Eeva-Jonna Panula

Towards More Accessible Android Applications

An Actionable Accessibility Checklist for Android Developers

School of Technology and Innovations
Master's Thesis
Master's Programme in Technical Communication

Vaasa 2024

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen akateeminen yksikkö
Tekijä: Eeva-Jonna Panula
Tutkielman nimi: Towards More Accessible Android Applications: An Actionable Accessibility Checklist for Android Developers
Tutkinto: Kauppatieteiden maisteri
Oppiaine: Teknisen viestinnän maisteriohjelma
Työn ohjaaja: Juho-Pekka Mäkipää
Valmistumisvuosi: 2024 **Sivumäärä:** 78

TIIVISTELMÄ:

Arvioilta 15 % maailman väestöstä on jonkinlainen vamma tai toimintarajoite. Jotta he pystyisivät käyttämään mobiilisovelluksia, täytyy sovellukset rakentaa saavutettavasti. Tällä hetkellä iso osa mobiilisovelluksista ei ole saavutettavia, ja yksi syy siihen on sovellusten kehittäjien osaamisen ja tiedon puute. Käyttöjärjestelmien välillä on eroja, ja kahdesta suosituimmasta mobiilikäyttöjärjestelmästä Android-sovelluksissa on selkeästi enemmän saavutettavuuspuutteita.

Tämän tutkielman tavoitteena on ymmärtää, miten Android-kehittäjät voivat kehittää sovelluksia, jotka ovat saavutettavampia kuin nykyiset sovellukset. Tavoitteen saavuttamiseksi tutkimuksessa käytettiin suunnittelutieteellistä tutkimusmenetelmää, ja sen prosessimallia. Työn tuloksena syntyi tarkistuslista Android-sovelluksen testauslista, jonka avulla sovelluskehittäjä voi testata sovelluksensa saavutettavuuden eri aspekteja.

Tarkistuslistan kehityksessä käytettiin hyväksi aikaisempaa tutkimusta, ja sen avulla luotiin listan ensimmäinen versio. Tämän jälkeen listan käytettävyyttä ja hyödyllisyyttä arvioitiin viiden Android-kehittäjän haastatteluilla. Haastattelut analysoitiin teemoittelun avulla. Analyysin pohjalta kehitettiin prototyyppi, sivusto, jolla tarkistuslista sekä sitä tukeva ohjeistusmateriaali on saatavilla. Tämän prototyypin hyödyllisyyttä ja toimivuutta testattiin kyselylomakkeen avulla, jossa Android-kehittäjiltä kysyttiin listan hyödyllisyydestä, ja siitä, miten sitä voisi parantaa. Tämäkin aineisto analysoitiin teemoittelun avulla.

Näiden kehitysehdotusten perusteella luotiin lopullinen tarkistuslista, joka sisältää kuusi teemaa, joiden alla on yhdestä kuuteen tarkistuskohtaa. Nämä teemat liittyvät automatisoituihin työkaluihin, visuaalisiin elementteihin (kuten kuvat), erilaisiin tapoihin navigoida, näytön suuntaan, käyttöliittymän rakenteeseen sekä näytön kokoon ja sen suurentamiseen. Lisäksi listalla on neljä teemaa, joihin kehittäjä ei välttämättä pysty vaikuttamaan, mutta jotka kannattaa testata – etenkin jos niihin pystyy vaikuttamaan. Nämä teemat liittyvät osaltaan visuaalisiin elementteihin (kuten kuvat), värien käyttöön, näytön suuntaan sekä audio- ja videosisältöön. Tarkistuslistan lisäksi sivusto, jolla tarkistuslista esitellään, sisältää ohjeistuksen tarkistusten tekemiseen sekä lisämateriaalia saavutettavuudesta oppimiseen.

AVAINSANAT: saavutettavuus, Android, ohjelmistokehitys, mobiilisovellukset, ohjelmistokehittäjät

UNIVERSITY OF VAASA
School of Technology and Innovations
Author: Eeva-Jonna Panula
Title of the thesis: Towards More Accessible Android Applications : An Actionable Accessibility Checklist for Android Developers
Degree: Master of Science in Economics and Business Administration
Discipline: Master's Programme in Technical Communications
Supervisor: Juho-Pekka Mäkipää
Year: 2024 **Pages:** 78

ABSTRACT :

Approximately 15% of the world's population has a disability or impairment. They need mobile applications to be developed with accessibility in mind to use them effectively. However, most mobile applications are not accessible because developers often lack the knowledge or skills to create them. Additionally, there are differences between operating systems, and Android apps have more accessibility issues compared to iOS.

The goal of this master's thesis is to understand how Android developers could develop more accessible apps than currently available. To reach this goal, the method of the study is design science research method. The outcome of this thesis is an artifact, a checklist for Android developers to use when developing an app to test app's accessibility.

For the development of the checklist prior research was used, and with its help, the first version of the checklist was created. The initial version was evaluated with interviews with five Android developers. Interview answers were analyzed with thematization. The results of the analysis contributed to a prototype, which was a website with the checklist and accompanying learning material and instructions. The usefulness and functionality of the prototype was tested with a questionnaire. The questions in the questionnaire were related to the usefulness of the checklist and how to improve the checklist. Thematization was used for the answers of the questionnaire as well.

The analyzed answers were used for improving the prototype and constructing the final checklist. It consists of six themes, which all have from one to six checks. The themes were related to automated tools, visual elements like images, alternative ways of navigation, screen orientation, structure of the user interface, and screen magnification. In addition to these themes, the list has checks that might be out of the developer's control, but worth checking. These checks are categorized into four themes: Visual elements like images, use of color, screen orientation, and audio- and video content. In addition to checks, the website with the checklist has instructions for conducting the checks, and additional material for learning about accessibility.

KEYWORDS: accessibility, Android, software development, mobile applications, software developers

Contents

Figures

Tables

1 Introduction

Mobile phones and apps are here to stay. On average, people use smartphones a lot: For example, an adult uses a smartphone on average for 3 hours and 15 minutes a day, and 79 % of adults carry their mobile phones for about 22 hours per day (Lee et al., 2022, p. 1). About 15 % of the world population is estimated to have a disability or an impairment (Di Gregorio et al., 2022, p. 145). People with disabilities or impairments often use assistive technologies such as screen readers, screen magnification, voice access, or switch devices. For applications to work for them, developers must build the apps with accessibility in mind.

From an assistive technology and accessibility settings perspective, there are differences between platforms and usage. For example, screen reader users tend to prefer iPhones or iPads as their primary mobile devices. For example, in WebAIM's screen reader survey in 2021, 71.9% answered so. VoiceOver, the screen reader Apple provides, was the most commonly used screen reader on mobile devices for 71.5% of the respondents. Talkback, the default screen reader for Android, was used by 29.1% of the respondents. (WebAIM, 2021) Two years later, the numbers are slowly shifting: in 2023, the percentage of VoiceOver users was 70.6%, and TalkBack users 34.7%. (WebAIM, 2023) This finding is interesting because the market share of Android phones is generally much higher: 83.8% of the global smartphone market (Lee et al., 2022, p. 1). When digging deeper into WebAIM's survey, the biggest reasons for the primary screen reader are the existing comfort, expertise, and features available, so it might suggest that Apple provides a better experience and more features for users (WebAIM, 2021). It is also important to note that participation in the surveys is mainly from the US, so the ratio between platforms and screen readers worldwide might be different.

Even though the companies behind mobile operating systems (like Google for Android) are improving built-in accessibility, many things depend on the developer to implement accessible solutions (Yan & Ramachandran, 2019, p. 4). It is not going well; many studies

report that most mobile apps lack accessibility. (e.g. Acosta-Vargas, Zarate-Estrella, et al., 2021; Chen et al., 2022; Da Silva et al., 2020; Di Gregorio et al., 2022; Ross et al., 2020; Yan & Ramachandran, 2019) Di Gregorio et al. (2022) surveyed developers to understand better why they do not implement accessibility guidelines. They found that the main driver for implementing accessibility guidelines was a personal willingness to develop accessible apps. Leite et al. (2021) came to the same conclusion when surveying Brazilian professionals. Other reasons for developing apps with accessibility in mind included widening the potential user base and the app targeting users with disabilities (Di Gregorio et al., 2022). Furthermore, Di Gregorio et al. (2022) concluded that mobile developers usually do not implement accessibility guidelines overall, even though they might have a chance to do so. (Di Gregorio et al., 2022, pp. 145–146)

In their research, Vendome et al. (2019, pp. 49–50) identified six lessons regarding supporting mobile developers. Each of these lessons contained actionable results. The first of these lessons was that developers leverage existing tools to support accessibility. When developers are made aware of the issues, they tend to learn more about them and support accessibility. The second lesson they concluded was that mobile developers lack experience and background in different accessibility features and tools, meaning they might think something is inaccessible because it is not working as they think it should – and not considering how people with disabilities interact with the features. The third lesson from their research was that there is limited support for automated accessibility testing for different disabilities. They observed that automated testing often concentrated on a limited scope of disabilities, and accessibility features often centered around visually impaired users. They also observed that even with the limited scope, developers often have difficulty implementing support for these accessibility features. Thus, the fourth lesson their research provided was a lack of support for developers automating the implementation of accessibility features. Vendome et al. (2019, p. 50) observed that the apps use accessibility APIs for more than just accessibility, and that was the fifth lesson. Finally, the sixth lesson in their research was that accessibility in mobile apps is not only about screen readers. Developers answering their study were

also concerned about other aspects; examples include readability for low-vision users and color transformations for color-blind users. (Vendome et al., 2019, pp. 49–50)

As mentioned above, these lessons each led to actionable suggestions. Two of them were particularly interesting for this research. First, semi-automated solutions for implementing and testing accessibility features could help mobile developers build more accessible applications than they are currently building. Second, as seen in example (1), Vendome et al. (2019, pp. 49–50) suggest that professionals should put more effort into a broader view of accessibility.

> (1) Practitioners, researchers, and educators should devote more effort to identify and promote best practices for designing and implementing accessible apps. (Vendome et al., 2019, p. 50)

This thesis aims to provide one solution for these two challenges: an easy-to-use solution that considers broader aspects of accessibility than just screen reader accessibility. I will discuss the goal and method of the research more extensively in the following subsection.

1.1 The Goal and Method of the Research

The goal of this master's thesis is to understand how Android developers could develop more accessible apps. The outcome of this thesis is an artifact, a set of instructions Android developers can use when developing an app.

The purpose is to answer the following research questions:

1. How can Android developers develop more accessible Android apps, than currently available?
2. What prior knowledge do Android developers have about creating accessible Android apps?

3. Which components define an artifact that helps Android developers develop more accessible Android apps?

Design Science Research (DSR) is used to achieve the goal and answer the research question above. DSR is a paradigm aiming to enhance knowledge by creating artifacts that solve problems and improve the setting in which they are instantiated (Vom Brocke, Hevner, et al., 2020, p. 1). This work utilizes the Design Science Research (DSRM) process method defined by Peffers et al. (2007).

I conduct the development of the artifact in three phases. The first phase includes a literature review, contributing to the initial guidelines set. The second phase will include evaluating the initial set of guidelines by interviewing five Android developers and then improving the checklist based on the answers. The third phase contains evaluating the improved checklist with the help of an online questionnaire for Android developers. The answers from the questionnaire will be used to construct the final version of the checklist.

For the interviews, I recruited five Android developers with different backgrounds and different amounts of experience in Android development and accessibility. For the questionnaire, I used Google Forms as a survey tool. I shared the link to the questionnaire for Android developers on social media platforms like LinkedIn and Slack communities. I analyzed the material collected from the interviews and the questionnaire using thematic analysis. These themes guided further development of the artifact, providing insights on what developers see as important as part of an artifact for accessibility guidelines for Android developers.

Hevner (2007) introduced a three cycle view on design science research activities. These three cycles (Relevance Cycle, Rigor Cycle, and Design Cycle) are visible in this work, too: the first phase mentioned above (the literature review) contributes to both the Relevance and the Rigor Cycles. The other two contribute to the Design cycle. The research questions mentioned above are aligned with these cycles as Thuan et al. (2019,

pp. 348–349) suggest: The research questions RQ2 and RQ3 are tied to the Rigor Cycle and Relevance Cycle, respectively. The first phase aims to answer them. The RQ1 is tied to the second and third phases, and they aim to answer it.

The evaluation framework used in this thesis is the build-evaluate pattern, which is typical for design science research as defined in Sonnenberg and vom Brocke (2012). As this research is part of a master's thesis work, that constraints the scope of evaluating the artifact. In practice, this means that the research aims to evaluate the proof of concept of the artifact. Testing the artifact in use (as defined as evaluation activity 4 in Sonnenberg and vom Brocke (2012)) is out of the scope of this thesis. It leaves the proof of validity to be evaluated in future research.

1.2 Structure of the Thesis

Chapter two discusses prior literature and research about the theme of the thesis. The main themes of that chapter are mobile accessibility guidelines and developers' accessibility knowledge. The chapter aims to provide a theoretical basis for the artifact through prior research and knowledge on the topic.

In Chapter three, I describe the process of the artifact development. The goal of the chapter is to provide a good look into the process and methods of artifact development.

Chapter four describes the results of the research. The first subsection examines the literature review and the initial checklist derived from it. The second subsection discusses the interviews and how the checklist was developed further based on them. The third subsection discusses the questionnaire and the final checklist based on the analysis of questionnaire answers. The chapter aims to present the artifact development.

The evaluation of the artifact is described in Chapter five. The first sections discuss the artifact evaluation activities, from evaluating the problem statement to evaluating the design specification and, finally, evaluating the checklist. The final section of the chapter concludes the evaluation.

Chapter six concludes the thesis, drawing together the findings. It also discusses the research's contributions to the practitioners and the knowledge base, limitations of the study, and suggestions for future research.

2 Accessibility and Android Developers

As the thesis aims to understand how Android developers could develop more accessible apps, the relevant topics to look at are accessibility and Android developers' relationship to accessibility.

In this chapter, I will first examine accessibility as a general concept. Then, I will zoom in on mobile and Android accessibility, deriving accessibility guidelines from prior research. Finally, I will discuss Android developers and their relationship to accessibility in the form of the challenges they face in developing more accessible applications and some solutions that prior research has suggested for these challenges.

2.1 Accessibility

Persson et al. (2015) suggest a definition of accessibility as

> (2) The extent to which products, systems, services, environments and facilities are able to be used by a population with the widest range of characteristics and capabilities (e.g. physical, cognitive, financial, social and cultural, etc.), to achieve a specified goal in a specified context. (Persson et al., 2015, p. 524)

They combine this definition in example (2) from multiple approaches that relate to accessibility – barrier-free design, design for all, universal design, accessible design, universal access, and cooperative design. They also use accessibility legislation and standards as the basis for their definition.

Even though it does not explicitly mention accessibility, the Convention on the Rights of Persons with Disabilities (United Nations, 2006) obligates universal access for all people

and calls for reasonable accommodations for people with disabilities to ensure equal access. (Persson et al., 2015, p. 514)

To take the definition of accessibility closer to this thesis' topic, Madeira et al. (2021, p. 557) define, that

> (3) A mobile application is accessible when any user, regardless of their functional diversity, can use it on their mobile device to their satisfaction. (Madeira et al., 2021, p. 557)

Alshayban et al. (2020)A define mobile accessibility similarly. This idea explained in example (3) of the app's usage to user's satisfaction can also be accomplished via assistive technology – as Desmond et al. (2018, p. 438) write, assistive technology is "an interface between the person and the life that they would like to lead".

In many instances, such as legislation, the question of "Is this app or website accessible?" is determined based on preselected criteria, often Web Content Accessibility Guidelines (WCAG) (See, for example, Khasawneh et al., 2023; Laamanen et al., 2022; Rygg et al., 2016). In the next section, I discuss guidelines related to mobile app accessibility.

2.2 Mobile Accessibility Guidelines

This section looks at existing guidelines about mobile and Android accessibility. I will start by looking at Web Content Accessibility Guidelines (WCAG) and how mobile devices and accessibility should be considered in the context of WCAG. After that, I will discuss the instructions Android platform documentation gives to developers regarding accessibility. Finally, I will examine prior research on guidelines for accessibility for Android applications. All the guidelines, that I've selected to use as basis for the initial checklist, are listed in Appendix 1 together with their codes.

2.2.1 Web Content Accessibility Guidelines (WCAG) and Mobile Accessibility

Web Content Accessibility Guidelines (WCAG) are an established set of criteria for evaluating digital accessibility. They are the basis of legislation in many countries. For example, in the USA, Section 508 of the Rehabilitation Act of 1973, as amended in 1998, which prohibits discrimination based on disability in federal programs or programs that receive funding from federal entities, defines digital accessibility through WCAG 2.0, level AA (Rehabilitation Act of 1973, 2017). In addition, EN 301 549's version 2.1.2, which is the standard used for defining what is accessible in the digital context in the European Union's Web Accessibility Directive and European Accessibility Act (European Parliament 2016; European Parliament, 2019).

WCAG's newest published version is 2.2. WCAG consists of four principles (perceivable, operable, understandable, and robust), which all divide into guidelines. In total, there are 13 guidelines, which all have different amounts of success criteria in each guideline, 87 in total. These success criteria can be of three levels: A, AA, and AAA, in which the A is the lowest, and the AAA is the highest. (Campbell et al., 2023; Kirkpatrick et al., 2023) Guidelines in WCAG were initially developed for the web (hence the name), but they do extend to mobile apps (be they native or web apps) as well. W3C, the community behind the WCAG, has developed a document to map how the success criteria in WCAG apply to mobile. (Patch et al., 2015) While the accessibility coverage for mobile apps has been enhanced from WCAG version 2.0, many things are still not covered (Alajarmeh, 2022).

The guidance on how WCAG maps to mobile is divided into four categories, following the four principles of WCAG. For the first one, perceivable, the document mentions three considerations: small screen size, zooming and magnification, and color contrast. For the second principle, operable, the considerations are keyboard control for touch devices, touch target size and spacing, touchscreen gestures, device manipulation gestures, and placing buttons where they can be easily accessed. For understandable category, the considerations are device orientation, consistent layout, positioning important content

before page scroll, grouping operable elements performing the same action, clearly indicating actionable items, and providing instructions for custom gestures. Finally, for the fourth principle, robust, the considerations are setting the virtual keyboard to the type that the data entry requires, providing an easy method for data entry, and supporting the characteristic properties of the platform. (Patch et al., 2015) Many of these considerations are based on, or related to, the fact that the device screen is smaller than, e.g., a computer screen. For example, zooming and/or magnification are more likely to happen on smaller screen sizes, and the important content placement requires more thought because the real estate for elements is smaller than on larger screen sizes.

Furthermore, some of these considerations are more design-related, and some are up to developers to take care of. As the thesis aims to create an artifact for developers to use, I will concentrate on the technical considerations, not design-related ones. This constraint leaves me with the following items: the small screen size (WCAG-1), zooming and magnification (WCAG-2), keyboard control (WCAG-3), touch target (WCAG-4), touchscreen and device manipulation gestures (WCAG-5), orientation of the device (WCAG-6), grouping operable elements that perform the same action (WCAG-7), setting the virtual keyboard to the type of data entry (WCAG-8), providing easy method for data entry (WCAG-9), and supporting the characteristic properties of the platform (WCAG-10). All of these are considered when developing the initial checklist for the artifact.

These considerations are closely related to the legislation requirements, so they are a good starting point for advancing the checklist. However, WCAG serves only as a minimum requirement for accessibility, and there are other considerations on accessibility as well. Next, let us discuss what the Android platform itself considers regarding accessibility.

2.2.2 Android Accessibility Principles

The official documentation for developing Android applications has a section about accessibility. In that documentation, they provide a list of principles for improving Android app accessibility. The topics covered are labeling elements (ANDROID-1), adding accessibility actions (ANDROID-2), extending system widgets (ANDROID-3), and making media content more accessible (ANDROID-4). It also includes a principle about using cues other than color, but I've left it out as this is not always up to the developer to change.

Labeling elements is about adding text alternatives and roles for different elements on the UI. It consists of, for example, adding a label for editable text fields, unique identifiers for items in a list, content descriptions for images and graphics, grouping items for easier readability and adding a custom label if necessary, annotating headings, and marking decorative elements so that assistive technology knows to ignore them. (Android Developers, 2022) From different considerations about how WCAG maps to mobile devices in Patch et al. (2015), the principles mentioned in labeling elements map to supporting the characteristic properties of the platform (WCAG-10) as, e.g., annotating headings, marking decorative elements, and adding content descriptions are about enabling the accessibility services to work properly. Also, it partly maps to grouping operable items (WCAG-7).

The next category is adding accessibility actions. Accessibility actions are actions exposed to accessibility services, providing an alternative for, e.g., gesture-based actions such as swiping. It is essential to make the actions available by providing an easy-to-understand label. (Android Developers, 2022) This principle also maps to the considerations in Patch et al. (2015): gesture-related considerations (WCAG-5) and partly grouping operable elements that perform the same action (WCAG-7).

The next category, extending system widgets, is about robustness and using the system-provided widgets as often as possible. However, sometimes, there may be a need to build a custom widget. In these cases, these custom widgets should extend the system widgets. This way, it is more likely that the new custom widget supports accessibility services provided by the Android platform – and thus conforms with the consideration mentioned in 2.1.1 about supporting the characteristic properties of the platform (WCAG-10). (Android Developers, 2022; Patch et al., 2015)

The final item on the list of categories is about making media content accessible. It reminds developers to add and enable controls, such as pausing and stopping the media, changing the volume, and toggling captions. Furthermore, it reminds developers to provide content in an alternative format if it provides information vital for completing a workflow. (Android Developers, 2022)

These categories provide an actionable list of actions for developers to improve the accessibility of Android applications. Some of them extend or make the considerations listed in Patch et al. (2015) more concrete. To complete my review of mobile accessibility guidelines, I will look at prior research on the topic in the following subchapter.

2.2.3 Mobile and Android Accessibility Guidelines from Prior Research

Díaz-Bossini and Moreno (2014) constructed guidelines for developing mobile interfaces for older people. One of the lists of guidelines they used was a filtered version of the Senior Friendly Usability Guidelines from Zaphiris et al. (2005). They filtered out the guidelines purely focused on the web and ended up with six categories that fit the mobile context. Of those six categories, three are suitable for my research, as they are under the control of the developer: Target Design (DIAZ-1), Use of Graphics (DIAZ-2), and User Cognitive Design (DIAZ-3). (Díaz-Bossini & Moreno, 2014, p. 61)

The guideline target design has instructions for providing larger targets and clear confirmation of target capture. This guideline builds on Patch et al.'s (2015) touch target guideline (WCAG-4). Use of graphics instructs to use only relevant graphics for the content and avoid animations. Images should have text alternatives, and icons should be meaningful. This guideline is related to guidelines WCAG-10 and ANDROID-1 about supporting platform characteristics and labeling elements (Android Developers, 2022; Patch et al., 2015). From the user cognitive design guideline's perspective, there should be enough time to read the information, and the user interface should support recognition rather than recall. (Zaphiris et al., 2005, p. 1899)

Ballantyne et al. (2018) extensively analyzed existing accessibility guidelines from WCAG and prior research. They divided them into 11 categories. These categories are text (BAL-1), audio (BAL-2), video (BAL-3), UI elements (BAL-4), user control (BAL-5), flexibility and efficiency (BAL-6), recognition rather than recall (BAL-7), gestures (BAL-8), system visibility (BAL-9), error prevention (BAL-10), and tangible interaction (BAL-12).

The text category contains guidelines related to the display of text and text alternatives. This guideline, too, builds on the previous guidelines about supporting platform characteristics and labeling elements, like WCAG-10, ANDROID-1, and DIAZ-1. Audio-category contains guidelines that instruct apps to allow users to control and access audio and alternative media for audio content. The video category, on the other hand, includes similar guidelines for video content. Together with ANDROID-4, these form a basis for more accessible multimedia. UI elements include guidelines for different user interface elements on the screen, which are clearly labeled, colored, and positioned on the screen. The app should be easy to navigate, find content, and understand where they are within the app. The user control category extends from the previous category. It is about having enough time to read and use content and ensuring the app respects the user's device settings, such as accessibility settings. This guideline relates to guidelines on supporting characteristics of the platform and labeling elements, similar to WCAG-10 and ANDROID-1, as well as adding accessibility actions, so ANDROID-2. The next category, flexibility and

efficiency, has guidelines for data entry and assessing that data can be inserted and accessed in multiple ways. It also touches the device orientation and keyboard accessibility and is related to WCAG-3, WCAG-6, WCAG-8, and WCAG-9. Recognition rather than recall guideline, on the other hand, guides the provision of necessary information to complete tasks inside the same screen, so it does not require recalling to be able to complete them, and builds on user cognitive design guidelines, such as DIAZ-2. The gestures category reminds developers to have alternative ways of interaction for actions initiated by user gestures. It relates to the WCAG-5 guideline for touchscreen and device manipulation gestures. System visibility requires the app to appear and operate in predictable ways and notify users in a way they can perceive. Furthermore, error prevention, the next category, has guidelines about helping users avoid and correct mistakes when they input data. Finally, the last category, tangible interaction, requires apps to work with tangible devices, such as external keyboards, and relates to WCAG-3. (Android Developers, 2022; Ballantyne et al., 2018, p. 309; Díaz-Bossini & Moreno, 2014; Patch et al., 2015)

Zaina et al. (2022) developed guidelines that relate to nine user interface design patterns. They collected their material from researching professional forums and blogs about developers' difficulties when developing applications with accessibility in mind. They chose the following mobile user interface patterns: Hamburger menu, tab navigation, icons, input, data tables, list and pagination, select and dropdown, sliders, and carousel. (Zaina et al., 2022) While most of these user interface patterns are not usable to this research as is, they provide insights into accessible development. Two of them are used as part of the checklist: Icons (ZAI-1) and inputs (ZAI-2).

Icons can create different accessibility barriers, and to mitigate these barriers, they suggest using icons familiar to users and already available on the mobile platform, adding text labels accompanying icons to explain what the icon means, and using neutral colors with icons. This guideline builds on the user cognitive design DIAZ-3 guideline. Inputs also need short, descriptive labels that provide only necessary information; placeholders

should contain information about the expected format, and all caps should not be used for either labels or placeholders. These both guidelines relate to supporting characteristics of the platform and labeling elements, so, WCAG-10 and ANDROID-1 (Android Developers, 2022; Díaz-Bossini & Moreno, 2014; Patch et al., 2015; Zaina et al., 2022, pp. 8–9)

Di Gregorio et al. (2022) also grouped guidelines for mobile accessibility. They listed the following categories: audio and video, design, editorial, focus, forms, images, links, notifications, scripts and dynamic content, structure, and text equivalents. (Di Gregorio et al., 2022, p. 10) Some categories are related to design or content creation, so they are not directly in developers' control. Thus, they are filtered out from my research. Such categories are design, editorial, images, links, notifications, and text equivalents. This filtering leaves me with five categories: audio and video (DIGR-1), focus (DIGR-2), forms (DIGR-3), scripts and dynamic content (DIGR-4), and structure (DIGR-5).

The video and audio guidelines relate to the controls' visibility, usability, and content presentation. They build on ANDROID-4 guideline about media content accessibility. For focus, the guidelines concentrate on the focus order, which should match the visual representation and be logical. It relates to three other guidelines: WCAG-3, DIAZ-3, and BAL-4. Forms require labels for every input, and the layout should be clear. This guideline backs up WCAG-10 and DIAZ-3. Scripts and dynamic content are about progressive enhancements, mostly related to the web. However, there are applications for native mobile as well for first building the core experience and then enhancing it for more capable users. The category of structure relates to identifying and annotating elements on the page to match the visual experience for screen reader users. This guideline, too, is in relation to WCAG-10. (Android Developers, 2022; Di Gregorio et al., 2022, p. 10; Díaz-Bossini & Moreno, 2014; Patch et al., 2015)

There is other similar research that's been made, but at this point, I have reached saturation point as all the guidelines start to resemble each other. Next, I will draw all these guidelines together.

2.2.4 Summary on Guidelines

The Web Content Accessibility Guidelines related to mobile accessibility, Android's documentation's accessibility guidelines, and prior research reveal a set of guidelines present in multiple sources and overlapping. From these sources, grouping them, doing a thematic analysis with them, and filtering out items that are usually not changeable for a developer, I was able to find eight categories that are useful for my work. Table 1 lists them with their content, and guidelines that are relevant to each category.

Table 1. Checklist categories with relevant guidelines.

Category	**Content**	**Guidelines**
Labels and text alternatives	• Icons should have visible labels • Form inputs should have visible labels • Images should have text alternatives if they convey information	WCAG-10, ANDROID-1, DIAZ-2, BAL-1, BAL-6, ZAI-1, ZAI-2, DIGR-3
Recognition rather than recall	• Use already available and familiar icons • Use patterns that are already familiar • Provide clear error messages and recovery tactics • App works in predictable ways	WCAG-10, DIAZ-3, BAL-7, BAL-9, BAL-10

Category	**Content**	**Guidelines**
Audio and Video content	• Content should be usable and accessible • Controls should be accessible	ANDROID-4, BAL-2, BAL-3, DIGR-1
Keyboard navigation and focus order	• Keyboard navigation should work • Focus order should match the visual representation • Extend platform-provided widgets instead of building from scratch	WCAG-3, WCAG-7, WCAG-8, WCAG-9, ANDROID-2, ANDROID-3, DIAZ-3, BAL-4, BAL-5, BAL-11, DIGR-2, DIGR-4
Gesture-related	• Touch target size should be big enough • Accessibility actions should be added for gesture-based actions • Gestures should not be the only way to interact, and there should be an alternative	WCAG-4, WCAG-5, WCAG-7, DIAZ-1, BAL-5, BAL-8
Orientation should not be locked into either position		WCAG-6, BAL-5
Structure	• Identify headings, containers, and landmarks	BAL-6, DIGR-5
Zooming and magnification	• The app should be usable when zoomed in or when using the magnification feature	WCAG-1, WCAG-2

It is worth noting that research on accessibility has been largely about disabilities that affect vision (Sandnes, 2022). However, as Vendome et al. (2019, p. 50) noted in one of the lessons they identified, mobile accessibility is not only about screen reader accessibility. I will discuss this and other developer accessibility knowledge-related issues in the next section.

2.3 Android Developers and Accessibility Knowledge

As mentioned in Chapter 1, Android apps are usually not accessible. In their analysis of the accessibility of 50 Android apps, Di Gregorio et al. (2022) found out that, at best, only 63 % of the accessibility guidelines they were testing were implemented. Furthermore, they concluded that mobile developers do not tend to implement accessibility guidelines even if they have a chance to do so (Di Gregorio et al., 2022, p. 145). Also, de Almeida and Gama (2021, p. 127) found out that 24% of developers had never been in contact with accessibility as a topic, and 64% did not know WCAG guidelines for developers. Leite et al. (Leite et al., 2021) discovered that developers who develop only for iOS were more aware of accessibility guidelines than developers who develop just for Android or both platforms.

In this section, I will examine the challenges developers face related to accessibility more closely. I will also discuss proposed solutions.

2.3.1 The Challenges

The first challenge in the literature is not knowing about accessibility and lack of awareness. Di Gregorio et al. (2022, pp. 30–31) discovered that developers often had problems understanding the exact needs of disabled users. On the other hand, they mentioned the (lack of) awareness of companies and customers and how that leads to accessibility requirements being ignored. Alshayban et al. (2020, p. 1330), de Almeida and Gama (2021, p. 128), and Leite et al. (2021, p. 19) had similar findings in their research. The discovery of companies ignoring accessibility requirements was visible in Patel et al. (2020, p. 4), too; they found that leadership often prioritized other things over accessibility when a deadline was approaching, and accessibility was often seen as an extra cost and not an opportunity. This attitude affected internal policies and prioritizing as well, meaning it was hard to get time to fix accessibility issues retroactively. Leite et al.'s (2021, p. 19) results suggest that the biggest barrier or limitation to accessibility is that it is not a part of the project requirements. Furthermore, as Di Gregorio et al. (2022, pp. 28–29) discovered, fixing accessibility issues as an afterthought can be considered hard and thus means that those issues are not fixed.

Vendome et al. (2019), analyzed accessibility-related questions on Stack Overflow, a platform developers use to ask questions and get help. They noticed that developers do not have much exposure or background with accessibility, leading to questions that revealed that they did not understand how, e.g., assistive tech users would use assistive technology. Leite et al. (2021) also confirmed this finding. Related to awareness and not understanding the needs of people with disabilities, one challenge is that it is hard to find ways to interact with people with disabilities. However, when interacting with disabled people, it was considered a helpful resource. (Patel et al., 2020, p. 5)

Another challenge raised in the literature was that accessibility is often about screen reader accessibility and ignoring other aspects. (Vendome et al., 2019, pp. 49–50) This conclusion also aligns with Acosta-Vargas et al.'s (Acosta-Vargas, Salvador-Acosta, et al.,

2021) findings that most disabilities considered for accessibility evaluation are sensory-related or, more precisely, visual or hearing impairments. One challenge, or rather a reason, why developers wouldn't implement accessibility guidelines is that they feel that some of them may have possible negative effects on the app's aesthetics and usability. (Di Gregorio et al., 2022, p. 27)

The lack of usable tools causes challenges in considering accessibility as well. There are not that many different tools available, and several developers have faced situations where tools have disappeared from use all of a sudden. (Alshayban et al., 2020, p. 1130; Leite et al., 2021, p. 19; Patel et al., 2020, p. 5; Vendome et al., 2019, p. 49) Other notable challenges were willingness to implement accessibility guidelines, lack of time, vague definitions of accessibility guidelines, and problems finding relevant and usable information from the web about accessibility features. (Alshayban et al., 2020, p. 1130; Di Gregorio et al., 2022, pp. 25–27; Leite et al., 2021, p. 19; Patel et al., 2020, p. 5)

2.3.2 Proposed solutions

In the prior research, several suggestions have been made to solve the challenges introduced in 2.2.1. One proposed way to solve these challenges is to have more and better tools for development to address accessibility in specific technical implementations. They could be integrated with IDEs (Integrated Development Environment such as Android Studio, which is a widely used for Android development). These tools could be (semi) automated. (Patel et al., 2020, p. 6; Vendome et al., 2019, pp. 49–50) Another proposal is to increase accessibility knowledge through formal education and beyond computing curriculums. Also, within computer science and related fields, learning about addressing accessibility issues during development would be beneficial. (Patel et al., 2020, p. 6; See also Bhatia et al., 2023)

Developers would generally benefit from education on the topic, learning more about universal design principles, and having targeted tutorials and workshops addressing specific accessibility guidelines. There should also be efforts from practitioners, researchers, and educators to promote best practices for accessibility. (Patel et al., 2020, p. 6; Vendome et al., 2019, p. 50)

So, to conclude, education in both formal and informal settings is the key. Also, developing and integrating tools that help developers build more accessible apps more easily can provide the help developers need.

3 Design Science Research

The method for conducting research in this master's thesis is Design Science Research. It is a method that aims to create and evaluate artifacts that should solve identified organizational problems. These artifacts are represented in a structured form, which varies between artifacts. Artifacts can be, for example, constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and representations), and instantiations (implemented and prototype systems). (Hevner et al., 2004, p. 77) Design Science Research consists of design activities that are composed of "build" and "evaluate" activities, which typically iterate multiple times (vom Brocke, Hevner, et al., 2020, p. 6).

The artifact built in the context of this research, the checklist, is a method that defines a process, guiding how to solve a problem. The final form of the artifact could also be described as an instantiation, as it shows that the model in this context can be implemented in a working system as defined by Hevner et al. (2004, p. 79). The checklist could also be described as a problem-specific aggregate of models and methods, as Winter (Winter, 2008, p. 471) defines an instantiation.

As vom Brocke et al. (vom Brocke, Winter, et al., 2020, p. 521) describe, design knowledge for a specific DSR project includes information about the problem, the designed solution, and the evaluation. Three components of design knowledge are problem space, solution space, and evaluation. While the first two exist independently, design knowledge can emerge only through their relation. Problem space consists of the context and goodness criteria, and solution space consists of representations and processes. (vom Brocke, Winter, et al., 2020, pp. 521–522) The solution and evaluation aspects will be discussed later in this chapter, but to summarize the problem concepts, Table 2 lays out the needs, goals, requirements, and stakeholders of the current research.

Table 2. Concepts of the problem, adapted from Maedche et al. (2019).

Concept	**Definition**
Stakeholders	• Android developers • Users who use assistive technology / accessibility settings
Needs	• Accessible Android applications
Goals	• Enable Android developers to develop more accessible apps
Requirements	• A tool to help Android developers to develop more accessible apps

3.1 Research Cycles in Design Science Research

Hevner (2007) presents the three cycle views for DSRM: the Relevance Cycle, the Rigor Cycle, and the central Design Cycle, as shown in Figure 1. It displays the application domains: people, organizational, and technical systems. Being part of the environment also presents opportunities and research problems. On the other end is the knowledge base, which provides scientific theories and methods, experience and expertise, and meta-artifacts such as design products and processes. Design science research is in the middle – building design artifacts and processes, and evaluation.

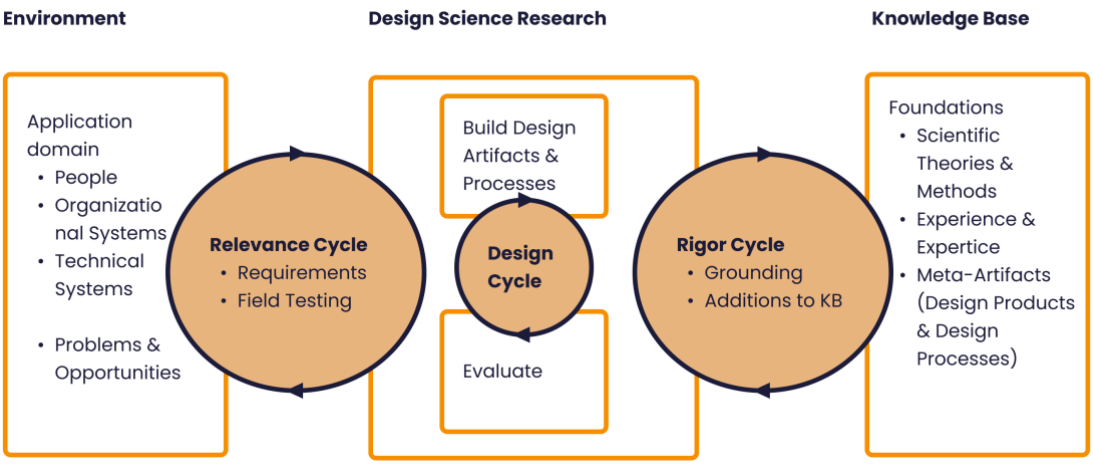

Figure 1. Design Science Research Cycles (Hevner, 2007, p. 2).

Three cycles connect the areas: The Relevance Cycle, which connects the environment with Design Science Research, and the Rigor Cycle, which connects the knowledge base with Design Science Research. Within the DSR, the Design Cycle connects the building and evaluation activities. The Relevance Cycle initiates DSR by providing the problem and requirements for the research. It also defines the acceptance criteria for the evaluation of the research results. The output, the final artifact, needs to be returned to the environment of the study to be fully evaluated. (Hevner, 2007, p. 3) In the context of this research, the environment that provides the problem consists of Android development and applications out in the wild. In a broader sense, it also contains Android users, both with and without disabilities. That environment then contributes to the problem and definition of the solution. The artifact has not been fully returned to the study environment for evaluation, as described in the next subsections.

The Rigor Cycle is about the theory and prior knowledge of the research subject. It utilizes prior research as well as experiences and expertise in the application domain and existing artifacts and processes in the application domain. It also grounds the research on theoretical knowledge. (Hevner, 2007, pp. 3–4) In my research, two components contribute to the Rigor Cycle as input: prior research, as described in 3.2.1, and my

expertise as an accessibility specialist. On the other hand, it also contributes to the knowledge base, as discussed in 6.1.

The Design Cycle is the core of the design science research project. It iterates between the construction of the artifact, its evaluation, and feedback to carry the design further. It requires input from the relevance cycle and the theories and methods for design and evaluation from the rigor cycle. It's important to remember that it is still an independent cycle, even though it has dependencies on the other two cycles. (Hevner, 2007, pp. 4–5) As generally in design science research project, Design Cycle serves as the core of the project. There are three cycles of building and evaluating in this research:

1. Evaluating and verifying the problem
2. Building the initial checklist based on the prior research
3. Building the prototype of the checklist and evaluating it

The process is discussed further in the following subsection.

3.2 Design Science Research Method (DSRM)

Peffers et al. (2007) introduce a design science research process (DSRM) with six phases. These phases are:

1. Problem identification and motivation
2. Definition of the objectives for a solution
3. Design and development
4. Demonstration
5. Evaluation
6. Communication.

Figure 2 demonstrates how my research adopts this process. The entry point for research is problem-centered initiation. As Peffers et al. (2007, p. 56) suggest, it is a relevant starting point "if the idea for the research resulted from observation of the problem…" In this thesis's case, that is precisely where the research results.

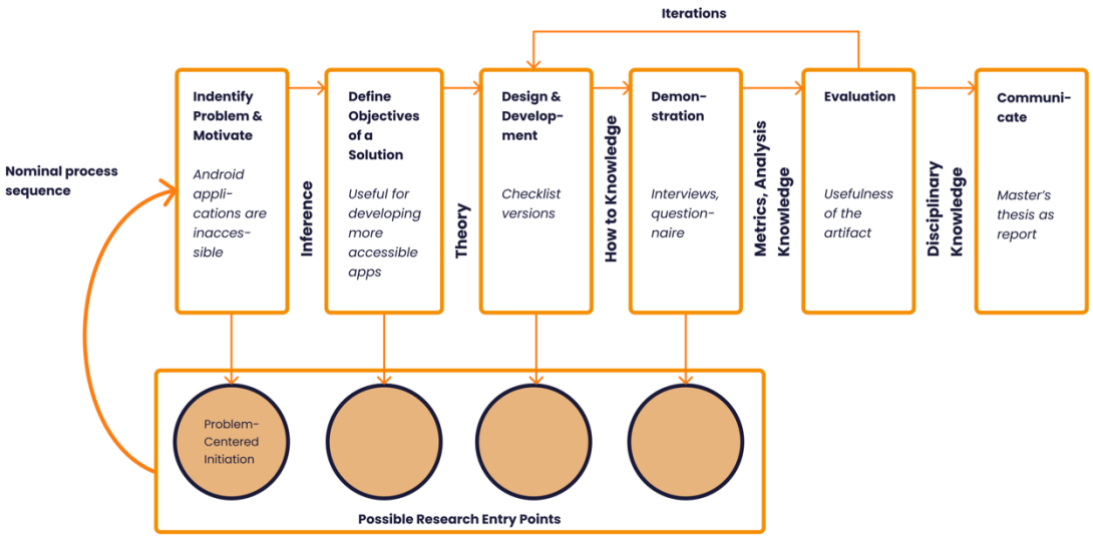

Figure 2. DSRM Process Model, adapted from Peffers et al. (2007).

From my experience as an accessibility professional and Android developer, I recognized the problem of many Android applications being inaccessible. This observation contributed to selecting the topic for this thesis and thus motivated and started the design research process. The importance of this topic is explained further in Chapter 1, which also concludes step 1 in the design research process.

The solution to the problem needs to be useful for Android developers in the context of creating more accessible Android applications. As the medium, a thesis sets some limitations from time and extent for the research. The feeling of usefulness for developing more accessible apps is used as the metric for the success of the artifact. For a more extensive study, it would be useful to evaluate if the artifact contributes to more accessible applications.

The design and development of the artifact started with the initial checklist constructed from prior research based on a narrative literature review and prior knowledge. This checklist was then demonstrated to five Android developers, and its usefulness, content, and form were evaluated. Based on the feedback, a new, improved checklist was created

and then demonstrated as a website accompanied by a questionnaire. The questionnaire results were used to create the final checklist and evaluate its usefulness.

This master's thesis report concludes the last activity in the design research process: communication. It communicates the problem, its importance, the artifact, and its utility, novelty, and effectiveness, as described by Peffers et al. (2007, p. 56).

3.3 Evaluation

This subsection explains the evaluation process in detail. While DSRM's evaluation is outcome-based (Peffers et al., 2018, p. 133), to evaluate the artifact during the process, I have followed Sonnenberg & vom Brocke's (2012) evaluation methodology. As seen in Figure 3, it is an iterative process that consists of ex ante and ex post evaluations. Ex ante evaluation, so evaluation before the artifact has been applied to a real-world problem, is closely related to the design process. Ex post evaluation, so, evaluation after an artifact has been constructed, on the other hand, is closely tied to the artifact in use. (Sonnenberg & vom Brocke, 2012, pp. 5, 7, 13–14)

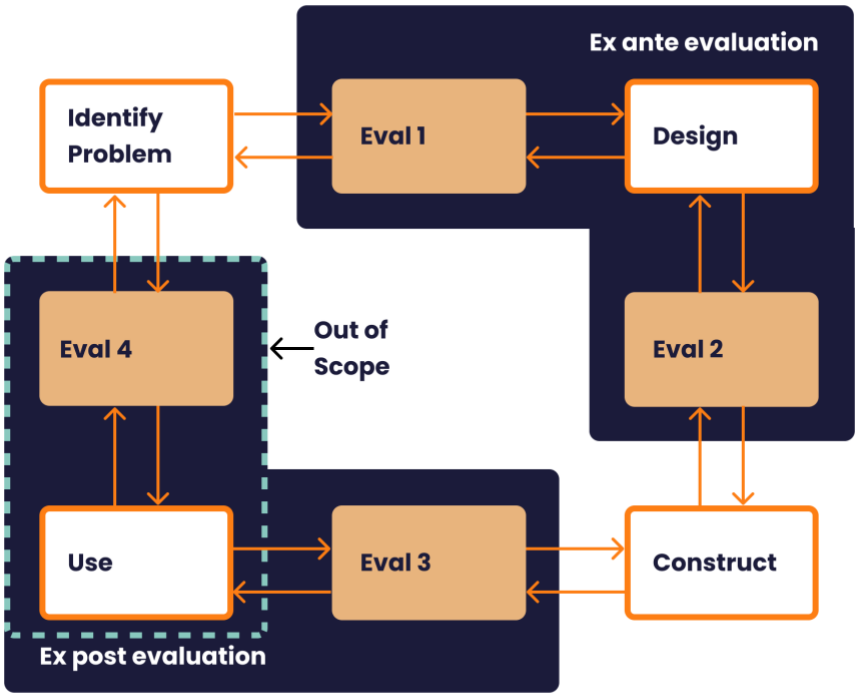

Figure 3. Evaluation activities within a DSR process, adapted from Sonnenberg & vom Brocke (2012).

As the evaluations in this research are an iterative process, they could also be described as formative evaluations. As Venable et al. (2016, p. 77) describe, formative evaluations are used to "produce empirically based interpretations that provide a basis for successful action in improving the characteristics or performance of the evaluand." At the end of the research, a summative evaluation could be constructed to measure the results of the completed development (Venable et al., 2016, p. 79).

This research conducts three evaluation activities – Eval 1, Eval 2, and Eval 3, described in Figure 3. The fourth evaluation activity is out of the scope of this research because of the limitations described in 1.1. Sonnenberg and vom Brocke (2012) describe the first evaluation activity as serving a purpose to ensure the meaningfulness of the DSR problem, its selection, and formulation. The goal is to demonstrate whether the design science research project is important for practice; it is novel and adds to the existing

knowledge base. (Sonnenberg & vom Brocke, 2012, p. 16) The first evaluation activity can include a problem statement or observation of a problem, research needs, design objectives, design theory, or existing solution to a practical problem. As an output, it should produce justification for the problem statement and justify the research gap and the design objectives. (Sonnenberg & vom Brocke, 2012, pp. 16–17)

The second evaluation activity aims to show that the artifact design progresses to a solution that solves the stated problem. Possible inputs for this activity include design specification, design objectives, information on the stakeholders of a design specification, or tools and methods used for creating the design specification. The output of this evaluation activity should be validated design specification and justified design tool or methodology. (Sonnenberg & vom Brocke, 2012, pp. 16–17) As Abraham et al. (2014, p. 6) suggest, this evaluation activity can build on top of the literature review and be evaluated against field data.

The third evaluation activity aims to initially demonstrate if and how well the artifact would perform in relation to organizational elements. This activity takes in an instance of the artifact, such as a prototype. The output of this activity is a validated artifact in an artificial setting and, possibly, proof of applicability. Evaluation of the artifact in this activity can be done within some of the realities, for example, with real users, real systems, or real tasks. Thus, the evaluation might prove applicability only concerning a task, within a system, or by real users – not all of them. (Sonnenberg & vom Brocke, 2012, pp. 16–17)

The research reported in this thesis contains these three evaluation activities described above. The first evaluation activity was conducted to justify the problem, and the method for evaluation was a literature review. The second evaluation activity was conducted after the literature review, and the input for the activity was the design specification in the form of the initial checklist. The method for evaluation was interviews with five Android developers. The third evaluation activity was performed

after a version of the artifact was created in the form of a website. The evaluation method was a survey for Android developers. Evaluation activities are described in depth in Chapter 5.

The next subsection discusses material collection methods used in this thesis.

3.4 Data Collection and Analysis

3.4.1 Narrative Literature Review

For the knowledge base and rigor cycle, a narrative literature review method is used. In general, there are four types of literature review: narrative review, integrative review, systematic literature review, and meta-analysis (Fan et al., 2022, p. 173). From these four, the narrative literature review suits my research best because its goal is to find out what is already known about the phenomenon, the main concepts, and the relationships between the concepts (Vilkka, 2023, pp. 39–40).

A narrative literature review is based on the mechanisms for organizing and analyzing literature. These mechanisms are often informal. The process starts with a small number of research papers and then increases as the researcher's understanding and knowledge of the topic increases. This approach has been criticized because of its informal and incremental nature, which can bring in the implicit biases of the author. Also, as the arguments can be based on flawed or selective assumptions, it can lead to a lack of thoroughness and systematicity. To address these limitations, researchers need to be transparent about their choices and approaches they follow. (Fan et al., 2022, p. 172)

The material collection for this thesis was twofold: First, the more official Android accessibility documents from Google and Web Content Accessibility Guidelines from

W3C were used. Second, prior research on the thesis topic was used. Web Content Accessibility Guidelines, and more concretely, their applications on mobile, were selected because these guidelines are, in fact, used in many countries in legislation about accessibility. In addition, Google's instructions were selected because they are the company developing Android. For prior research, I started by searching for articles about "Android" and "Accessibility." I found a couple of articles about Android accessibility and continued my research from those papers as my understanding increased. After much reading, four papers with accessibility guidelines for Android were selected. Table 3 introduces these papers.

Table 3. Selected articles with accessibility guidelines.

Article	**Guidelines**
Ballantyne et al., 2018	An exhaustive list of guidelines to gauge mobile app accessibility, with a mobile-spesific framework for categorizing the guidelines.
Zaina et al., 2022	Guidelines to prevent problems in nine user interface patterns with accessibility barriers.
Díaz-Bossini and Moreno, 2014	A set of accessibility guidelines for mobile interface accessibility for older people.
Di Gregorio et al., 2022	A set of accessibility guidelines as technological agnostic best practices for mobile web content, hybrid, and native apps.

3.4.2 Interviews

As Hyvärinen et al. (n.d.) write, when one wants to know more about others' thoughts and actions, why not ask? That is why I chose to interview developers and ask them questions about the checklist.

There are different types of interviews. Hirsjärvi et al. (2009, p. 208) list three types of interviews: structured, half-structured and unstructured interview. A structured interview is usually based on a form which has the questions in a structured format (hence the name). The assumption is that everyone understands the questions in the same way. (Hirsjärvi & Hurme, 2022, Chapter 4.2.1) An unstructured interview uses open-ended questions. The interviewer is responsible for deepening the answers with new questions, and the interview often resembles a conversation. (Hirsjärvi & Hurme, 2022, Chapter 4.2.2)

A half-structured interview has different definitions. For example, Fielding (1993, in Hirsjärvi & Hurme, 2022) defines that in a half-structured interview, the questions are the same for everyone, but the order of the questions might change. On the other hand, Eskola and Suoranta (1998, in Hirsjärvi & Hurme, 2022) state that the questions are the same, but answers can be open-ended. Robson (1995, in Hirsjärvi & Hurme, 2022) sees the nature of a half-structured interview such that the questions are pre-defined, but the interviewer might change the phrasing of the questions. The common theme for half-structured interviews is that some aspect of the interview is identical for all, but not everything needs to be similar. (Hirsjärvi & Hurme, 2022, Chapter 4.2.3)

For this research, I chose to use the half-structured interview. The questions were the same for each interview, and the answers were open-ended. The phrasing of the questions was subject to change. Also, I already knew the themes I wanted to have information about, so an unstructured interview did not help me. The interview questions can be found in Appendix 2.

Five Android developers were interviewed for this research. They were recruited through Slack (a messaging app used by the company). Most of them (3) had more than ten years of Android development experience; one had 9, and one had about three years. Their background in accessibility varied; some had more experience and had been developing apps with accessibility requirements, and some described their knowledge to be just the "basic stuff".

3.4.3 Questionnaire

The final iteration of the research process was conducted with the help of a questionnaire. For the questionnaire, Google Forms was used for its familiarity for me and wide usage as a tool for surveys. The questionnaire consisted of two pages: First, a page asking for participant's consent, and second, a page with actual questions. The questions on the second page are listed in Appendix 3. The questionnaire was tested with two Android developers before sharing to verify it.

The answers to the questionnaire were collected from January 26th, 2024, to February 23rd, 2024. I shared the questionnaire in different Slack groups and social media channels together with a link to the accessibility checklist. Multiple people reshared it. The questionnaire received 13 answers in total.

The questionnaire's aim was to understand two things: whether the checklist would be useful for Android developers and whether something would be missing from it. The responses gathered answered these questions well and produced new material for the checklist, which I then utilized in the final version of the checklist.

I analyzed the questionnaire answers with the help of theming. First, I went through the whole material multiple times and then started to look into it on a more granular level,

as Puusa et al. (2020, Chapter IV) suggest. There were several similarities in the answers, which I constructed into themes. These themes were then extracted from the material and used to improve the checklist further.

While the amount of material might seem small, with this sample, the themes in the answers started to repeat themselves, and the material was no longer providing new, relevant information. In other words, it can be said it was saturated, as defined by Tuomi and Sarajärvi (2017, Chapter 3.4.1).

The next chapter discusses the artifact development process in more detail. It discusses each step of the development and, finally, presents the checklist in its final form.

4 Developing Android Accessibility Checklist

In this chapter, I will discuss the development of the final artifact, the Android accessibility checklist. In the first subsection, I will look at the results of the literature review and how they were used to create the initial checklist that was evaluated in the interviews. In the second subsection, I will discuss the interviews with five Android developers and how their answers further influenced the checklist development. The final subsection of this chapter is about demonstrating the checklist by sharing it with a broader audience, collecting feedback, and developing the final version of the Android accessibility checklist.

4.1 Literature Review and Initial Checklist

4.1.1 Findings of Literature Review

The literature review results provided a set of accessibility guidelines for Android applications. These guidelines derive from prior research and different accessibility guidelines, such as Web Content Accessibility Guidelines and Android Accessibility principles. The guideline categories found were labels and text alternatives, recognition rather than recall, audio and video content, keyboard navigation and focus order, gesture-related guidelines, orientation (should not be locked into one position), structure, and zooming and magnification. All the relevant guidelines are listed in chapter 2.2.4.

Another topic the literature review highlighted was Android developers' challenges when implementing accessibility. As mentioned in section 2.2.1, many different aspects

prevent or cause challenges for Android developers when it comes to developing accessible apps. Some of these include a lack of usable tools, a lack of knowledge, and problems finding relevant and usable information (Di Gregorio et al., 2022, pp. 25–27; Patel et al., 2020, p. 5; Vendome et al., 2019, p. 49).

If there is material available, it tends to be heavily focused on screen reader accessibility, often ignoring other aspects of accessibility (Vendome et al., 2019, pp. 49–50). Also, Patel et al. (2020, p. 6) and Vendome et al. (2019, p. 50) conclude that developers would benefit from education about accessibility—both in formal education and outside of it.

Di Gregorio et al. (2022, pp. 28–29) found in their study that fixing accessibility issues as an afterthought is not straightforward and usually takes more time, leading to a situation where accessibility issues are not prioritized. Also, Alshayban et al. (2020, p. 1130) found that about 72% of the survey respondents had no accessibility evaluation as part of their app development process.

These points about the lack of tools and education, the need for processes, and ways to integrate accessibility evaluation into the development process serve as the motivation for this study. They present a need for a resource for easy-to-use and educative tools that could be integrated into the development processes. Together with these findings and my prior knowledge as a software developer, the form of the checklist started to be clear. In the next subsection, we discuss the contents and form of the checklist.

4.1.2 Initial Checklist

As discussed in the previous subsection, the relevant guidelines were divided into eight categories with differing numbers of subitems. Using my knowledge as an Android developer and accessibility specialist, I constructed the initial checklist based on these

guidelines. The goal was that the checks be actionable, clear, and relevant to modern Android development.

Looking at the guidelines provided by the literature review, it became clear that three are usually things out of a developer's control. Of course, this varies per company, but generally, the selection of icons and the design of error messages are not in the developer's control. The following guidelines were filtered out:

- Icons should have visible labels
- Use already available and familiar icons
- Provide clear error messages and recovery tactics

After this, I had 15 guidelines to construct into actionable, relevant checks. I grouped the checks into eight categories: automated tools, Images and other visual elements, Use of color, Alternative ways of navigation, Orientation, Semantics and structure, and Zooming and magnification. I will discuss next what each section contains and which guidelines contributed to it.

4.1.2.1 Automated Tools

The automated tools section of the initial checklist consists of two checks: "Ran Accessibility Scanner for the feature" and "Ran color blindness simulation for the feature."

Accessibility Scanner is an Android application and tool for testing some aspects of the app's accessibility. It scans the app screen or screens. It suggests improved accessibility in the following categories: Content labels, touch target size, clickable items, and text and image contrast. (Google Help, 2024)

From the prior literature's guidelines' point of view, it covers the "Touch target size should be big enough" and provides some checks for "Focus order should match the visual representation." While it does not fully cover this guideline, checking with the Accessibility Scanner can help to find some problems with the focus traversal order.

The other check for running a color blindness simulation for the feature (or app) does not cover any of the guidelines from the prior research. I added it based on my expertise on the topic and my experiences with testing and developing applications. Also, Pinheiro et al. (2023, p. 25) noted that color blindness simulation tools can help developers catch color blindness-related problems, sometimes even better than a color-blind person.

4.1.2.2 Images and Other Visual Elements

This section covers the guideline “Images should have text alternatives, if they convey information”. I chose to break this into two checks: “Every visual element, that conveys meaning has a descriptive text alternative,” and “Purely decorative images have an empty content description”.

This decision was based on my previous experiences: It is not always clear when text alternatives (or, as in Android, the attribute called content description) are required and when they are not. This distinction underlines that there are two cases: adding text alternatives when needed and not having them when they are redundant.

4.1.2.3 Use of Color

The initial checklist's "Use of Color" section consists of one check: "The feature works even if the user can't see colors." It does not particularly cover any of the guidelines from the literature review, but based on my experience, it is an important check.

One way to complete this check would be with a color blindness simulation tool, and one could argue that this check is redundant because of that. However, I see a distinction between running the test automation tool and the feature working if the user does not see color. There is also an opportunity to combine these checks in further iterations.

4.1.2.4 Alternative Ways of Navigation

The “Alternative Ways of Navigation”-section is the largest, and it contains the following checks:

- The feature works with the keyboard
- The feature works with Switch-control
- The feature works with Voice control
- Focus order matches visual representation
- Added accessibility actions for gesture-based actions (e.g., swipe), or there is another way to interact than the gestures (e.g., buttons)

These checks cover or contribute to the following guidelines:

- Keyboard navigation should work
- Focus order should match the visual representation
- Content should be usable and accessible
- App works in predictable ways

- Controls should be accessible
- Use patterns that are already familiar
- Gestures should not be the only way to interact, and there should be an alternative

My approach for this section was introducing different assistive technologies for testing and providing concrete checks to complete. As Youngblood (2013, p. 20) suggests, a novice developer should be taught to use multiple tools because they work in different ways. Testing with these tools also has the potential to reveal different accessibility issues. Some issues might not be discovered if the developer tests with only one tool. While these checks cannot guarantee finding every possible issue, using multiple tools increases the possibility of finding most of the issues.

4.1.2.5 Orientation

The checklist's section "Orientation" covers the guideline "Orientation should not be locked into either position". It contains two checks: "Orientation is not locked to either mode" and "Experience is similar for landscape and portrait modes".

Many apps do not support both horizontal landscape and vertical portrait modes. For example, Madeira (2021, p. 564) found that of the 11 applications tested in their study, only two supported both orientations. Also, Alajarmeh (2022, p. 520) discovered that some participants reported problems with locked screen orientation. These findings are also something I have observed and talked a lot about with Android developers: The consensus seems to be that there is no need (or time) to support both modes, and it is ok to lock the orientation to vertical portrait mode.

However, I have seen apps that utilize the landscape mode showing one part of the screen (a graph, for example) in the landscape mode but not the whole screen. These experiences led me to add the second check about similar experiences for both modes.

4.1.2.6 Semantics and Structure

The section “Semantics and Structure” contains three checks: “Semantics (e.g. headings) are identified”, “Form fields have labels programmatically associated to them”, and “Custom components have correct semantics, state information and interaction patterns”. From the guidelines from the prior literature, this section covers three:

- Identify headings, containers, and landmarks
- Extend platform-provided widgets instead of building from scratch
- Form inputs should have visible labels

This section could have contained more concrete checks, but I did not want to make it too long. One example of these possible checks would have been that only interactive elements are focusable – as Zhong et al. (2015, p. 8) found out in their research, that several apps had non-interactive widgets flagged as touch targets, and thus would complicate the usage of the app for users using different assistive technologies.

I also contemplated adding an explicit check for extending platform-provided widgets instead of building from scratch. While it might have been useful, it also might have been too vague and, due to the nature of Android development, even redundant. Based on my experience, compared to web development, where that check is critical, most widgets are built on top of the system-provided widgets in Android development. So, given the need to keep the list as short as possible and the reasons listed above, I deemed this check redundant and left it out.

4.1.2.7 Zooming and Magnification

The last section, “Zooming and Magnification,” contains two checks: “The feature works when using zoom or magnification” and “The feature works with font size set to the biggest size.” These both relate to the guideline “The app should be usable when zoomed in or when using the magnification feature.”

I wanted to call out the case specifically with larger font sizes. From my experience, this is something that rarely gets tested before a user reports a problem with larger font sizes. This behavior was confirmed by Eler et al. (2019, p. 7) , as they found in their analysis of Play Store reviews that about 10% of the analyzed reviews mentioned font size adjustments in 30% of the analyzed apps.

4.2 Interviews And Developing the Checklist Further

After the initial checklist was developed, I interviewed Android developers to confirm that it was going in the right direction. Based on the analyzed interview responses, I developed it further. In the next subsections, I’ll first discuss the interviews and their content and then the further developments on the checklist.

4.2.1 Findings of Interviews

The interviewed Android developers had various levels of accessibility knowledge. Some had worked with accessibility before, and some knew the basics, but they admitted that they knew little. Even the ones with more knowledge recognized that it had been a while since they worked specifically with accessibility.

These interviews were conducted remotely through a video meeting service (Google Meet) and recorded. Notes were also taken during the interviews. Both types of materials, recordings, and notes, were used for further analysis.

These interviews revealed several interesting points about the checklist. Overall, all participants agreed that this kind of tool would be helpful. However, they also recognized that they have limited time when they are developing features, so unless accessibility is not prioritized by management, they probably would not have time to think about accessibility. They also raised points like designs should already contain a lot of accessibility-related decisions and information and that they do not have much exposure to how people with disabilities use digital devices – which aligns with prior research (Patel et al., 2020, p. 5).

Most of the interviewed developers also mentioned that some of the proposed checks are out of their control. For example, color-related decisions are often made by the designer or dictated by color themes, and a developer does not have much to say about that. Another common theme was that more information and instructions were needed. Depending on the person answering, some of the checks seemed straightforward, but it became clear that most checks needed accompanying instructions for testing.

A couple of notions were made when I asked about the checklist's format and how it would serve them best. First, the checklist itself must be brief. It should not contain too many instructions but links to instructions on completing the tests in the checklist. Second, it could be part of a pull request template[1]. However, some interviewees also mentioned that they would prefer it to be in a separate place, where they could check it throughout the development, not just when it is time to merge their code to the general codebase.

[1] When a developer wants to merge their code to the common codebase, they create a pull request. Different cloud-based version control providers provide a way to define a template developers should use when creating the pull request.

Also, many interviewees mentioned that it would be good to include why a particular check is done — as one of the interviewees said, "to build empathy". Studies support this. For example, Di Gregorio et al. (2022, p. 23) found out in their study that 39% of developers implement accessibility guidelines because of their personal ethics. Deriving from that, if the checklist can appeal to the developer by building empathy, that would increase the application of accessibility guidelines.

Based on the interviews, I developed the checklist further. In the following subsection, I will discuss the changes in the content and format and other decisions I made.

4.2.2 Developing the Checklist Further Based on Interviews

As discussed in the previous section, the interviews illuminated some aspects of the checklist and called for improvement. Areas to improve were the content and structure of the checklist, adding instructions for testing, and creating a website for the checks. Let us discuss these next.

First, based on the interviews, I moved some checks under a section called "Checks That Might be Out of Your Control (But Worth Checking!)". The checks I moved were "Every visual element that conveys meaning has a descriptive text alternative", "The feature works even if user can't see colors", and "Orientation is not locked to either mode".

The reason for the move was that these are checks that the developer does not have control over in many instances. In many companies, decisions around color are made in branding or the design phase. Also, from what I have noticed, there has been a long-standing custom to develop only for portrait mode—and changing it might not be in the control of one developer and might require much work. That's why I decided to move that check to this section.

Another change related to the content and structure of the checklist was to modify the wording of some checks. This included, for example, using the correct names for switches (so, Switch Access instead of Switch-control) and changing the check for form fields to check that the labels are available—the Accessibility Scanner checks for the programmatic association.

Finally, I merged some parts of the checks, as those were overlapping or redundant. I moved the checks for zooming (so having a screen setting set to the biggest option) under the font size tests and added them to the instructions I wrote. I also decided to remove the check for the color blindness simulator, as even if it could be useful, it is something the developer often does not have control over, as discussed above.

In addition to the changes to the checklist itself, I developed a website that contains the checks and instructions for completing them. As the goal is to provide a development tool, I added the checks in markdown format, which is used in many development processes, so that they could be copied and used as part of the development flow.

After finishing the website containing the checklist and accompanying instructions, I created a questionnaire to survey Android developers. In the next section, I will discuss its results and the final checklist.

4.3 Questionnaire And the Final Checklist

4.3.1 Questionnaire Results

The respondents' experience with Android development ranged from about one year to 13 years. The average experience was 6.5 years, and the median was five years. Eight of

the respondents described their accessibility knowledge as being minimal or as being new to accessibility. Two respondents described their knowledge as good, and two as medium.

The question of whether the checklist would be useful received an almost unanimous answer of yes. Only one respondent answered no. In addition to answering yes, some respondents provided longer answers, explaining that the checklist is a good reminder of things to consider and test. One respondent pointed it out as a tool, especially for developers who haven't had much experience with accessibility.

The final question in the questionnaire was about possible missing things from the checklist. A couple of themes were visible in several answers: Adding example images or graphics, adding code examples, and adding unit test examples. In addition, the following themes were mentioned once: Adding priorities to checklist items, adding a test with TalkBack to the list of tested accessibility services, adding a check about Time to React, extending the check to test with a keyboard to include D-pad, adding something about dark and light modes being available, and adding information about live regions.

In addition to these, one respondent pointed out that the content description should be set to null, not an empty string, for purely decorative images. I had missed this while improving the initial checklist, so I added it. Another oversight on my side was that I had added instructions to test with a screen reader, but I had not included them as part of the checklist.

In addition, text size and fonts were mentioned, but as mentioned previously, developers often lack control over these, so I intentionally left them out of the checklist.

4.3.2 Improvements for the Final Checklist

After analyzing the questionnaire results, I improved the checklist. I added two more checks: one about having captions with audio and video and one about testing with a screen reader. The screen reader check was purely forgotten from the previous version—I had added instructions to the page but not the check itself.

I added the new check about having captions as part of the checks that might be out of the developer's control. This decision was made because often, supporting the captions is technically possible, but the captions themselves are not available—and in that case, it might not be in the developer's power to add them.

In addition, I extended the check about the keyboard to include a directional pad (D-pad). It's often used for TV-applications as the navigation method, but it can also be used as navigation input for mobile devices. I also changed the wording of the check about purely decorative images having content descriptions set to empty to be set to null, as described in the previous subsection.

I added images to several places where they are suitable. For example, the Accessibility Scanner page now has an example image of the results of the items found when scanning, and the font size testing instructions include an example of the problems testing might reveal. I also added some test code examples for the custom components, giving an idea of how UI tests (or instrumentation tests, as they are called in Android development) could be used for testing accessibility. The tests demonstrate how to test the WCAG criteria 4.1.2 "Name, Role, Value" for custom components.

Finally, I added a link to Appt.org's repository of accessible code samples (Appt.org, 2021). I was pondering whether code samples should be in the checklist's accompanying materials but decided that adding them there in this current form would make it harder

to read. As a future improvement, the checklist could have a section containing examples of how to build custom components in an accessible way.

The questionnaire also gave ideas for future improvements. The checklist or the accompanying material could include supporting time to read-setting, something about live regions, dark and light mode-related checks, and prioritizing the checklist items. The last one is important, as one of the participants (QR3) said:

(4) "… for example font scaling can cause headaches, whereas a seizure from video can be fatal."

As seen in the example (4), the prioritization of the checks would highlight that some of the checks can have bigger impact – while some things listed as problems might be inconvenient, others could be outright fatal.

4.4 Android Accessibility Checklist

Table 4 introduces the design principle of developing more accessible apps as constructed based on Gregor et al. (2020, p. 1633).

Table 4. Design principle of developing more accessible Android apps.

Design principle title	**Design principle of developing more accessible Android apps**
Aim, implementer, and user	For Android developers (implementers) to develop more accessible apps (aim) for users who use assistive technology or accessibility settings
Context	In Android app development

Design principle title	**Design principle of developing more accessible Android apps**
Mechanism	Ensure Android developers have knowledge and tools to develop more accessible apps
Rationale	Because having those tools would help Android developers to develop more accessible apps for more broader group of users than just screen reader users as mentioned in Vendome et al. (2019, pp. 49–50)

The Android Accessibility Checklist, the final artifact, consists of checks and accompanying material. The material provides instructions on how to perform the checks on the checklist and links to learn more about accessibility in general and specifically on Android.

The checks are grouped into two larger categories: The checks that should be performed and those that would be good to perform but might be out of the developer's control. These categories have subcategories: the first has six, and the second four. Each subsection has from one to six checks, with a link to instructions (if relevant).

The following list contains the checks in the checklist. It has two sub-lists, "Checks" and "Checks that might be out of developer's control," and in each of them, the subcategories and their checks. I have left out the links to the accompanying material to save space.

The checks are:

- Automated Tools

    - Ran Accessibility Scanner for the feature
- Images and Other Visual Elements
    - Purely decorative images have content description set to null
- Alternative ways of navigation
    - The feature works with keyboard and D-pad
    - The feature works with Switch Access
    - The feature works with Voice control
    - The feature works with a screen reader
    - Focus order matches visual representation
    - Added accessibility actions for gesture based actions (e.g. swipe or drag and drop), or there is another way to interact than the gestures (e.g. buttons)
- Orientation
    - Experience is similar for landscape and portrait modes
- Semantics and Structure
    - Semantics (e.g. headings) are identified
    - Form elements (such as text fields, radio inputs and switches) have a text label
    - Custom components have correct semantics, state information and interaction patterns
- Magnification
    - The feature works when using magnification
    - The feature works with font size set to the biggest size

And the checks that might be out of developer's control are:

- Images and Other Visual Elements
    - Every visual element that conveys meaning has a descriptive text alternative
- Use of Color

    - The feature works even if user can't see colors
- Orientation
    - Orientation is not locked to either mode
- Audio and Video
    - Audio and video clips have captions

Completing these checks and fixing the issues they reveal can serve different users. This group includes users with permanent disabilities and impairments, and non-disabled users. The issues that the Accessibility Scanner can find are related to problems people with vision impairments might encounter. Some findings can be issues for people with physical disabilities that affect movement and coordination. In addition, orientation-related issues affect both of these groups – users might mount their phone on landscape mode on, for example, their wheelchair. Some low-vision users might use landscape mode to zoom in to have more width for the screen than what is available with the portrait mode. Magnification-related checks benefit low-vision users, and semantics and structure-related checks users using built-in assistive technologies that utilize the Accessibility API under the hood. Checks for images and visual elements, as well as for the use of color, benefit people who cannot see the images and who cannot see or distinguish colors. Finally, the checks for alternative navigation benefit many different user groups: Keyboard users, Switch Access users, screen reader users, and Voice Access users, to name a few.

The next chapter discusses evaluation of the artifact in each of the iterations.

5 Evaluation

This chapter discusses the evaluation of the artifact created in the research. As discussed in 3.1.3, the artifact's evaluation in this research context consists of three evaluation activities. It leaves out the fourth evaluation activity described in Sonnenberg and vom Brocke (2012).

In the first subsection, we will discuss the first evaluation activity—evaluating the problem statement. The second subsection discusses evaluating the design specifications, and the third subsection evaluates the checklist (so, the artifact) for proof of applicability. Finally, the last subsection of this chapter concludes the evaluation.

5.1 Evaluation of the Problem Statement

The first evaluation activity was conducted in conjunction with the literature review to justify the problem statement and the need for a solution. The problem statement was that Android applications are not accessible. A hypothesis for why that is included the assumption that Android developers don’t know much about accessibility and thus are not creating accessible applications. The design objective is to create an artifact that would help developers develop more accessible Android applications. The solution is a checklist with additional learning material.

The evaluation method for this hypothesis and problem statement was a literature review. A more extensive description of the literature review can be found in Chapter 2. The literature review illustrated that, indeed, Android applications are not as accessible as they can be. For example, the analysis of 50 Android applications revealed that these applications had only 63% of the accessibility guidelines implemented at best (Di Gregorio et al., 2022). Other studies have similar findings; Acosta-Vargas et al. (Acosta-

Vargas, Zarate-Estrella, et al., 2021) reviewed 50 apps with a tool called Accessibility Scanner, and only one of the reviewed apps did not have any accessibility problems. Chen et al. (2022) carried out a large investigation of Android applications and found that 88,99% of the apps had accessibility issues. Yan and Ramachandran (2019) analyzed 479 Android apps for accessibility issues, and their results showed that 94,8% of the apps contained violations of accessibility guidelines, and 97,5% contained potential violations.

Furthermore, as de Almeida and Gama (2021, p. 127) describe in their study, 24% of developers had never been in contact with accessibility as a topic, and 64% do not know the WCAG guidelines for developers. Di Gregorio et al. (2022) also concluded that mobile developers do not tend to implement accessibility guidelines in their apps – even if they have a chance to do so. (Di Gregorio et al., 2022, p. 145)

Prior research has proposed two types of solutions to the challenges Android developers face with developing accessible applications: easy-to-use tools and education. Vendome et al. (2019, p. 50) propose that developers would benefit from detailed guidance from tools identifying accessibility issues. They also recommend (semi-)automated tools for testing accessibility features to help developers gain accessibility knowledge. (Vendome et al., 2019, p. 50) Patel et al. (2020, p. 6) suggest creating useful resources for technology professionals. Particularly, they propose that there should be quick and targeted tutorials addressing accessibility in specific technical implementations and include modular tools that could be easily applied to development. (Patel et al., 2020, p. 6)

These findings from the literature review support and justify the problem – Android applications are not always accessible. Furthermore, they support the hypothesis that one reason Android apps are not accessible is the lack of knowledge of accessibility on Android developers' end. Finally, they side with the artifact being an easy-to-use tool that could be integrated into development processes, and that it should provide education in the targeted, specific tutorials.

5.2 Evaluation of the Design Specification

The second evaluation activity was conducted after creating the initial checklist based on the literature review's accessibility guidelines. The checklist acted as the input to this activity, a design specification with initial principles of form and function. This evaluation activity aimed to assess its usefulness and fit correspond to the stated design objectives. (Sonnenberg & vom Brocke, 2012, p. 17) The evaluation method was interviews with Android developers, as described in 4.2.1.

The interviewed developers agreed that this kind of checklist would be useful. While some of the proposed checks were out of their control, they felt that most were useful. The structure of the checks was seen as clear. Also, the need for more instructions accompanying the checklist was brought up, which justified the idea of adding separate educational pages for the checks.

Furthermore, as Sonnenberg and vom Brocke (2012, p. 17) suggest, the understandability and meaningfulness of the stakeholders should be assessed. One of the interview questions was whether the checks were understandable. All the participants answered that they were, some adding that to complete the checks, they'd need more information and instructions.

5.3 Evaluation of the Checklist

The third evaluation activity for this research was performed after the checklist had been constructed to its final form as a website. As Sonnenberg and vom Brocke (2012, p. 17)

define, this evaluation activity serves to demonstrate whether and how well the artifact performs when it touches organizational elements.

The prototype, the website with the checklist, and the accompanying educational material were the input for this validation activity. The method for evaluating the artifact was a survey about the checklist for Android developers. The survey had a question about the checklist being useful for the respondent. The answers to this question were almost unanimous – only one respondent answered no. The other answers varied from "Yes" to more elaborate, pointing out that it's a comprehensive tool and a reminder to test accessibility. One of the respondents also mentioned, that the checklist would be useful not just for them, but their whole team of developers.

Also, another respondent answered:

> (5) "I think it can be useful as a reminder to test as many things as possible, and to explain why something is important to consider. … So it's good to be reminded of things I may not be aware of until I become more competent in the area of accessibility in general."

In the example (5) the respondent underlines the importance of being reminded of things to consider, and that the checklist could act as a reminder to remember to test many aspects of accessibility. Vendome et al. (2019, pp. 49–50) mentioned screen-reader accessibility being only, or mainly, considered as one of the challenges for developers creating more accessible apps; the checklist can act as a reminder to test other aspects as well.

These survey responses demonstrate that the checklist is useful for real users and, thus, applicable to real users. The next step would be to evaluate it in a real setting—but that is not in the scope of this research, as described in Chapter 1. The objective of the solution for the thesis was defined so that the artifact should be useful for creating accessible Android applications. As described in the previous sections of this chapter, the usefulness of the application has been evaluated throughout the development process.

As Hevner et al. (2004, p. 85) describe, “A design artifact is complete and effective when it satisfies the requirements and constraints of the problem it was meant to solve”. While this solution probably does not solve all Android applications’ accessibility issues, it can help developers create more accessible applications than are currently available.

Hevner et al. (2004, p. 87) also discuss the research contributions in their article and that design-science research needs to provide contributions in the area of the design artifact. The next chapter will discuss these contributions, limitations of the study, and possible future research.

6 Discussion

This chapter discusses the research findings, its contributions to practitioners and knowledge base. After that, the limitations of the study and future research opportunities are discussed.

6.1 Contributions to Practitioners

The goal of this research was to understand how Android developers could develop more accessible apps. The research also aimed to understand the challenges of Android developers not developing accessible apps. As a solution to these problems, an accessibility checklist was built using the principles of the design research method.

The accessibility checklist with accompanying instructions and learning materials provide a tool for Android developers to develop more accessible apps than they're currently developing. Prior research also suggests that easy-to-use tools are part of the solution for creating more accessible apps. (Patel et al., 2020, p. 6; Vendome et al., 2019, pp. 49–50)

Patel et al. (2020, p. 6) also suggest that learning about addressing accessibility issues during development would be beneficial. The accessibility checklist serves as an educational tool as well. One of the interviewees mentioned that they would use the checklist during the development to ensure they would consider everything. Furthermore, the instructions and other educational materials enable learning during the development.

Android developers tend to have little prior knowledge of building accessible applications. They tend to lack awareness and do not know about accessibility

requirements. Furthermore, companies they work for tend not to prioritize accessibility, so it is not a requirement to learn about it. (Alshayban et al., 2020, p. 1130; de Almeida & Gama, 2021, p. 128; Di Gregorio et al., 2022, pp. 30–31; Patel et al., 2020, p. 4)

The interviews and the questionnaire for Android developers suggest that to help developers develop more accessible apps, a straightforward checklist that could be integrated into their development processes could be one solution. Furthermore, instructions and explaining why the individual checks need to be done were considered helpful.

Other possible components could be ways to interact with people with disabilities and ways to understand their needs and issues they face better, as suggested by Patel et al. (2020, p. 5).

6.2 Contributions to Knowledge Base

To my knowledge, there are no similar checklists available for Android developers. There are multiple accessibility checklists for web development, and there are some checklists for mobile web development. However, I have never seen one that is purely for Android development. Some websites contain instructions for testing different aspects of accessibility, but the information is scattered and is not that easy to use and actionable for a developer.

The checklist, so, the artifact, contributes to the knowledge base as the design artifact as defined by Hevner et al. (2004, p. 87). As Gregor and Hevner (2013, p. 346) describe, the artifact as improvements may be in the form of quality measures, so, in this case, accessibility of the Android applications.

6.3 Limitations of the Study

As the study was part of a master's thesis, the artifact was not evaluated in use as described in 3.3. Thus, it has yet to be evaluated in full organizational settings. While the artifact and its usefulness have been evaluated artificially, a need for the final evaluation would be needed to fully conclude the usefulness and validity. So, to put it into other words, this study concludes a proof of concept, and proof of validity is yet to be evaluated.

The questionnaire's sharing methods included me, the author, sharing the link to my LinkedIn networks and online communities where I am part. This limits the possible respondents to those circles, depending on my network and their networks on LinkedIn and the people who happen to be part of the communities the link was shared with.

As an accessibility professional, my networks tend to consist of people more interested in accessibility than an average developer. Thus, it is possible that the respondents are, on average, more interested in accessibility and tend to see the checklist as more useful and important than the others. This is also in line with, for example, Saleh and Bista's (2017, p. 70) findings that respondents tend to answer surveys more likely if they are interested in the topic.

6.4 Suggestions for Future Research

As this thesis did not include the fourth evaluation action described in Sonnenberg and vom Brocke (2012), a clear recommendation for future research would be to test the artifact in a real organizational environment to evaluate the proof of validity. Some possible metrics to measure could be if the application's accessibility increases after using the constructed checklist and the developers' confidence in their accessibility skills

after using the artifact in development processes. Possible research questions could include, for example, the following:

- Does the artifact usage improve the accessibility of the developed Android application?
- In which way can the accessibility checklist be improved?

It is also good to note that, as described in 6.3, it is possible that the respondents of the questionnaire are more interested in accessibility than an average developer and might be more susceptible to answering questionnaires about accessibility. Researching a wider group of developers to confirm the findings would be good. So, when conducting further research, attention should be paid to recruiting the participants to ensure that they include developers who are unfamiliar or not that interested in accessibility.

References

Abraham, R., Aier, S., & Winter, R. (2014). Fail Early, Fail Often: Towards Coherent Feedback Loops in Design Science Research Evaluation. *Proceedings of the 35th International Conference on Information Systems*.

Acosta-Vargas, P., Salvador-Acosta, B., Salvador-Ullauri, L., Villegas-Ch., W., & Gonzalez, M. (2021). Accessibility in Native Mobile Applications for Users with Disabilities: A Scoping Review. *Applied Sciences*, *11*(12), 5707. https://doi.org/10.3390/app11125707

Acosta-Vargas, P., Zarate-Estrella, S., Mantilla-Vaca, F., Novillo-Villegas, S., Chimbo, C., & Luján-Mora, S. (2021). Towards Accessibility and Inclusion of Native Mobile Applications Available for Ecuador in Google Play Store. *Sustainability*, *13*(20), Article 20. https://doi.org/10.3390/su132011237

Alajarmeh, N. (2022). The extent of mobile accessibility coverage in WCAG 2.1: Sufficiency of success criteria and appropriateness of relevant conformance levels pertaining to accessibility problems encountered by users who are visually impaired. *Universal Access in the Information Society*, *21*(2), 507–532. https://doi.org/10.1007/s10209-020-00785-w

Alshayban, A., Ahmed, I., & Malek, S. (2020). Accessibility issues in Android apps: State of affairs, sentiments, and ways forward. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 1323–1334. https://doi.org/10.1145/3377811.3380392

Android Developers. (2022). *Build accessible apps | Android Developers*. Retrieved 2024-01-05 from https://developer.android.com/guide/topics/ui/accessibility

Appt.org. (2021). *Accessibility code samples for mobile apps*. Retrieved 2024-02-20 from https://github.com/appt-org/accessibility-code-examples

Ballantyne, M., Jha, A., Jacobsen, A., Hawker, J. S., & El-Glaly, Y. N. (2018). Study of Accessibility Guidelines of Mobile Applications. *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia*, 305–315. https://doi.org/10.1145/3282894.3282921

Bhatia, J. S., P D, P., Tiwari, S., Nagpal, D., & Joshi, S. (2023). Integrating Accessibility in a Mobile App Development Course. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 1021–1027. https://doi.org/10.1145/3545945.3569825

Campbell, A., Adams, C., Bradley Montgomery, R., Cooper, M., & Kirkpatrick, A. (2023). *Web Content Accessibility Guidelines (WCAG) 2.2*. Retrieved 2024-03-03 from https://www.w3.org/TR/WCAG22/

Chen, S., Chen, C., Fan, L., Fan, M., Zhan, X., & Liu, Y. (2022). Accessible or Not? An Empirical Investigation of Android App Accessibility. *IEEE Transactions on Software Engineering*, *48*(10), 3954–3968. https://doi.org/10.1109/TSE.2021.3108162

Da Silva, H. N., Endo, A. T., Eler, M. M., Vergilio, S. R., & Durelli, V. H. S. (2020). On the Relation between Code Elements and Accessibility Issues in Android Apps. *Proceedings of the 5th Brazilian Symposium on Systematic and Automated Software Testing*, 40–49. https://doi.org/10.1145/3425174.3425209

de Almeida, V. L., & Gama, K. (2021). Mobile Accessibility Guidelines Adoption under the Perspective of Developers and Designers. *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 127–128. https://doi.org/10.1109/CHASE52884.2021.00028

Desmond, D., Layton, N., Bentley, J., Boot, F. H., Borg, J., Dhungana, B. M., Gallagher, P., Gitlow, L., Gowran, R. J., Groce, N., Mavrou, K., Mackeogh, T., McDonald, R., Pettersson, C., & Scherer, M. J. (2018). Assistive technology and people: A position paper from the first global research, innovation and education on assistive technology (GREAT) summit. *Disability and Rehabilitation: Assistive Technology*, *13*(5), 437–444. https://doi.org/10.1080/17483107.2018.1471169

Di Gregorio, M., Di Nucci, D., Palomba, F., & Vitiello, G. (2022). The making of accessible Android applications: An empirical study on the state of the practice. *Empirical Software Engineering*, *27*(6), 145. https://doi.org/10.1007/s10664-022-10182-x

Díaz-Bossini, J.-M., & Moreno, L. (2014). Accessibility to Mobile Interfaces for Older People. *Procedia Computer Science*, *27*, 57–66. https://doi.org/10.1016/j.procs.2014.02.008

Eler, M. M., Orlandin, L., & Oliveira, A. D. A. (2019). Do Android app users care about accessibility? An analysis of user reviews on the Google play store. *Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*, 1–11. https://doi.org/10.1145/3357155.3358477

European Parliament (2016). Directive 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the Accessibility of the Websites and Mobile Applications of Public Sector Bodies (Text with EEA Relevance), CONSIL, EP, 327 OJ L (2016). Retrieved 2024-01-05 from http://data.europa.eu/eli/dir/2016/2102/oj/eng

European Parliament. (2019). Directive 2019/882 of the European Parliament and of the Council of 17 April 2019 on the accessibility requirements for products and services (Text with EEA relevance). *Official Journal of the European Union.* Retrieved 2024-01-05 from http://data.europa.eu/eli/dir/2019/882/oj/eng

Fan, D., Breslin, D., Callahan, J. L., & Iszatt-White, M. (2022). Advancing literature review methodology through rigour, generativity, scope and transparency. *International Journal of Management Reviews*, *24*(2), 171–180. https://doi.org/10.1111/ijmr.12291

Google Help. (2024). *Accessibility Scanner—Android Accessibility Help*. Retrieved 2024-02-02 from https://support.google.com/accessibility/android/faq/6376582?hl=en&sjid=15077242903809662356-EU

Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly*, *37*(2), 337-A6.

Gregor, S., Kruse, L., & Seidel, S. (2020). Research Perspectives: The Anatomy of a Design Principle. *Journal of the Association for Information Systems*, *21*, 1622–1652. https://doi.org/10.17705/1jais.00649

Hevner, A. R. (2007). A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, *19*(2).

Hevner, A. R., March, S., Park, J., & Sudha, R. (2004). Design Science in Information Systems Research. *Management Information Systems Quarterly*, *28*(1), 75–105.

Hirsjärvi, S., & Hurme, H. (2022). *Tutkimushaastattelu: Teemahaastattelun teoria ja käytäntö* (2nd ed.). Gaudeamus.

Hirsjärvi, S., Remes, P., Sajavaara, P., & Sinivuori, E. (2009). *Tutki ja kirjoita* (15. uud. p). Tammi.

Hyvärinen, M., Suoninen, E., & Vuori, J. (n.d.). Haastattelut. In J. Vuori, *Laadullisen tutkimuksen verkkokäsikirja*. Tampere: Yhteiskuntatieteellinen tietoarkisto. Retrieved 2023-06-19 from https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus

Khasawneh, A., Gallagher, P. B., Jacobson, E. I., & Riley, L. (2023). Enhancing Ada Compliance for Websites and Online Applications: A Heuristic Evaluation Approach. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *67*(1), 1457–1463. https://doi.org/10.1177/21695067231192722

Kirkpatrick, A., O Connor, J., Campbell, A., & Cooper, M. (2023). *Web Content Accessibility Guidelines (WCAG) 2.1*. Retrieved 2023-06-19 from https://www.w3.org/TR/WCAG21/

Laamanen, M., Ladonlahti, T., Puupponen, H., & Kärkkäinen, T. (2022). Does the law matter? An empirical study on the accessibility of Finnish higher education institutions' web pages. *Universal Access in the Information Society*. https://doi.org/10.1007/s10209-022-00931-6

Lee, H., Park, J., & Lee, U. (2022). A Systematic Survey on Android API Usage for Data-driven Analytics with Smartphones. *ACM Computing Surveys*, *55*(5), 104:1-104:38. https://doi.org/10.1145/3530814

Leite, M. V. R., Scatalon, L. P., Freire, A. P., & Eler, M. M. (2021). Accessibility in the mobile development industry in Brazil: Awareness, knowledge, adoption, motivations and barriers. *Journal of Systems and Software*, *177*, 110942. https://doi.org/10.1016/j.jss.2021.110942

Madeira, S., Branco, F., Gonçalves, R., Au-Yong-Oliveira, M., Moreira, F., & Martins, J. (2021). Accessibility of mobile applications for tourism—Is equal access a reality? *Universal Access in the Information Society*, *20*(3), 555–571. https://doi.org/10.1007/s10209-020-00770-3

Maedche, A., Gregor, S., Morana, S., & Feine, J. (2019). Conceptualization of the Problem Space in Design Science Research. In B. Tulu, S. Djamasbi, & G. Leroy (Eds.), *Extending the Boundaries of Design Science Theory and Practice* (Vol. 11491, pp. 18–31). Springer International Publishing. https://doi.org/10.1007/978-3-030-19504-5_2

Patch, K., Spellman, J., & Wahlbin, K. (2015). *Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile*. Retrieved 2023-04-29 from https://www.w3.org/TR/mobile-accessibility-mapping/

Patel, R., Breton, P., Baker, C. M., El-Glaly, Y. N., & Shinohara, K. (2020). Why Software is Not Accessible: Technology Professionals' Perspectives and Challenges. *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–9. https://doi.org/10.1145/3334480.3383103

Peffers, K., Tuunanen, T., & Niehaves, B. (2018). Design science research genres: Introduction to the special issue on exemplars and criteria for applicable design science research. *European Journal of Information Systems*, *27*(2), 129–139. https://doi.org/10.1080/0960085X.2018.1458066

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Persson, H., Åhman, H., Yngling, A. A., & Gulliksen, J. (2015). Universal design, inclusive design, accessible design, design for all: Different concepts—one goal? On the concept of accessibility—historical, methodological and philosophical aspects. *Universal Access in the Information Society*, *14*(4), 505–526. https://doi.org/10.1007/s10209-014-0358-z

Pinheiro, M., Viana, W., & De Gois Ribeiro Darin, T. (2023). Why Should Red and Green Never Be Seen? Exploring Color Blindness Simulations as Tools to Create Chromatically Accessible Games. *Proceedings of the ACM on Human-Computer Interaction*, *7*(CHI PLAY), 165–196. https://doi.org/10.1145/3611026

Puusa, A., Juuti, P., & Aaltio, I. (Eds.). (2020). *Laadullisen tutkimuksen näkökulmat ja menetelmät*. Gaudeamus.

Rehabilitation Act of 1973, 29 U.S.C. § 798 (2017).

Ross, A. S., Zhang, X., Fogarty, J., & Wobbrock, J. O. (2020). An Epidemiology-inspired Large-scale Analysis of Android App Accessibility. *ACM Transactions on Accessible Computing*, *13*(1), 4:1-4:36. https://doi.org/10.1145/3348797

Rygg, M., Rømen, D., & Sterri, B. R. (2016). Norway's ICT Accessibility Legislation, Methods and Indicators. *Studies in Health Technology and Informatics*, *229*, 471–481.

Saleh, A., & Bista, K. (2017). Examining Factors Impacting Online Survey Response Rates in Educational Research: Perceptions of Graduate Students. *Journal of MultiDisciplinary Evaluation*, *13*(29), 63–74.

Sandnes, F. E. (2022). Is there an imbalance in the supply and demand for universal accessibility knowledge? Twenty years of UAIS papers viewed through the lens of WCAG. *Universal Access in the Information Society*, *21*(2), 333–349. https://doi.org/10.1007/s10209-021-00834-y

Sonnenberg, C., & vom Brocke, J. (2012). Evaluations in the Science of the Artificial – Reconsidering the Build-Evaluate Pattern in Design Science Research. In K. Peffers, M. Rothenberger, & B. Kuechler (Eds.), *Design Science Research in Information Systems. Advances in Theory and Practice* (Vol. 7286, pp. 381–397). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-29863-9_28

Thuan, N. H., Drechsler, A., & Antunes, P. (2019). Construction of Design Science Research Questions. *Communications of the Association for Information Systems*, *44*, 20. https://doi.org/10.17705/1CAIS.04420

Tuomi, J., & Sarajärvi, A. (2017). *Laadullinen tutkimus ja sisällönanalyysi*. Tammi.

United Nations. (2006). *Convention on the Rights of Persons with Disabilities*. OHCHR. Retrieved 2024-02-04 from https://www.ohchr.org/en/instruments-mechanisms/instruments/convention-rights-persons-disabilities

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: A Framework for Evaluation in Design Science Research. *European Journal of Information Systems*, *25*(1), 77–89. https://doi.org/10.1057/ejis.2014.36

Vendome, C., Solano, D., Liñán, S., & Linares-Vásquez, M. (2019). Can Everyone use my app? An Empirical Study on Accessibility in Android Apps. *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 41–52. https://doi.org/10.1109/ICSME.2019.00014

Vilkka, H. (2023). *Kirjallisuuskatsaus metodina, opinnäytetyön osana ja tekstilajina*. Art House.

vom Brocke, J., Hevner, A., & Maedche, A. (2020). Introduction to Design Science Research. In *Design Science Research. Cases* (pp. 1–13). Springer International Publishing. https://doi.org/10.1007/978-3-030-46781-4_1

vom Brocke, J., Winter, R., Hevner, A., & Maedche, A. (2020). Special Issue Editorial – Accumulation and Evolution of Design Knowledge in Design Science Research: A Journey Through Time and Space. *Journal of the Association for Information Systems*, *21*(3), 520–544. https://doi.org/10.17705/1jais.00611

WebAIM. (2021). *WebAIM: Screen Reader User Survey #9 Results*. Screen Reader User Survey #9. Retrieved 2024-01-03 from https://webaim.org/projects/screenreadersurvey9/

WebAIM. (2023). *WebAIM: Screen Reader User Survey #10 Results*. Screen Reader User Survey #10. Retrieved 2024-02-23 from https://webaim.org/projects/screenreadersurvey10/#mobilescreenreaders

Winter, R. (2008). Design science research in Europe. *European Journal of Information Systems*, *17*(5), 470–475. https://doi.org/10.1057/ejis.2008.44

Yan, S., & Ramachandran, P. G. (2019). The Current Status of Accessibility in Mobile Apps. *ACM Transactions on Accessible Computing*, *12*(1), 3:1-3:31. https://doi.org/10.1145/3300176

Youngblood, S. A. (2013). Communicating Web Accessibility to the Novice Developer: From User Experience to Application. *Journal of Business and Technical Communication*, *27*(2), 209–232. https://doi.org/10.1177/1050651912458924

Zaina, L. A. M., Fortes, R. P. M., Casadei, V., Nozaki, L. S., & Paiva, D. M. B. (2022). Preventing accessibility barriers: Guidelines for using user interface design patterns in mobile applications. *Journal of Systems and Software*, *186*, 111213. https://doi.org/10.1016/j.jss.2021.111213

Zaphiris, P., Ghiawadwala, M., & Mughal, S. (2005). Age-centered research-based web design guidelines. *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, 1897–1900. https://doi.org/10.1145/1056808.1057050

Zhong, Y., Weber, A., Burkhardt, C., Weaver, P., & Bigham, J. P. (2015). Enhancing Android accessibility for users with hand tremor by reducing fine pointing and steady tapping. *Proceedings of the 12th International Web for All Conference*, 1–10. https://doi.org/10.1145/2745555.2747277

Appendices

Appendix 1. Accessibility Guidelines From Literature Review

Accessibility guidelines from literature review.

Source	Guideline	Code
Patch et al., 2015	Small screen size	WCAG-1
Patch et al., 2015	Zooming and magnification	WCAG-2
Patch et al., 2015	Keyboard control	WCAG-3
Patch et al., 2015	Touch target	WCAG-4
Patch et al., 2015	Touchscreen and device manipulation gestures	WCAG-5
Patch et al., 2015	Device orientation	WCAG-6
Patch et al., 2015	Grouping operable items	WCAG-7
Patch et al., 2015	Seting the virtual keyboard to the type of data entry	WCAG-8
Patch et al., 2015	Providing an easy method for data entry	WCAG-9
Patch et al., 2015	Supporting the charactheristics of the platform	WCAG-10
Android Developers, 2022	Labelling elements	ANDROID-1
Android Developers, 2022	Adding accessibility actions	ANDROID-2
Android Developers, 2022	Extending system widgets	ANDROID-3
Android Developers, 2022	Making media content more accessible	ANDROID-4
Díaz-Bossini and Moreno, 2014	Target design	DIAZ-1
Díaz-Bossini and Moreno, 2014	Use of Graphics	DIAZ-2

Source	Guideline	Code
Díaz-Bossini and Moreno, 2014	User cognitive design	DIAZ-3
Ballantyne et al., 2018	Text	BAL-1
Ballantyne et al., 2018	Audio	BAL-2
Ballantyne et al., 2018	Video	BAL-3
Ballantyne et al., 2018	UI Elements	BAL-4
Ballantyne et al., 2018	User Control	BAL-5
Ballantyne et al., 2018	Flexibility and efficiency	BAL-6
Ballantyne et al., 2018	Recognition rather than recall	BAL-7
Ballantyne et al., 2018	Gestures	BAL-8
Ballantyne et al., 2018	System visibility	BAL-9
Ballantyne et al., 2018	Error prevention	BAL-10
Ballantyne et al., 2018	Tangible interaction	BAL-11
Zaina et al., 2022	Icons	ZAI-1
Zaina et al., 2022	Inputs	ZAI-2
Di Gregorio et al., 2022	Audio and video	DIGR-1
Di Gregorio et al., 2022	Focus	DIGR-2
Di Gregorio et al., 2022	Forms	DIGR-3
Di Gregorio et al., 2022	Scripts and dynamic content	DIGR-4
Di Gregorio et al., 2022	Structure	DIGR-5

Appendix 2. Interview Questions

Background Info

- How long have you developed Android?
- How much experience do you have with accessibility? Please, describe your experience with accessibility.
- What do you think, that would help you to develop more accessible Android apps?

Accessibility Guidelines

- Let's look at the initial guidelines. How does the list look like for you?
- Are these checks understandable? What do you think about the language?
- Is the list too short? Too long?
- Would you need more info? What kind of things would be interesting? In what format?
- Is there anything else you'd like to add?

Appendix 3. Questionnaire Questions

Background

How many years of Android development experience do you have?

What is your level of accessibility knowledge? Can be with Android or generally.

Android Accessibility Checks

Can you see the site and the checklist being useful to you?

Is there anything missing that should be part of the checklist and/or site? Anything tha would make it more useful?