Echoboard

- An Android keyboard showcasing how a screen reader interprets your message -

Bachelor Project

Goran Kirovski
Karina Mary-Anne Botes
Magne Frank Dinesen
Pernille Knudsen

May 29, 2024

Aalborg University
Software

The report is written in LaTeX Overleaf, with Zotero integrated for managing sources and citations. The program is developed in Android Studio, utilizing Java and Kotlin, and is shared via GitHub. Google Docs served as the platform for note-sharing. Diagrams are either sourced from external references and properly cited, or created by group members using Microsoft Excel or Draw.io, based on data from cited sources. The prototype design is created in Photoshop.

Software
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Echoboad - An android keyboard showcasing how a screen reader interprets message

Theme:
Accessibility support for vision impaired users in text-based communication

Project Period:
Spring Semester 2024

Project Group:
8

Participant(s):
Goran Kirovski
Karina Mary-Anne Botes
Magne Frank Dinesen
Pernille Knudsen

Supervisor(s):
Carla Florencia Griggio

Copies: 1

Page Numbers: 26

Date of Completion:
May 29, 2024

Abstract:

Approximately 2.2 billion people are to some degree VI (visually impaired), and 43 million are blind. To navigate the digital world, screen readers are often used by this group of people. They often experience misunderstandings when messaging with sighted people due to the way screen readers interpret messages. In this paper, we study sighted users understanding of how messages are interpreted and vocalized by a screen reader, and whether the text which the screen reader reads aloud, corresponds to their intended meaning. We also designed and implemented Echoboard, an extension to an opensource Android keyboard that displays how a screen reader reads messages. We conducted two user studies within this project: (1) testing prototypes and designs, and (2) testing the usability of Echoboard. We found that most sighted people were unfamiliar with how text-to-speech services would vocalize most symbols and emojis. We also found that live previewing how a screen reader would interpret the users' text message, significantly benefits the users' understanding of screen readers and how they can adapt their messages to be more compatible with screen readers. We recommend Androids text-to-speech service to, besides providing the speech, also to provide a textual interpretation as output, allowing developers to use it. Another implementation is messaging apps to implement an API that provides access to messaging field data and sender identifiers. If Google and OpenBoard could provide more extensive documentation, the implementation process will be streamlined.

Summary

In this project, it has been our objective to research the theme accessibility support for vision impaired users in text-based communication, then create a software solution to propose a solution for the given problem. After researching the theme, we decided to scope the project into focusing on how we could make sighted users more aware of how screen readers interpret and read text messages and emojis for vision impaired.

The following paper examines this theme and presents our findings through related work and our own test studies. These studies include an interactive prototype test before the implementation and a usability test conducted after the implementation of our software solution. The interactive prototype was based on our findings in researching the area and our own speculations within the topic and provided us great insight into sighted users' knowledge and experience of screen readers. It also gave us insight into how the users thought a screen reader worked, and to their surprise, how little they knew about them.

Based on our findings from the prototype test, we started designing the Echoboard - our proposal for a solution to this problem. We started by defining some requirements using the MoSCoW model and then implementing them. We used Android Studio and based the Echoboard on the open source keyboard; OpenBoard. We approached this project with little knowledge about the development of Android keyboards and encountered a lot of difficulties implementing the Echoboard based on the OpenBoard, however despite the difficulties we have completed a functional version of the Echoboard that provides the user with a live preview of how their text message, including emojis, will be received by a screen reader.

After the first iteration of implementing the Echoboard, we conducted a usability test on 6 participants, the goal of this test was to get feedback on implementation, design, and to test intuitiveness and usability. Based on the feedback gained from the usability test, it can be concluded that the Echoboard is relatively intuitive as the test subjects found it easy and could easily maneuver and use the product, we also gained some insight into what could be improved or changed in the product. There were only some minor, serious, and cosmetic errors in the usability test, such as how it handles large numbers or how the participants liked the look of the product. No crucial or critical errors that could obstruct using Echoboard were found during the usability test.

Although the project had a deadline we managed to fulfill all the must have requirements and some of the should have requirements that we defined early in the project.

Contents

1 Introduction

Problem statement
The use of emojis, elongated words, all caps etc. can enhance the messaging experience and textual communication in the digital world today. However, for people with vision impairments that use screen readers, Emojis can be confusing and be the source of misunderstanding and miscommunication. This problem occurs because screen readers today aren't advanced enough to handle the unspoken context and non-verbal cues of Emojis, and sighted users might not consider how screen readers handle and read the Emojis they write. There might occur other issues in text-based communication when sighted users use text as a visual means to express tone and emotions, such as with all caps, repeated exclamation/question marks, elongated words which screen readers cannot handle or interpret correctly.

When screen readers read Emojis, they read the descriptive name of the Emojis, however many of the descriptions of the Emojis are long and not known to sighted users and sighted users rarely select Emojis based on anything else than their visual look. For example, the descriptor for this emoji:👩🏽‍⚕️, is *woman-health-worker-medium-skin-tone* which is very long, unintuitive, and most likely unknown to most sighted people. Another issue that can occur with screen readers are how they read elongated words and how it does not necessarily match how sighted users would interpret that, an example of this is *"Heeeey"*. Or how all caps messages such as *CONGRATULATIONS*, is interpreted differently by sighted users and screen readers.

Based on this problem we wanted to focus on creating a solution and working towards solving the problem statement:

How can a software solution be designed help sighted users write more inclusive and understandable messages to visually impaired users in messaging apps?

How can sighted users be made aware of how their messages are interpreted by visually impaired people?

2 Background and Related Work

2.1 Emojis in communication

Emojis are graphic illustrations used in much of the world's textual communication and they are defined by the Unicode Consortium[1]. Emojis visualize and convey many different concepts and ideas such as emotions, people, animals, food, flags, objects, symbols, etc., and new Emojis are added yearly [1] [2]. Beyond this wide array of Emojis, some are customizable by having skin tone and gender modifiers that support improved self presentation [3] [4].

Emojis are managed by the Unicode Consortium, how they are managed is important for us to understand since their visual difference based on platform might change the users interpretation and use of them [5]. In the Unicode Consortium, emojis are identified by Unicode characters and then rendered differently based on the platform's font package and therefore there remains a stylistic divergence in the design of emojis across different platforms, as shown in Figure 1 [1] [6]. The Emojis displayed to users vary depending on the messaging platform they use. This nuanced difference can potentially alter the interpretation of the conveyed message, adding another layer of complexity based on the platforms used by both the sender and recipient[7].

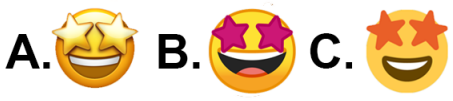


Figure 1: The same Emoji on three different platforms, A) IOS, B) Android, and C) Discord

The topic of how people communicate using Emojis has been extensively researched, revealing a significant variation in their usage. Emojis range from being used literally, where the illustration serves as a substitute for words, to clarifying or enhancing the intent behind a message, or even as a comedic tool within a message[8]. However, emojis are also used to indicate nonverbal cues and allow users to extend the ways that they can express themselves in textual conversations [5]. Emojis can help users to replace nonlinguistic cues such as facial expression or body

language [9] that might have gotten completely lost otherwise over textual communication.

Emojis can also be used in different ways other than to express feelings. Kelly et al. have drawn findings based on interview data and have found that emoji can have multiple useful roles in conversation, and these roles are not always associated with discrete expressions of emotion [9]. Kelly et al. have primarily focused on three categories of how emojis can be used beyond expressing emotions. One way emojis can be used is to maintain a conversation or end it with a low effort when there are no words left [9]. Emojis can also be used as an outlet for playful interaction which may add to the "atmosphere" of the conversation making it more serious, playful, or relaxing [9]. Lastly, Kelly et al.'s final category is that emojis allow users to create shared and secret meanings in emojis. The meaning of the emojis might vary based on who the user sends it to, an example of this from Kelly et al. is how one test subject described the difference in the meaning of the kiss face emoji when sent to a best friend or a person they are dating. The meaning of the emoji changes based on the relationship with the author and receiver of the messages [9].

2.2 Visual impairment

Visual impairment (VI) is a condition referring to a decrease in one's vision, VI covers a spectrum over vision, that slightly deviates from normal levels of vision, to complete blindness [10]. According to the World Health Organization (WHO), approximately 2.2 billion people globally are affected by some degree of VI [10]. The severity of VI is measured by visual acuity and is categorized under, "Near", "Mild", "Moderate", "Severe" and "Blindness"[11]. VI may be treatable, although, in the more severe cases, VI is permanent [11]. The International Agency for the Prevention of Blindness (IABP), has the following estimates for the amount of people in the different VI categories [12]:

- "510 million people have near vision problems"
- "258 million people have mild visual impairment"
- "295 million people have moderate to severe visual impairment"
- "43 million people are blind"

VI impacts people on a personal level, various aspects of one's life are of substandard quality, e.g. lower quality of life, lower employment rates, social exclusion, depression [10] [13]. It should be mentioned that there is a correspondence with VI severity and personal impact. Communication is an important part of life, and digital communication has become a necessity in today's world, however, digital communication may not always be as accessible to minority groups with specific conditions. It is important to ensure accessibility in digital communications, for all individuals, so they can equally participate, regardless of conditions, this inclusion can improve aspects of one's life.

2.3 Screen readers

Screen readers are tools used by people with visual impairments. It converts text, buttons, images, and other screen elements into speech [14]. Screen readers are available on the computer, in browsers, and on mobile devices [14].

In December 2023 and January 2024, WebAIM surveyed the preferences of screen reader users [15]. 89.9% of the participants use screen readers due to a disability where 76.6% responds with that disability being blindness, and 19.9% responds with that disability being low vision/visual impairment [15].

The latest WebAiM survey also shows that 91,3% of respondents use a screen reader on a mobile device[15]. Many of the different options for screen readers are based on what mobile device you use or if you use your computer or a browser. The most commonly used screen reader for mobile devices submitted in the survey by WebAIM is "VoiceOver" which has the highest popularity with 70,6% followed by "TalkBack" with 34,7% popularity [15]. These two choices are distributed like this due to "VoiceOver" being the primary screen reader for Apple products and "TalkBack" being the primary screen reader for Android products.

We will look into developing on Android due to their default keyboard being open source and iOS's is not open source making implementation far more difficult, However, the distribution of mobile device preference by visually impaired can benefit our choice.

But how does a screen reader work? When looking at "TalkBack" more precisely and how it works, Google describes it as an accessibility feature that helps people who are blind or have low vision to interact with their Android devices

using touch and spoken feedback [16]. If TalkBack is turned on, the user can hold down both volume buttons as a shortcut to turn TalkBack on and off. When TalkBack is enabled, it will outline the items on the screen with a focus box and read the text within the focus box aloud [16]. When TalkBack is activated it can be navigated around the screen through touch interactions, these touch interactions can be customized to the user if necessary, and these interactions can toggle select for the focus box and scroll horizontally or vertically on the screen [16].

2.4 Enhancing text communication for vision-impaired users

When sighted users write text messages to vision-impaired people, miscommunication can occur and the message may not be interpreted as intended. For instance, using Emojis can often hinder communication for screen reader users [17]. Emojis are popular to use, which can lead to social exclusion and a reduced quality of life for visually impaired. [17] Therefore Tigwell et al. suggest sighted people to be aware of the needs of screen reader users and consider some best practices. They should consider that using repeating Emoji causes the screen reader to read the name of that Emoji multiple times. Besides this, Emojis should not be placed in the middle of a sentence but rather at the end of the content, because the screen reader will read the Emoji as if it were text. The descriptors for Emojis do often not accurately represent the visual design. Therefore, messages should be communicated through clear communication as text rather than a potentially ambiguous Emoji. An example on this is seen from Tigwell et al. in their interview: "P6: "Emoji is something fun for sighted texters ...but for me it's just an extra string of words. ...like the grinning face emoji[😃]; it looks fun and cute when you look at it, but Voiceover describes it as 'grinning face with clenched teeth emoji' which sounds more like a grimace than a big smile)." Lastly, if you write on a public platform, you should be aware that the visually impaired might read it and perceive Emojis differently [17].

A study by Griggio et al. [18] has explored how to raise awareness of how messages sound in text-to-speech. Griggio et al. [18] have explored the inaccessibility when sighted users write text messages with Emojis to individuals who are vision-impaired as recipients. Griggio et al. created two accessibility support design options. They named the first design "PREVIEW" which is the design that their participants preferred, as it gives better opportunities for subjective communication. In the PREVIEW design, a text is shown above the message input box, to illustrate how a screen reader would read a message with an Emoji. The other design is called "ALERT" which composes a summary of potential accessibility challenges arising from the use of emojis in a message. They conducted an online questionnaire with 116 sighted participants whereas 88 preferred the PREVIEW design, 12 preferred the ALERT design and 11 preferred no support at all. A reason why some users didn't prefer the ALERT design, was that they wanted to solve the accessibility problem themselves.

Some users are not aware of the names of the Emojis. Those who are unaware of the name of the Emoji they send cannot predict how it will be read aloud by a text reader. A comment from a participant in the survey in Griggio et. al.'s paper [18] was:"For the first and second set of questions, I had no idea how Emojis were described. So only used fairly simply happy / sad expressions so as not to confuse Taylor." Taylor is the persona in Griggio et. al.'s scenario who uses a screen reader. It is relevant to take into consideration that people may not necessarily be familiar with the description of Emojis.

Other examples that indicated participants who were not aware of Emoji descriptors, were some who replaced emojis with longer descriptions. An example of this is a participant who changed "Congratulations! 👏👏🎉🎉" to "Congratulations! 👏🏻 ("Clapping Hands Light Skin Tone"). Griggio et al. [18] believe that this might be because the participant has used an Emoji from their recently used list and are not aware of the added skin-tone modifier. They point out that it can be beneficial to draw attention to the fact that added modifiers (such as skin tone) in an Emoji extend the text read aloud by a screen reader.

Figure 2 shows an example of an Emoji with a name that is not obvious with three different skin-tone modifiers.

Figure 2: The emoji descriptors for the emojis are: A.) Sign of the Horns. B.) Sign of the Horns: Medium-light Skin Tone. C.) Sign of the Horns: Medium-Dark Skin Tone.

Griggio et al.'s paper's [18] online questionnaire reveals participants' tendencies to alter messages by replacing Emojis with textual markers for better textual expressiveness. However, the changes are sometimes less compatible with screen readers due to the way screen readers interpret the text, and having a monotone voice, leading to the opposite effect and causing more misunderstandings. By utilizing the screen reader VoiceOver, Griggio et al. [18] have identified the following textual markers.

Capitalizing words. When the participants replace the Emoji with its name in capital letters, e.g. *"TEARS OF JOY!"* to express emotions as excitement, however a screen readers of "TEARS OF JOY!" or "tears of joy!" would be the same, since it does not account for case-sensitivity. [18]

Repeating question and exclamation marks. Screen readers have a monotone voice, meaning that there is a lack of intonation, naturally, this means they do not account for the intonation of punctuation's, and the pronunciation will be flat. So when participants replace emojis with punctuation to try and convey emotions, e.g. from "I know right 😉" to "I know right??", it will have the reverse effect since the screen reader won't take into account the punctuation, a part of the message is lost and it will not convey the same meaning. [18]

Using ASCII emoticons When participants had replaced Emojis, with their counterpart ASCII emoticon, e.g. an 😀 emoji to an ":D" emoticon. The screen reader won't pronounce the punctuation in the emoticon, and only pronounce the character "D" ("Dee"), or emoticons only composed of punctuation ":)" may not be pronounced at all. This may either cause confusion since only a part of the emoticon will be pronounced, or the message won't convey the same meaning if the emoticon is not pronounced. [18]

Elongating words Participants elongate words when they try to convey a certain effect through the message, e.g. this could be to convey emotions or emphasis. Nonetheless, screen readers do not interpret the elongation of words with the expected effect. E.g. a participant had written "Soooo... how did last night goooo?", and this was interpreted by the screen reader as "Soo [pause] how did last night goo?", ignoring the added effect. [18]

Separating the main text from text representing non-verbal cues with punctuation Participants were observed replacing emojis with a text descriptor encapsulated with punctuation, such as parenthesis, asterisk, etc. The text descriptor could be, e.g. an action, such as "I got an A in the exam *drops mic*", and the screen reader will pronounce it like "I got an A in the exam star drops mic star". Depending on the symbols used to encapsulate the descriptor, the screen reader might either literally read the symbols, or ignore them, and could therefore cause confusion or misunderstandings. [18]

2.5 Extending an open source keyboard

Griggio et al. have implemented "DearBoard", by extending the AOSP (Android Open Source Project) LatinME [19]. AOSP is the default keyboard on many Android phones. Unlike typical private customizations in messaging apps, DearBoard allows two users to collaboratively modify the keyboard's color theme and a toolbar of expression shortcuts, including Emojis and GIF. DearBoard enables co-customizations for messaging applications and allows the customized keyboard to be used seamlessly across different messaging platforms. A notable aspect of DearBoard is its contact recognition mechanism, which identifies whether the user is conversing with their "keyboard contact" by sampling and comparing pixel-wide screenshots of the messaging app interface. This ensures that co-customizations are only displayed during conversations with the designated partner, maintaining the intimate context of these customizations.

2.6 Opportunities and applications of piggyback prototyping

Epstein et al. [20] have explored the benefits of using the "piggyback prototyping" method in the CSCW (computer supported cooperative work) community. According to Epstein et al., piggyback prototyping is a great way to explore the potential and opportunities with certain ideas without risking great amounts of effort for unknown value [20]. It is also a great way to take a look at existing limitations and proposing alternative solutions. The piggyback prototype method is a great way for us to explore and expand on Griggio et al. PREVIEW design [18] while piggyback prototyping on an open source keyboard and thereby expanding and contributing to its design potential. Epstein et al. has defined 5 challenges where piggyback prototyping has been primarily used, however, they state that this may not include all challenges in which this method has been used, we have looked at these challenges to gain insight into which might be useful for our project. One of these defined challenges is *Supporting creation and personalization of social content* [20], it can be discussed how the PREVIEW design proposed by

Griggio et al. explores how a keyboard can provide support and insight for sighted users into how their messages will be interpreted by screen readers, this provides the user the opportunity to customize their message with the insight knowledge of how it will be received. However, it could also fit under the challenge *Content moderation and quality control* [20] as the PREVIEW design helps the user control the quality of the content of their messages in regards to the screen readers interpretation. These are great for us to keep in mind as we piggyback on an already existing open source keyboard and explore the PREVIEW design by Griggio et al.

To sum up the background and related work, the use of Emojis by sighted users in text messages can lead to misinterpretation by screen reader users, causing social exclusion and reducing the quality of life for visually impaired individuals. Some users are not familiar with the names or descriptions of Emojis, which can lead to unpredictable interpretations by screen readers. Replacing Emojis with textual markers like capitalized words, repeated punctuation, ASCII emoticons, elongated words, or encapsulated descriptors can sometimes lead to misunderstandings or loss of intended meaning when interpreted by screen readers. A study by Griggio et al. [18] explores the implementation of DearBoard, an extended version of the AOSP keyboard. Piggyback prototyping offers a method for exploring and expanding upon existing designs, such as the OpenBoard keyboard and PREVIEW design proposed by Griggio et al., to enhance keyboard accessibility and support for vision-impaired users.

3 Defining product goals and requirements through MoSCoW

The requirement specifications of a product dictate the implementation order of the functionalities. In this section, the requirements of the product will be defined and described via the MoSCoW model. The MoSCoW model consists of four "categories" in which the requirements are categorized, primarily based on importance, but also criteria such as complexity, dependencies, and development time. [21] The categories of the model are:

"Must have" - contains the most crucial requirement for the product to be completed and must be focused on completing to have the minimum viable product.
"Should have" - contains requirements that are not as crucial for the product as the "Must have" requirements, but are still essential. These requirements can highly improve the capability of the product and add to the quality of life.
"Could have" - contains requirements that only add smaller improvements to the quality of life and overall user experience of the product.
"Won't have" - contains requirements that the product won't have, due to it being too difficult, time-consuming for the project, or beyond the scope of the project goals.

The research conducted in section *2 Background and Related Work*, section *5 User study 1: Prototype testing*, and the design choices discussed in section 10.2 in *Appendix A* serve as a background for the creation and prioritization of our requirements. All our MoSCoW requirements can be found in section 10.1 in*Appendix A*, where Figure 9 furthermore visualizes the MoSCoW requirements categorization and prioritization of our product. A description of each requirement can also be found in that section.
We will briefly go through the 7 must have requirements we have defined for this project as they are the most crucial goals which the product must achieve. This is followed by a short description of the 4 should have requirements defined as well as they are the second most important requirements for the product.

The main requirements for the product is that the product must be a *custom keyboard extension* which has a *orange box below the input field* where the*orange box is scrollable* in case it has to display longer messages. The orange box should have a *translate/transcribe functionality* which displays a version of the user input so it matches what a text-to-speech system would vocalize, the box should also *transcribe emojis into their descriptive name* and display that correctly so it matched a text-to-speech's interpretation of emojis. Lastly, we will first focus on implementing the product to work when the user writes messages with their phone in *portrait mode*, and to scope the project we have limited it to only work with the *English language*.
Following the most crucial requirements, we have defined these should have requirements that can strongly improve the usability and user experience of the product. One of these features is a *speaker button* which would read the user's transcribed input aloud for the user to hear. It can also be a *expand box button* functionality, which could allow the user to enlarge or minimize the orange box. To provide more guidance for the user, the orange box can *highlight* problems in the user's text to help them prevent misunderstanding. Lastly, the product could have a *Long press for emoji names* functionality, which can allow the user to learn the descriptive name of the emoji pressed, before using it in their message.

4 Overview of methods

We will base our approach on the PREVIEW design from Griggio et. al.'s paper [18] to raise awareness on how a message sounds on a screen reader. By incorporating a box on the keyboard that demonstrates how a screen reader interprets a message with an emoji, we will aim to inform users about the descriptors of emojis and their modifiers such as skin-tone modifiers. We will also take the textual markers, described in this section, into consideration. The paper we draw inspiration from is limited to a survey with static images and lacks testing of the actual experience of using such a tool. We will expand on the design, seek improvements, and thereafter implement and test it.

We have chosen to look at how sighted users send messages to screen reader users, rather than looking at how screen readers can be improved, due to our own interest in the subject, we have also made this choice due to how little the field has been researched from the screen readers perspective. Some have researched the area such as Das et al. [22], however the main take away from them are that there has not been much work done to improve screen reader access in mainstream collaborative writing tools. The nature of how screen readers present text-based content in combination with the lack of well-designed auditory representations amongst other things make it especially difficult for blind writers and readers [22], this should be researched more. But mainly due to our own interest in taking the perspective from sighted users, we have focused our project on that angle.

Throughout this project, we have focused on following a user-centered design approach, where we have iterated between design and user studies. A step-by-step list can be seen below of our approach and course of action.

1) High-fi prototype design.
Firstly we created some design ideas for a potential product and made prototypes of those ideas. These are mainly based on our research in the background and related work.
2) User study on High-Fi prototypes.
Then we conducted a user study with high-fi prototype tests based on the design ideas, this can be read in section 5.
3) Defining product goals and requirements
After the first user study, we explored and defined product goals and requirement for our project. This was done using the MoSCoW model which is described in section 3 and explored more in-depth in *Appendix A*
4) Implementation of prototype
Based on the product goals, requirements and prototype user study including the findings thereof, we created a more 'final' design and then went on designing and implementing that, this can be read in section 6.
5) User study on usability
After the implementation iteration, we made our second iteration of user studies. This user study was a usability test where we wanted to test the current product's usability and gain insight into how a user would interact with the Echoboard.

In the original plan, we wanted to have a second implementation iteration after the usability test, however, due to delays and the project deadline we decided to cancel that iteration and focus on writing and describing the findings of the usability test, and further development. The findings from the user studies was analyzed through the Instant Data Analysis (IDA) method, which will be described in the following section 4.1.

4.1 IDA method for data analyzing

The Instant Data Analysis method (IDA) allows us to analyze the test data straight after the tests and highlight the most important key parts gathered from the tests while everything from the test is still "fresh in mind" [23].

The IDA technique aims to complete the entire test valuation in one day and the test only has to be based on 4-6 user tests [23]. This will provide enough data for a foundation for the IDA technique [23]. We have used this IDA technique to not only shape our analysis of both user studies but to shape our entire user study design for both of our user studies.

During the IDA session, all these roles should be present, these roles are described in *Appendix A* in section 10.3, these roles present their findings and in unity analyze, and evaluate and their findings throughout the test. With each role having their own unique focus during the test, the evaluation should be effective and produce very useful information and results.

5 User study 1: Prototype testing

5.1 Test design description

We have discussed multiple design options for the user interface, these include a scroll function option in the orange text field and the option of allowing the user to hide the orange text field box with a speech bubble icon. these can be read in more depth in *Appendix A* in section 10.2. We have decided to create high fidelity interactive prototypes of the first design concept with the orange text box seen in Figure 13 and the second design concept with the preview page on the keyboard which can be seen in Figure 14, both of these can be found in *Appendix C - Design.* The method and usage of high fidelity prototypes are also described in section 10.3.3. These two prototypes will be tested and based on the test results we will determine which design to continue with. For this project, we have decided to create interactive prototypes through the page figma.com.

The benefit of creating interactive prototypes is that the closer the prototype resembles the final product, the better and more accurate the test results will be [24].

5.2 Test design rationale

We have decided to test 5 people to get insight into their thoughts and how they use the current design prototypes we have created. The test is structured as a usability test where they are provided with background for the test and given the prototype, which is on a smartphone, in hand, afterwards we will ask them to complete tasks that give us insight into how intuitive our design is and if there should be made any changes.

We created 3 prototypes, which can be seen below in figure 3. The first prototype given to the test participant was A) the default keyboard, this was given to gather control data and insight if there actually is a problem or difficulties in the field we are researching.

Then the test participant was given prototype B) Orange Box and emoji-text. This prototype was created to explore how this specific design of the orange preview box design would be received and handled by the user. We also wanted to test a feature on the emojis, this feature is simply if the user holds down on an emoji the descriptive name would pop up, we also used this test to gain insight if users actually know the names of commonly used emojis.

We discussed if there were any other possible design options that might be better, and came up with a new idea we would like to test. The third prototype C) Keyboard preview page was created to test how the design of the eye button and a separate preview page in the keyboard would be received by the user. We want to test how the user would react to this design and then we can evaluate the difference between the prototypes.

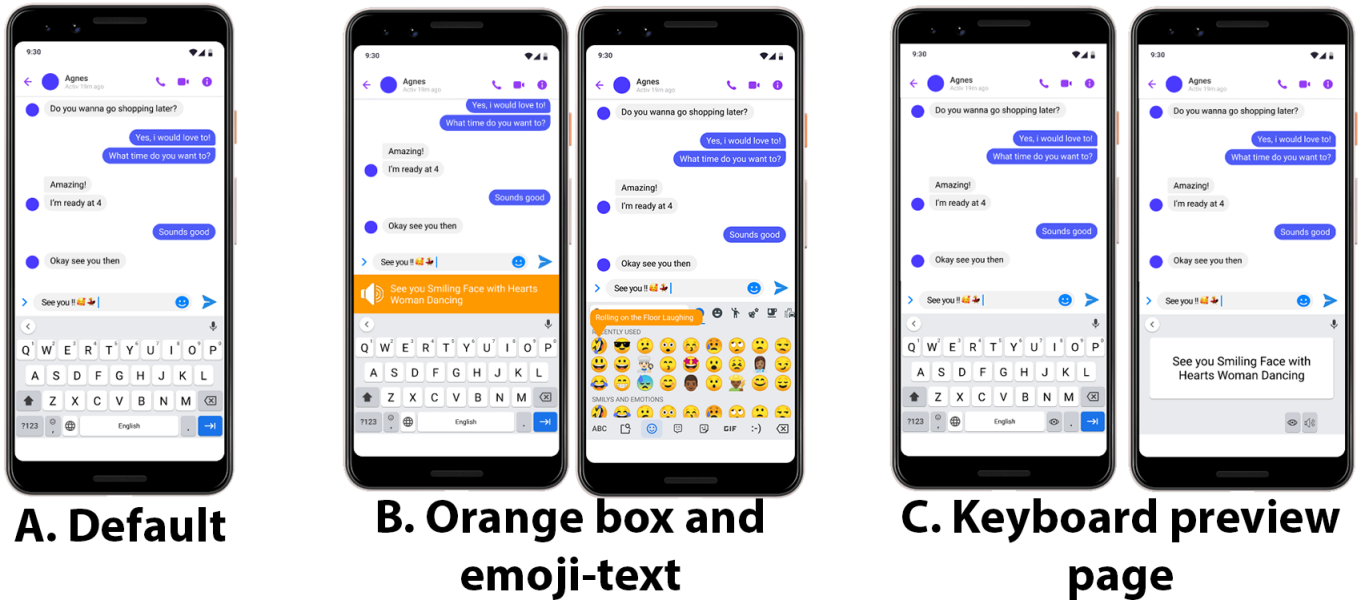


Figure 3: Variations of prototypes for the test A) Default keyboard B) Orange preview box and emoji-text test C) Keyboard preview page

5.3 General goal of the study

The goal of this study is to test different design options and to gain insight into how people with normal vision would consider emojis and messages when texting a vision impaired person. The secondary purpose is to test the designs as an interactive prototype created in Figma, which can be seen in figure 3. Based on the testing of the three designs, we hope to get insight into which design would fit the project and its purpose best and iterate further on the design based on the test results.

5.4 Participants

We tested the prototypes on 5 participants, 1 woman and 4 men who ranged from the age of 22 to 29. These participants were selected based on their familiarity with text messaging. All the participants are Danish students at Aalborg University taking the software bachelor and are in the 6 semester. None of the participants have experience with writing messages to vision impaired and therefore can give us great insight into how people with no prior experience would assume a screen reader to interpret emojis and messages.

5.5 Procedure and user stories

We prepared a user story and tasks before the test to ensure the test person had enough background information to understand the test purpose and questions correctly.

The user story for this test was:
You are in the process of writing a message to Agnes, who you know is visually impaired/blind and uses a text-to-speech program to have her messages read out.

We prepared these questions for each of the 3 prototype tests, the test was originally conducted in Danish but the questions here have been translated to English:

Questions for A) Default

- Is there anything you would change about your current message?
- How did you come to the conclusion that this should be changed
- How do you think the message will be read to Agnes?

- Can you check it?

Questions for the orange box in prototype B) and for prototype C. Keyboard preview page

- How will you find a preview of the text as it will be read out by Agnes screen reader?
- Now that you see your message in this new design, is there anything you would change about your current message?
- Why do you want to change this?
- How did you decide that this should be changed?
- How do you think the message will be read to Agnes?
- Can you check it?
- Is it clearer in this design how the message will be read?

Questions for the emoji-text in prototype B)

- Can you figure out the name of the first emoji?
- You want to insert a smiley that is laughing and slanted
- Did you know it was called that?

Final statements

- Which solution do you prefer?
- Would you consider using one of these designs?
- What do you like about the different designs?
- What do you think is less good?
- How could they be improved?

5.6 Results and conclusion

After the prototype tests, we analyzed the data through the Instant Data Analysis (IDA) method [23] also described in section 4.1.

It can be concluded based on prototype **A) default** which can be seen in figure 3 that the participants were aware that the emojis might be a problem for the screen readers to read to the vision-impaired Agnes. However, none of the participants knew how the "🥰💃" emojis would sound and 2-3 of the participants would rather just remove the Emojis than confuse. They could not find a way to check how the message would be read aloud to Agnes, mainly because most default keyboards do not have that option, and if they do they are not easy to find. This showed that in general, the default keyboard used on Android phones does not provide the user with enough options to consider their messages to vision-impaired people using screen readers.

Based on the results we gathered from questions regarding prototype **B) Orange box** also seen in figure 3 was that the orange box was very visible and easy to find. The orange preview gave great insight into how a screen reader would receive the written message. The preview gave insight into how the emojis would be translated from "🥰💃" into "Smiling Face with Hearts Woman Dancing", which the participants said changed their perception of the context of the original message. The participants were also not aware that screen readers do not read "!!" when it is not standing alone or used as a punctuation in a sentence, the preview made them aware of this and they learned that through looking at the preview. They found the speaker button seen to the left of the preview text to be somewhat intuitive, one of the participants stated that the button could be more visible as a "button". The participants also thought the orange box was huge and consumed the screen too much for their liking, which we also predicted in the Design section 10.2 in *Appendix A*, some participants wished that the orange box was optional and could be hidden so it was not always present, or maybe it could be automated to be shown on specific people and not everybody.

The test of the emoji-text is also seen in prototype **B) Orange box and emoji-text** seen in figure 3 showed us that 4 out of 5 of the participants were unaware of the name of the 🤣 Emoji, or any of the Emojis in particular.

They all found the ”long tap” on the Emojis very intuitive and an easy and quick way to find the Emojis’ descriptive name. This gave the participants more insight into considering the emojis when knowing the receiver was using a screen reader.

The test on the prototype **C. Keyboard preview page** seen in figure 3 showed that the participants liked the space given on the extra page for the preview, they thought the eye icon button was clear but 2 of the participants said an ”ear” icon would maybe make more sense, one participant thought that it would help make the button more visible as it disappeared a bit too much into the normal keyboard. This design was not consuming the screen the same way as the orange box design, however, some of the participants missed seeing the preview while they were writing and found it annoying they had to switch pages to see the whole preview. This design also gave the participants great insight into how the message was going to be received by the screen reader and the speaker button was clearer for the participants.

When asking the participants for their **Final thoughts** of both designs individually and in comparison, there were very mixed opinions on which design they preferred. The orange box prototype was easy to spot, and seeing the ”live” feedback was more intuitive and felt better than switching to a different page. However, the speaker button were more visible and the space in prototype C was a smaller area of the screen needed to support the features.

Based on the data gathered in these prototype tests, we have created a new design that combines the best design choices of the prototypes **B. Orange box and emoji-text** and **C. Keyboard preview page**, the new design can be seen in figure 5. This design keeps the orange box however there is an expand and minimize feature added through an arrow button, this allows the preview to handle longer text messages without the orange box expanding upwards and blocking more of the screen. The speaker button has also been changed in color and a grey box has been added around it, this makes the button look more intuitively ”clickable”. We have also added an ear button that allows the user to toggle whether the preview is shown or hidden. This button was previously an eye icon, however, 3 of the participants commented that an ear icon would be more fitting, the change of this button can be seen in figure 4.

Figure 4: Eye and ear icon in discussion

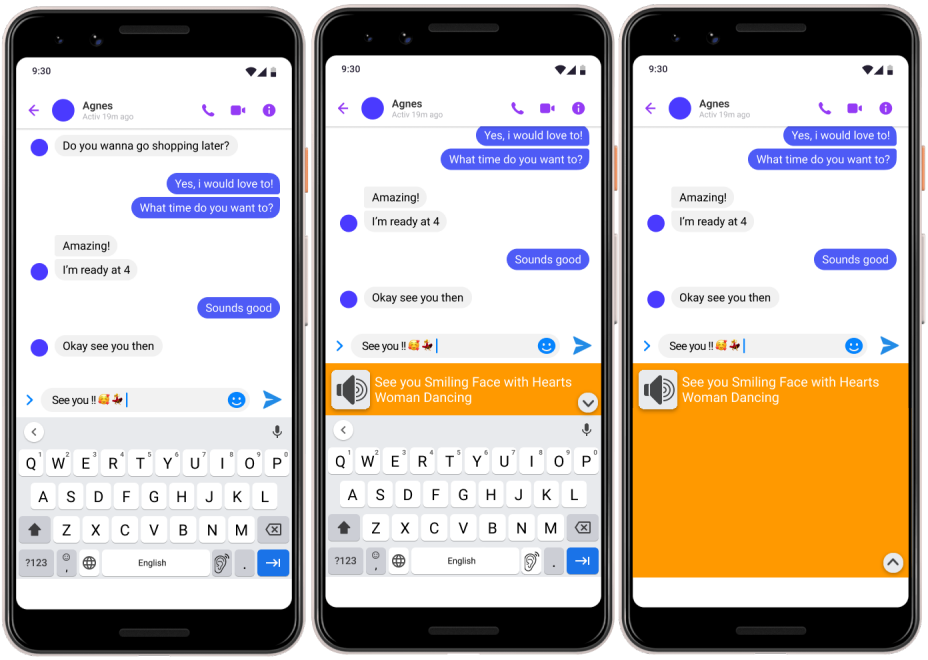


Figure 5: Design based on test results

6 Implementation

Before going into the specific details of our implementation, it is essential to outline the primary goals that guided our development process. These goals describe the main challenges that should be solved in the implementation. Firstly, we need to extend an open source keyboard by adding new functionality. Secondly, we need to access any input field the user is typing in and get its contents. Thirdly, we need to show a transcription while the user types to provide a real-time, on-screen transcription of what a screen reader would vocalize as the user types. This includes all characters, symbols, words, emojis, and their variations. Lastly, we want to have a button that reads the messages aloud to allow the user to preview how their message will be vocalized.

6.1 Open source keyboards

We have looked into a variety of open source keyboards to build and implement on. We have looked at keyboards based on certain requirements, these requirements include; *Modifiability*, *Design*, *Completeness*, *Ease of Development* and *Emoji implementation*. With these requirements in mind, we have looked at the following keyboards, AndroidCustomKeyboard, Google SoftKeyboard, Anysoftkeyboard, and florisboard. However, these either lacked easy modifiability or did not support emoji implementation, in the end, we decided to develop on the open source keyboard "OpenBoard". The selection of open source keyboards has been described more in depth in *Appendix B* in section 10.2.1.

6.2 Extending the keyboard

When we extended the OpenBoard keyboard with our own orange box component and its functionality we started by splitting it into two "objects". One reason we did was so we could split up and have two group members focusing or the visual XML object and another two group members could focus on the functionality and the transcriber object. This would allow for more productivity in developing the product from "multiple" sides at once.

The visual orange_preview_box XML file is extended in the main_keyboard_frame XML file, the orange preview box element is therefore instantiated by this component. We also had to extend the height of the keyboard to allow space in the main keyboard frame's input field, otherwise the orange box would be on top of the existing keyboard but not a part of the input field in which the keyboard would register the users input. The OrangeScript class is responsible for handling interactions with the buttons in the orange box i.e. increasing and decreasing the size of the Orange box, and reading aloud.

The functionality of the orange box is the Text Visualizer component which is responsible for storing and updating the text in the orange box. Text that should be put into the text field is given by our Text-to-speech service, to allow this behavior across the service lifecycle and activity lifecycle we made static methods for setting and updating the text field string.

6.3 The implementation of the Accessibility service

We extend the built-in accessibility service provided by Android to access user input in any text field. This approach was selected for its simplicity in accessing text fields across various applications. Our accessibility service is a distinct object from Android that extends the abstract class AccessibilityService. Within this service, we override the onAccessibilityEvent method. Here, we begin by verifying that the accessibility event is a change in a text field. Upon detecting such a change, we capture the modified string and tokenize it, separating each word by a comma. Subsequently, we invoke the text-to-speech functionality to initiate the translation process, which will be elaborated upon in section 6.4.

6.4 The implementation of the transcription

One of the fundamental issues that shaped the design of this program was how to translate input from any text field into a string composed solely of words that accurately reflect what a text-to-speech system would vocalize, including characters, symbols, words, and their variations. In order to ensure the system's adaptability to diverse scenarios, we utilized the built-in text-to-speech functionality in Android. This feature generates an audio file, which is then processed by a speech-to-text model to produce the text displayed in the orange box. This approach enabled us to address the myriad of cases efficiently and effectively. To optimize the speech-to-text conversion process, we modified the input to the text-to-speech system to be separated by commas. This adjustment facilitates the speech-to-text model's understanding of each individual word. For instance, in cases such as "can butter fly" where the absence of comma separation could result in misinterpretation ("in butterfly"), the use of comma separation ensures accurate transcription, yielding the output "can butter fly".

6.5 Rationale for using Androids text-to-speech functionality alongside Vosk for Speech-to-Text conversion

When deciding how to implement the code, we wanted to make the right choices for optimizing our engineering efforts by choosing appropriate frameworks and libraries instead of "reinventing the wheel". Some of our considerations regarding our translator and the incorporation of text-to-speech and speech-to-text will be discussed below.

Since the text-to-speech accessibility service is the primary screen reader for Android products, we decided to use its capabilities for converting text-to-speech within our implementation. We needed this to provide an audio preview of the typed text from the user and also to afterward convert it to speech-to-text. To get this integration, we incorporated Google's open source code[25] into our codebase. By using the text-to-speech in Android, we can ensure consistency between users of Android devices, which means that our resulting translation, of the written text on the keyboard, will match what would be heard on a screen reader by someone with a visual impairment.

For the text that is displayed in the orange box of Echoboard, we decided to use speech-to-text, because it demonstrates how a screen reader would read the text. Unfortunately, Android's own API for speech-to-text conversion called SpeechRecognizer, does not provide support for speech conversion out of audio files, which is a core requirement of our implementation. This limitation of SpeechRecognizer makes it unsuitable for our purpose and ultimately resulting in us opting not to utilize it.

Therefore, for converting speech-to-text, we have chosen Vosk, an offline speech recognition API. This implies that it works locally on the device and doesn't have to constantly communicate with external servers through an internet connection. By using an offline solution, the app will work even when there isn't a reliable internet connection, contributing to a better user experience, for example when using SMS. There will also be less delay between each call to the transcription service, which we prioritized because we aimed for a live translation. Vosk is an automatic speech recognition (ASR) which means that it makes speech sounds into text[26]. Vosk is open-source and works with more than 20 different languages[27]. A study by Pereira et al. has compared Mozilla DeepSpeech, CMU Sphinx, and Vosk with Google Speech AP concluding that they recommend using Vosk and that it has the best accuracy results of the three and it is similar to Google Speech API[26].

Vosk provides both smaller and larger models, the larger models are intended for servers, while the smaller ones are intended for use on devices with limited capabilities/resources[27]. Vosk offers two small English models, one that is 40MB, and another that is 128MB[27]. We've decided to utilize the 128MB model, despite the increase in computation time, due to the substantial improvement of accuracy which we prioritized.

6.6 How everything comes together

In Figure 6 we illustrate the comprehensive process of capturing and displaying user input in the orange box. When a user starts typing in the input text field, our accessibility service detects the change and captures the modified string. This string is then tokenized, separating each word by a comma to improve the vocalization by the text-to-speech (TTS) system. The TTS functionality converts the tokenized text into an audio file, which is subsequently processed by a speech-to-text (STT) model to generate a transcription. This transcribed text is sent to the Text Visualizer object, responsible for storing and updating the text displayed in the orange box. The orange box, integrated into the main keyboard activity, then shows the updated text. Users can resize the text field to view more of their text and get the finalized transcripted string read aloud by text-to-speech.

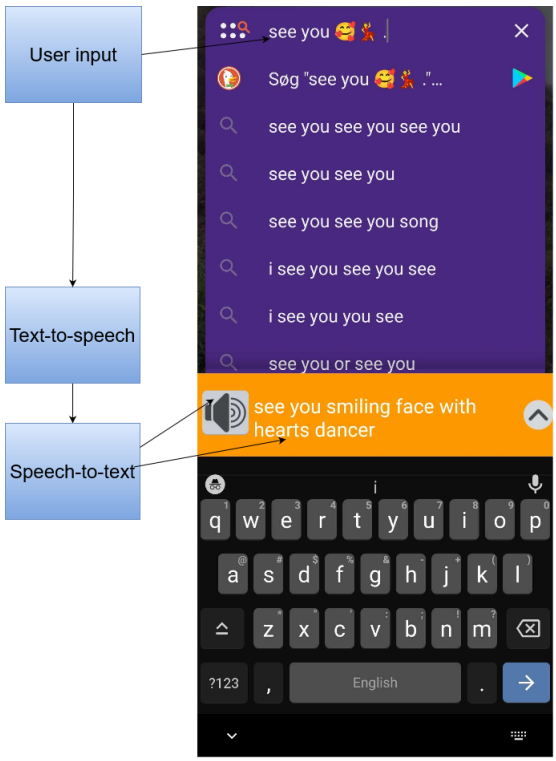


Figure 6: A figure of the user input and how text-to-speech is linked to speech-to-text which is then outputted to the speaker button in the keyboard and the transcription in the orange box.

7 User study 2: Usability testing

7.1 Usability test design description

For the usability test, we wanted to test our implementation of Echoboard in person, and the participants were able to test it physically on one of our smartphones. We wanted to take a qualitative approach and observe how the participants interacted with the keyboard.

7.2 General goals of the study

The goals of the usability test were to test how the users would interact with the product, determine whether the UI/UX was intuitive, and ask the participants if they liked the design. In addition to testing the design, we also wanted to find out if Echoboard helps them understand how a screen reader reads text aloud and if they prefer to

change the text messages after they see the transcription in the orange box.

7.3 Participants

For this project, we conducted tests on 6 participants, 1 woman and 5 men, all of whom are in their twenties (age range between 21 to 27). All the participants are students from Denmark studying Software at Aalborg University, Copenhagen. A criterion was that they were not visually impaired, since the keyboard is designed for sighted individuals.

7.4 Procedure

Before the test, several questions were prepared which we asked the participants during the tests.

Similar to the prototype test we conducted earlier, we presented a user story for the participants, to guarantee that the participants had enough background information to understand the questions as well as the purpose of the test correctly.

The user story for this test was:
You are in the process of writing a message to Agnes, who you know is visually impaired/blind and uses a text-to-speech program to have her messages read out.

We decided to ask the participants to write four different messages to test Emojis, capitalized words, repeating exclamation marks, and elongating words. Lastly, we wanted to allow the participants to write whatever they wanted, to find errors that we did not know about.

Steps

- We wanted to test how much the participants knew about screen readers before showing them our keyboard. Through this approach, we can evaluate whether the keyboard has enhanced their comprehension of how screen readers work and how a screen reader would voice the messages by the end of the test.

  We prepared varied messages which each had unique translations. We chose to have one of each example to get a broad evaluation of the keyboard, while still keeping the test process effective and focused. With these examples, we were able to see the participants' reactions to various messages. We chose not to test multiple types of messages for each scenario, as it could result in repetition and possibly excessive data collection. The messages we selected for the beginning of the test, were messages that we knew would be translated correctly by Echoboard.

  We used the message: *"see you 🥰💃"* since we also used it in our prototype test. However, we also wanted to explore another Emoji in the usability test, to find out if they are more aware of the name of that Emoji. We chose the Emoji 🙏 because it is ambiguous as to its meaning, and we want to check if the participants either know or can guess it is called "Folded Hands", based on its design.

  Therefore, we started the user test by showing them examples of four messages, one by one, and asked the participants:

  How do you think these messages will be read aloud by a screen reader?
  "Hey, can you take my shift this weekend 🙏🙏"
  "see you 🥰💃"
  "PLEASE HELP!!!!!"
  "Miss yooooooooou"

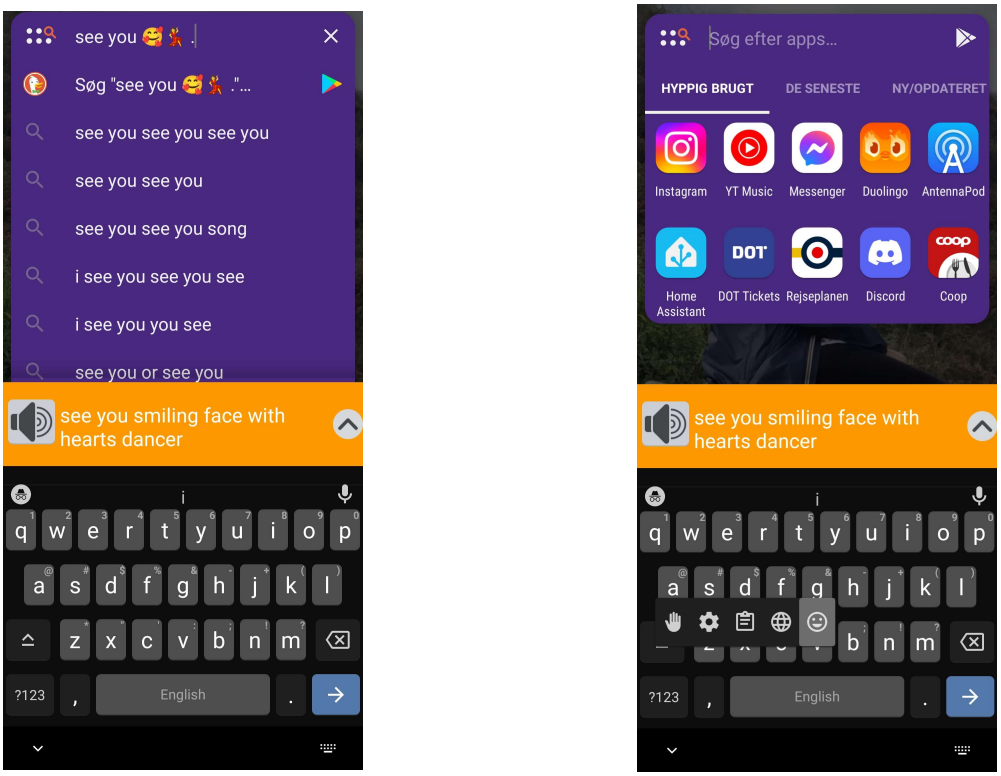
- Afterwards, we asked the participants to write the messages in a text input field on the keyboard of an Android device one by one. After each message, we asked if they had any comments:
  Write the messages above to Agnes
- The subsequent question was asked within the same context as they typed on the keyboard:
  Can you see on the screen how the messages will be read by a screen reader?
- To see if the purpose and design of the speaker button were intuitive we asked:
  If you want to hear the message read aloud for you, is there a button that can do that?
- After the participants were able to see the text in the orange box, we wanted to examine if our keyboard alters their perspective on the messages and makes them reconsider how they are writing. Therefore we asked

them:
What do you think about the text that is displayed and is there anything you would change?

- After this, we wanted to give them a chance to write whatever they wanted to discover any edge cases that we were not yet aware of:
  Write a long message
- We wanted to see if the participants would notice the scroll bar:
  Now that the text is very long, what will you do to be able to see the entire text?
- Then we wanted to see if the expand button was intuitive:
  Is there a way to expand the preview so you can see the whole text?
- Final question was:
  Can you minimize it again?

Evaluation

- Was it easy to find and read the preview?
- Was the final message in the preview the same as you expected
- Was the speaker button easy to understand and use?
- Was the expand/minimize button easy to understand and use?
- How much do you think the orange box helps understanding how a screen reader reads messages aloud?
- Would you use Echoboard?



(a) An example of how *"see you 🥰💃"* appears when typed into a text field.

(b) To open Emojis, the user has to press the comma button.

Figure 7: Echoboard

7.5 Results from the usability test

To evaluate the results from the usability test, we have used the Instant Data Analysis (IDA) method [23], also described in section 4.1.

We were already aware of some issues with Echoboard, requiring us to give some extra explaining to the participants. First and foremost, you need to press the space key before a call is made to the API, which is not very intuitive. One participant suggested that it should refresh when you pull up, which is something to consider for future improvements. Besides this, for inserting Emojis the comma button must first be pressed, see Figure 7b. We knew about these issues, so we decided to explain the inadequacies and not let them figure it out themselves. The reason behind this was to make the test run more smoothly and to make sure that the participants mainly focused on other areas of interest.

We asked the participants if they knew how the messages would be read aloud by a screen reader. We began with this question to subsequently observe if Echoboard helps them understand how a screen reader works.

"see you 🥰💃" & *"Hey, can you take my shift this weekend 🙏 🙏"*
Four out of the six participants knew that the text would be read aloud followed by the names of the Emojis, however, they did not know how the Emojis would be translated. The last two participants thought that the screen reader somehow would emphasize that it was reading an Emoji e.g. "see you loving emoji dancing emoji".

"PLEASE HELP!!!!!"
Only one participant predicted that the preview would remove the exclamation marks. Five out of six participants thought that the screen reader in some way would indicate that there were several exclamation marks. One of these five participants assumed that the screen reader would scream the text, while two other participants likewise thought that it would sound more definite. Two out of the five participants thought that it would read "exclamation mark" five times repeatedly.

"Miss yooooooooou"
All participants thought that a screen reader would read every "o" in "yooooooooou", which implies that it would extend the word. They were surprised when noticing that the text in the orange box did not include the extra "o's". One participant assumed that the screen reader would read the text in a teasing voice.

When we asked the participants to write the four different text examples on their own, they noticed a change in the text displayed in the orange box. For each text message, we asked them about their thoughts before moving on to the next message. One thing that surprised all the participants was the translation of the Emoji "🙏", the majority believed that the name of the Emoji was "praying" and some thought that it was "high-five". They also noticed that the text turned into lowercase, that the exclamation marks were removed, and that the word "yooooooooou" was shortened. Most participants pointed out that these changes could possibly result in missing certain aspects of the intended context. One participant suggested writing "please help this is really important" to replace the exclamation marks to maintain the expression and emphasize the urgency of the message. Another participant suggested writing "Miss you sweety" instead of "Miss yooooooooou". One person wrote "yooooouh" instead to try to trick the translator, but it was translated to "yoga" which was not the person's intention. These three proposals are all examples of adding additional text to try and keep the context and tone of the text to avoid misunderstandings.

All participants knew right away what the speaker button was for, it seemed very intuitive for them to press it and they accurately anticipated the text to be read aloud. However, a participant suggested that it would be even better if the speaker changed its color when active to clarify that it is playing a sound.

The translator did not always work correctly when the participants were asked to write their own message. When asking the participants to write a long text, we observed that e.g "tekken" was mistakenly converted to "check in", and when a participant tried to write a very long number it was also translated wrong, see Figure 8.

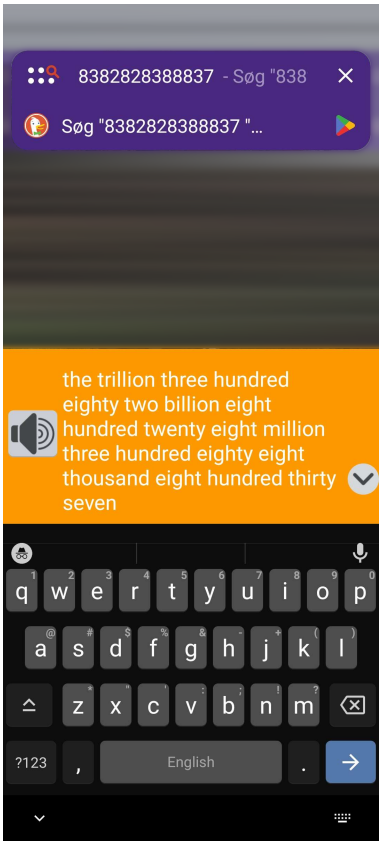


Figure 8: An example of how *"83828388837"* appears when typed into a text field.

The expand/minimize button was intuitive for all the participants, but most participants did not notice the scroll bar and it was required for us to point it out before they became aware that it existed. To improve this issue, we could make the scroll bar visible at all times.

We got several comments on the design of the keyboard. One participant would prefer the speaker button to be smaller since it takes up a lot of space. Another person also brought up a point about the space limitation and suggested that the padding around the text field should be reduced. One person did not like that the arrow was visible when they did not need it. The person thought that the arrow should only appear when the text was too long to be able to see it. The person said the same thing about the scroll bar, i.e. that it should not be there until it is needed. Generally, the participants liked the design and found it intuitive, only two persons were a bit skeptical about the visual aspect of the keyboard. One mentioned liking the text size and the orange background color, however, the person expressed that the design did not look modern because of the speaker button and the grey color. The other person expressed that the orange and grey colors did not match that well.

All participants thought that Echoboard improves their understanding of a screen reader and said that they would use Echoboard if they had a friend with impaired vision. One participant mentioned that it provides the opportunity to adjust your message for the receiver of the message. Another person mentioned that it helped to realize what different Emojis are called.

8 Discussion

This section contains a more in-depth discussion regarding the thoughts, work process, and issues encountered and solved throughout this project.

8.1 Implementation of MoSCoW requirements

Earlier in the project, the group defined the products requirements through the MoSCoW model in section 3. These can also be seen in section 10.1 in*Appendix A* When looking at the final product, its completion can be concluded based on its fulfillment of the must have and the other requirements.

It can be seen that the Echoboard does fulfill all the must have requirements determined and described in section 3. The Echoboard works in *Portrait mode* and works with the *English language*. In terms of visual design, the Echoboard has a *Orange box below the input field*, whereas the orange box is *Scrollable*. The Echoboards *translator function* works as it writes the user's input into the orange box. One caveat is the must have requirement*CLDR short name of emojis* written about in section 10.1.1 was made before we discovered that the text-to-speech service in Android used its own translations for emojis that sometimes differ from their CLDR name an example being the emoji 💃 with the code "U+1F483" which has the CLDR short name "woman dancing"[28], but is read as "dancer" by the text-to-speech service in Android seen in Figure 7a. We still believe that this requirement was achieved as the translations made by our system closely match how any emoji would be vocalized by the text-to-speech system in Android.

Therefore the final Echoboard developed throughout this project fulfills all the must have requirements.

Since the group successfully implemented all the must-have requirements, they moved to implementing the should have requirements. However, due to time consumption and the project deadline, only two of the should have requirements were implemented. The Echoboard has a *Speaker button* which reads the users' text input from the orange box aloud. The Echoboard does also have a *Expand box toggle button* which allows the user to expand and minimize the orange box to their liking.

None of the won't have requirements were implemented as they were not meant to. Based on that and the successful implementation of the requirements listed above, it can be concluded that the Echoboard fulfills the requirements to be deemed as a successful product within our project. However there could still be a lot of improvements, these will be discussed further in this section.

8.2 Broader Implications and Recommendations

8.2.1 New findings based on the user study

Through our user studies, we learned that some sighted users are not aware of the names of emojis and they cannot predict how a text reader will be read aloud. This matches with what we expected based on our background research in section 1, even though we only tested 11 people across both user studies it is telling that all of our test subjects struggled with this exact issue revealing that it can be a problem for sighted users when texting vision-impaired users.

In our prototype user study, when testing the complete default Android keyboard, 2-3 of the participants would rather remove the emojis than cause unnecessary confusion because most default keyboards do not show the user how their message would sound. However, as researched in section 1, Emojis are graphic illustrations used to visualize and give the user indirect context and convey emotions or concepts. This indirect context and concepts are completely lost when the sighted user decides to remove the emojis, we want to prevent this and make emojis more accessible. Through the prototype test, we find that it is possible to improve the sighted users' understanding of screen readers.

In our usability test conducted after the development of the Echoboard, our aim with the product to improve in the sighted users' understanding of screen readers' interpretation of emojis. When testing the participants, we first tested what they thought various sentences would be translated to, such as the examples; *"Miss yooooou", "PLEASE HELP!!!"*. All the participants were surprised by what they thought the translation would be and how the screen reader would interpret it. This only indicates further that the sighted users did not understand screen readers enough and that their messages could cause confusion for vision impaired. However, through our usability test, we found all the participants experienced that the Echoboard significantly improved their understanding of screen readers and how they interpret messages.

8.2.2 The Text-to-Speech and Speech-to-Text approach

The approach we ultimately selected is extensively discussed in section 6. It involves initially converting the user input into an audio file using a text-to-speech service. Subsequently, we transcribe this audio file using a speech-to-text service and present the output to the user. While this approach is functional, its transcriptions suffer from imperfections and demand significant computational resources.

Ideally, we would get the text directly from the text-to-speech service ensuring consistency with what it would read aloud, but Google does not include this in the text-to-speech API. It would ease our process if Google provided access to the preprocessing step, which we assume exists, within Android's text-to-speech service. We think this step transforms the input string to consist solely of words before synthesizing it into sound. This would address the aforementioned issues by reducing the amount of services involved and having the precise string made by the text-to-speech service. Without access to this preprocessing step, the transcription differs from the expected outcome in some cases.

8.2.3 Addressing external factors to Echoboard's success

The current version of Echoboard is in the form of a minimum viable product, for us to streamline the implementation process, and develop Echoboard as we envisioned. There are a few important considerations to be addressed. We will discuss issues with each of these considerations, and then propose suggestions for improvements.

Currently, Echoboard obtains user input, through the use of accessibility services, though this solution obtains data everywhere there is an input field, e.g. search bars, additionally, it serves as a barrier, since the user has to give accessibility permissions before using Echoboard. This could be prevented, if messaging apps provided an API, where data can be obtained from the messaging field, as well as identifiers of the people messaging, which will allow to only show the orange box preview when desired.

Google's documentation on Android development lacks detailed explanations, especially in terms of integration of accessibility services, this posed an issue since we deemed the best approach for implementing a big part of Echoboard's logic was through accessibility services. In the design and implementation phases, we were uncertain if the accessibility service approach was possible, and if the data granted would suffice for our needs. This uneasiness was further fueled by the lack of resources on the accessibility implementation on the internet by others sources. Had Google provided more detailed explanations, it would have greatly streamlined our development process.

A challenge we ran into while implementing is that none of the group members had experience with development on Android keyboards, however, some of the group members have worked with Android Studio prior to this project. We have not worked with OpenBoard before or worked with the structure that OpenBoard is written in. We have some experience further developing on a project, however, the lack of documentation on OpenBoard posed as a slight challenge. This especially proved difficult when we worked on extending the visual design of the keyboard and implementing our orange box preview, it would have been more straightforward to develop on with more documentation and guide on the structure of the already existing OpenBoard project.

8.2.4 Implications for other target groups

While this study focuses on people with visual impairment, we also see other groups that could be better accommodated by using the Echoborad for their textual communication.

Firstly, both iOS and Android allow for messages to be read aloud while users are using headphones, which is particularly useful in situations where they are unable to read their notifications [29] [30]. As observed in both the prototype tests (see Section 3) and user tests (see Section 7.5), most users were surprised by how the text-to-speech system read their messages. Therefore, we propose that mobile phone developers could incorporate a feature similar to Echoboard into their keyboards. This feature would activate while a user is writing a message for someone who is using text-to-speech, ensuring the message keeps the intended meaning when read aloud.

8.3 Limitations

During the project duration, we have experienced several factors, that have posed as a limitation. These factors are classified into their respective groups, "Technical Limitations" and "User Study", and we will discuss their effects on the project.

Technical limitations
We faced multiple limiting factors during the development process. Screen readers do not provide a function that outputs an textual interpretation, we had to create our own, and we decided to create the Dual Conversion solution, by making use of TTS and STT services. For TTS, we wanted to utilize an API from Android's default screen reader, TalkBack, giving us a one-to-one interpretation, but a TalkBack API simply doesn't exist, and we opted out to use the next closest thing, being Android TTS. We decided to use free APIs for TTS and STT, even though the accuracy they provide is of subpar quality to that of paid alternatives, additionally for STT, the API we chose, Vosk, works offline. Hardware constraints on mobile devices, limited the scope of available Vosk models to lightweight, resulting in lower translation accuracy.

User Study
In hindsight, some aspects of the user study could have been more comprehensive. The sample size is inadequate in terms of broadness, in contrary to our product not having a specific target group bound by age or technical knowledge, the study focuses on a tech savvy target group, with ages between 20-27. The knowledge gained from the user study is insufficient in terms of encompassing the target group as a whole, meaning that the current result may not completely align with the overall results of the entire target group. The frequent inaccuracy of Echoboard captures the focus of the user study participants and diverts their attention from evaluating other aspects of the keyboard. As mentioned above in *Technical limitations*, due to having implemented Androids TTS service, the user study tests on a TTS service, and not a screen reader. Even though there is a strong similarity between the two, we should be somewhat skeptical of the user study results as they may not exactly representing a screen reader's interpretation.

8.4 Further development

In considering the current state of our project, we envision numerous opportunities for further development and enhancement. Building upon the foundation laid so far, there are several promising features that could be implemented to expand the functionality and usability of our system. In this section, we explore these potential enhancements, ranging from refining existing features to introducing new functionalities. By continually advancing our application, we aim to deliver an enriched user experience and extend the utility of our solution.

8.4.1 Highlight problems

The first feature we wanted to highlight is the red underline functionality, designed to highlight errors within the text displayed in the orange box. This visual cue will draw users' attention to areas where revisions may be needed, facilitating easier identification of potential issues. By integrating this feature, users can quickly identify problematic sections of text and proceed to make necessary adjustments, thereby enhancing the overall clarity and quality of their written content (See Moscow model: section 10.1.2).

8.4.2 Error suggestions

Building upon the red underline feature, we plan to incorporate a suggestion mechanism accessible upon clicking the highlighted text. Upon clicking the red underline, users will be presented with a menu of suggested corrections tailored to the identified error. This interactive feature aims to assist users in refining their text by offering helpful suggestions for improvement. By providing users with actionable recommendations, we aim to further streamline the editing process and empower users to produce more polished and refined content. (See Moscow model: section 10.1.4).

8.4.3 Long press for emoji descriptive name

This feature would simplify the process of identifying emoji names, as demonstrated in the prototype tests in section 3 sighted users do not know how emojis are interpreted and read aloud by screen readers, or in general what the descriptive names of the emojis are. By incorporating this functionality, users can conveniently ascertain the name of an emoji before incorporating it into their textual communication. This preemptive knowledge empowers users to make informed choices regarding emoji usage, ensuring clarity and alignment with their intended message.

8.4.4 Landscape mode

Another feature that would enhances the versatility of the Echoboard is the introduction of landscape mode. This functionality enables users to utilize the Echoboard while their device is oriented sideways, providing a more flexible and comfortable writing experience. By supporting landscape mode, users have the freedom to interact with the Echoboard that suits their preferences and workflow. Whether typing out lengthy documents, writing down quick notes, or engaging in collaborative brainstorming sessions, landscape mode offers added convenience and accessibility. This feature expands the range of which the Echoboard can be effectively utilized. Furthermore, landscape mode enhances usability on devices with larger screens, maximizing screen real estate and optimizing the user interface for improved productivity.

9 Conclusion

Throughout the project, we have created an android keyboard to make sighted users more aware of how their textual messages are interpreted by screen readers and therefore how their messages are received by people who use screen readers.

The Echoboard product created throughout this project proposes a solution to this problem, which is also stated in the problem statement in section 1. By looking at related work and findings, the group has based their project on contributing to the area of Human Computer Interactions in perspective of accessibility for vision impaired users.

For this project, we have created some requirements that the product must achieve, these requirements were based on the prototype test conducted written in section 5. The fulfillment of these requirements are discussed in section 8.1, where it can be concluded that the product fulfills the most crucial requirements. Following the implementation of the Echoboard, a user study was conducted where the usability and understandment of the Echoboard was tested. The conclusion from the user study test was that the Echoboard gave the participants more insight into how screen readers would interpret their messages and use of emojis, this is written more in-depth in section 6.

10 Acknowledgments and the integration of AI

We want to thank our supervisor, a tenure-track assistant professor at the Department of Computer Science at Aalborg University in Copenhagen, we appreciate her support throughout the project, utilizing her expertise in Android app development and the subject of the paper. Additionally, we would like to thank all the participants from our prototype test and usability test.

When writing the paper, we have used the AI technology ChatGPT to contribute to correct grammar and rearrange sentences. We only used ChatGPT for some sentences and after using the tool, we still edited the text to fit our preferences. We take full responsibility of the paper and its contents.

References

[1] Unicode Consortium, *About Emoji*, en-US, Aug. 2022. [Online]. Available: https://home.unicode.org/emoji/about-emoji/ (visited on 02/21/2024).

[2] K. Broni, *New Emojis In 2022-2023*, en, Jul. 2022. [Online]. Available: https://blog.emojipedia.org/new-emojis-in-2022-2023/ (visited on 02/22/2024).

[3] A. Robertson, W. Magdy, and S. Goldwater, "Self-Representation on Twitter Using Emoji Skin Color Modifiers," en, *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 12, no. 1, Jun. 2018, ISSN: 2334-0770. DOI: 10.1609/icwsm.v12i1.15055. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/15055 (visited on 02/26/2024).

[4] M. Davis, The Unicode Emoji Subcommittee, and R. BEEN, *Gender Emoji ZWJ Sequences*, 2016. [Online]. Available: https://www.unicode.org/L2/L2016/16181-gender-zwj-sequences.pdf.

[5] G. W. Tigwell and D. R. Flatla, "Oh that's what you meant! reducing emoji misunderstanding," in *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, ser. MobileHCI '16, New York, NY, USA: Association for Computing Machinery, Sep. 2016, pp. 859–866, ISBN: 9781450344135. DOI: 10.1145/2957265.2961844. [Online]. Available: https://doi.org/10.1145/2957265.2961844 (visited on 05/21/2024).

[6] M. Davis and N. Holbrook, *UTS #51: Unicode Emoji*, Sep. 2023. [Online]. Available: https://www.unicode.org/reports/tr51/#Design_Guidelines (visited on 02/26/2024).

[7] H. Miller, J. Thebault-Spieker, S. Chang, I. Johnson, L. Terveen, and B. Hecht, ""Blissfully Happy" or "Ready toFight": Varying Interpretations of Emoji," en, *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 10, no. 1, pp. 259–268, 2016, ISSN: 2334-0770. DOI: 10.1609/icwsm.v10i1.14757. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14757 (visited on 03/05/2024).

[8] H. Cramer, P. de Juan, and J. Tetreault, "Sender-intended functions of emojis in US messaging," in *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, ser. MobileHCI '16, New York, NY, USA: Association for Computing Machinery, Sep. 2016, pp. 504–509, ISBN: 9781450344081. DOI: 10.1145/2935334.2935370. [Online]. Available: https://doi.org/10.1145/2935334.2935370 (visited on 03/05/2024).

[9] R. Kelly and L. Watts, "Characterising the inventive appropriation of emoji as relationally meaningful in mediated close personal relationships: Experiences of Technology Appropriation: Unanticipated Users, Usage, Circumstances, and Design," Sep. 2015.

[10] WHO, *Vision impairment and blindness*, en, Aug. 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (visited on 02/23/2024).

[11] IABP, *Definitions*, en-GB, Feb. 2024. [Online]. Available: https://www.iapb.org/learn/vision-atlas/about/definitions/ (visited on 02/26/2024).

[12] IABP, *Global Estimates of Vision Loss*, en-GB, Feb. 2024. [Online]. Available: https://www.iapb.org/learn/vision-atlas/magnitude-and-projections/global/ (visited on 02/23/2024).

[13] T. Bonsaksen, A. Brunes, and T. Heir, "Quality of life in people with visual impairment compared with the general population," en, *Journal of Public Health*, Jul. 2023, ISSN: 1613-2238. DOI: 10.1007/s10389-023-01995-1. [Online]. Available: https://doi.org/10.1007/s10389-023-01995-1 (visited on 02/26/2024).

[14] D. Göransson, *What is a screen reader?* en, Nov. 2019. [Online]. Available: https://axesslab.com/what-is-a-screen-reader/ (visited on 02/23/2024).

[15] WebAIM, *WebAIM: Screen Reader User Survey*, Feb. 2024. [Online]. Available: https://webaim.org/projects/screenreadersurvey10/ (visited on 02/23/2024).

[16] Google Support, *TalkBack - Android Accessibility Help*, Feb. 2024. [Online]. Available: https://support.google.com/accessibility/android/answer/6007100?hl=en (visited on 02/23/2024).

[17] T. et al., "Emoji Accessibility for Visually Impaired People," Apr. 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3313831.3376267.

[18] T. W. G. Griggio C. Gorman G., "Party Face Congratulations! Exploring Design Ideas to Help Sighted Users with Emoji Accessibility when Messaging with Screen Reader Users," 2024.

[19] G. et al., "Mediating Intimacy with DearBoard: A Co-Customizable Keyboard for Everyday Messaging," May 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3411764.3445757.

[20] D. A. Epstein, F. Liu, A. Monroy-Hernández, and D. Wang, "Revisiting Piggyback Prototyping: Examining Benefits and Tradeoffs in Extending Existing Social Computing Systems," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, 456:1–456:28, Nov. 2022. DOI: 10.1145/3555557. [Online]. Available: https://dl.acm.org/doi/10.1145/3555557 (visited on 05/20/2024).

[21] ProductPlan, *What is MoSCoW prioritization? Overview of the MoSCoW Method*, en-US, May 2024. [Online]. Available: https://www.productplan.com/glossary/moscow-prioritization/ (visited on 05/02/2024).

[22] M. Das, A. M. Piper, and D. Gergle, "Design and Evaluation of Accessible Collaborative Writing Techniques for People with Vision Impairments," *ACM Transactions on Computer-Human Interaction*, vol. 29, no. 2, 9:1–9:42, Jan. 2022, ISSN: 1073-0516. DOI: 10.1145/3480169. [Online]. Available: https://dl.acm.org/doi/10.1145/3480169 (visited on 05/28/2024).

[23] J. Kjeldskov, M. B. Skov, and J. Stage, "Instant data analysis: Conducting usability evaluations in a day," in *Proceedings of the third Nordic conference on Human-computer interaction*, ser. NordiCHI '04, New York, NY, USA: Association for Computing Machinery, Oct. 2004, pp. 233–240, ISBN: 9781581138573. DOI: 10.1145/1028014.1028050. [Online]. Available: https://doi.org/10.1145/1028014.1028050 (visited on 03/06/2024).

[24] UXPin, *A Quick Guide to Interactive Prototyping*, en-US, Jan. 2023. [Online]. Available: https://www.uxpin.com/studio/blog/interactive-prototype-setting-user-interactions-without-coding/ (visited on 02/27/2024).

[25] Google, *Github Talkback*, Apr. 2024. [Online]. Available: https://github.com/google/talkback.

[26] T. F. P. et al., "A web-based Voice Interaction framework proposal for enhancing Information Systems user experience," 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705092102233X.

[27] A. Cephei, *Vosk API*, Apr. 2024. [Online]. Available: https://alphacephei.com/vosk/.

[28] unicode, *Full Emoji List, v15.1*, May 2024. [Online]. Available: https://www.unicode.org/emoji/charts/full-emoji-list.html.

[29] Google, *Manage Google Assistant headphone notifications - Google Assistant Help*, May 2024. [Online]. Available: https://support.google.com/assistant/answer/7515695?hl=en (visited on 05/22/2024).

[30] Apple, *Announce notifications with Siri on AirPods or Beats*, en, May 2024. [Online]. Available: https://support.apple.com/en-us/102536 (visited on 05/22/2024).

[31] C. Black, *Building a custom android keyboard*, Mar. 2016. [Online]. Available: https://www.blackcj.com/blog/2016/03/30/building-a-custom-android-keyboard/.

[32] *AnySoftKeyboard*, en. [Online]. Available: https://anysoftkeyboard.github.io/ (visited on 05/02/2024).

[33] *FlorisBoard*, en-US. [Online]. Available: https://florisboard.org/ (visited on 05/02/2024).

[34] *Openboard-team/openboard*, original-date: 2019-12-31T17:17:45Z, May 2024. [Online]. Available: https://github.com/openboard-team/openboard (visited on 05/02/2024).

[35] X. Jiang, Y. Li, J. P. Jokinen, V. B. Hirvola, A. Oulasvirta, and X. Ren, "How We Type: Eye and Finger Movement Strategies in Mobile Typing," en, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA: ACM, Apr. 2020, pp. 1–14, ISBN: 9781450367080. DOI: 10.1145/3313831.3376711. [Online]. Available: https://dl.acm.org/doi/10.1145/3313831.3376711 (visited on 02/26/2024).

[36] M. W. V. S. et al., *The think aloud method: A practical guide to modelling cognitive processes*. 1994. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fe711ff90228ccfa1495d2deafb11c

[37] Android Developers, *AccessibilityService*, en, Apr. 2024. [Online]. Available: https://developer.android.com/reference/android/accessibilityservice/AccessibilityService (visited on 05/06/2024).

[38] Android Developers, *The activity lifecycle*, en, Jan. 2024. [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle (visited on 05/06/2024).

Appendix A - Initial analysis of design and implementation requirements and alternatives

10.1 MoSCoW

Figure 9, visualizes the MoSCoW requirements categorization and prioritization of our product, furthermore, a description will be provided for each requirement.

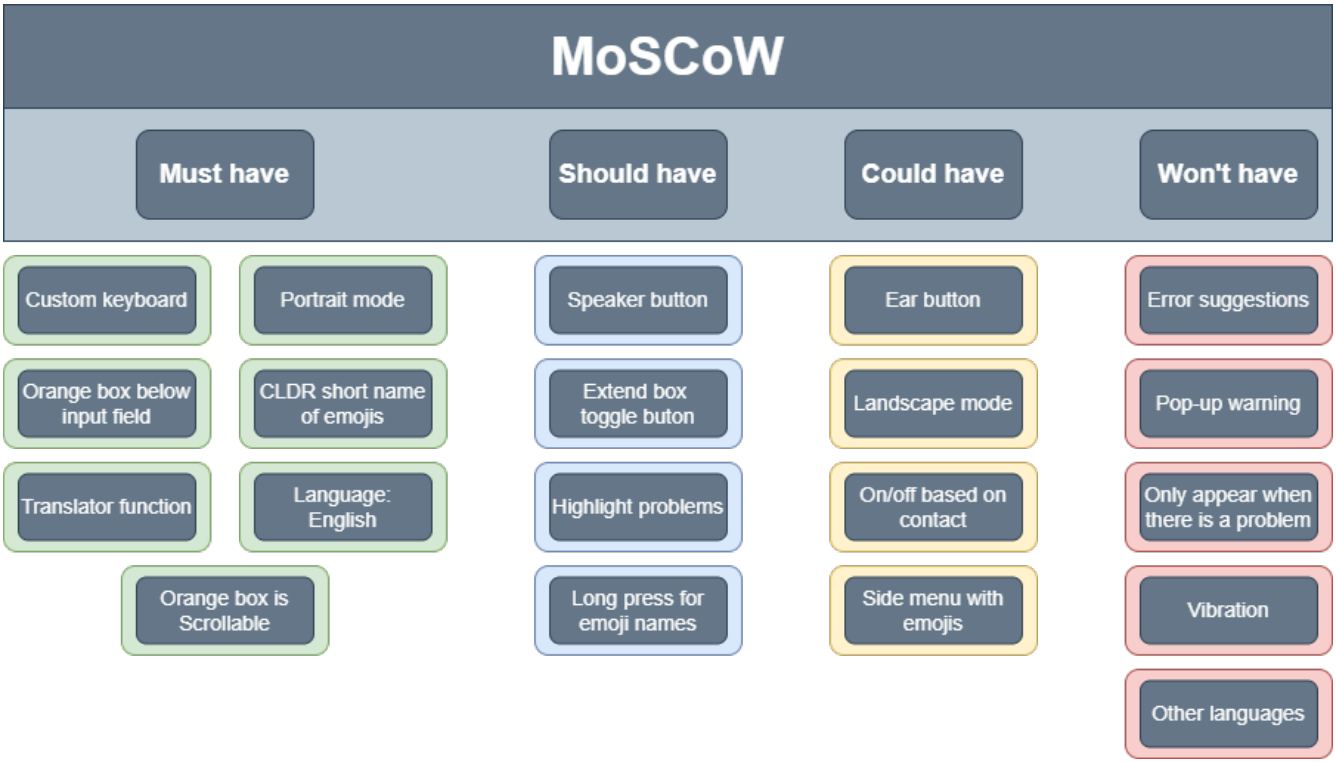


Figure 9: MoSCoW

10.1.1 Must have requirements

Custom keyboard extension:
One of the main requirements of this project is that the Echoboard has to be a custom keyboard extension that we will develop and is uniquely ours.

Orange box below input field:
This requirement describes that the orange box in the preview design has to be based on Griggio et al.s paper [18]. This requirement has been iterated and improved through section 10.2. The orange box will become the container for the screen reader preview and it can include the speaker button and extend toggle button described in the should have requirements.

Translator function:
The translator describes the whole function of the Echoboard, it is responsible for taking the input written by the user, both text and emojis, then rewriting it such that it matches what a text-to-speech system would vocalize and placing it into the orange preview box. This requirement is a must have requirement since it is the essential part of the product.

Portrait mode:
When users write messages on the phone it is usually done with the phone in portrait mode, therefore the group will first focus on ensuring the Echoboard works in this mode.

Descriptors of emoji: When a user writes emojis in the input field, this requirement describes the functionality of

translating the emoji into the emoji's descriptive name and placing that into the orange box. This is essential for the user to gain a preview of a screen reader's input.

Langauge: English
To limit the scope of the project, we decided to make the Echoboard with a focus on the English language, correctly translating into English based on the input in a screen reader, therefore this requirement has been placed as a must have requirement.

Orange box is scrollable:
This requirement describes that if the text in the orange box is longer than the box, it should be scrollable. If it is not it could cause issues and make the keyboard harder to read, therefore this requirement has been placed as a must have requirement.

10.1.2 Should have requirements

Speaker button:
This requirement was based on the results from the prototype testing in section *4 Prototype*. It was first explored in Carla's paper as a method for the user to hear how their message would be read aloud by a screen reader. This requirement has been marked as a should have requirement since it is not crucial for the product to function but it would add a lot of usability and "quality of life" to the product.

Expand box toggle button:
This requirement is based on the results from the prototype testing in section *4 Prototype*, the expand toggle button allows the user to make the orange box bigger allowing the product to become more usable and user-friendly, however, this is not a crucial requirement for the product and has therefore been placed as a should have requirement.

Highlight problems:
The group discussed that if the product could highlight problems for a screen reader in the user's input, that would help the user even further to create more screen reader "friendly" messages.

Long press for emoji names:
To help the user create "better" messages for screen readers using emojis, it would benefit the user to be able to read the names of the different emojis more easily, this requirement describes that if the user does a long press on an emoji, the name should pop-up. This is based on the results from the prototype testing seen in section *4 Prototype*.

10.1.3 Could have requirements

Ear button:
The "ear button" is a feature that allows the user to hide/show the orange box from the keyboard. This feature has been explored and is based on the results from the prototype testing seen in section *4 Prototype*.

Landscape mode:
Since the group has chosen to focus the keyboard to work in *Portrait mode*, as a must have requirement, the landscape mode has been placed as a could have requirement. It could be implemented but it is not crucial to the product but only adds to the "quality of life" aspect of the product and will only be implemented if there is enough time in the project.

On/off based on contact:
In the results of the prototype testing seen in section *4 Prototype*, the participants expressed a desire for the orange box to be automatically shown/hidden based on the contact that the user was typing. Say if the orange box was hidden by default but could be set to be shown when texting certain people, thereby only being shown when the user was messaging certain people. The group recognized this problem might be a great feature to add to the product, however, it was deemed to possibly being too difficult and time-consuming for the time period of this project. However it has been placed as a could have requirement and if time allows, the group will look at this requirement.

Side Menu with emojis:
The group discussed that a side menu with the emojis next to their description name could be just as useful as the *Long press for emoji names* function described prior. This would give the user a similar amount of insight into how a screen reader would interpret the different emojis, however, this has been placed as a could have requirement since it is not a key feature in the product.

10.1.4 Won't have requirements

Error suggestions:
This feature is an extension of the Highlight problems10.1.2 feature, the Error suggestions will be accessed when users click a highlighted problem, which will display a menu of suggested corrections for the error. The error suggestions were a feature discussed but it was placed in won't have due to it being too time-consuming within the project deadline.

Pop-up warning:
The pop-up for warnings was a functionality that would add to the usability of the product, however, this requirement was deemed to be too time-consuming. Therefore it was marked as a won't have requirement.

Only appear when there is a problem:
This feature describes that the orange box should only appear if the product could detect an error in the user's message regarding screen readers. This feature was placed in won't have because it would be too time-consuming within the project deadline.

Vibration:
The feature that the orange box should appear with vibration was discussed within the group, however, this feature will not be added to the final product and therefore has been placed as a won't have requirement.

Other languages:
The group considered making the Echoboard compatible with multiple languages, however, due to the time limit and scope of the project, the Echoboard will only be compatible with the English language, also described in the must have requirements. Therefore, other languages have been marked as a won't have requirement for this project.

10.2 Design Choices

Now that we have a clear picture of the products goal and functionality, we have to explore the various approaches for implementation, while considering both their possibilities and limitations. This exploration/evaluation is a crucial part that provides us with the most effective course of action, to realize our product. Throughout this section, we will evaluate options both for the UI (User Interface) and the backend.

10.2.1 Backend

In this case, the backend is the backbone of the product and will dictate to what extent we can achieve the functionalities, and thereby the product's effectiveness. The following part will encompass the considerations on the following points; "keyboard choice", "Input Data", and "Screen reader Interpretation".

Keyboard choice The essence of our product is to be an extension of an Android keyboard. To achieve this, we have evaluated different open-source keyboards so that the most suitable match can be found. The selection process consists of ranking the keyboards based on specific criteria, each criterion is assigned a weight to determine its importance. The criteria are:

- EI (Emoji Implementation) - Evaluates the presence of emoji support, the diversity of emojis, and if it supports skin-tone modifiers.
- Completeness - Evaluate the overall functionality and features the keyboard has to offer.
- Open Source - It has to be an open source keyboard.
- Design - Evaluates the visual appeal of the keyboard, and for familiarity and ease of use, it should resemble a standard keyboard.
- Modifiability - Evaluate how easily the keyboard can be modified.

- EoD (Ease of Development) - Assesses the difficulty to integrate the product as an extension of the keyboard.

The keyboards that were chosen to be explored are Custom Keyboard, AndroidCustomKeyboard[31], AnySoftKeyboard[32], FlorisBoard[33], OpenBoard[34]. A grading over the keyboards with criteria points can be seen in table 1.

Table 1: A requirement matrix over open-source keyboards

	Criteria and weights						
Keyboard Name	EI (weight 5)	Completeness (weight 2)	Open Source (weight 2)	Design (weight 3)	Modifiability (weight 3)	EoD (weight 4)	**Total Points**
Custom Keyboard	0	0	5	5	5	1	39
AndroidCustomKeyboard	2	4	5	3	4	5	66
Google SoftKeyboard	0	4	5	3	4	4	52
Anysoftkeyboard	5	5	5	5	2	2	69
Florisboard	5	3	5	5	2	3	69
OpenBoard	5	4	5	5	4	4	81

Due to emojis being a vital part of the product, all keyboards lacking emoji support are disregarded, and based on the total points, the OpenBoard keyboard poses as the best candidate to expand on.

Input Data

Another core feature of our product is to obtain the input data of the user which can then be modified to simulate how it would be interpreted by a screen reader. Due to uncertainty regarding permissions for accessing the data of a mobile device, we have discussed different approaches to implement a feature that will allow us to obtain the input data. Below are listed the following approaches, ordered from optimal to sub-optimal method of implementing the feature.

Accessibility permissions - is the most optimal solution, where data is directly grabbed from the input field of messaging applications.

Through the implementation of accessibility services, we can obtain the required data from the mobile device, and by restricting how and what data we obtain, we ensure that only data from input fields is accessed.

Keystroke event with input position - this solution captures each keystroke the user performs on the keyboard, and additionally takes into account the user moving the input position of the cursor, though there is an uncertainty if the user position in the input field can be obtained. In this context, user keystrokes signify actions that will affect the input data, like typing, deleting, cutting, and pasting. This approach will always register the input, therefore a safeguard needs to be developed, that only allows registering of input when the user is typing in an input field.

Macro - this solution mimics the user periodically going into the input field and copying the contents to be transcribed and then shown over in the orange box.

Custom input field - opposite of the macro solution, the user uses the orange box as an input field to write their messages, and then they are copied over into the input field of the messaging app. Since the users will be writing in our product, we can more easily gain access to the data needed.

Keystroke event - this solution derives from the ”Keystroke event with input position” solution, with the distinction of excluding the cursor position. This solution will only work correctly if the user does not move their cursor at all, otherwise, any movement will cause an incomprehensible outcome.

App extension - this solution suggest that we develop our product as an extension of an open-source messaging application, such as Telegram. Gaining access to the input field data should be straight forward, although there's a downside, being that our product can only be used for the one application we develop it for.

Screen reader Interpretation

The remaining key feature involves processing the obtained input data, to generate an output that simulates how a screen reader would interpret it. Below we have different ways this feature can be implemented.

Direct approach - involves using screen reader functions that give a textual output. Because the output comes directly from the screen reader, it's highly accurate, since it exactly reflects how the screen reader interprets it. This approach is exceptionally light on computational resources and requires no maintenance. However, it's important to mention that screen readers may not provide a textual output, since the primary function of a screen reader is to provide audible output.

Regular Expressions - allows us to identify specific patterns and modify the input accordingly. This approach will take more time to implement since research needs to be done as regarding to how a screen reader interprets content. The accuracy of this solution reflects the quality of research and implementation and requires maintenance to account for changes in a screen reader's content interpretation. This approach is relatively light on computational resources since it is tailored specifically to our problem.

Dual Synthesis - involves utilizing services for TTS (text-to-speech) and STT (speech-to-text) by transforming the input into audio, and then back into text. The implementation should be relatively simple, though it requires service compatibility. The accuracy will be highly dependent on the quality of the services, and how well they interact. Maintenance will not be necessary unless there are fundamental changes in how a service operates. This approach will be more taxing on computational resources, as it involves dual synthesis.

10.2.2 User Interface

We propose a design based on the "PREVIEW" prototype by Griggio et al. [18] which we explored in section 1 since this paper has taken inspiration from it and will be a continuation of that project.

As already mentioned in section 1 *Related Work* the PREVIEW design shows a text box that lets the user preview how a screen reader would read the whole message. This text box is orange and the box has the same placement on screen, which can be seen in our concept designs in Figure 13 in *Appendix A - Design*.

Two design options have been considered, the difference being the placement of the orange text box. The first design is the original PREVIEW design from Griggio et al. [18], which places the text box on top of the input field, this placement inspiration can be seen in Figure 13a.

The second design consideration is based on the research found in the paper by Jiang et.al. [35], this design option can be seen in Figure 13b. Jiang et.al. [35] have researched eye and finger movement strategies in mobile typing through eye tracking and have found that when typing on the keyboard the eye movement is on or straight above the keyboard, or at the top of the screen. The difference in the concept design in Figure 13a is that the orange text box is directly above the keyboard. This placement of the box could prove to be more visible and eye-catching for the user according to Jiang et.al. [35], which is beneficial for catching the attention of the writers even further.

However, if this design is created, the space in the orange box for the preview text would be very limited if the input text is long with multiple Emojis. We have explored a variety of design options that could solve this issue, one of them making the orange text box scrollable, this can be seen in Figure 10. This is an option however if the text is too long the scroll option might be inconvenient for the user.

Figure 10: A.) Orange Text box with scroll function concept.

Another option we explored was making the preview a speech bubble that folds out when pressed. This gives the user the option to hide the orange preview box if they don't want to see it. This concept can be seen in Figure 11, this design option was explored but was deemed to be taking too much space on the screen and the function should be integrated into the keyboard.

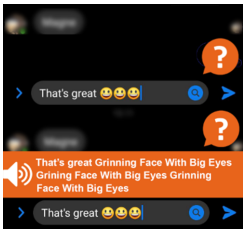


Figure 11: A.) Speech bubble concept.

Based on the speech bubble concept, we considered the possibility of creating a different design where the preview is showcased on a separate page of the keyboard toggled by a button with an eye icon. This design possibility could solve the issue of the text being too long for the orange preview box. If the preview got too long to be on the separate page, a scroll option could be discussed and implemented. This design can be seen in Figure 14 which can be seen in *Appendix C*

However, a problem with a design where the user has to go to a different page on the keyboard is that they are not able to see the preview update while writing as in the other design with the orange box placed above the keyboard. A live update of the text is a very convenient feature for the users because it shows their potential ”mistakes” which immediately change their context and intention with their message.

10.3 Test methods and details

We did two iterations of a user-centered design process; a high-fidelity prototype test, and a usability test. We had certain roles and used a think-aloud approach for both tests.

10.3.1 Roles

During the user tests the group delegated these roles between them: Test monitor, observer, and data logger. The test monitor’s role is to be the one ”in contact” with the test subject and give them their tasks during the test, it is crucial that this role only guides and helps the test subject as little as possible and allows the test subject to ”think on their own” for the most optimal data. The data logger’s role is to log down how the subject approached the problems, if and how they solved them, and how they interacted with the test. The observer’s role is to primarily focus on the test subject’s behavior, their physical reactions, and thought process throughout the test [23].

10.3.2 Think aloud

Throughout both tests, we asked the participants to ”think aloud”, which implies that they were asked to share their mental processes when interacting with Echoboard. The participants should say all their thoughts out loud that come to mind. By using the think-aloud protocol [36], we were able to get subjective thoughts and data from the participants.

10.3.3 High fidelity prototype

For the first iteration, we created some high-fidelity prototypes in Figma, the purpose was to evaluate three different design options. We chose a high-fidelity approach leveraging our research findings as a solid starting point for the design from the early stages of the project. By using previous experiences from researchers, we avoided repeating design processes that have already been made. With Figma, we were able to make an interactive prototype and it would closely resemble how they would interact with the finished keyboard.

10.3.4 Usability testing

After we had a working implementation of Echoboard, we evaluated the final design by making a usability test. Before the usability test, we established the tasks the participants should perform based on certain goals we had established, and we installed Echoboard on a phone.

Appendix B - Implementation details and challenges

10.4 Implementation

This section contains how the Echoboard has been implemented and which methods and tool were used to accomplish this. Firstly, in section 10.2 the open source keyboard chosen to be the foundation of the Echoboard is described. This is followed by section 10.5, were Android studio and the use of it for the Echoboard will be discussed. Then in section 10.7 some of the challenges we encountered throughout the project will be described and the solutions will be explained and reasoned.

10.5 Android studio

Android Studio was chosen as the tool to implementing the Echoboard due to the group choosing to focus on creating a keyboard for Android phones, we decided this because we have prior experience with Android Studio this seemed like the obvious choice. Android Studio is compatible with the Open Board keyboard which was written in the programming language Java and Kotlin, therefore the group implemented the Echoboard in Java and Kotlin.

10.6 Class diagram

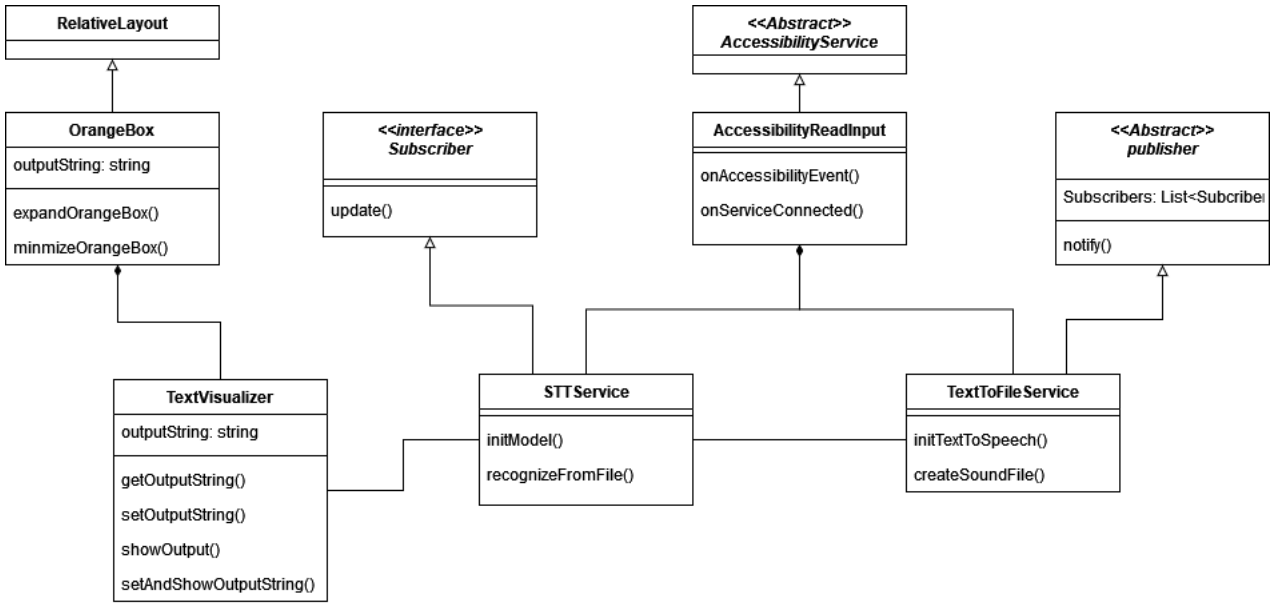


Figure 12: Class Diagram

10.7 Challenges and solutions

10.7.1 Regular expressions, text-to-speech, and speech-to-text

In the initial stages of the project, our aim was to develop custom Regular Expressions to approximate the output of Android's text-to-speech system for users with visual impairments. We opted for this solution primarily for its computational efficiency, which promised faster performance and compatibility across a wider range of devices. Additionally, we believed it would afford us greater flexibility for customization. However, as we progressed with this approach, we encountered a significant challenge: numerous symbols were either incorrectly vocalized or not vocalized at all, depending on the context of the message.

Consequently, we shifted our focus to a thorough examination of Android's text-to-speech functionality, hoping to identify an intermediate step in its processing pipeline that would generate a string closely resembling the final vocalized output. Regrettably, we discovered that either such a step did not exist or it was inaccessible from outside the Android text-to-speech system.

Our final solution involved using the text-to-speech system in Android to generate an audio rendition of the text

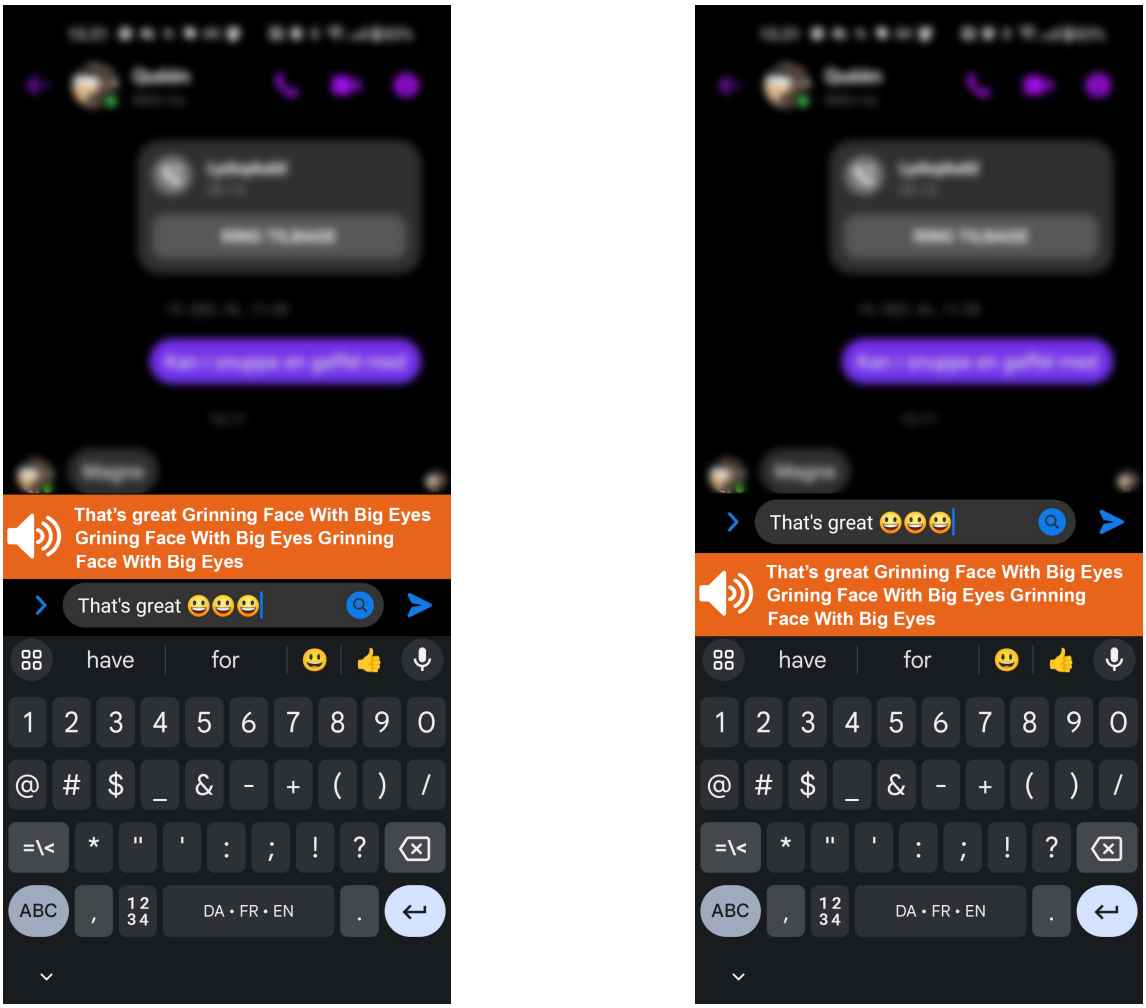
entered into the field. Subsequently, this audio file is then used as input to a speech-to-text model, which transcribes the spoken content. The transcribed text is then displayed in the designated orange box.

10.7.2 Text in the orange box

After creating the system to generate a string approximating the output of the text-to-speech system, our next task was to implement a mechanism to display it within the orange box. However, during the development of this system, we encountered a challenge: our accessibility service, AccessibilityReadInput, which manages both the text-to-speech and speech-to-text services, is instantiated within the ”Service lifecycle”, because all accessibility services extend the ”Service” abstract class, and therefore it is controlled by the ”Service lifecycle” [37]. Conversely, the orange box, responsible for visualizing the text, is created within the ”Activity lifecycle”[38]. This mismatch rendered it impossible to access the orange box from the AccessibilityReadInput script.

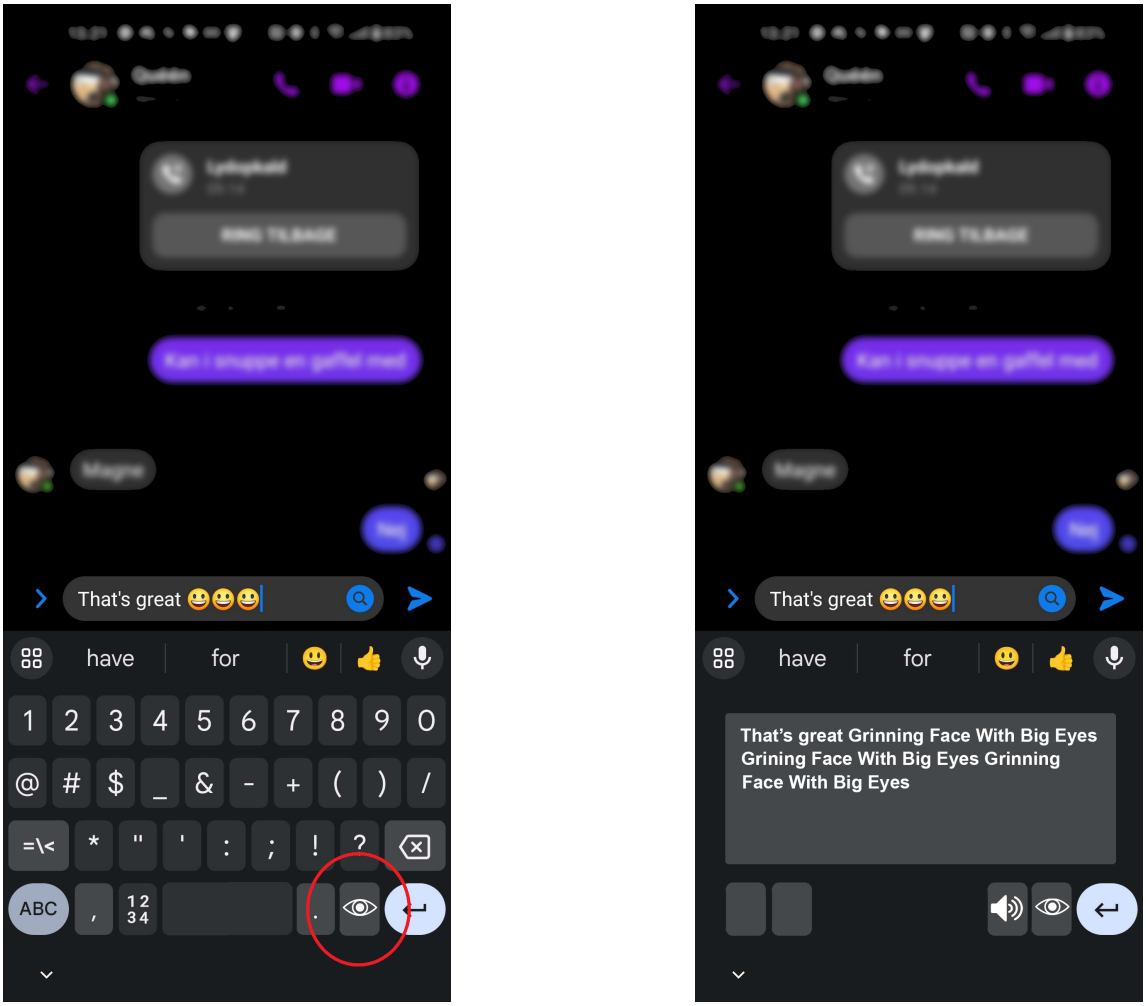
To address this issue, we adopted a solution whereby we made the variables and methods in the text visualizer, responsible for displaying text in the orange box, static. This adjustment ensured that the variables and methods became accessible in any context, enabling seamless integration with the AccessibilityReadInput script.

Appendix C - Design



(a) Top placement of preview. **(b)** Bottom placement of preview.

Figure 13: First Design consideration with orange preview box



(a) Eye icon as a page toggle button.

(b) Concept of preview page on keyboard

Figure 14: Second Design consideration with the preview page.