Bilkent University

Department of Computer Engineering

Senior Design Project

Project Short Name: Clerk

Project Specifications Report

Ahmet Malal, Ensar Kaya, Faruk Şimşekli, Muhammed Salih Altun, Samet Demir

Supervisor: Uğur Doğrusöz

Jury Members:

Innovation Expert: Mehmet Surav

Project Specifications Report
Oct 14, 2019
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1 Introduction

Microsoft Word is one of the most popular word-processing programs around the world. Microsoft Word is used primarily to create various types of documents that you can print and publish, such as books, papers and reports. When you open a file in Microsoft Word, it can be edited using various features that Word provides. Currently, these features are accessible through use of some input devices, such as mouse and keyboard.

In this way, it is assumed that people who are going to use Microsoft Word, should able to use these devices in a conventional way. However, some people can't use these devices effectively or at all. Some might even be too lazy to use them. Some people can't use their hands or are visually impaired. We should not ignore the fact that these people also may want to create some documents and write reports, poems, and maybe a book from scratch.

Let us consider a visually impaired person. They should be able to create, delete, and save files; type and delete sentences; change the font and type of the text; change the position of the cursor and the alignment of the paragraph; change the color of the words; insert images. Briefly, they should be able to use basic functionalities of Microsoft Word. There is no tool currently available that provides use of major functionalities of Microsoft Word without the use of mouse and keyboard together.

In the market, there is an add-in Dictation for Microsoft Word, which enables you to write without using the keyboard using speech recognition. However, this is the only feature of this add-in. Still, there is a need to use a mouse to configure your text in the template. Therefore, we will enhance this by going further so that there won't be any need to use a mouse either.

1.1 Description

Clerk will be an add-in in Microsoft Word designed for users who can't or don't want to type using the keyboard. The user will be able to dictate their commands by giving voice input from an input device. Clerk will take commands from the user's voice input and apply them in Microsoft Word. For instance, when the user says, "Dear John", the program will insert "Dear John" to where the cursor is. It will not only write whatever it hears, but it will also allow users to copy, paste, delete, select a text as well as save the file, make selected text bold or italic and so on. For, instance when the user says, "Save the file as PDF", it will save the file in PDF format.

Clerk will be useful especially for visually impaired people who are not able to use the keyboard and mouse devices. Our application will let them write reports, articles.

When visually impaired users use our add-in, it is likely for the voice recognition algorithm to misunderstand the user. To solve this problem, the program will read the specified text/paragraph or spell a specific word if the user commands the program to do so. Also, for visually impaired people it will be hard to use the mouse as well. To ease the use of Word

for them, we will have commands to set the position of the cursor by specifying the location where they want the cursor to be. For example, "Second Paragraph First Sentence" will take the cursor to the first sentence of the second paragraph.

We are going to use Word JavaScript API to develop the Microsoft Word add-in [1]. HTML, JavaScript and CSS will be used for implementation of our add-in. In short, Clerk will allow users to use Microsoft Word without using a keyboard or mouse.

1.2 Constraints

1.2.1 Implementation Constraints

- Clerk will be developed as a Microsoft Word Add-in application.
- Javascript, HTML, and CSS will be used to develop the add-in.
- Git and Github will be used for version control and collaboration.
- Web Speech API will be used for speech recognition and synthesis [2].
- The application will be built around Object Oriented Programming paradigms.

1.2.2 Economic Constraints

- Clerk will be an open-source Microsoft Word Add-in that will be available to use for free to all Word users and developers.
- Free and open-source libraries and APIs will be used.
- The web page of the application will be on the Github domain, which is free.

1.2.3 Ethical Constraints

- Personal information and private data will not be shared with any third parties.
- Microsoft Word already encrypts the user data, accessible only after authentication.

1.2.4 Sustainability Constraints

- To improve the usability of the application, feedback from the users will be considered.
- The application will be updated by the developers according to feedback taken from the users.

1.2.5 Social Constraints

- Even though the main audience of the application is visually impaired people, the application can be used by everyone.

1.2.6 Language Constraints

- Turkish or English languages will be supported by the application.

1.3 Professional and Ethical Issues

- The consent of the user is required since we will take their voice input and process it.
- The application will not store the voice input content in any way. So, there is no way we can share data with third parties.
- Licenses for third-party APIs and libraries will be checked before usage.

2 Requirements

2.1 Functional Requirements

2.1.1 Receiving Voice Input

- The user will be able to give voice input to the application using any input device.

2.1.2 Converting Voice Input to Text Commands

- The input received will be converted to text to be parsed.
- The text input will be parsed to understand the contents. The application will figure out which functionality the user wants to use, such as typing, copying, pasting, and saving the file.

2.1.3 Providing Microsoft Word Functionality

- After understanding what the user wants to do by parsing the command, the application will be able to execute it.
- The user will be able to command for any functionality provided by Microsoft Word. These functionalities include copy/paste, saving files, inserting images/figures/tables, navigating the contents, adding/removing text, changing format of the text, e.g. font.

2.1.4 Read-back Functionality for Error Checking

- The user will be able to request a read-back of a word, sentence, or paragraph from where the cursor currently is. This means that the program will convert the text in the document to speech and read it to the user.
- The user will be able to stop the read-back and ask for a spelling of a word. This will help find typos.

- The read-back functionality will be integrated with errors found using Microsoft Word's error checking mechanisms, e.g. underlining a word in red that isn't in the dictionary, so that the user will be alerted to these errors real time.

2.1.5 Document Navigation

- The application should provide satisfactory navigation inside the document. It should have commands to be able to navigate to certain sections, pages, paragraphs or sentences. It should also provide functionality to search for phrases within a document.

2.2 Non-functional Requirements

2.2.1 Reliability

- The application must be able to convert voice to text with acceptable accuracy. Errors and misunderstandings must happen rarely.
- The application must not randomly crash. It must be able to recover from most errors. It must be able to communicate the errors and the reasons they occured to the user as much as possible.

2.2.2 Efficiency

- The application must not take more than 5 seconds to process and respond to any voice command. This requirement includes speech to text delay and parsing.

2.2.3 Extensibility

- It must be easy to develop new features and add new functionality into the application. This requires the application to be modular.

2.2.4 Compatibility

- The application will be able to work on multiple platforms that can run Word such as Windows, Mac, iPad and web browser. This is provided by Microsoft Office themselves [3].

2.2.5 Usability

- Once the application is launched, the user should be able to perform all the functionality using only voice commands from an audio input device. There should not be a need to use any other devices.

References

[1] "Word JavaScript API Overview - Office Add-ins | Microsoft Docs," https://docs.microsoft.com/en-us/office/dev/add-ins/reference/overview/word-add-ins-reference-overview?view=word-js-preview, Accessed: 2019-10-11.

[2] "Web Speech API - Web APIs | MDN," https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API, Accessed: 2019-10-12.

[3] "Office Add-ins Documentation - Office Add-ins | Microsoft Docs," https://docs.microsoft.com/en-us/office/dev/add-ins/, Accessed: 2019-10-12.