Bilkent University

Department of Computer Engineering

Senior Design Project

Project Short Name: Clerk

Analysis Report

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

Supervisor: Prof. Uğur Doğrusöz

Jury Members: Prof. Varol Akman and Prof. Çiğdem Gündüz Demir

Innovation Expert: Mehmet Surav

Project Specifications Report
Nov 11, 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. It 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 people might not want to use them. Some people can't use their hands or are visually impaired. There should be a reliable way for these people who also may want to create documents and write reports, poems, and maybe a book.

Let us consider a visually impaired person, for instance. 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 or insert images into the document. Briefly, they should be able to use basic functionalities of a program like Microsoft Word. There is no tool currently available that provides use of major functionalities of Microsoft Word without the use of mouse and keyboard, or similar input devices.

Microsoft allows developers from around the world to develop applications that will extend the functionality of Word. These applications are called "Word add-ins". A Word add-in is an application which is essentially embedded inside of Word and can be launched from within a running Word instance. We want to build our application as an add-in for Word because it is the most popular text editor there is and it fits best with the nature of the work we want to do, which is to extend the conventional text-editor functionalities and make text-editing more accessible for people.

We will build an add-in for Word that will allow users to utilize Word features with voice commands.

2 Current System

Currently, there is a Word add-in called Dictate for Microsoft Word [1] that was developed by a team within Microsoft. It enables you to write without using the keyboard using Microsoft's speech recognition technologies, and also recognizes some punctuation. However, this is the only feature of this add-in. It doesn't provide any other functionalities that we would like to provide in an add-in that enables users to work using voice commands. It was also discontinued as a project recently.

3 Proposed System

3.1 Overview

Clerk will be an add-in for Microsoft Word designed for users who aren't able to or don't want to edit documents using keyboard and mouse. It will be a multi platform application since add-ins can work on any platform that can run Word. 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 Word. For instance, when the user says, "Dear John", the program will insert the words "Dear John" to where the cursor currently is. Additionally, it will allow users to execute common functionality like copy, paste, delete, select a portion of text, as well as save the file, make selected text bold or italic. For instance, when the user says, "Save the file as PDF", it will save the file in PDF format.

Clerk will also have features to detect errors and misunderstandings, and confirm spelling of words. These features will work by reading the specified text/paragraph or spell a specific word if the user commands the application to do so. If Word detects a spelling error or encounters a word that is not in the dictionary, the application will alert the user. Also, for visually impaired people it is not possible to use the mouse to navigate inside the document. 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 [2]. HTML, JavaScript and CSS will be used for implementation of our add-in, which is the standard way add-ins are built for Microsoft Word.

In short, Clerk will allow users to use Microsoft Word without using a keyboard or mouse.

3.2 Functional Requirements

3.2.1 Receiving Voice Input

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

3.2.2 Adjusting Settings for the Input Device

- The user will be able to choose their input device for sound.
- The user will be able to set a threshold value for the lowest intensity of sounds it will receive similar to many voice chatting programs such as Skype or Discord.

3.2.3 Switching Modes

- Clerk will have three modes, i.e. states.

- In sleep mode voice input will be taken but not analyzed unless it is a special, mode switching sequence or a button to switch modes is pressed.
- In speech mode Clerk will take voice input and convert it into text, i.e. it will let the user dictate what they want to type. Commands will not be recognized in this mode.
- In command mode, only commands will be recognized and applied, speech won't be converted to text.
- The application will start in sleep mode. The user can switch to speech mode by using the "Wake Up" command or use a button that will be provided in the user interface.
- When the application is in speech mode, the user will be able to switch to command mode by pressing and holding the space bar.

3.2.4 Converting Voice Input to Text Commands

- Clerk will receive voice commands in command mode.
- 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, or saving the file. Then the command will be executed.

3.2.5 Dictating Text with Voice Input

- In speech mode, the user will be able to dictate the words getting typed into the document with their voice.
- Clerk will recognize punctuation alongside words.

3.2.6 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.
- The user will be able to set the speed at which the words are read by the read-back feature.

3.2.7 Saving a File

- Clerk will provide a command for saving the current file in the format they request.

3.2.8 Document Navigation

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

3.2.9 Selecting a Range of Text

- The user will be able to select and highlight a range of text using voice commands. They can select a specific word, sentence or paragraph, as well as selecting all text within some range specified by them.

3.2.10 Reading a Range of Text to the User

- The user will be able to request Clerk to read them a range of text.
- The application will read words to the user by converting the text in the document to speech.

3.2.11 Deleting a Range of Text

- The user will be able to delete the currently highlighted range of text by giving voice commands.
- The user will be able to delete the word, sentence or paragraph that the cursor is currently on by giving voice commands.
- The user will be able to delete the last sentence of the document or a certain paragraph by giving voice commands.
- The user will be able to delete all text in a document. The user will be prompted to confirm this request.
- The user will be able to delete all text and objects in a document. The user will be prompted to confirm this request.

3.2.12 Cutting, Copying and Pasting Text

- Clerk will provide commands for cutting, copying and pasting text in command mode.
- The user will be able to cut or copy the selected range of text, as well as request the last word, sentence or paragraph to be cut or copied using voice commands.
- The user will be able to paste the portion of text that was previously cut or copied.

3.2.13 Adding Headers and Footers

- The user will be able to add headers and footers to the current page by using voice commands.

3.2.14 Numbering the Pages

- Clerk will provide commands to turn page numbers on and off.
- The user will be able to dictate the position of the page numbers.

3.2.15 Changing Font

- Clerk will provide commands to change font, font size, color; set text to be bold, italic, underlined; put a subscript or superscript as well as to change the color of the text.
- The user can format their text in any way they want, and can also clear all formatting from a range of text.

3.2.16 Adjusting Margins and Spacing

- Clerk will provide commands to adjust margins and spacing inside of a document.

3.3 Non-functional Requirements

3.3.1 Reliability

- The application must be able to convert voice to text with acceptable accuracy. Errors and misunderstandings must happen rarely. When they do happen, the user needs to be alerted appropriately.
- 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.

3.3.2 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. Longer delays could turn the user away from use of the application.

3.3.3 Extensibility

- It must be easy to develop new features and add new functionality into the application. This requires the application to be sufficiently modular, which will be achieved by using Object Oriented Programming paradigms and appropriate design patterns.

3.3.4 Compatibility

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

3.3.5 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. This is especially important since our target audiences include people who aren't able to use these devices.
- Despite working only with voice commands, Clerk will still provide a user interface to its users. Some functionality such as setting the mode and changing voice input options can also be used from the user interface.

3.3.6 Error-Handling

- The application must handle errors as much as possible and provide acceptable feedback to the user about them.

3.3.7 Licensing

- The application will be built using Microsoft Developer Licenses to test in Word.
- Licensing for the third party libraries will be adhered to.

3.4 Pseudo Requirements

- Clerk will be developed as a Microsoft Word Add-in application.
- Clerk will work only when connected to the Internet.
- English language will be supported.
- Javascript, HTML, and CSS will be used to develop the add-in.
- Git and GitHub will be used for version control and collaboration.
- Word Javascript API will be used in development of the add-in [2].
- Common API from Office365 will be used to control the objects and metadata in Word documents [4].
- The application will work on the following platforms: Word 2013 or later on Windows, Word on the web, Word 2016 or later on Mac, and Word on iPad.
- The application will be developed under Object Oriented Programming paradigms.

- Clerk will be an open-source Microsoft Word Add-in that will be available to use for free to all Word users.
- Licenses for third-party APIs and libraries will be checked before usage. Free and open-source libraries and APIs will be used.
- The following speech recognition libraries will be tested in terms of accuracy and response time: Web Speech API [5], Google Cloud Speech API [6], Microsoft Bing Voice Recognition [7]. There are also other options included in the PyPi Speech Recognition library [8].
- Personal information and private data will not be shared with any third parties.
- Microsoft Word already encrypts the user data, which is accessible only after authentication.
- 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.
- 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.

3.5 System Models

3.5.1 Scenarios

Scenario 1 Use Speech Mode with Punctuation
Actors: User
Entry Conditions:

1. The user opens the add-in and clicks "Start Dictation" button or says "Wake Up Clerk".

Exit Conditions:

1. The user says "Shut Down Clerk".
2. The user clicks "Exit" button.

Main Flow of Events:

1. The user opens the add-in and says "Wake Up Clerk".
2. The users says "hello world exclamation mark new line I am the first user of Clerk period the following marks are some of the available punctuation marks colon comma semicolon open quote question mark close quote period".
3. The system prints "Hello world! I am the first user of Clerk. The following marks are some of available punctuation marks: , ; "? " . ".

4. The user says “Go to sleep Clerk”.

5. System stop writing and waiting for input such as “Wake up Clerk” statement or click “Start Dictation” button or “Shut Down Clerk” statement or “Exit” button.

6. The user says “Shut down Clerk”.

Alternative Flow of Events:

1. The user opens the add-in and clicks “Start Dictation” button.

2. The user says “hello world exclamation mark new line I am the first user of Clerk period the following marks are some of the available punctuation marks colon comma semicolon open quote question mark close quote period”.

3. The system prints “Hello world! I am the first user of Clerk. The following marks are some of available punctuation marks: , ; “? “ . ”.

4. The user clicks “Stop Dictation” button.

5. System stop writing and waiting for input such as “Wake up Clerk” statement or click “Start Dictation” button or “Shut Down Clerk” statement or “Exit” button.

6. The user click “Exit” button.

Scenario 2 Use Select, Read and Delete Text Commands
Actors: User
Entry Conditions:

1. The user opens the add-in and clicks “Start Dictation” button or says “Wake Up Clerk”.

2. The user press space button while saying proper select, read, or delete commands.

Exit Conditions:

1. The user says “Shut Down Clerk”.

2. The user clicks “Exit” button.

Main Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.

2. The user press space button and says “Select Word”.

3. The system selects the word that the cursor stands or the word before the cursor.

4. The user presses space button and says “Select Sentence”.

5. The system selects the sentence that the cursor stands.

6. The user presses space button and says “Select Paragraph”.
7. The system selects the paragraph that the cursor stands.
8. The user presses space button and says “Select all text”.
9. The system selects all document.

Alternative Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user presses space button and says “Read Word”.
3. The system reads the word that the cursor stands or the word before the cursor.
4. The user presses space button and says “Read Sentence”.
5. The system reads the sentence that the cursor stands.
6. The user presses space button and says “Read Paragraph”.
7. The system reads the paragraph that the cursor stands.
8. The user presses space button and says “Read all text”.
9. The system reads all document.

Alternative Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user presses space button and says “Delete Word”.
3. The system deletes the word that the cursor stands or the word before the cursor.
4. The user presses space button and says “Delete Sentence”.
5. The system deletes the sentence that the cursor stands.
6. The user presses space button and says “Delete Paragraph”.
7. The system deletes the paragraph that the cursor stands.
8. The user presses space button and says “Delete all text”.
9. The system deletes all documents.

Scenario 3 Use Add Header Footer Command
Actors: User
Entry Conditions:

1. The user opens the add-in and clicks “Start Dictation” button or says “Wake Up Clerk”.
2. The user says “Add header” or “Add footer”.

Exit Conditions:

1. The user says “Shut Down Clerk”.
2. The user clicks “Exit” button.

Main Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user press space button and says “Add header”.
3. The system asks to user the type of header.
4. The user press space button and says “Blank”.
5. The system creates a blank header.
6. The user says “This is a header”.
7. The system writes “This is a header” into header section.
8. The user says “Shut down Clerk”

Alternative Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user press space button and says “Add footer”.
3. The system asks to user the type of footer.
4. The user press space button and says “Blank”.
5. The system creates a blank footer.
6. The user says “This is a footer”.
7. The system writes “This is a footer” into footer section.
8. The user says “Shut down Clerk”.

Scenario 4 Use Add Page Number Command
Actors: User
Entry Conditions:

1. The user opens the add-in and clicks “Start Dictation” button or says “Wake Up Clerk”.

2. The user press space button while saying “Add page number”.

Exit Conditions:

1. The user says “Shut Down Clerk”.
2. The user clicks “Exit” button.

Main Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user press space button and says “Add Page Number”.
3. The system asks the place of page numbers: bottom or top and left, center, or right.
4. The user press space button and says “top right”.
5. The system add page numbers on the top right of the page.
6. The user says “Shut down Clerk”.

Scenario 5 Use Find Replace Command
Actors: User
Entry Conditions:

1. The user opens the add-in and clicks “Start Dictation” button or says “Wake Up Clerk”.
2. The user press space button while saying “Find” or “Find and Replace”.

Exit Conditions:

1. The user says “Shut Down Clerk”.
2. The user clicks “Exit” button.

Main Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user press space button and says “Find”.
3. The system asks the word that is going to be found.
4. The user says “hello”.
5. The system says “The cursor is in the first occurrence”.
6. The user press space button and says “Find Next”.
7. The system says “The cursor is in the second occurrence”.
8. The user says “Shut down clerk”.

Alternative Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user press space button and says “Find and Replace”.
3. The system asks the word that is going to be found.
4. The user says “hello”.
5. The system asks the word that is going to be replace word.
6. The user says “goodbye”.
7. The system says “The cursor is in the first occurrence”.
8. The user press space button and says “Find Next”.
9. The system says “The cursor is in the second occurrence”.
10. The user press space button and says “Replace all”.
11. The system replaces all the “hello” words except first occurrence with “goodbye”.
12. The user says “Shut down Clerk”.

Scenario 6 Use Start Stop Listening Command
Actors: User
Entry Conditions:

1. The user opens the add-in and clicks “Start Dictation” button or says “Wake Up Clerk”.

Exit Conditions:

1. The user says “Shut Down Clerk”.
2. The user clicks “Exit” button.

Main Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user says “Go to sleep Clerk”.
3. The system keeps listening but stop writing.
4. The user says “Wake up Clerk”.
5. The system becomes active again.
6. The user says “Shut down Clerk”.

Scenario 7 Use Align Margin Command
Actors: User
Entry Conditions:

1. The user opens the add-in and clicks "Start Dictation" button or says "Wake Up Clerk".
2. The user selects some text and press space button while saying "Align Margin".

Exit Conditions:

1. The user says "Shut Down Clerk".
2. The user clicks "Exit" button.

Main Flow of Events:

1. The user opens the add-in and says "Wake up Clerk".
2. The user press space button and says "Select all text".
3. The system selects all text.
4. The user press space button and says "Align Margin".
5. The system asks left, right, center, or justify.
6. The user says "left".
7. The system aligns selected text to left.
8. The user says "Shut down Clerk".

Scenario 8 Use Change Font Command
Actors: User
Entry Conditions:

1. The user opens the add-in and clicks "Start Dictation" button or says "Wake Up Clerk".
2. The user says "Change Font".

Exit Conditions:

1. The user says "Shut Down Clerk".
2. The user clicks "Exit" button.

Main Flow of Events:

1. The user opens the add-in and says "Wake up Clerk".
2. The user press space button and says "Select all text".

3. The system selects all texts.
4. The user press space button and says “Change Font”.
5. The system asks user to change at least one of the font, font style, size, font color, underline style and underline color.
6. The user presses space button and says “Font style Bold italic”.
7. The system changes the font settings of the selected text.
8. The user presses space button and says “Font Size 12”.
9. The system changes the font size to 12.
10. The user says “Shut down Clerk”.

Scenario 9 Open How to Use Section
Actors: User Entry Conditions:

1. The user opens the add-in and clicks “Start Dictation” button or says “Wake Up Clerk”.
2. The user says “How to Use” or clicks “How to Use” button.

Exit Conditions:

1. The user says “Shut Down Clerk”.
2. The user clicks “Exit” button.

Main Flow of Events:

1. The user opens the add-in and says “Wake up Clerk”.
2. The user press space button and says ”How to Use”.
3. The system opens How to Use screen which shows how to use which command and how to add punctuation.
4. The user says “Shut down Clerk”.

3.5.2 Use Case Model

General Use Cases

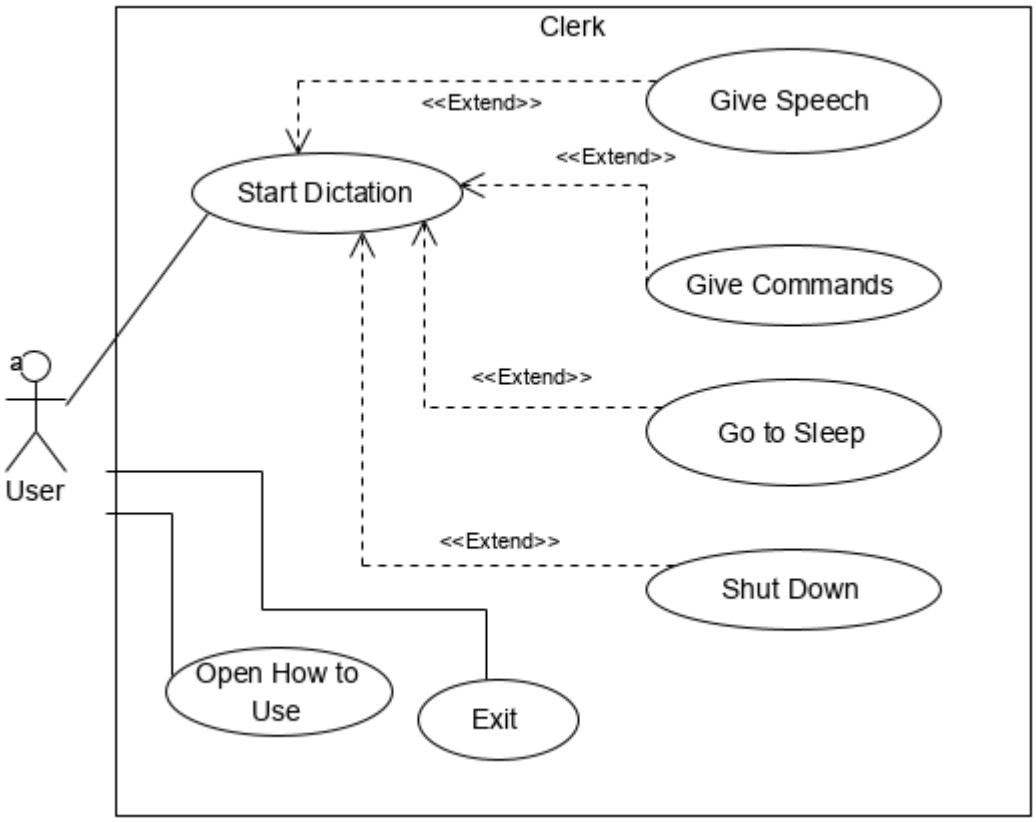

Figure 1: General use case diagram.

Command Mode Use Cases

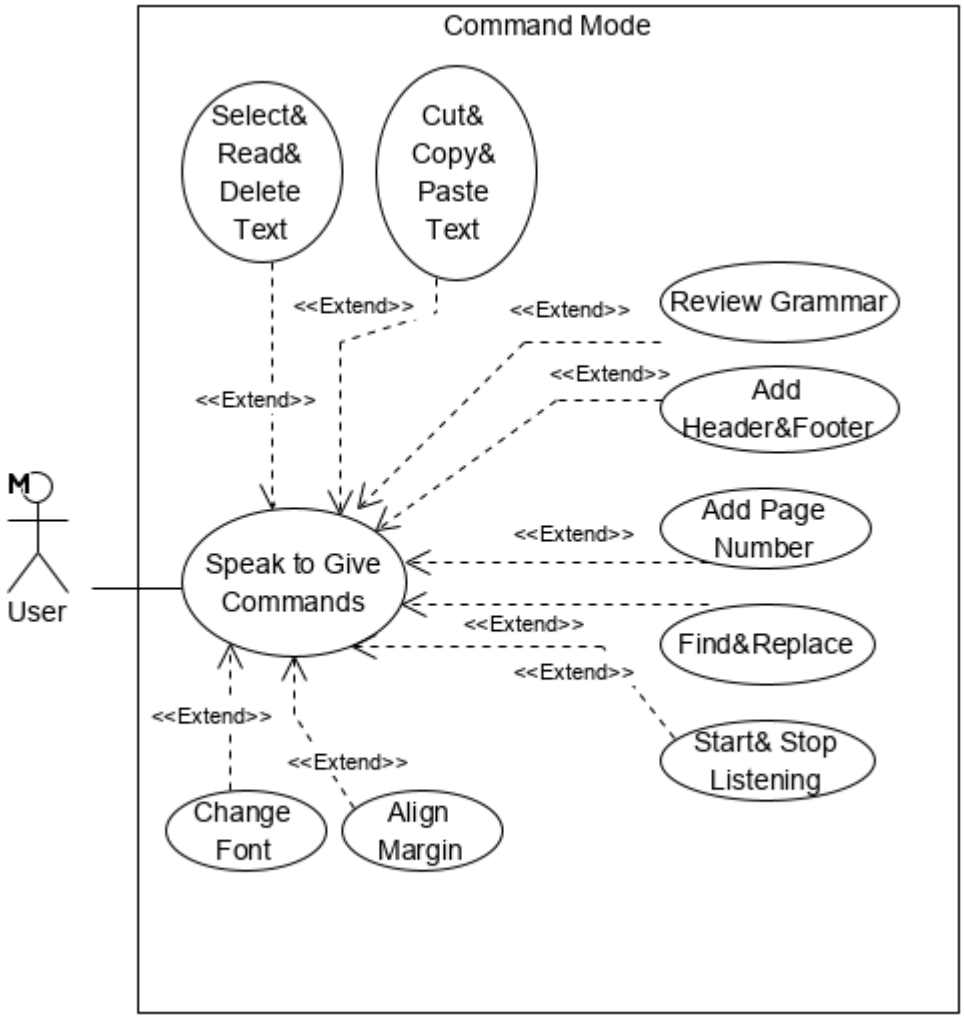

Figure 2: Command mode use case diagram.

Speech Mode Use Cases

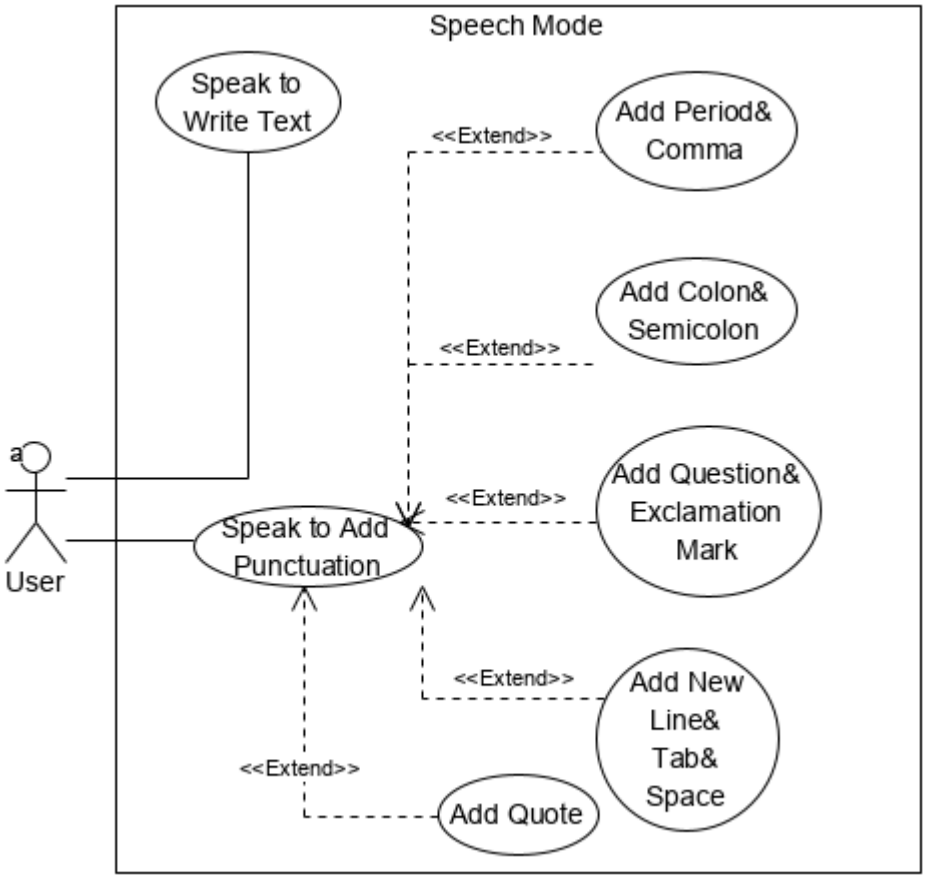

Figure 3: Speech mode use case diagram.

3.5.3 Object and Class Model

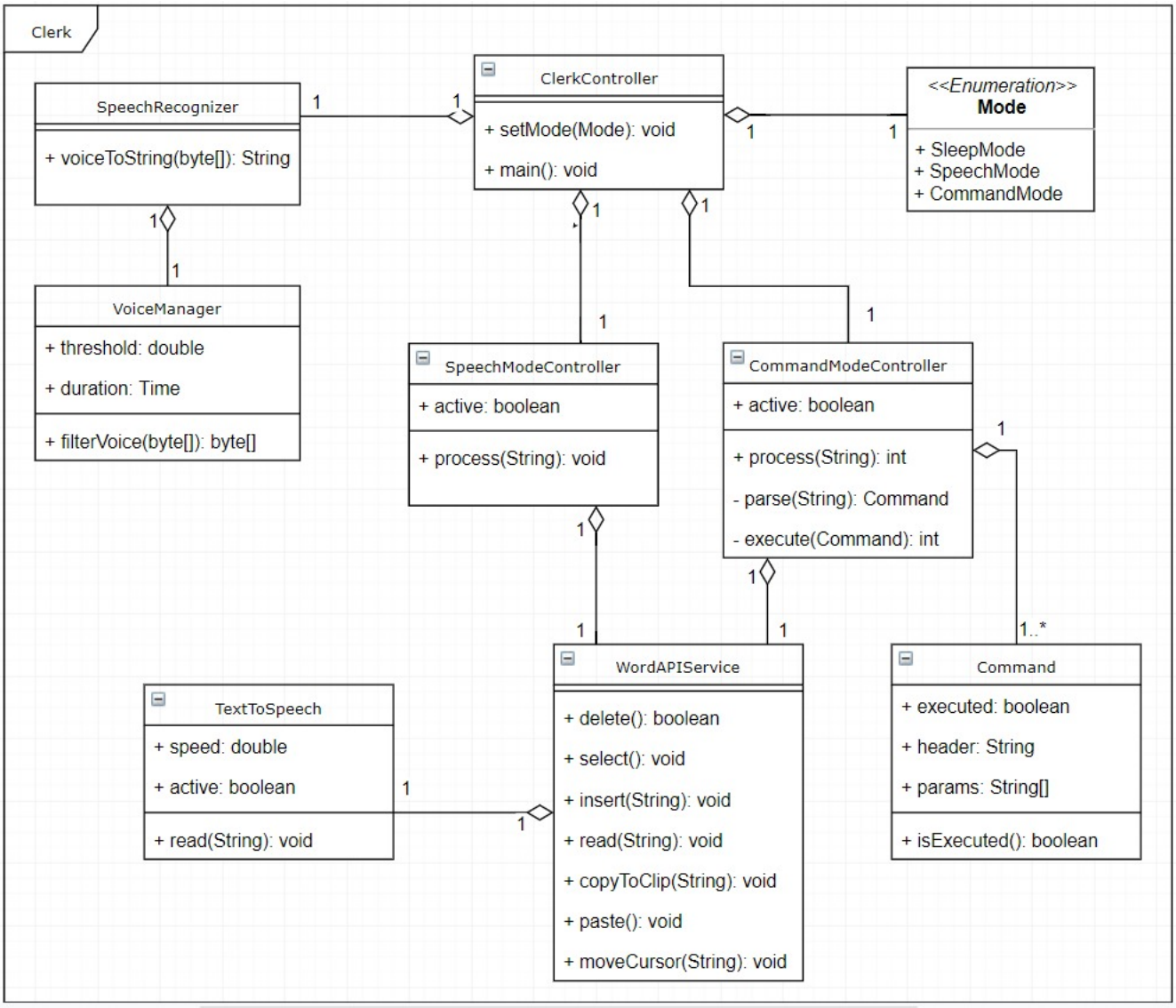

Figure 4: Class diagram for Clerk.

3.5.4 Dynamic Models

Sequence Diagrams

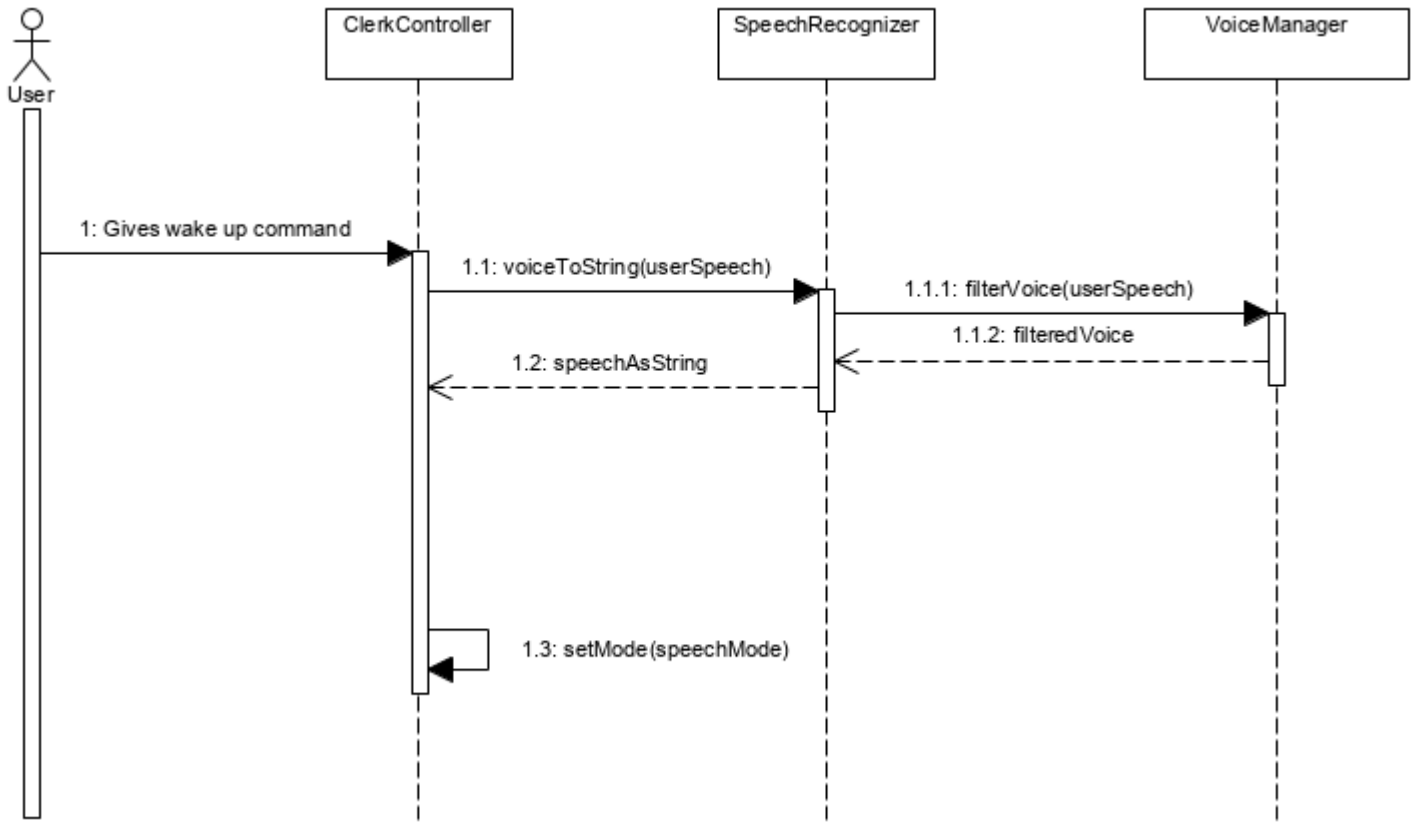

Figure 5: Change to speech mode sequence diagram. When the user gives a "Wake Up" command, Clerk will set itself to speech mode and start listening to dictation and commands.

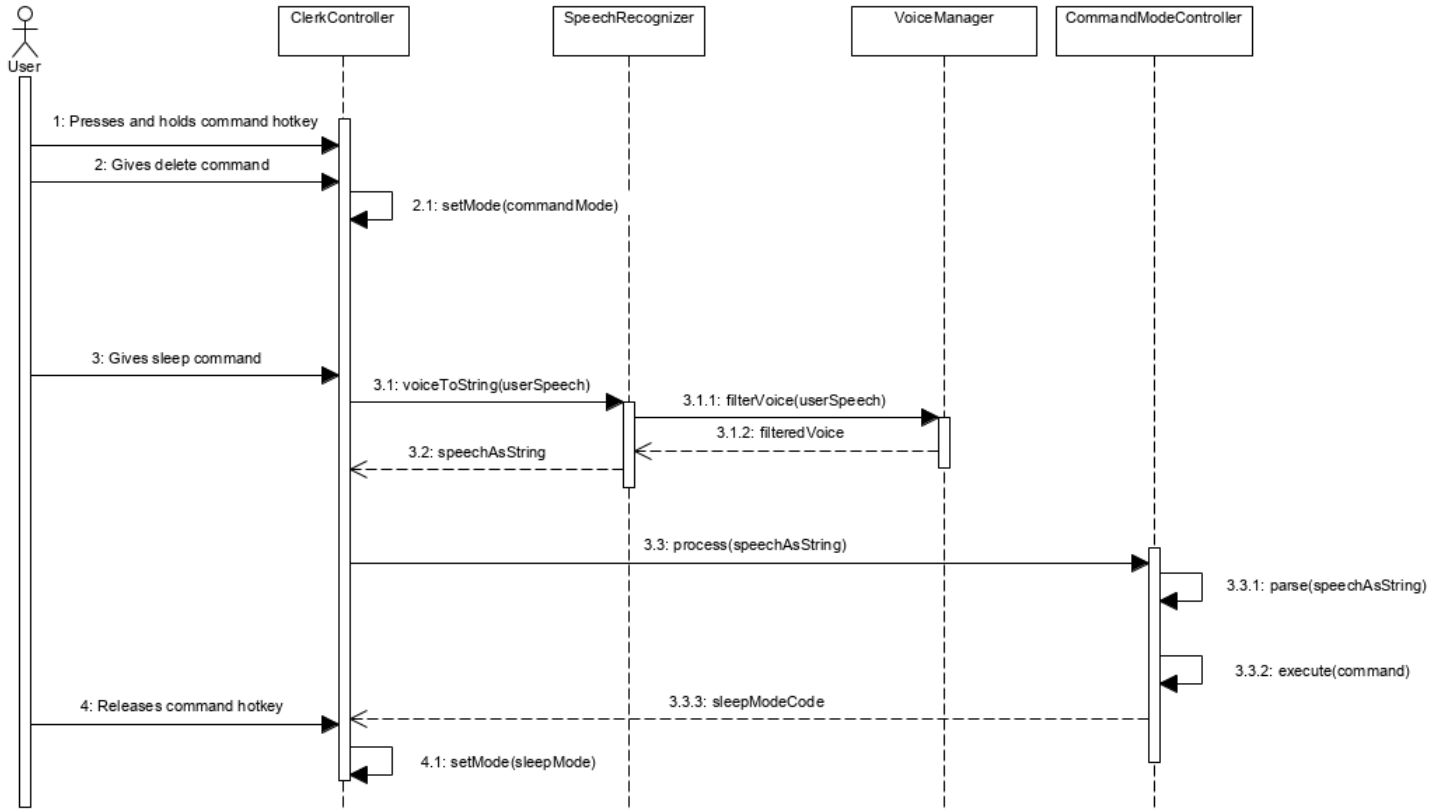

Figure 6: Change to sleep mode sequence diagram. When the user gives a "Go to Sleep" command, Clerk will set itself to sleep mode and stop listening to dictation and commands.

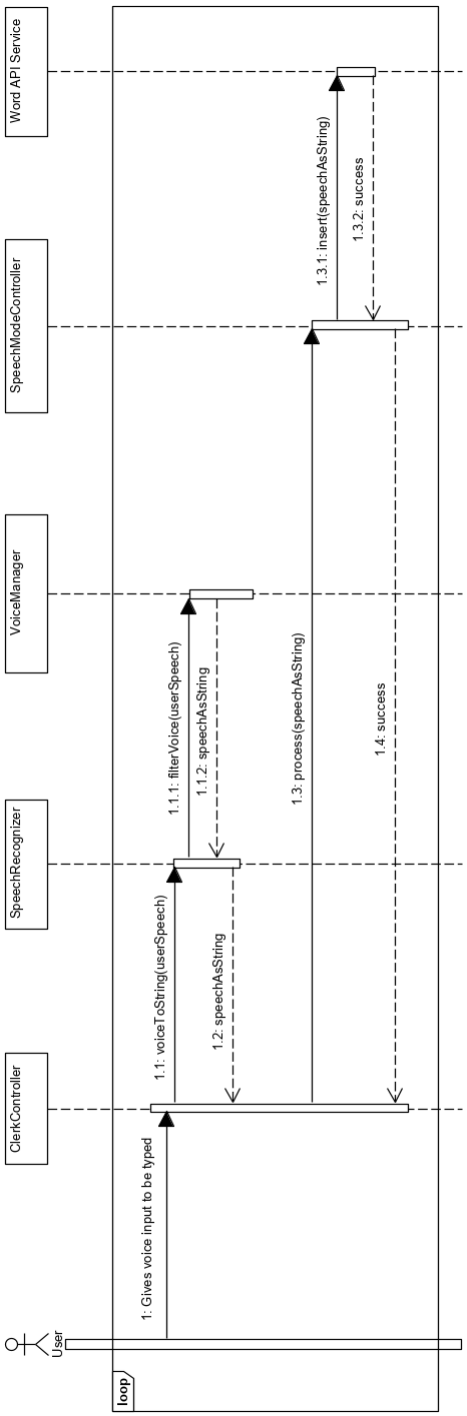

Figure 7: Voice dictation sequence diagram. The diagram shows what is essentially the main program loop for dictation. As the user keeps giving commands in speech mode, Clerk will keep inserting text. Commands will be ignored in speech mode.

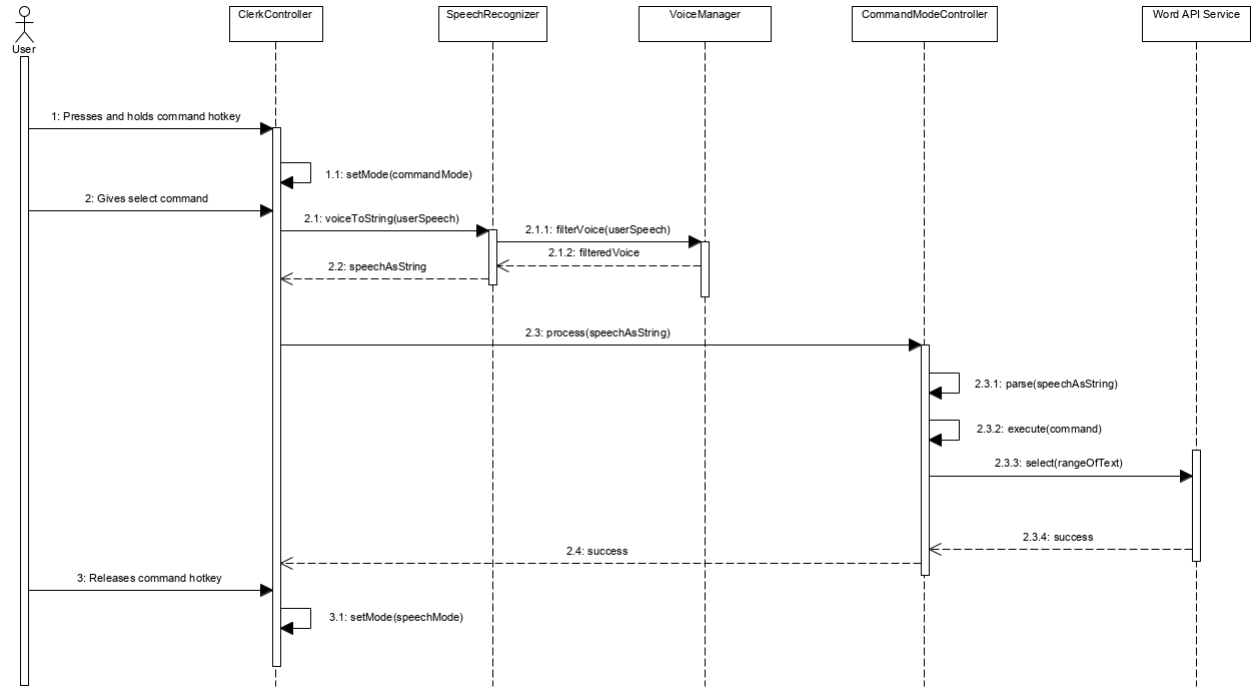

Figure 8: Select command sequence diagram.

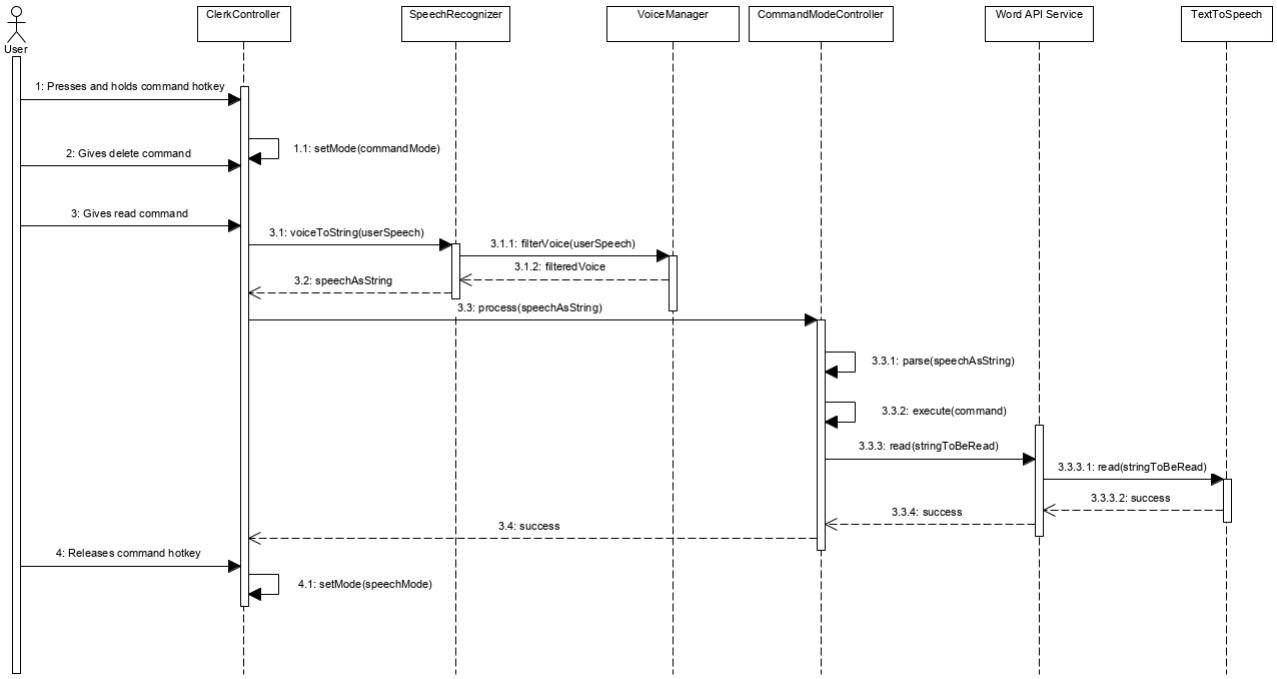

Figure 9: Read command sequence diagram.

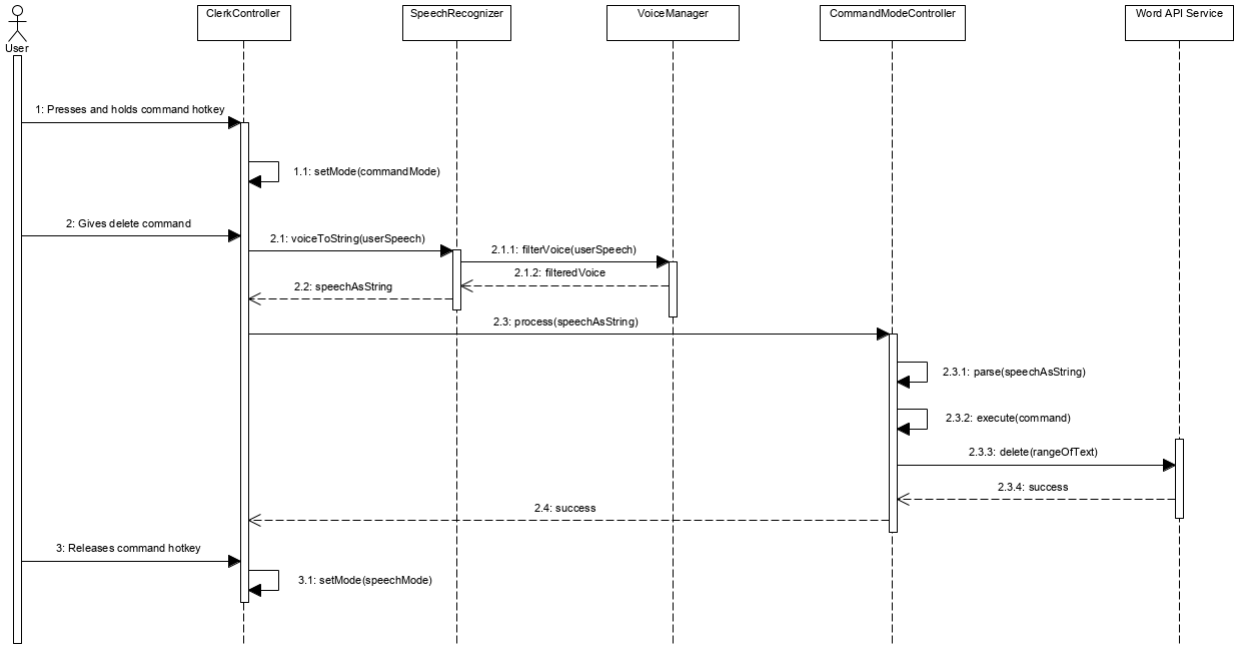

Figure 10: Delete command sequence diagram.

Activity Diagram

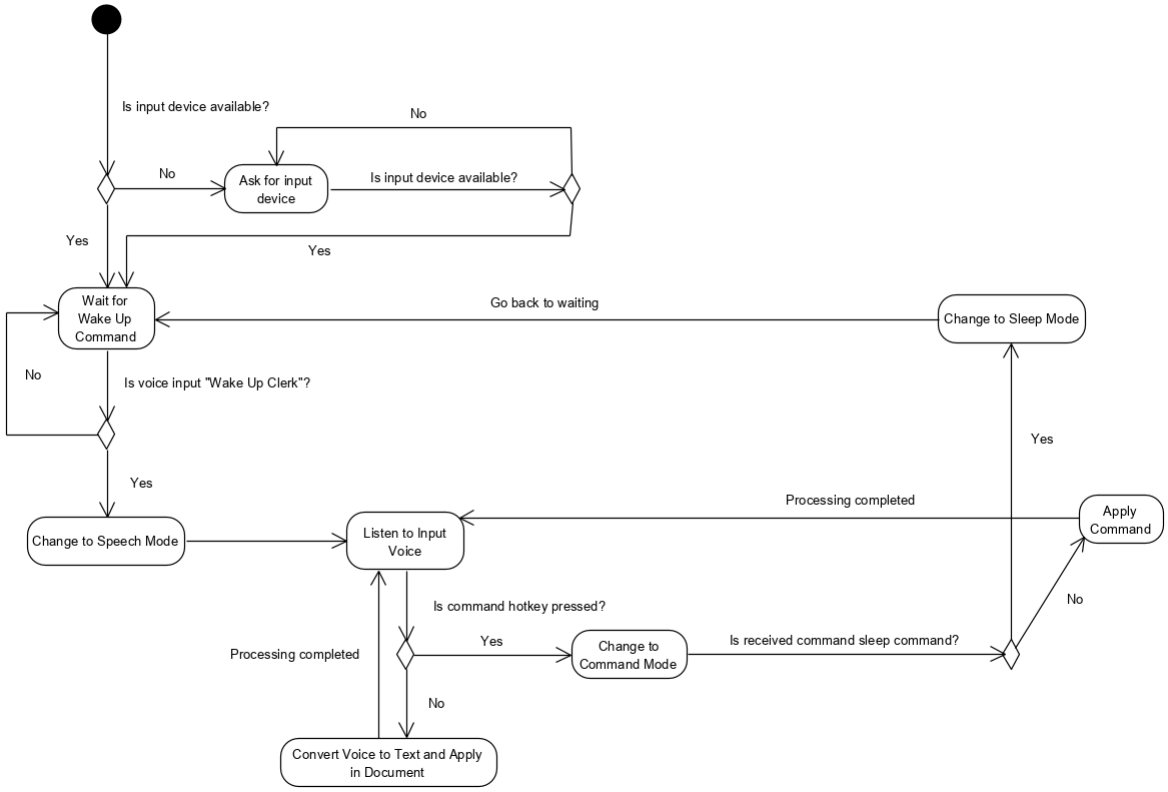

Figure 11: Activity diagram that shows the general flow of our application.

State Diagram

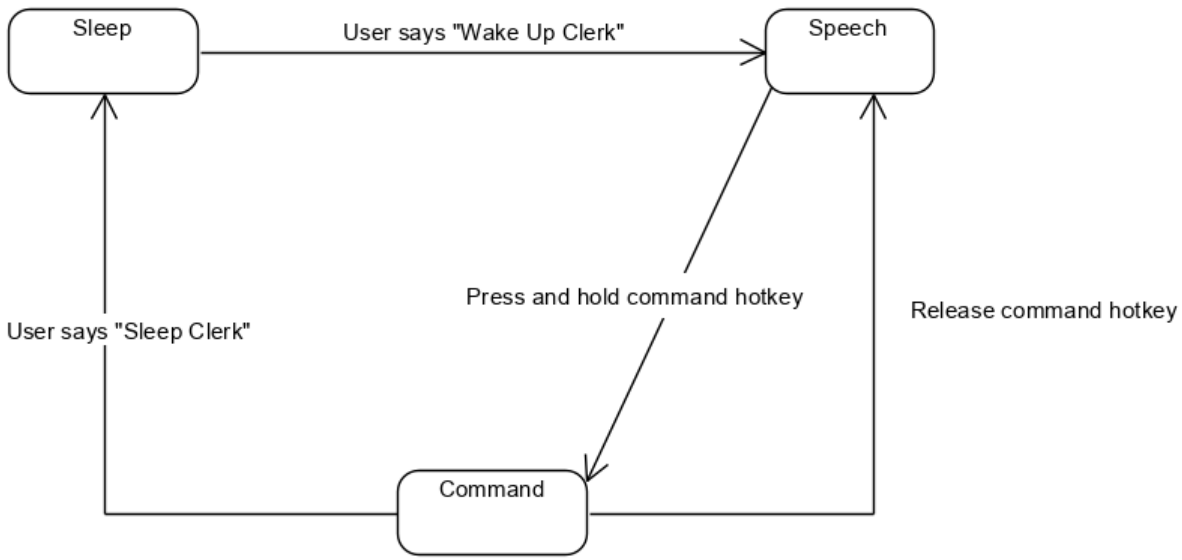

Figure 12: State diagram. The start state is Sleep. The transitions happen with commands like "Wake up Clerk" and "Sleep Clerk". To switch to command mode the user must be in speech mode and use the command hotkey.

3.6 User Interface and Navigational Paths

3.6.1 User Interface Mock-ups

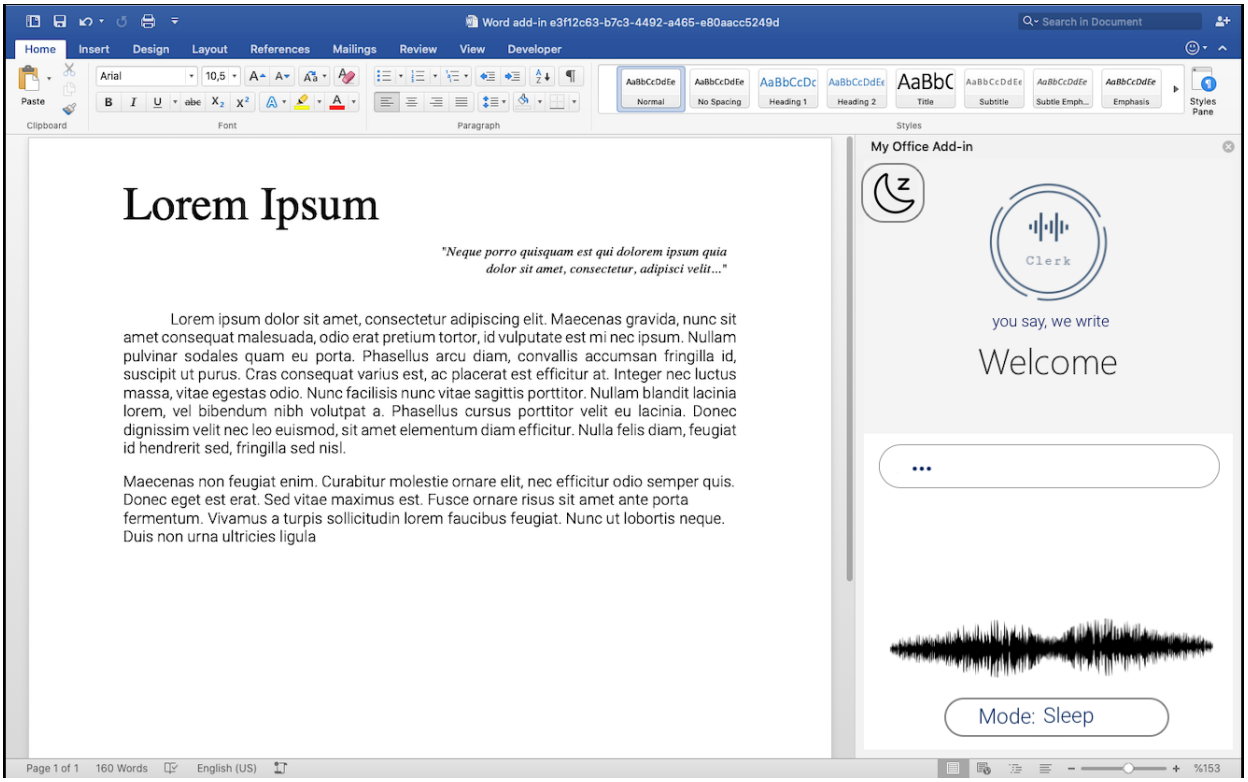

Figure 13: A user interface mock-up showing the interface in sleep mode.

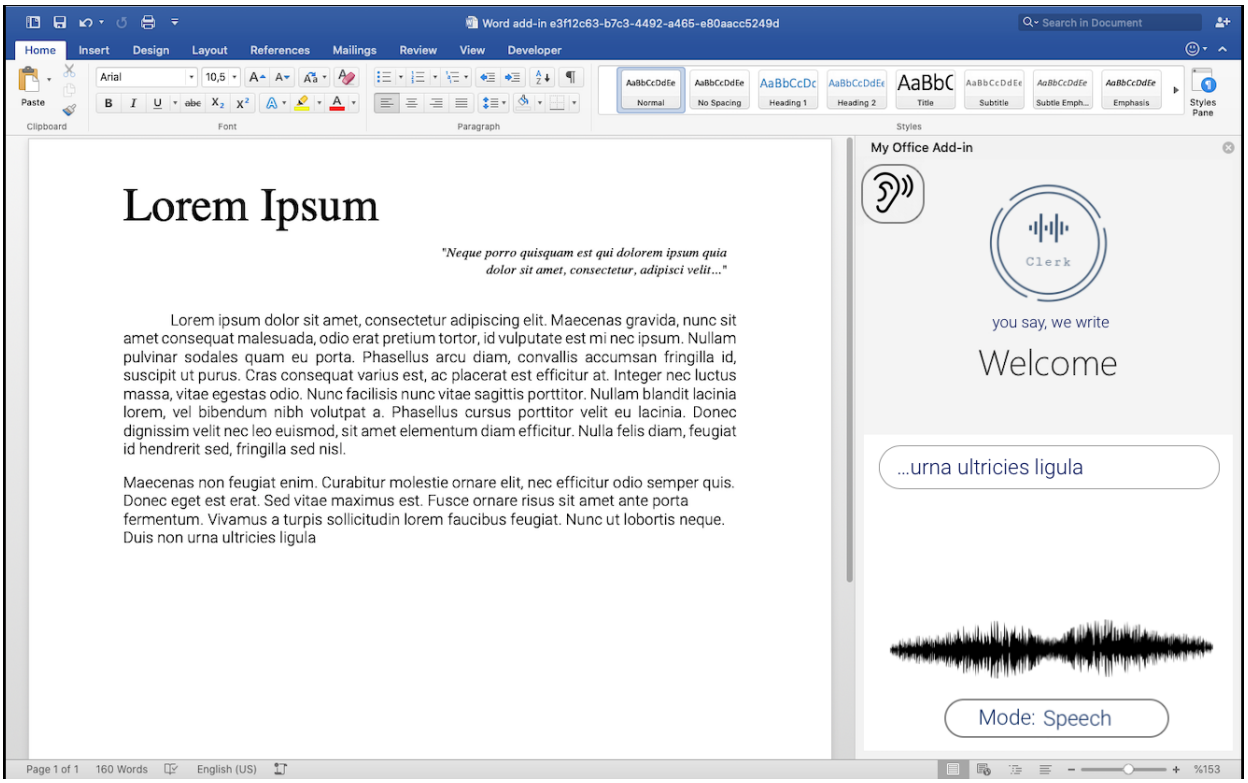

Figure 14: A user interface mock-up showing the interface in speech mode.

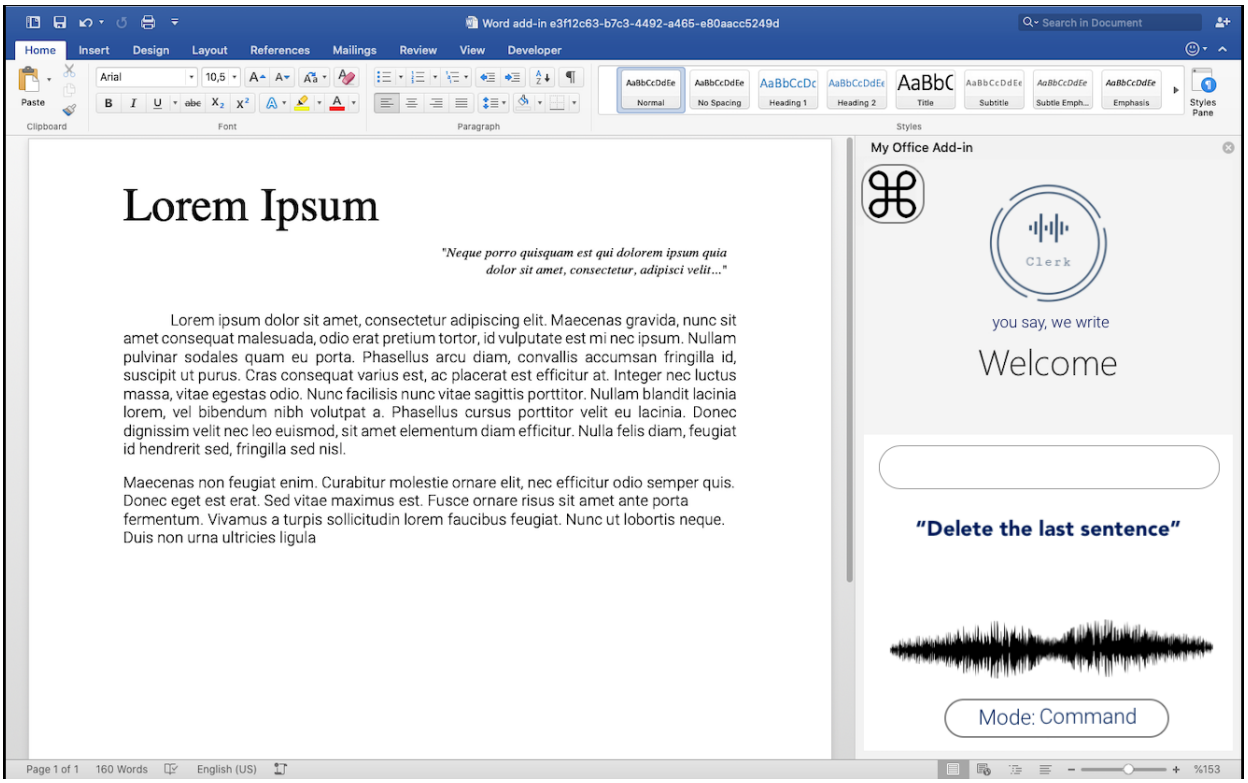

Figure 15: A user interface mock-up showing the interface in command mode.

3.6.2 Navigational Paths

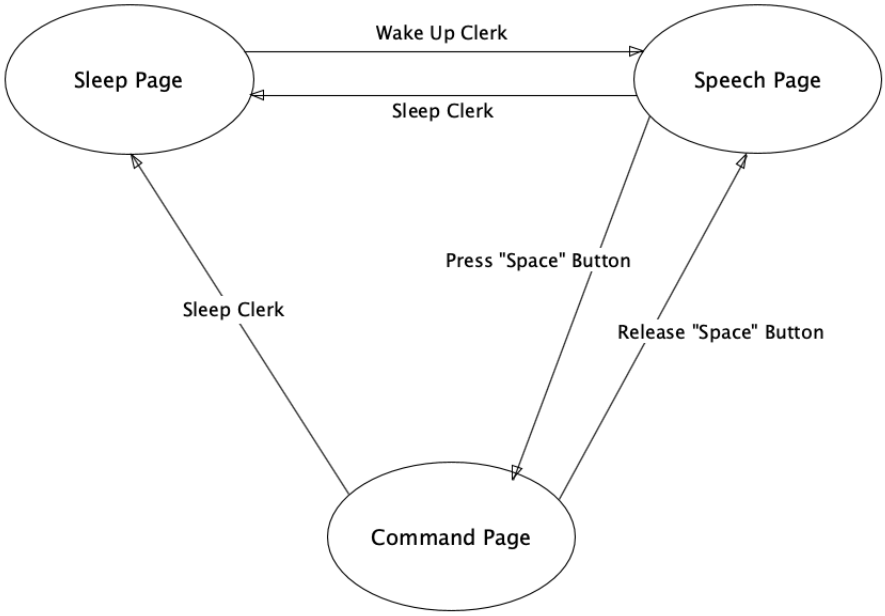

Figure 16: A state diagram showing navigational paths of Clerk.

4 Other Analysis Elements

4.1 Consideration of Various Factors

Public Health Considerations

Provided that it works efficiently and accurately enough, Clerk will have a positive impact on public health. When people are using a program such as Word and typing consistently for long periods of time, they tend to sit in ways that are bad for their posture. Adding to that, they also need to sit in front of a screen and directly look at it for long periods of time. But with this type of add-in users can have the freedom of sitting or standing however they want while they are working on editing a document and they don't have to look at the screen to the same degree. If more of the same type of programs are built for different platforms, then the impact will be even greater.

Additionally, for people with visual impairment problems or other disabilities who can't use keyboard and mouse effectively, Clerk and similar applications can make their day-to-day activities easier by removing some restrictions from their lives. It is a well known fact that empowering people with disabilities results in happier individuals and happier people are generally healthier.

Public Safety Considerations

Our project does not have any clear contributions to public safety; nor does it pose any threat to it. We also don't have any safety related concerns or considerations at this time.

Public Welfare Considerations

Clerk can improve the welfare of society by making its users healthier and happier as mentioned in the public health considerations. Empowering disabled people will also improve our welfare. Human beings have an inner motive to work and create things. Enabling more people to participate in different tasks such as writing reports or creative tasks like writing books will improve the society as a whole.

Global Scale Considerations

We have considered that more people can use our add-in on a global scale if we provide it mainly in English language, rather than our native tongue of Turkish, as we have five Turkish members. Therefore, at this point in time we are going to build it in a way that works with English. However, speech recognition API's and libraries are advanced enough to recognize more languages than just English. Though we want to prioritize making the application work with English language first, it could be a consideration later to make it multilingual in the future.

Environmental Considerations

Our project doesn't have any clear, significant effects to the environment, negative or positive.

Economic Considerations

We have decided that we will provide our add-in to Word users free of charge. However, it should be considered that it already costs a certain amount of money to obtain an Office365 subscription which is required to use Word with all its functionality. At the same time, our add-in will also be available on Office Online which can be used free of charge, though it doesn't quite provide all the functionality that subscribers get on desktop version.

Our decision to make the add-in free hinges on the fact that right now we are considering a design that won't cost us any money to maintain. This design decision may change in the future which would mean the economic considerations may be revisited.

Cultural Considerations

We considered that a very popular application like Word almost comes with its own culture and user behaviour. We considered what a typical Word user would want a feature to work like with voice inputs. We thought about the keywords to use and the convenience of them. Since our application's goals are aligned with improving the user's quality of life while using Word, these are very important considerations.

We also tried to consider how visually impaired people's culture and expectations would intersect with other Word users. We needed to know what it was like to use accessibility features already provided by such applications from their point of view. We needed to consider which features would be most critical for a visually impaired person to be able to use Word effectively.

Social Considerations

Our considerations in terms of social impact of our project and the effect of society on it have already been mentioned in public health, welfare, global, economic and cultural considerations sections.

	Effect Level	Effect
Public Health	6	More relaxed ways of using a text editor and empowerment of people with disabilities.
Public Safety	0	None
Public Welfare	5	Empowerment of people with disabilities.
Global Scale	6	Use of English language.
Environmental	0	None
Economic	4	Providing the add-in free of charge.
Cultural	6	Thinking about how users would use Word with voice.
Social	8	Combination of health, welfare, global scale, economic and cultural effects.

Table 1: Factors that have affected the design and their severity.

4.2 Risks and Alternatives

- Time may not be enough to complete all requirements. In this case, we can keep the most major requirements while removing the minor requirements from the project in the next phases.
- Team members may leave. In this case, the rest should be able to work harder and compensate for the workload of the member who has left. Some functionality may not be implemented.
- Team members might be unfamiliar with technology that will be used in the project. In this case, they will try to get familiar with it using any learning strategy they want to start working on the project as soon as possible.
- Design and architecture of the project may not be suitable or may not be efficient for the desired product. In this case, suitable design and architecture for the project should reformed in the next phases to make it suitable and increase the efficiency.
- The accuracy of Speech to Text library or API may be lower than expected. In this case, we will try different sources and compare their result against each other to find the most suitable one for our project.
- The deadlines for the releases we set may not be met. In this case, we will start spending more time on the missed deadline as soon as possible and try to work on the next release earlier not to miss the next deadline.
- The leader of a specific work package may not have the necessary qualifications. In this case, they will try to improve themselves according to feedback from the team members. Also, we will assign a co-leader for that work package to ensure the work gets done.

- As the project progresses more requirements that were not identified at the beginning of the project may emerge which could threaten estimates and timelines. In this case, since changes and requirements inflation is accepted as a fact of agile project plan we have for our project, we will include those in the next phases.
- Developer License for MS Word, which is a 90-day free trial and seems renewable at this moment, may cause a problem later. In this case, we may have to acquire an official MS Word to continue to work on.

Risk	Impact Level	B Plan
No enough time	High	Keep major requirements. Remove minor requirements.
No suitable or efficient design	High	Reform the design and architecture.
Low accuracy of Speech Recognition	High	Try different sources. Find the most suitable one.
Missing deadline	High	Spend more time on the missed deadline and start working on the next deadline.
No valid Developer License	High	Acquire a new license for MS Word.
Absence of member	Medium	Other members will work harder and compensate the workload.
Unfamiliarity with technology	Medium	He will learn it as soon as possible.
Unqualified leader	Medium	He will improve himself. Assign a co-leader.
More requirements needed	Medium	Include those in the next phases.

Table 2: Possible risks with their impact level and B Plan

4.3 Project Plan

Project Goals

- Speech-to-Text libraries will be tested in terms of accuracy and response time in order to decide which one to use.
- Proficiency in HTML, CSS and Javascript will be achieved.
- Infrastructure of the add-in will be developed.
- High-Level Design Report will be completed.
- The capabilities of MS Word API will be learned and the sequence of API calls in order to execute Clerk's functionality will be learned.
- A wrapper class for MS Word API will be created for our functional requirements.
- SpeechModeController class will be implemented to write the converted text into MS Word.
- Low-Level Design Report will be completed.
- CommandController class, which will parse and execute the commands, will be implemented.
- User Interface will be prepared.
- Text-to-Speech will be accomplished using Web Speech API.
- Tests will be done.
- Final Report will be completed.

Work Package 1

- Leader: Muhammed Salih Altun
- Start Date: 02.12.2019
- End Date: 31.12.2019
- Milestone: Specific speech recognition libraries will be chosen. Proficiency in HTML, CSS and Javascript will be achieved. Infrastructure of the add-in will be developed. High-level design report is ready to hand in.
- Deliverable: High-Level Design Report
- Description: Currently we are not proficient with HTML, CSS and JavaScript. The team members will work on improving on these languages. We are also not certain about the accuracy and response time of the available speech recognition libraries. We will test them, learn and choose one to use. The bases of the add-in will be developed. High-level design report will be written.

Work Package 2

- Leader: Samet Demir
- Start Date: 10.01.2020
- End Date: 30.01.2020
- Milestone: We know exactly what we can or cannot do with MS Word API. The states are managed in the project. We are successfully converting speech to text.
- Deliverable: None
- Description: Currently we are not fully aware of the capabilities of MS Word API. We will read documentation and implement some programs to learn the ins and outs of this API. Speech-to-Text is needed to convert to the voice of the user into text and is crucial for our application. It is one of the first pieces we need to get working correctly.

Work Package 3

- Leader: Ensar Kaya
- Start Date: 31.01.2020
- End Date: 09.03.2020
- Milestone: We are able to write the speech into Word document. We have a wrapper class for MS Word API and this base class for CommandController class to execute commands, is ready. Low-level design report is ready to hand in.
- Deliverable: Low-Level Design Report.
- Description: Since we have the transcript of the text from Speech-to-Text, we will write it into Word. And since we know the capabilities of MS Word API, we will write a wrapper class for this API that can execute the complete functionality of our program. Low-level design report will be completed.

Work Package 4

- Leader: Ahmet Malal
- Start Date: 10.03.2020
- End Date: 20.04.2020
- Milestone: CommandController class will be implemented. Now, we are able to execute the commands given by the user.
- Deliverable: None
- Description: Our structure is ready for our CommandController class. CommandController class will be implemented.

Work Package 5

- Leader: Faruk Şimşekli
- Start Date: 21.04.2020
- End Date: 08.05.2020
- Milestone: The user interface is ready for use. The program is able to speak back to the user. Testing is completed. The final report is ready to be handed-in. The program is ready to be presented.
- Deliverable: Final Report and Clerk the Word add-in.
- Description: Related mock-ups will be implemented in the User Interface. The text written in Word document should be spoken to the user in case of any request to modify the text. Necessary tests will be done for debugging purposes and correct them. The final report will be written.

Gantt Chart

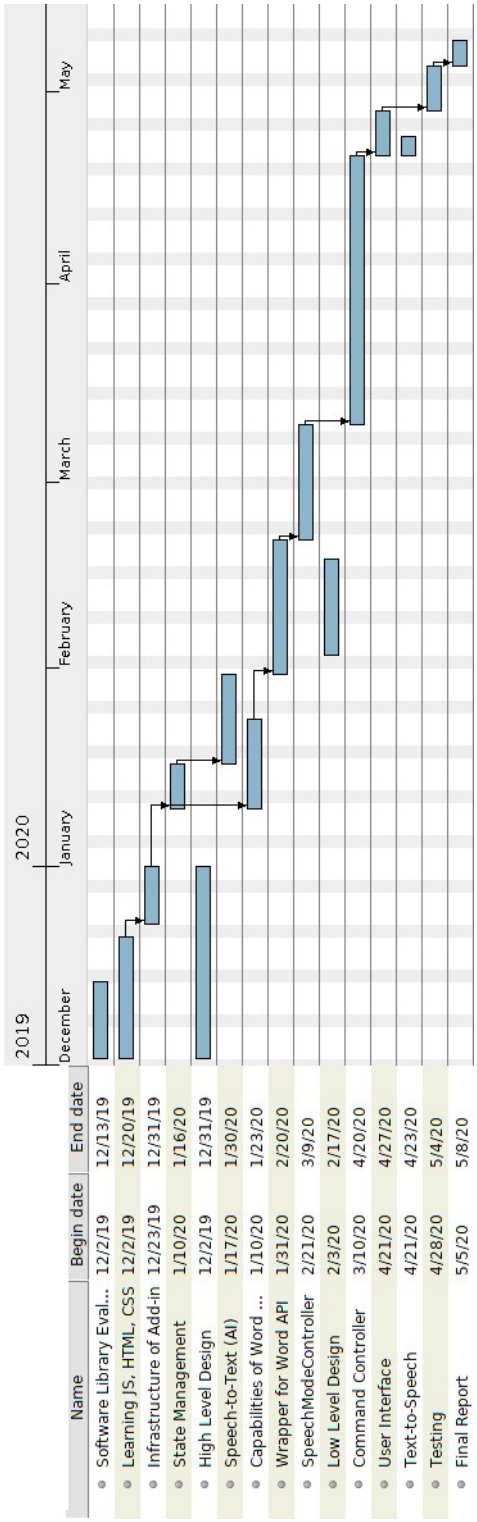

Figure 17: Gantt Chart of our Project Plan.

4.4 Ensuring Proper Team Work

For our project we have investigated GitHub, Jira, and BaseCamp. The project management tools we investigated usually come with some sort of price associated with them and we wanted to use free software if possible. After some consideration, we have decided to go on only with GitHub. Accounts have been created and the supervisor has been notified about the project account in GitHub. For creating the Gantt Chart we have used a tool called GanttProject [9].

To ensure proper team work in the group, leaders will do the most work. They will establish some ground rules. They should be aware of their biases and assumptions, take actions, empower the team members, hold them accountable, and be courageous. They will practice participatory leadership. This way, we will create an inclusive and collaborative environment for the project. They will be a communication channel in the group where you can get the updates and changes; and they will be in constant communication with the team members whether the project is going smoothly or not. They will provide an ongoing feedback for the group as a whole in that work package. Necessary steps will be taken immediately. The leader should be careful about the division of work among the team members as well so that each member could contribute more or less the same to the project. They should know the strengths and weaknesses of the team members and they should take this into account so that each member will do what they do best. This makes team members more motivated along the work package. For our team, in each work package the leader will find an activity that all members of the team can attend so that social bond between the members are strong and this can have a positive effect on the team work in the project.

The team members also has some work to ensure a proper team work. First of all, they should complete their assignments successfully. They should be aware of the problems that may emerge and warn the team. They will be in constant communication with the team in case of any issue that has occurred in the project. They should be productive within the team. They should question a possible decision in the team.

4.5 Ethics and Professional Responsibilities

Law number 6698 on the Protection of Personal Data known as KKVK in Turkey is holding us responsible to keep personal information private and not to share this personal information with any third-party. Since, we will not hold any personal data, we will not have any problem related to this.

One of the very first responsibilities is to accept individual responsibility. Even though work-packages mentioned above have leaders, following orders does not justify approving bad designs for other members. Having a leader in a work-package does not necessarily mean that he does what he wants. One cannot always be a team player. At this point professional standards have priority. If there is something to argue, we will make it subject of a meeting to make the product better.

As software engineers to be, we are responsible for looking beyond the customer's opinions. We should have a precise description of any problem that may occur in the project. We should get that problem reviewed by other members before building to actually solve the real problem.

Another professional responsibility that we have to meet is that we should be honest about the capabilities of our project. We will not offer technical solutions where there are none. And if there is solution for a specific problem at this moment, we will talk about the possibility of a solution, avoid giving precise statements about a solution.

Another important responsibility for us is to produce reviewable designs for our projects. This will allows us to keep a good documentation throughout the project to create a communication channel among the members and producers and customers.

One of the users of our project will be visually impaired people. We have a responsibility for them. Since we have promised an easy and usable product, we have to produce a maintainable product, which means that this product can be maintained without us. Because this product is not our personal product any longer.

4.6 New Knowledge and Learning Strategies

The main API we will use throughout the project is MS Word JavaScript API. For this reason, first of all, the members will learn how to use the JavaScript programming language. To create an add-in for MS Word we should use HTML and CSS as well. Since we do not know how to use them, we will learn them also. We will use online learning and learning by doing as the strategies to learn JavaScript, HTML, and CSS.

We should get familiar with the necessary APIs such as MS Word JavaScript API, Web-Speech API, Google Cloud API, Microsoft Bing Voice Recognition, and other options in the PyPi Speech Recognition library. We will learn these APIs in the following days using online learning and hands-on-experience as the strategies.

Last but not the least, we should learn software project management to be a capable leader in the work-package that we are assigned. Specifically in this part, we should learn what a leader or member are supposed to do, how the division of labor should be, how to manage time, and how to ensure the team work effectively and efficiently.

5 Glossary

Word add-in: An application that can be built and embedded inside Microsoft Word and work inside a Word instance that can extend the available functionality.

References

[1] "Microsoft Garage: Dictate - an add-in for Microsoft Office on Windows," https://www.microsoft.com/en-us/garage/profiles/dictate/, Accessed: 2019-11-08.

[2] "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.

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

[4] "office package - Office Add-ins | Microsoft Docs," https://docs.microsoft.com/en-us/javascript/api/office?view=word-js-preview, Accessed: 2019-10-13.

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

[6] "Cloud Speech-to-Text - Speech Recognition | Cloud Speech-to-Text | Google Cloud," https://cloud.google.com/speech-to-text/, Accessed: 2019-11-08.

[7] "Cognitive Speech Services | Microsoft Azure," https://azure.microsoft.com/en-us/services/cognitive-services/speech-services/, Accessed: 2019-11-08.

[8] "SpeechRecognition · PyPI," https://pypi.org/project/SpeechRecognition/, Accessed: 2019-11-08.

[9] "Gamtt Project: free desktop project management app," https://www.ganttproject.biz/, Accessed: 2019-11-10.