then there is the toolbar to the left with all the interactive features. Upon uploading the file, the user is presented with information about the document at the bottom of the page, that is the information bar. These include character, line, and word counts as well as grammatical and spelling errors. More information can also be found in the metrics feature. The reason behind this design is to cluster related information and features together to make it more intuitive.

After the spell checker successfully does its job of returning an error-free document, the user can then use the save option to either overwrite the current file or save it in a new location. Errors are shown by a blue or red underline to indicate the type of error suggestions for correcting the error then show up when you click on it. How the file is spell-checked will depend on the settings set by the user, which can either be set to use the default or the user dictionary. The user dictionary allows the user to add or remove any words of their choice. This includes a reset option where the user dictionary could be set back to the default state. If the user were to encounter any problems with the application, they can access the information tab. Here, users could get help by either reviewing frequently asked questions or accessing the how-to-use feature, which includes a detailed summary of how the application works. Finally, the users could learn more about the developers behind the spell checker in the About section. Below is a step-by-step guide on how the application works alongside its features and the reasons behind specific design choices.

3.2 Wireframe




Upon opening the spell checker application, on the bottom is what you first see as the user. Features like file uploading, saving and metrics are immediately available for use. This was chosen to make the user experience straightforward.





The first step to using a spell checker is to upload a file. Clicking the upload feature found at the top left enables a pop-up, allowing the user to upload their file of choice with a markup language option. The reason icons are used rather than words is to make the application more universal and intuitive.


SPELL CHECKER




File Name: Sample







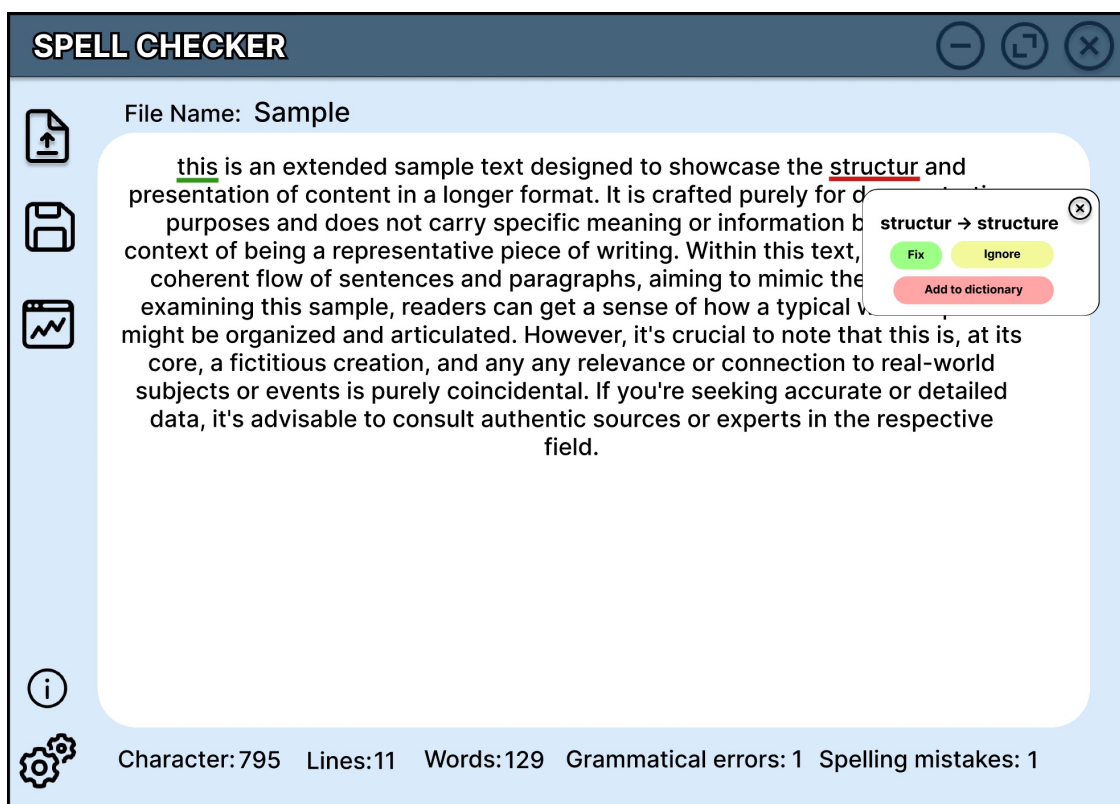


this is an extended sample text designed to showcase the structur and presentation of content in a longer format. It is crafted purely for demonstration purposes and does not carry specific meaning or information beyond the context of being a representative piece of writing. Within this text, you will find a coherent flow of sentences and paragraphs, aiming to mimic the rhythm. By examining this sample, readers can get a sense of how a typical written piece might be organized and articulated. However, it's crucial to note that this is, at its core, a fictitious creation, and any any relevance or connection to real-world subjects or events is purely coincidental. If you're seeking accurate or detailed data, it's advisable to consult authentic sources or experts in the respective field.

Character: 795 Lines: 11 Words: 129 Grammatical errors: 1 Spelling mistakes: 1

After uploading the file, it is spell-checked and information like character, line, and word count as well as grammatical and spelling errors are updated and displayed to the user at the bottom. The file name will also be updated and shown at the top. The placement was chosen to have the information available without the user having to look for it.

8



9

SPELL CHECKER

File Name: Sample

this is an extended sample text designed to showcase the structur and presentation of content in a longer format. It is crafted purely for demonstration purposes and does not carry specific meaning or information beyond the context of being a representative piece of writing. Within this text, you will find a coherent flow of sentences and paragraphs, aiming to mimic the rhythm. By examining this sample, r might be organized and ar core, a fictitious creation subjects or events is pure data, it's advisable to co

Metrics

Total characters, lines, and words.

Number of each type of error detected.

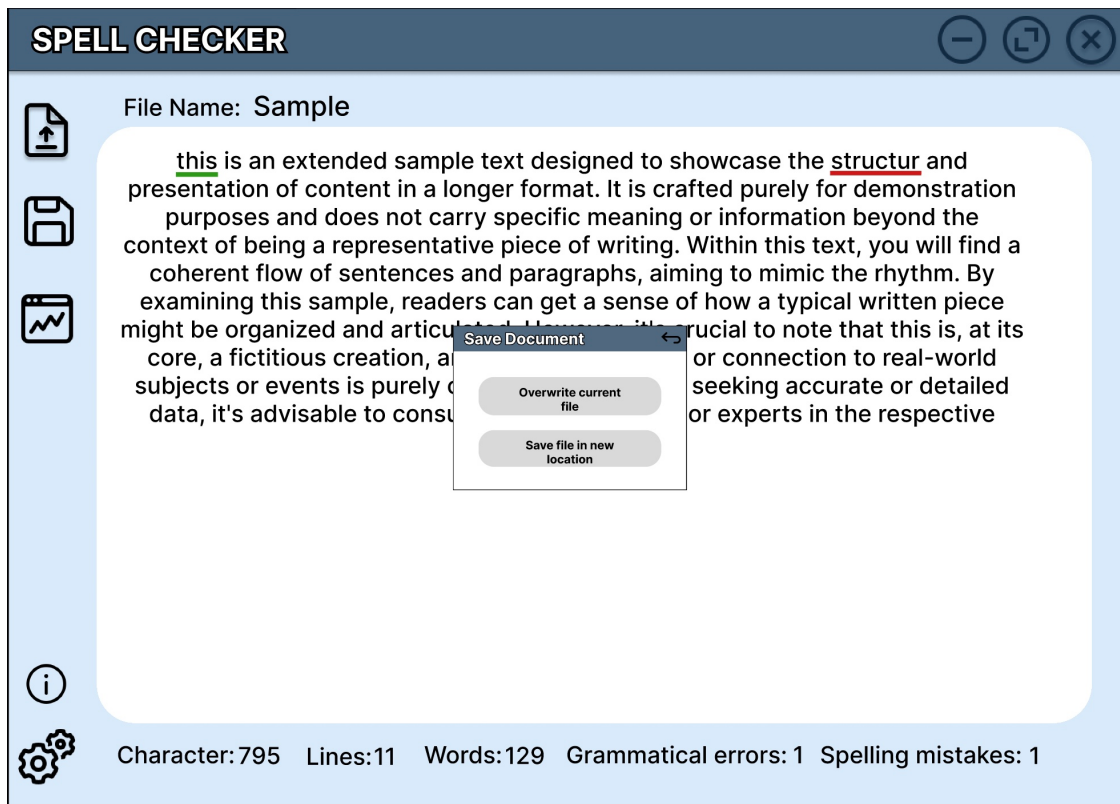
The number of each type of correction applied.

ow a typical written piece al to note that this is, at its connection to real-world eking accurate or detailed experts in the respective

Character: 795 Lines: 11 Words: 129 Grammatical errors: 1 Spelling mistakes: 1

A metrics feature on the left side of the screen has also been added if the user wishes to view additional information about the file. When clicked on, a tab opens and shows additional characteristics of the uploaded document if the user is interested. The reason for choosing popups rather than new pages is to not overload the user's main memory with concurrent activity that can be avoided.

10



After the user has spell-checked their document, they can proceed to save it. A save feature was included under the file upload feature, which when clicked on opens a tab that presents the user with two saving options based on the user's preferences.

Settings

Dictionary selection

☐ User Dictionary

☐ Application Dictionary

User Dictionary settings

Enter a word, press add to append into dictionary or remove to delete from dictionary

enter word..

Add

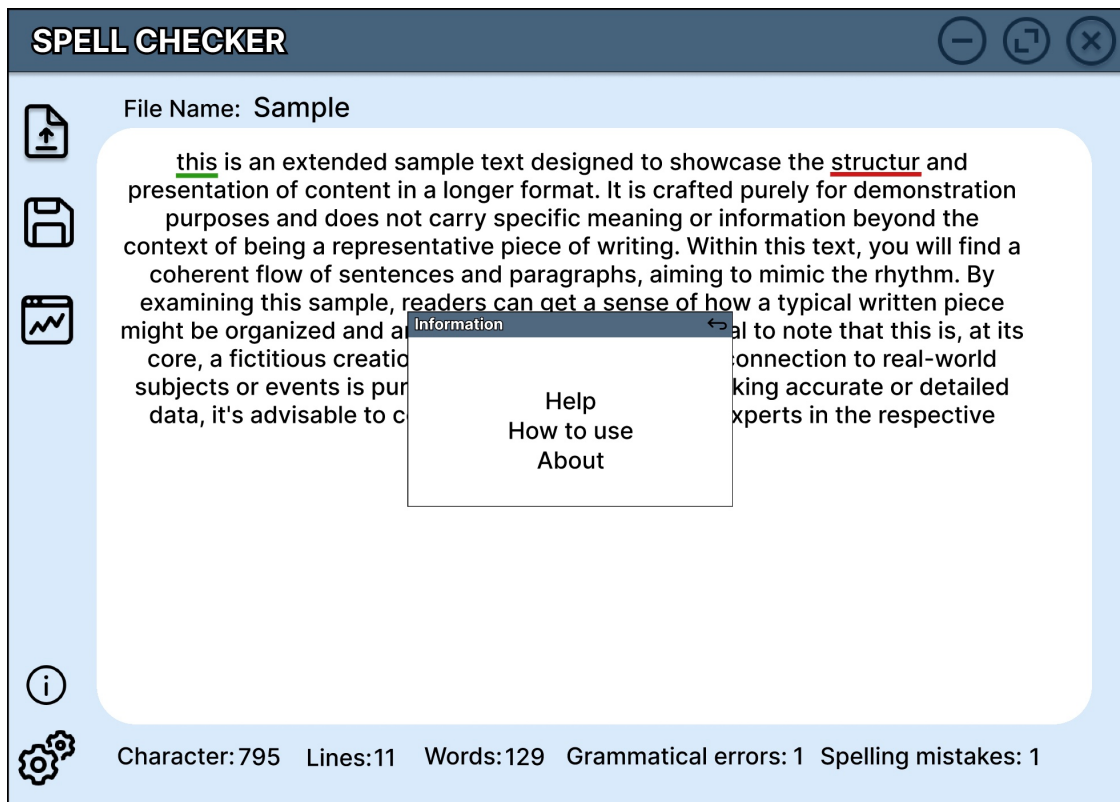
Remove

Erase all entries in the dictionary to default state

Reset dictionary

A settings feature can be found at the bottom left of the screen which opens a whole new tab when clicked on. The reason it is the lowest of all features is because it usually is the one accessed the least. This was added to allow the user to choose between using a User or Application dictionary or both. You can also edit the user dictionary for new words that would not be found in the application dictionary to be introduced.

12



An information feature has also been included right above the settings. This feature was included to assist the user if they encounter any issues when using the spell checker. Clicking this feature opens a tab to present the user with a few options that will help guide them.

Help



Clicking on the Help option guides the user to frequently asked questions about the application. This opens a new tab for the user to find answers to similar questions that they might have had about the application.

FAQ

How to Use



Upload:

Upload text file using upload button. The file needs to be a text file written in English.

Save:

Files can be saved by pressing the save button. The changes can be saved in the same file or a new file in a different location.

All unsaved changes will be lost.

Dictionary:

Preferred dictionary can be set through settings.

Words can be added and deleted from the user dictionary by going to the settings under "User dictionary settings".

Metrics:

Data collected throughout the spell-check can be found in metrics.

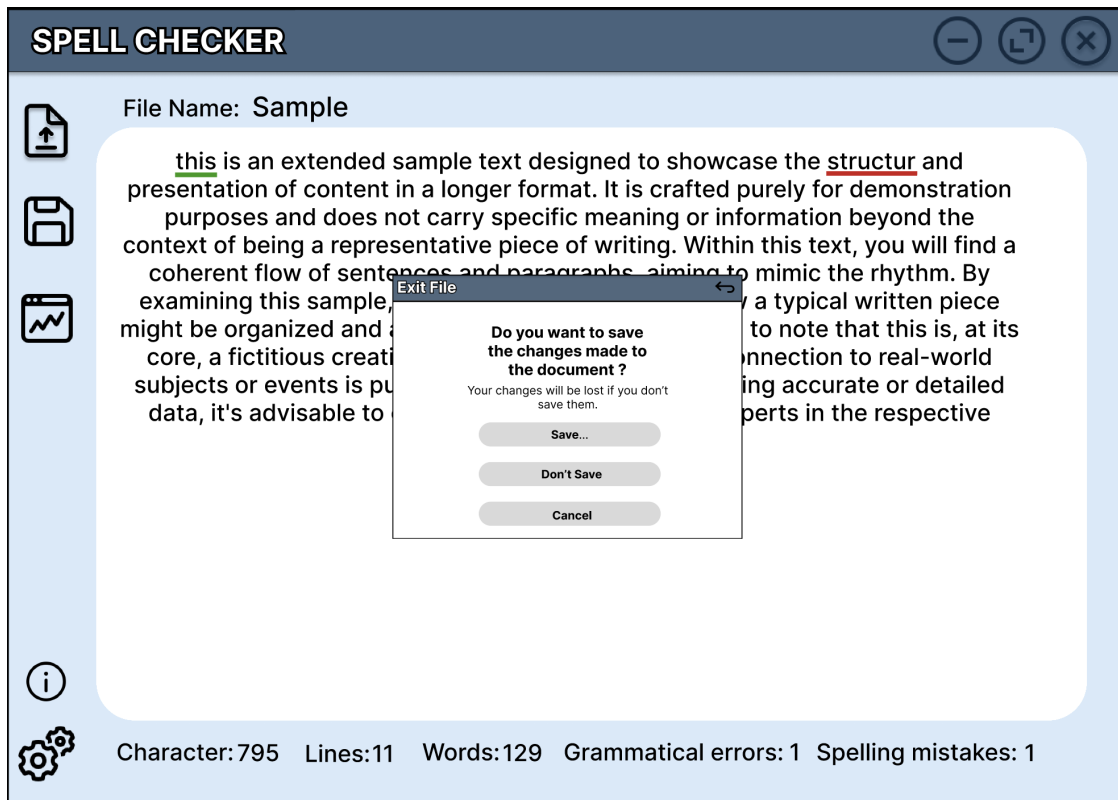
The How to Use option is also included in the information feature, making it easy to find. Upon clicking this option, a new tab displays a summary of how the spell checker works to the user in a way that makes it easy for them to understand and follow through.

About



**This is a spell checker created by
Alishba Farhan, Yazan Hunjul,
Fuad Ghareeb,
Mohammad Halwani, and
Adam Seif for COMPSCI 2212 in
the Fall of 2023.**

In the information tab is also the About option. This was included here for the user to user to easily find the team behind the spell checker, which is displayed in a new tab when clicked on.



Finally, if the user wishes to exit the application, they can do so by clicking on the conveniently placed exit button found at the top right of the application. Upon clicking the button, a new tab is opened. This tab was included to allow the user to save their file if they already haven't or exiting without saving. We also included a warning sign as a heads-up to save.

4. Summary

4.1 Brief Description

Our development team is currently working on a desktop application with spell-checking features for text documents. The application allows users to upload text files for processing. The software verifies each word in the document against a dictionary of correct words and is capable of identifying and correcting improper capitalization and repetition errors. In case of an error, the application underlines the text, with a green underline indicating a grammatical error and a red underline representing a spelling error. Upon clicking on the underlined text, a pop-up window will present users with correction options such as suggestions, ignore, and adding the word to the dictionary. The software also supports a user dictionary, enabling users to choose their preferred dictionary for use within the application settings. The user can easily edit the dictionary with the instructions provided in the application. The application features a metrics function that provides users with data such as word count, misspellings, and repeated words in the document. Our software allows users to exit the application at any time and prompts them to save changes to prevent data loss. Users can save the file in a new location or overwrite the existing file. Our application comes with accessible help resources, including Frequently Asked Questions (FAQs) and instructions on how to use the application.

4.2 Table of Terms

Term	Definition
UML Diagram	A UML diagram is a diagram based on the Unified Modeling Language, with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.
Linked List	A linear collection of data elements whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence.
Delimiters	Something such as a mark or symbol used to show the beginning or end of separate elements.