A First Look at Dark Mode in Real-World Android Apps

SUYU MA, Monash University, Australia
CHUNYANG CHEN*, Monash University, Australia
HOURIEH KHALAJZADEH, Monash University, Australia
JOHN GRUNDY, Monash University, Australia

Android apps often have a “dark mode" option used in low light situations, for those who find the conventional color palette problematic, or because of personal preferences. Typically developers add a dark mode option for their apps with different backgrounds, text, and sometimes iconic forms. We wanted to understand the actual provision of this dark mode in real-world Android apps through an empirical study of posts from Stack Overflow and real-world Android app analysis. Using these approaches, we identified the aspects of dark mode that developers implemented as well as the key difficulties they experienced in implementing it. We performed a quantitative analysis using open-coding of more than 300 discussion threads to create a taxonomy regarding the aspects discussed by developers with respect to dark mode in Android. Our quantitative analysis of over 6,000 Android apps highlights which dark mode features are typically provided in Android apps, and what aspects developers care about during dark mode design. We also examined four app development support tools to see how well they aid Android app development for dark mode. From our analysis, we distilled some key lessons to guide further research and actions in aiding developers with supporting users who require such assistive features. For example, developers should be aware of the potential risks in using unsuitable dark mode design schema and researchers should take dark mode features into consideration when developing app development support tools.

Additional Key Words and Phrases: Graphical User Interface, Dark Mode, Accessibility, Android

ACM Reference Format:
Suyu Ma, Chunyang Chen, Hourieh Khalajzadeh, and John Grundy. 2022. A First Look at Dark Mode in Real-World Android Apps. *ACM Trans. Softw. Eng. Methodol.* 1, 1 (June 2022), 27 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Mobile apps have now become the most popular way of accessing the Internet as well as performing daily tasks, e.g., reading, shopping, banking, and chatting [7, 19]. As one of the most important components of the app, its Graphical User Interface (GUI) provides a visual bridge between a software application and end-users through which they can interact with each other. A good GUI design makes an application easy, practical, and efficient to use, which significantly affects the success of the application and the loyalty of its users [34]. For example, computer users often view

*Corresponding author.

Authors' addresses: Suyu Ma, Monash University, Faculty of Information Technology, Melbourne, Australia, suyu.ma1@monash.edu; Chunyang Chen, Monash University, Faculty of Information Technology, Melbourne, Australia, chunyang.chen@monash.edu; Hourieh Khalajzadeh, Monash University, Faculty of Information Technology, Melbourne, Australia, hourieh.khalajzadeh@monash.edu; John Grundy, Monash University, Faculty of Information Technology, Melbourne, Australia, john.grundy@monash.edu.

1049-331X/2022/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

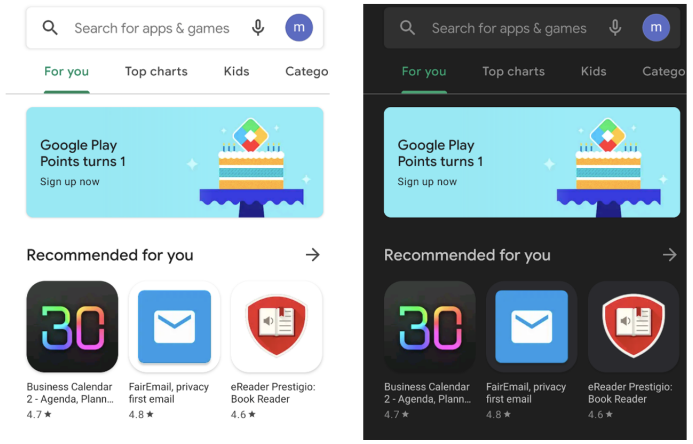

Fig. 1. Example of dark mode in Google Play

Apple's Macintosh system as having a better GUI than the Windows system, therefore their positive views are almost double that of Windows users, leading to 20% more brand loyalty [55].

Due to the wide usage of mobile apps in different scenarios, one single standard GUI cannot always cater to all users' needs and preferences all of the time. To make the GUI more user-friendly, the "dark mode" (also called dark theme) of an Android app GUI is often provided, in addition to the default "light mode". Unlike light mode, dark mode is a design scheme with a dark palette, including text and sometimes iconic differences to conventional light mode. For instance, the white background in light mode is converted into black in dark mode, as shown in Figure 1, which shows the screenshots of Google Play app in light and dark modes. The principle behind dark mode is that it reduces the light emitted by device screens while maintaining the minimum color contrast ratios required for readability. dark mode was first introduced to web development [2] and has recently become more common in mobile apps. For example, all system-default apps in iOS and Android began to support a dark mode in 2019 and [3, 17] and both of them encourage developers to design a dark mode theme for their apps.

A dark mode brings some advantages to mobile apps. First, dark mode can be used in low-light settings (e.g., night) for better readability and user experience, which may also lead to better sleep when using the phone near bed time [49]. Second, dark mode may reduce eye strain in low-light conditions, and make the app more accessible especially to users with light sensitivity or visual impairment [10]. Third, dark mode consumes less energy because pixels with black color in Organic Light-Emitting Diode (OLED) displays require no power. As the screen display typically uses most of the energy in the mobile devices [23], using dark mode makes the phone battery last longer [54]. In a recent informal survey[1], around 82.7% of participants stated that they use dark mode on their devices.

Awareness of the need to provide a dark mode option for apps has been growing. In 2019, Google and Apple have released developer design guidelines for dark mode [4, 17]. Despite these dark mode-focused efforts, there are no studies to investigate dark mode features in real-world Android apps. Furthermore, it is not clear to what extent developers utilize dark mode features in their apps. This suggests a need for an increasing dark mode awareness among researchers, developers of mobile platforms (e.g., Apple, Google), and developers of individual apps. To the best of our

[1] https://medium.com/dev-channel/let-there-be-darkness-maybe-9facd9c3023d

knowledge, this is the first work to investigate the development practices about dark mode in mobile apps, by answering questions including: What issues are there with developing dark mode? What features are there in apps with dark mode? How do Android development tools support dark mode? We answer these questions by mining real-world Q&A discussions, android apps.

Our analysis demonstrates good potential to aid developers with enhancing dark mode design of their apps. In detail, we aim to address this gap in research by providing a holistic view of dark mode in Android apps. We investigate dark mode features in Android apps, what aspects developers care about in dark mode design and the performance of app development support tools in light and dark mode. First, we analyze dark mode-related discussions from Stack Overflow to determine what aspects of dark mode developers care about. Next, we conduct a mining study based on Android apps collected from the Google Play store to investigate the extent to which dark mode features are present. We then analyze how well app development support tools aid dark mode vs light mode development. This work makes the following key contributions:

- Our work is the first to investigate the key dark mode features and issues in Android applications, and key difficulties faced by developers in the provision of a dark mode;
- We analyze aspects of dark mode that developers implemented as well as those they experienced difficulty through an investigation of posts from Stack Overflow;
- We report on the first large-scale dark mode analysis features in over 6,000 real-world Android apps across 33 different application categories;
- We investigate how well four app development support tools work on supporting light and dark mode app development;
- We discuss actionable implications and future work for developers and distill some key lessons to guide further research.

2 BACKGROUND

Dark mode, which is also called dark theme, black mode, or night mode, is a UI design scheme that commonly uses light colors in text, icons, and graphical components but dark colors in the background [6]. These dark modes are supported by many modern websites and operating systems, such as IOS and Android, for better user interface support. Predecessors of modern computers drew graphs and show text on a black background. With the appearance of teletext, researchers investigated which color combinations worked best for users. Cyan or yellow on black was widely used as the optimal option [9]. However, the dark-on-light scheme became the norm with the rise of WYSIWYG (What You See Is What You Get) word processors which is a computer program that supports editing and formatting text to simulate ink on paper [6].

However, these dark on light approaches produce very high contrast displays; potentially disrupt sleep patterns; may cause eye strain in low light conditions; and some users simply prefer more light on dark-oriented displays for their web sites and apps [14]. Thus in recent times, dark mode options have been added to many websites and smartphones. For example, dark mode options were supported in Firefox[2] and Chromium[3]. In 2019, Apple announced that all native applications in iOS 13 and iPadOS would support dark mode and third-party developers could implement special dark mode on their applications [17]. Designer Sylvain Boyer added dark mode into the interface of smartphones with OLED screens [8]. To provide users a dark mode choice, a "prefers-color-scheme" was created as CSS property [6].

In the Android platform, dark mode is officially supported in APIs which are higher than level 29 (Android 10) [3]. To enable dark mode in the Android system, users can either change the display

[2]https://support.mozilla.org/en-US/kb/using-dark-theme-firefox-android
[3]https://support.google.com/chrome/answer/9275525?hl=enco=GENIE.Platform%3DAndroid

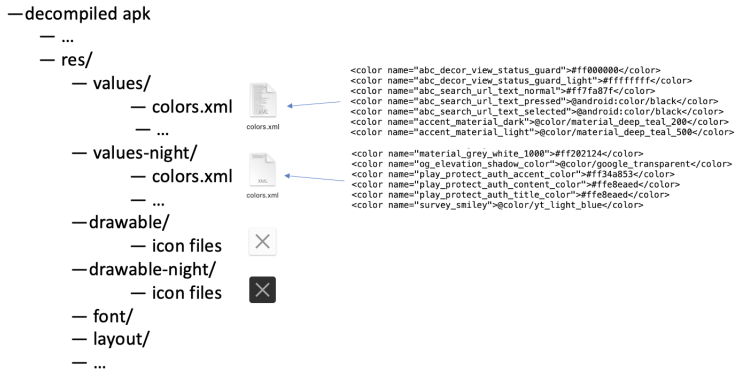

Fig. 2. Example for dark mode development

theme option in the system setting or switch themes from the notification tray. Selecting the battery saver mode also automatically enables dark mode in Pixel devices [1]. dark mode implementation process for Android applications is as follows. First, to enable dark mode in Android applications, developers should set the app themes to inherit a DayNight theme to support dark mode:

```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
```

Via inheritance, applications have a default dark mode and dark mode option is controlled by a system setting. Second, to avoid hard-coded colors or icons that are designed for light mode, developers usually create dark mode resources. An example of dark mode resources is shown in Figure 2. "values" and "drawable" folders contain light mode resources. Dark mode resources are stored in "values-night" and "drawable-night" folders with the same filenames. "values-night" folder contains color resources named "colors.xml". "drawable-night" folder contains dark mode icon resources. The icon files are usually png or VectorDrawable files. A VectorDrawable is an XML file that represents a vector graphic. A VectorDrawable contains points, lines and curves, and color information. Unlike normal image files, VectorDrawable files can be scaled into different sizes without losing display quality. This advantage makes it suitable for APK files [11].

Implementing a dark mode is not just a matter of changing the color palette. Care needs to be taken that the colors in dark mode work under low light and contrast is not lost. Similarly, iconic elements need to be translated into a dark mode form which often requires some redesign of the element and regeneration of an image format. Finally, resizing some components may be necessary to get an aesthetically pleasing result.

3 STUDY DESIGN

In this study we aim to find out (i) how developers discuss building dark mode for Android apps, including key difficulties and challenges they mention that they face in doing this; (ii) how dark mode is implemented in real-world Android apps; and (iii) how dark mode is supported in current Android app development support tools. To this end, our study consists of three parts: a qualitative study of Stack Overflow posts, a quantitative study of real-world Android applications, and a quantitative study of app development support tools.

3.1 A quantitative study of Stack Overflow posts

Motivation: The goal of this study is to identify the common issues that developers encounter when implementing dark mode Android applications. To achieve this goal, we analyzed developers'

discussions on Stack Overflow, which is the most popular Q&A website in the software engineering area. We investigated the following RQ:

RQ1: What dark mode aspects are discussed by developers in Stack Overflow?

Answering this RQ helps developers understand dark mode features and the common challenges and solutions for them in dark mode implementation.

Approach: We identified posts from Stack Overflow related to dark mode features and concepts. We collected the posts (i.e., questions and answers) from the Stack Exchange Data Explorer website[4]. SQL queries can be used on this website to search and download posts. Retrieved posts contain information such as questions, answers, number of received votes, and view count of a post.

Stack Overflow requires a question to be labeled with one to five tags before it is asked by developers. By analyzing the tags, we can identify the topic of a question. Therefore, to select dark mode related posts, we searched for posts whose tags or title contain keywords “dark” or “night”. Note that substrings matching is supported in Stack Exchange Data Explorer, which means keywords like dark could match with "dark-mode" or "dark-theme". We limited the query to retrieve posts with creation time before December 2022. In addition, we focused only on Android-related posts, i.e., we select the posts with the tag "android". Based on the above-mentioned criteria, we collected 618 dark mode-related posts. We filter out posts with 0 votes, because low votes imply that few in the community have faced similar problems. Some of the keywords used in the query are prone to false positives (e.g., the term dark pattern is misleading). The first and another experienced Android mobile app developer went through the discussions and discarded the ones not related to dark mode features/concepts in Android. After filtering out the false positives, the number of posts was reduced to 324. Note that we delete all the code snippets because code snippets often contain meaningless information for our analysis. By analyzing these posts, we can understand the challenges and best practices in implementing dark mode, which contribute to the development of better support tools and frameworks.

We first analyzed the posts by building a word cloud of these dark mode posts to understand what the posts are about. We applied standard text processing steps to posts such as removing punctuation, lower-casing all characters, and excluding stop words. We counted the word frequencies and we display the top 50 most frequent words in a word cloud [51]. The size of a word shows word frequency in our dataset i.e., the larger words with higher frequency while smaller words with lower frequency.

We then adopted the Latent Dirichlet Allocation (LDA) [21] model to extract common types. LDA is a statistical model for discovering abstract topics that occur in a collection of documents in which each topic consists of a set of keywords. Compared to other NLP and IR topic modelling techniques, LDA is unsupervised and easy to train. It can automatically distill interpretable topics from documents. A significant limitation of LDA is that it considers only single words (i.e., unigrams). However, a single word may not capture the exact semantics of the post-edit comments. In contrast, phrases that are composed of several words are more intuitive to understand the intention behind post edits, such as “dark mode” instead of “dark”. Therefore, these multi-word phrases must be recognized and treated as a whole in the LDA model. We adopted a simple data-driven and memory-efficient approach [41] to detect multi-word phrases in posts. In this approach, phrases are formed iteratively based on the unigram and bigram counts, using the following formula:

$score(w_i, w_{i+1}) = \frac{count(w_i, w_{i+1}) - \theta}{count(w_i) \times count(w_{i+1})} \times N.$

The w_i and w_{i+1} are two consecutive words. θ is a discounting coefficient to prevent infrequent bigrams to be formed. That is, the two consecutive words will not form a bigram phrase if they appear as a phrase less than θ times in the corpus. N is the vocabulary size of the corpus. In this

[4]https://data.stackexchange.com/stackoverflow/query/new

Fig. 3. Workflow of real-world Android applications analysis

work, we experimentally set θ as 10 and the threshold for a score as 10 to achieve a good balance between the coverage and accuracy of the detected multi-word phrases.

Finally, we conducted a quantitative study. We studied each post based on the question itself and the associated answers to identify the challenges that developers have when implementing dark mode. Based on the topics generated by the LDA model, we build a taxonomy to further identify dark mode-related issues discussed by developers in Stack Overflow. The manual classification is inspired by eight topics from the LDA model, but the classification results are purely based on the post content. The taxonomy was built by the first and another experienced Android developer based on an open-coding process.

3.2 A quantitative study of dark mode in real-world Android applications

Motivation: To further investigate dark mode features and issues in Android applications, we conducted a detailed empirical study on real-world Android applications from the Google Play store[5]. We analyzed dark mode-related resources in Android applications. With the dataset from real-world Android apps, we aimed to conduct an empirical study to reveal fine-grained insights. We investigated the following RQ:

RQ2: What are dark mode features found in real-world Android applications?

Answering this RQ helps developers understand how real-world Android apps make use of dark mode and the common issues in this process. The findings also have far-reaching implications in software engineering.

Approach: As shown in Figure 3, we developed a pipeline that can be used to collect real-world Android apps, extract dark mode resources and accessibility issues from these apps. We selected 6,600 apps in the form of Android Package (APK) from the Google Play Store, which is an official app store for certified devices running on the Android operating system. We chose the top 200 apps from 33 different categories, such as music, health, finance, and productivity, etc. Our data collection process consisted of the following steps: 1) We used a Node.js module named *google-play-scraper*[6]

[5]https://play.google.com/store
[6]https://github.com/facundoolano/google-play-scraper

to scrape application data from the Google Play store, which includes meta-data including app ID, category, name of the developer, etc. For each category, we collected the top 200 app IDs ranked with rating scores. 2) Based on the collected app IDs, we downloaded the corresponding Android applications from the Google Play store. In this step, we used a tool named *PlaystoreDownloader*[7] which can download applications by specifying their package name.

To analyze the apps' dark mode-related resources, we decoded these apps with *apktool*[8], a tool for reverse engineering 3rd party apps. Each decoded app has resources with the structure in Figure 2. We further selected the apps with dark mode implementation. For each app with a dark mode, we analyzed its colors in light mode and dark mode to investigate the color conversion in dark mode design. To understand the design principles for dark mode icons, we analyzed the icons in these apps that are different in light mode and dark mode.

To collect the screenshots of apps, we used a tool named Xbot [13]. Note that Xbot is a tool that can simulate user interaction to collect accessibility issues, but we use it to collect screenshots for quantitative dark mode feature analysis. Xbot instruments the apk files to enable launching by other third-party components. Compared to other existing testing tools (i.e., Google Monkey) which can only achieve a low activity coverage, Xbot can achieve nearly 80% activity coverage [13]. To enable activity launching from other entries, Xbot instruments each apk by manipulating the Android Manifest file and repackaging it to a new one. Specifically, Xbot first decompiles the app, extracts each activity together with its required fields such as "action", and sets the *"exported=true"* to enable the launching process from other components. It extracts the required Activity Intent parameters for launching each activity. Xbot then repacks it to a new apk and signs it to ensure usability. To dynamically launch each activity, we installed the new repackaged apk on the Android emulator and attached the Intent parameters extracted by Xbot to the current activity. When it is launched successfully, it uses Accessibility Scanner[9] to take screenshots of each app page, collect the accessibility issues and the corresponding layout hierarchy. The Accessibility Scanner tool does not report any false positive results. Because it is a rule based tool which can precisely detect accessibility issues with confidence. For activities that need input to load the screen, the screen will crash. If such a crash happens, Xbot will stop the app and set it to the original state which allows another activity to launch. For each app, we ran Xbot with it twice in both light mode and dark mode settings.

3.3 A quantitative study of app development support tools working in dark mode

Motivation: There are many tools developed by the community to help facilitate the android app (especially GUI) development such as code generation, usability and accessibility testing [24, 37, 56]. Although they are well verified in the normal app setting i.e., light mode, it is rarely to explore their performance in dark mode. Therefore, we conduct experiments of four app GUI development support tools Accessibility Scanner, UIED, UI2code and Owleye in dark mode. These four tools are all related to Android GUI development and open released. For each tool, we check the outputs in both light and dark mode settings. With the results of tools in light mode and dark mode, we investigated the following RQ:

RQ3: How well do app development support tools aid dark mode vs light mode development?

Answering this RQ helps developers better understand the dark mode features supported by app development support tools and the common issues of these tools in dark mode setting.

[7]https://github.com/ClaudiuGeorgiu/PlaystoreDownloader
[8]https://ibotpeaches.github.io/Apktool/
[9]https://support.google.com/accessibility/android/answer/6376570

Approach: Based on the screenshots collected by Xbot, we detected consistency issues of different app development support tools when analysing light vs dark mode apps. For Accessibility Scanner, we directly analyzed the results of Xbot. With Xbot, we collected different types of accessibility issues for each app using Accessibility Scanner. We then analyzed if the collected issues are consistent in dark mode and light mode settings. For the other three tools, we randomly selected 50 apps from our collected real-world apps. We further randomly selected 4 light mode screenshots and 4 corresponding dark mode screenshots for the same activities in each app. For each of these app development support tools, we run the tool in collected light mode screenshots and dark mode screenshots, and compare the results. These four tools are introduced in detail as follows.

Accessibility Scanner is an Android development tool that can help developers find accessibility issues of Android apps. When running Android apps, Accessibility Scanner scans the screenshots and looks for usability issues during the current activities.

UIED [56] is a tool developed for detecting GUI elements. Traditional computer version techniques and deep learning models are both used in UIED to deal with complicated GUI elements. It contains two parts: text elements detection and non-text elements detection. Based on traditional computer version techniques, a top-down coarse-to-fine approach is used extract non-text elements. Text elements are extracted with deep learning model.

UI2code [24] is a tool that can translate a UI design image into a GUI skeleton. This tool uses Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) as encoder, uses another RNN as decoder. The CNN encoder extracts features from input image with convolution and pooling operations. Then, the spatial layout information of the image features are summarized by RNN encoder. Based on the summarized image features, the RNN decoder generate the GUI skeleton in token sequence representation.

OwlEye [37] is a tool that can automatically detect and localize UI display issues in the screenshots of the application. A CNN-based model is used in OwlEye to classify the display issue type of the input image. OwlEye further utilizes Grad CAM-based model to localize the issue position of the input UI screenshots to guide developers to fix the bug.

4 RESULTS

4.1 RQ1 – What dark mode aspects are discussed in Stack Overflow posts?

Figure 4 shows the word cloud for dark mode-related posts. According to our observation, these keywords align well with the Android application development such as UI design (e.g., “theme”, “color”, “bar"), Android programming (e.g., “activity”, “override”, “components”). Apart from these frequent types, other dark mode-specific keywords also emerge such as “dark”, “nightmode”, “dark-setting”. These keywords are highly related to dark mode implementation in the Android platform and they are more frequent than other keywords.

To further analyze the exact topics discussed by developers in Stack Overflow, we use the LDA model to extract 8 topics with corresponding keywords shown in Table 1. Note that we annotated the topic’s name with our summarization based on the topic keywords. Different from individual frequent keywords, these topics provide finer-grained information. These topics are all domain-specific to dark mode implementation. Some are related to GUI design (#1 Android widget, #2 app layout, and #3 color design), some to android implementation (#5 disabling dark mode, #6 developing dark mode, #7 errors in implementation), and others to dark mode implementation.

Figure 5 shows the manually classified question types. In total, we uncovered 10 types of questions that developers encounter, which include 8 basic types and 2 high-level types that are included

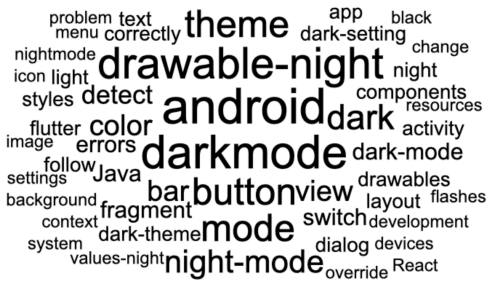

Fig. 4. Word cloud of dark-mode posts

Table 1. Topics of dark mode-related posts

ID	Topic Name	Keywords
1	Android widget	bar, text, button
2	app layout	Constrain Layout, Linear Layout, List View
3	color design	ARGB, alpha, RGB
4	dark system	dark device, dark system, system-wide
5	disabling dark mode	switch to light, turn off, close
6	developing dark mode	implementation, development, programming
7	errors in implementation	not working, not loading, exception
8	detecting dark mode	detect dark, check, test

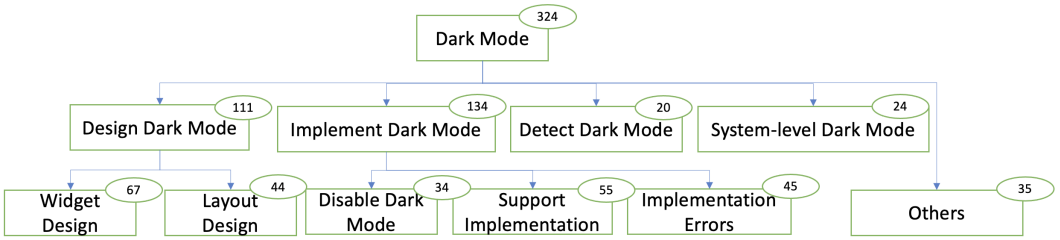

Fig. 5. Our manual classification of dark mode-related posts on Stack Overflow

from basic types. Below, we discuss these uncovered 8 basic types and analyze the root reasons in detail. Because of space limitations, we include qualitative examples for some categories.

Widget design in dark mode is the most prevalent issue that developers care about (67 posts). This is the top-1 discussed aspect, in particular, these posts are related to dark mode UI features that describe what is displayed in the app's GUI in dark mode. Developers often face difficulties related to widget design in dark mode. For example, the SO (Stack Overflow) question 6155888 asks about *designing button style in dark mode*. The user wants to design a button with rounded corners, and white colors when pressed. However, the system ignores the button background, the user asks for help with this problem. The SO question 66043540 asks about *changing the text color in the spinner* in dark mode. The user aims to implement the dark theme in his/her Android app. His/her problem is the text and background colors are both blacks, which harms the readability. This problem occurs if he/she sets the values of the spinner programmatically. The analyzed posts also show that some developers are lacking knowledge on changing the colors of widgets in dark mode, as in the case of SO questions 65383610, 58210151, 66562092. These developers want to *change the theme of one activity* to dark mode. In those questions, the developers request

help to design specific colors for widgets in dark mode Android application. ***The root cause for these questions is that developers do not know how to correctly design and implement widgets in dark mode.*** The complexity of designing widgets in dark mode, combined with insufficient or unclear documentation, often poses challenges to developers. Transitioning an app to dark mode isn't just about changing colors—it requires a deep understanding of how the colors, layout, and shapes of individual elements interact in the new theme. This complexity can be overwhelming, especially for developers new to dark mode design. Developers require more comprehensive guidance and clearer documentation on implementing dark mode features to avoid such issues.

Developers often face issues when they design dark-mode app layout (44 posts). Dark mode UI is implemented in Android apps by specifying it in layout files. Based on our analysis, we found that Android developers look for help in Stack Overflow on how to design the entire UI in dark mode. There are examples of questions related to customizing the UI for an app in dark mode. For example, SO question 58382876 describes the need for *setting app UI to dark mode programmatically*. The developer wants to find a way of interacting with the buttons and set the UI dark theme programmatically through an app. SO question 66241056 asks *how to change UI in a dark mode in Android 10*. The developer would like to apply dark mode to an app in Android 10. In the same line of thought, SO question 58199283 discusses dark mode UI. The developer is *trying to implement dark mode in his/her application and faces strange issues with dark mode settings.* ***The root cause of these posts is the difficulty of designing a whole dark mode application.*** Designing a dark mode application entails intricate understanding of various design parameters like layout, colors, and icon shapes. This task is challenging as it goes beyond mere color alteration to involve a comprehensive understanding of element interactivity in a dark setting. The difficulty lies in creating an aesthetically pleasing yet functional interface that ensures consistency and a seamless user experience across the application. This requires a depth of knowledge some developers may lack, resulting in difficulties in designing dark mode applications.

Developers often encounter issues when they start to implement Android dark mode design (55 posts). Unfamiliar with dark mode development, many developers cannot find the start point for implementing dark mode in Android applications. For example, SO question 66241056 asks *how to force apply a dark theme on Android Studio*. The developer faces errors in modifying the activity manifest and inheriting the theme from the Theme for his/her app. SO question 59720952 requests *help for setting the dark theme for WebView*. The developer cannot set dark mode for an Android WebView and asks for a solution that is backward compatible before API 29. SO question 59569926 asks *how to get dark mode to work in one Android Q application*. The developer has set up a ListPreference with different values for dark mode preference. He/she wants the app to work with the new dark mode implementation introduced in Android Q. ***The root cause of these problems is the lack of basic knowledge in dark mode implementation.*** Despite available guidelines and tutorials, many developers remain unaware of these or struggle to understand them due to limited foundational knowledge in Android development. This gap in knowledge is amplified when developers of varying experience levels attempt to create dark mode applications without a strong understanding of the underlying principles and practices.

45 posts are related to implementation errors in dark mode applications. Many Android developers discuss the errors they encountered in dark mode development. Some developers ask why dark mode does not work in their applications. For example, SO question 60860240 asks *why colors do not change in dark mode*: The developer follows google's blog post to make a dark theme for his/her app, but don't get support on how to get the elevation effects on views (buttons, app bar, etc) to work. He/she wants to know how to implement this elevation functionality. SO question 65749561 requests suggestions for *fixing the app crash issue in dark mode*. Once the app setting is switched to dark mode, the app crashes. ***The root cause is the diversity of dark mode***

developments. There is no standard for dark mode development in the Android platform. Developers often employ different techniques, leading to inconsistencies and compatibility issues, particularly when certain methods are only effective on older Android versions, thus creating problems when applied to newer ones.

Developers not only care about dark mode in individual applications, but also the system-level dark mode (24 posts). Some SO questions focus on dark mode design in the scope of the entire system rather than a single application. SO question 64908385 discusses *how to set dark mode for the entire device in Android 10 or 11.* The developer is trying to create an app that toggles dark mode (Night mode) for the whole system, rather than a single app. SO question 64358104 asks *how to set the default dark theme for the system.* He/she has looked in all resources and does not find the parameters that can force the app to dark mode. ***The root cause is that developers sometimes do not know the relation between individual application themes and system theme settings.*** Developers have difficulty in binding their application themes to the broader system theme, leading to inconsistencies and problems. This issue is further compounded by a lack of robust community support and clear, accessible educational resources. Developers require more comprehensive and user-friendly guidelines that can effectively guide them through this complex process.

Developers sometimes have issues on how to detect dark mode (20 posts). We found some dark mode-related questions were oriented to detect dark mode in Android applications. In this category, the most common questions *relate to detecting if the Android device is in dark mode.* In SO question 57685088, the developer is trying to support dark mode for his/her Android app and can not figure out the current theme. Some other SO questions, such as SO question 59408538, discuss *if dark mode exists in Android OS.* The developer asks for a way to check programmatically if OS dark theme is available or not. ***The root cause is that there is no official support for dark mode detection.*** Currently, the resources provided officially are majorly related to the design and implementation aspects of dark mode, but they overlook the need for detection tools. This leaves developers to manually inspect the theme of the device and verify the existence of dark mode, a process that can be cumbersome and prone to error. In essence, this issue could be addressed by providing robust, official resources and tools that assist developers in accurately detecting dark mode, thereby simplifying their workflow and improving the overall user experience.

Some developers want to *disable* dark mode (34 posts). Rather than focusing support dark mode in Android applications, some SO questions discuss how to *disable* dark mode. SO question 66167450 asks *how to disable night mode in Android 11.* The developer tries to avoid color changes in dark mode. SO question 64403722 requests *advice for avoiding forced dark mode in Material theme.* The developer wants the app to be only in a light theme. Some devices force dark mode on apps and the developer wants to avoid that. SO question 64052086 discusses *how to disable automatic change of day-night theme*, based on android system settings. The developer is not willing to make this app to support the dark theme. ***The root cause is that dark mode may harm the function of these apps.*** Some developers have concerns regarding the default force dark theme, which might produce undesirable visual effects in their applications. These issues highlight the need for more flexibility and control in the implementation of dark mode, allowing developers to opt-out or customize it according to their application's specific requirements, without compromising on the overall user experience or the app's functional integrity.

35 posts do not belong to the above-mentioned categories and are assigned to the Other category. The analyzed samples also include dark mode-related posts but were not mentioned in the above categories. SO question 64878577 discusses *how to set an app into dark mode on the first-time launch.* SO question 61310165 discusses *how to save dark mode state when an app is closed.* The developer wants to save dark mode state in android when the app is completely closed. SO

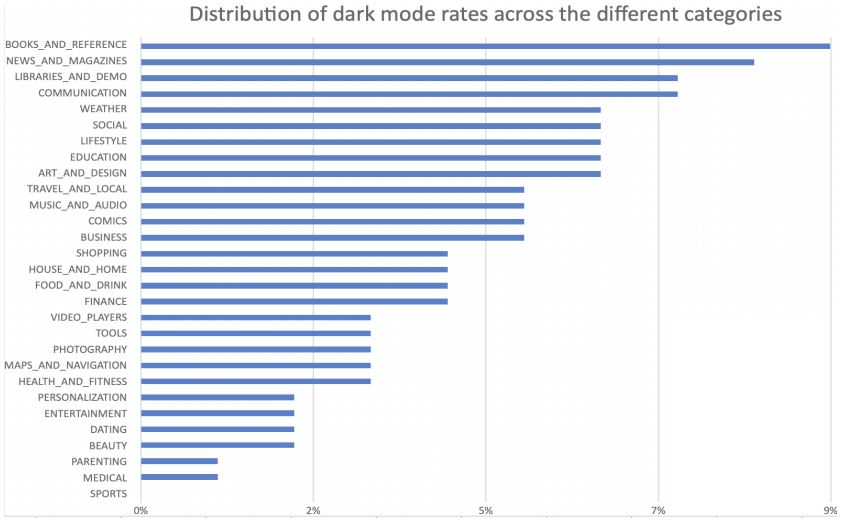

Fig. 6. Dark mode rates in different categories

question 60666892 introduces *the attributes in dark mode related packages.* **The root cause of some dark mode-related SO questions and discussions are varied.**

> *After analyzing 324 dark-mode related Stack Overflow posts, we manually classified them into eight types of issues that developers encounter. The most common issues were widget design, layout, implementation approach and implementation errors. Less common ones were issues related to disabling dark mode, detecting dark mode, and managing dark mode state on launch/close.*

4.2 RQ2 – What are dark mode features found in real-world Android applications?

Out of the 6,600 apps we downloaded from the Google Play store, **only 236 had implemented dark mode among the top 200 apps across 33 categories**. We calculated dark mode rates of each category as shown in Figure 6. dark mode rate for a category is calculated as the number of apps with dark mode divided by the number of all apps in the category. Some categories have a high dark mode rate. For instance, in categories "NEWS_AND_MAGAZINES" and "BOOKS_AND_REFERENCES", the dark rates are 8% and 9%. These apps are related to reading, and likely that developers have to design dark mode to render a better user experience in a low-light environment. However, dark mode rate is quite low in some categories. For instance, in the category "SPORTS", the dark rate is 0%. The reason is likely that the functionality of these apps is less related to reading. In the category "SPORTS", most apps are mainly designed for watching videos. The lack of a dark mode thus likely does not affect the usability of these apps.

To analyze dark mode features in Android applications, we decompile applications and analyze source files. We analyze the *colors.xml* in the *res/values/* and *res/values-night* folder to analyze the color features in dark mode.

First, by analyzing the colors in light mode and dark mode for the same Android applications, we collected 3,302 color mapping relations, which map the light color into dark color. To better investigate the differences in colors in light mode and dark mode, we use the k-means method to analyze the most frequent color clusters and calculate the mean color for each color cluster in light mode and dark mode. We choose k-means rather than simple frequency ranking because some high-frequency colors are quite similar and hard to detect by human eyes and k-means can group

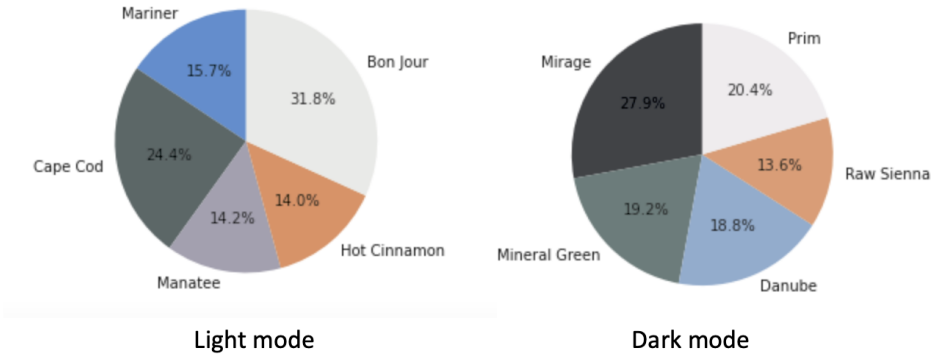

Fig. 7. Kmeans result for light and dark mode colors

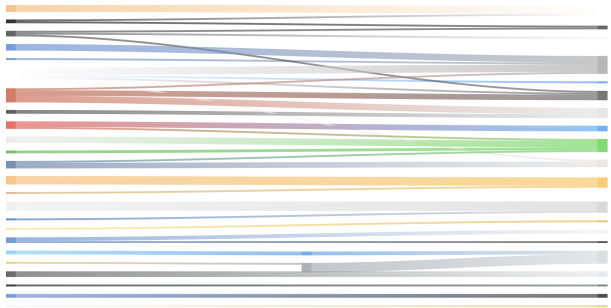

Fig. 8. Color mapping between light and dark mode

similar colors together, reducing the complexity of color differentiation. The results are shown in Figure 7. In light mode, the **5 most frequent colors are Mariner, Cape Cod, Manatee, Bon jour, and Hot cinnamon**. In dark mode, the **5 frequent colors are Mirage, Prim, Mineral Green, Danube, and Raw Sienna**. The frequent colors in light mode have higher brightness compared to the colors in dark mode. Because developers adopt different color schemes in light mode and dark mode. In dark mode, developers have to consider the low-light environment the color contrast between text and background.

Second, we found that **the same dark-mode color can be mapped into multiple light-mode colors, and the same light-mode color can also be mapped into multiple dark-mode colors**. Figure 8 is an example, the left side lists light mode colors and the right side lists dark mode colors. For instance, light mode color #000000 is mapped into dark mode color #1fffff and #2f2f2f. On the other hand, dark mode color #3a76d0 is mapped into light mode color #0084ff and #0596fa . The same color in light mode could be mapped into different dark mode colors in different apps. For instance, Figure 9 (a) and 9 (b) have the same white color in background, but are mapped into different dark mode colors. The reasons are as follows. First, different developers use different design schemes. There is no standard for GUI design in the Android platform, but only abstract guidelines [33]. Therefore, developers may add their design preference in dark mode design. Second, even the same developer may adopt different design schemes in different apps. When designing app GUIs, developers have to choose the proper color scheme considering various elements, such as the content of the app and the aimed user groups. For example, a reading app may have higher color contrast than a music app. Third, the developers have to consider the relations between different

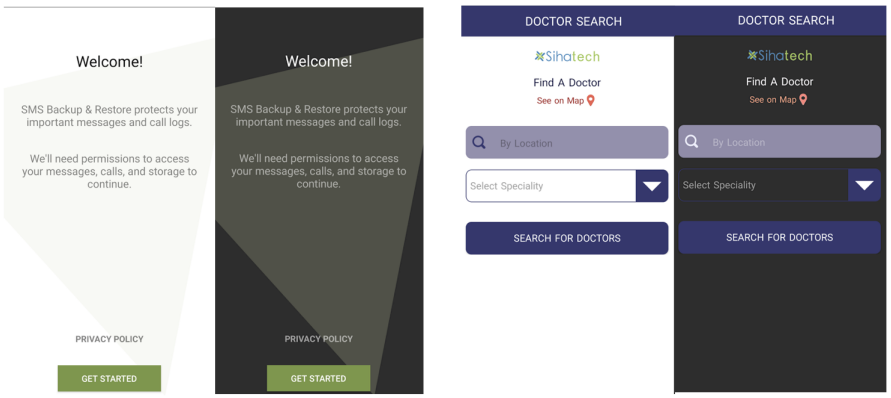

Fig. 9. Android app examples in light and dark mode

Fig. 10. Icon examples for different types

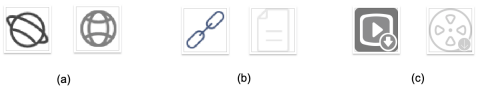

Fig. 11. Icon examples for shape change

components. For example, if the background is in black, the text cannot use dark colors to avoid readability issues.

We also analyzed the icons images in *res/drawable* and *res/drawable-night* folders to analyze the widget features in dark mode. For apps with dark mode, **developers usually design two versions of their app icons**, one for light mode and one for dark mode. Among the 4,213 icons specially designed for dark mode, 4,042 (95.94%) icons are designed as vector drawable (Figure 10 (a)). 171 (4.05%) icons are specially designed as png files (Figure 10 (b)). Only one gif icon is specially designed for dark mode (Figure 10 (c)).

Most icons only have color differences in light mode and dark mode. As examples in Figure 10, colors in light mode are changed into the different colors in dark mode. **A small part of icons (10 icons) have shape changes in dark mode.** As shown in Figure 11, (a) represents web, (b) represents input paste, (c) represents videos. The shapes of these icons are changed in dark mode. The reason for these shape-changing icons is that developers want to improve the UI effects of apps in dark mode.

We calculated and analyzed dark mode rate for 6,000 real-world Android apps. We found that only 236 of these apps have dark mode and dark mode rate varies across different categories – books, news, weather, social and education have higher number of dark mode (higher than 7%), whereas sport, medical, dating and entertainment have a lower number (less than 2%). In dark mode, the 5 frequent colors are Mirage, Prim, Mineral Green, Danube, and Raw Sienna. The same dark-mode color can be mapped into multiple light-mode colors, and the same light-mode color can also be mapped into multiple dark-mode colors. We also found that most icons only have color differences in light mode and dark mode, only a small number of icons have change of shape in dark mode.

4.3 RQ3 - How well do app development support tools aid dark mode vs light mode development?

In this experiment, we wanted to investigate how well four app development support tools work when applied to apps in dark mode vs light mode. We focused on the performance and consistency of app development support tools in light and dark modes. We checked if each tool generates the same results in light vs dark mode settings. Our focus remains on the performance and consistency of app development support tools in light and dark modes, and the implications of any observed differences. We conducted detailed experiments with four tools: Accessibility Scanner, UIED, UI2code, OwlEye as follows. Based on their functionality, we choose suitable metrics to compare the performance of these tools in dark mode and light mode. We analyzed the consistency issues for each tool. The consistency issue means the results are different in light and dark mode settings. If the result of a tool is highly consistent in light mode and dark mode, this tool has the goodness of consistency in light mode and dark mode.

4.3.1 What are the consistency issues in Accessibility Scanner? We examined the consistency issues of Accessibility Scanner when analysing dark mode apps. Based on the Xbot results, we filtered out the screenshots that have accessibility issues in dark mode or light mode. In total we collected problematic 1,672 screenshots in dark mode and 1,612 problematic screenshots in light mode. We actually analyze pairs of screenshots when either the light mode or the dark mode screenshots are problematic. We analyze 1731 screenshot pairs in total. Then we analyzed the accessibility issues claimed by Accessibility Scanner for these app screens in both dark mode and light mode settings. As shown in Table 2, 6 issue types appear in dark and light mode. In dark mode, there are 342 text contrast issues, 15 image contrast issues, 635 touch target issues, 1,427 item label issues, 57 item description issues, and 76 clickable item issues. In light mode, there are 315 text contrast issues, 6 image contrast issues, 635 touch target issues, 1,427 item label issues, 57 item description issues, and 76 clickable item issues.

Based on this result, we found that the Accessibility Scanner results differ in terms of text contrast and image contrast issue in light and dark mode. The text contrast issue corresponds to visible text, where there is a low contrast ratio between the text color and background color. Image contrast issue identifies images that are visible on the screen but with a low contrast ratio between the foreground and background colors. The best practice of text contrast ratio is 4.5 and higher. But we found that most wrong instances are located between 2 to 4 contrast ratios in both dark and light mode. **The contrast ratio in dark mode is lower compared to light mode setting.** The average contrast ratio is 3.57 for light mode, and 3.45 for dark mode. We list the top-5 most frequent wrong pairs of foreground text and background color in light and dark mode in Table 3 and Table 4. Using these color pairs will negatively influence the readability of the text. The wrong pairs in dark mode are more difficulty to read compared to light mode pairs. **The results of Accessibility**

Table 2. Accessibility issues in Android apps

ID	Issue type	Description	#Issues (dark/light)
1	text contrast	texts with a contrast ratio lower than 3.0 between the text color and background color	342/315
2	image contrast	images with a contrast ratio lower than 3.0 between the foreground and background color	15/9
3	touch target	clickable and long-clickable Views that are smaller than 48dp x 48dp in either dimension	635/635
4	item label	views that a screen reader could focus and that have an empty spoken description	1427/1427
5	item description	views with a redundant description	57/57
6	clickable item	more than one item share the same on-screen location	76/76

Table 3. Text Demo of the top 5 contrast issues in dark mode

Demo	Foreground	Background	Contrast Ratio	#Issues
Text	#76777A	#1A1D21	3.78	31
Text	#818084	#312F34	3.37	27
Text	#737373	#000000	4.43	22
Text	#02537A	#0C0C0C	2.35	17
Text	#5677FC	#3A3A3A	2.95	12

Table 4. Text Demo of the top 5 contrast issues in light mode

Demo	Foreground	Background	Contrast Ratio	#Issues
Text	#8E8D8E	#FFFFFF	3.31	33
Text	#AC96AC	#582D59	3.99	28
Text	#FFFFFF	#0081C2	3.16	22
Text	#FFFFFF	#00A82D	3.16	18
Text	#FFFFFF	#F0A00D	2.16	11

Scanner differ in dark mode and light mode because the different color schema are used in dark mode UI display.

To further analyze consistency issues, we analyze the difference between dark mode and light mode in the same apps. We found that many dark mode apps have accessibility issues involving item label issues, item description issues, and clickable item issues. **These apps have more text and image contrast issues in dark mode compared to light mode.** The reason may be that changing the app to dark mode only involves a change in color attributes. We found that **about half of the apps have the same issue number in light and dark mode** (52.6% apps). In some apps, such as figure 12 (a), while the app has the same accessibility issues in light and dark mode, the inaccessibility degree may be different. For example, for the same accessibility issues in the home button, this button has text contrast issues in both light and dark mode, but the contrast ratio is 4.1 in light but 3.7 in dark mode, which means dark mode is more inaccessible. Other apps have different issues in dark and light mode (47.4% apps), as shown in figure 12 (b). In this example, all mail text has text contrast issues in light but not in dark mode. The mail icon has an image contrast issue in dark but not in light mode. That indicates that the developers do not consider the color contrast problems in both light and dark mode.

Some dark mode-specific issues cannot be detected by Accessibility Scanner. According to dark mode guidelines in Material design [4], developers should not use bright colors for large

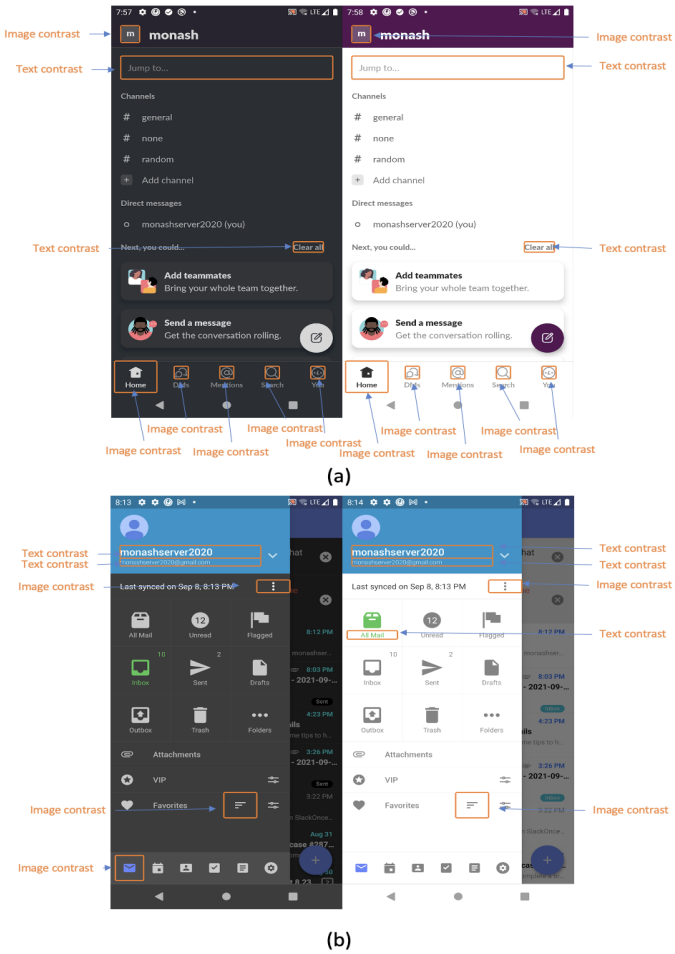

Fig. 12. Examples of accessibility issues

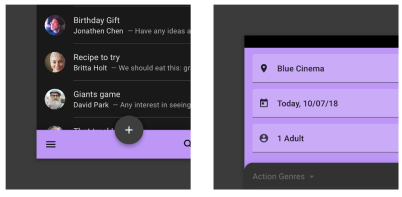

Fig. 13. Issues not detected by Accessibility Scanner

surfaces because they can emit too much brightness. In Figure 13 (a) and (b), there are bright color issues, but these issues cannot be detected by Accessibility Scanner, since Accessibility Scanner only cares about general accessibility issues rather specific dark mode features.

Tool	Evaluation Metric	Score
UIED	Consistency Ratio	0.18
	F1-score (light mode)	0.51
	F1-score (dark mode)	**0.42**
UI2code	Consistency Ratio	0.49
	BLEU score (light mode)	0.82
	BLEU score (dark mode)	**0.65**
OwlEye	Consistency Ratio	0.70
	F1-score (light mode)	0.77
	F1-score (dark mode)	**0.60**

Table 5. Evaluation metrics for app development support tools

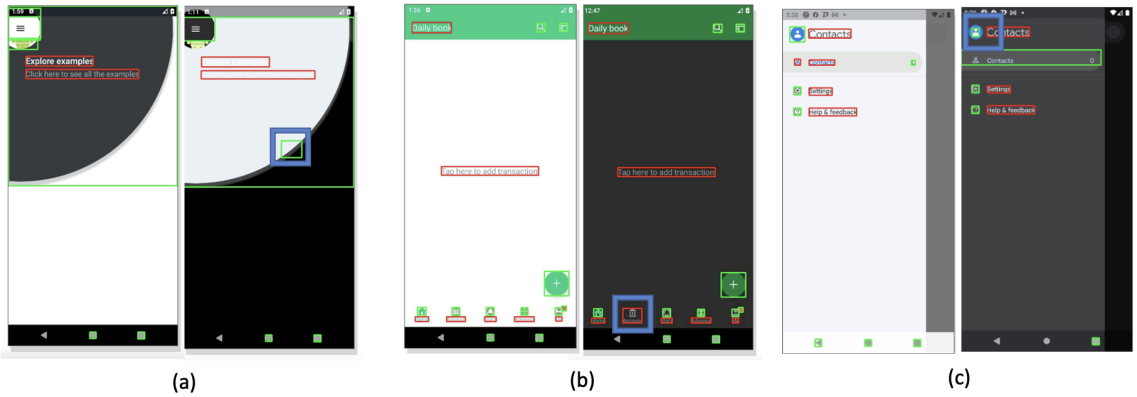

Fig. 14. Examples of consistency issues of UIED

4.3.2 What are the consistency issues in UIED?. We collected 200 pairs of UI screenshots from 50 Android apps (on average 4 pairs per app) in light and dark mode. dark mode UI screenshots and light mode UI screenshots are collected from same activities of android apps. Then we ran the UIED tool on 200 light-mode Android screenshots and 200 dark-mode Android screenshots. We further analyzed the consistency issues in dark mode and light mode results. We used F1-score as an evaluation metric which is interpreted as a weighted average of the precision and recall. The true positive (TP) means that the tool correctly detects the GUI elements in the screenshot. The FP means that the tool detects the GUI elements in the screenshot, but they are incorrect. Note that the measurements are evaluated on IoU > 0.9, where the IoU is the intersection area over union area of the detected bounding box and the ground-truth box. We manually compared the tool results in light mode and dark mode and calculated the consistency ratio of UIED, which is the ratio of UI images that have the same GUI elements in light and dark mode.

As shown in table 5, among 200 screenshots pairs, 36 screenshots have same result in light mode and dark mode, which means the consistency ratio for screenshots is 0.18. We further analyzed the difference of F1-score in light and dark mode. In light mode, the F1-score is 0.51 (IoU > 0.9). In dark mode, the F1-score is 0.42 (IoU > 0.9). The F1-score in dark mode is lower compared to light mode, which means the UIED has worse performance in dark mode.

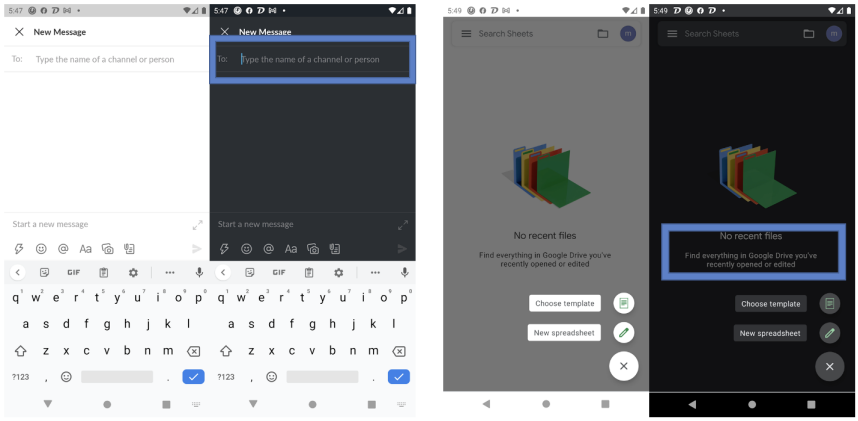

Fig. 15. Examples of consistency issues of UI2code

We further analyzed the types of consistency issues for UIED. Figure 14 shows three examples of the consistency issues of UIED, these issues are marked with blue box. **The first issue is the incorrect bounding box in dark mode (19%), which means the bounding box location is incorrect in dark mode screenshots.** As shown in figure 14 (a), the bounding boxes are correctly annotated in light mode screenshot, but an incorrect bounding box appears in the result of dark mode screenshot. **The second consistency issues is failing to divide small components in dark mode (66%), which means some small components are not detected in dark mode.** As shown in figure 14 (b), the icon and text are wrongly grouped as a icon in dark mode screenshot. **The third consistency issue is wrong box shape in dark mode (15%), which means the bounding box interacts with the component but does not contain the whole component.** As shown in figure 14 (c), the bounding box in dark mode only include parts of the icon. These issues indicate that dark mode features are not well considered in the UIED tool development.

4.3.3 What are the consistency issues in UI2code? We collected 200 pairs of UI screenshots and GUI skeletons from 50 Android apps (on average about 4 pairs per app) in light and dark mode. Each GUI skeleton is mapped to one light mode screenshot and one dark mode screenshot. We used BLEU score and exact match as evaluation metric to analyze the consistency issues in light mode and dark mode. BLEU score evaluated the quality of generated skeleton by calculating the correspondence of generated skeleton and original skeleton. Exact match checked if the generated skeleton is same as original skeleton. We manually compared the tool results in light mode and dark and calculated the consistency ratio of UI2code, which is the ratio of UI images that have the same GUI skeletons in light and dark mode.

We analyzed the consistency issues of UI2code in light and dark mode setting. Among 50 apps, 16 apps have same result in dark and light mode. Among 200 screenshot pairs, 97 screenshot pairs have the same results in light mode and dark mode. As shown in table 5, the consistency ratio is 0.49. In light mode, the generated GUI skeletons for 133 (66.5%) UI images exactly match the ground truth GUI skeletons, and the average BLEU score over all test UI images is 0.82, for only 1 of all test UI images, the model fails to generate closed brackets. In dark mode, the generated GUI skeletons for 85 (42.5%) UI images exactly match the ground truth GUI skeletons, and the average

Fig. 16. Examples of consistency issues of OwlEye

BLEU score over all test UI images is 0.65, for 8 of all test UI images, the model fails to generate closed brackets. This result shows that the UI2code tool is not consistent in terms of exact match and BLEU score in light and dark mode. Compared to light mode apps, this tool fail to capture the composition information of container components in dark mode apps.

We manually analyzed the difference of these generated GUI skeletons in dark and light mode. We summarized the main cause of this consistency issue. Low contrast background in dark mode could be more problematic compared to light mode background. Figure 15 shows two examples in which UI2code generates wrong GUI components in dark mode. In Figure 15 (a) and (b), the color contrast of text color and background color in dark mode is lower than light mode. UI2code may mistake the content displayed as part of the UI to implement, and generate wrong GUI components.

4.3.4 What are the consistency issues in OwlEye? We collected 50 pairs of non-duplicate screenshots with UI display issues and equal number of bug-free non-duplicate screenshots in light and dark mode. We run OwlEye on these collected screenshots and analyze the results. In order to evaluate the issues detection performance of OwlEye, we employed three evaluation metrics, i.e., precision, recall, F1-score, which are commonly-used in image classification and pattern recognition. The true positive (TP) means that the tool correctly detects the GUI issues in the screenshot. The FP means that the tool wrongly detects issues in the screenshot. We manually compared the tool results in light mode and dark mode and calculated the consistency ratio of OwlEye, which is the ratio of UI images that have the same visual issues in light and dark mode.

As shown in table 5, the consistency ratio is 0.7, which means 70% screenshots have the same OwlEye detection results in light mode and dark mode. In light mode, the recall score is 0.75 , precision score is 0.81 and F1 score 0.77. In dark mode, the recall score is 0.5, precision score is 0.75 and F1 score is 0.6. The OwlEye has lower score in recall, precision and F1 score in dark mode, which indicates it performs worse in dark mode compared to light mode.

To further analyze the inconsistency in light and dark mode, we applied Grad-CAM model for the localization of UI display issues by visualizing the attention. We found that some UI display issues are detected by OwlEye in light mode, but not in dark mode. Figure 16 is the visualization of attention in light mode and dark mode screenshots. In Figure 16 (a) and (b), the text components are blocked by image in light and dark mode. In light mode, attention are placed in the interaction of text components and images, which indicates OwlEye can easily focus the UI display issues. But in dark mode, OwlEye pays attention to areas which are irrelevant to the display issues. OwlEye uses

deep learning based approach to classify the UI display issues, which means the tool performance mainly rely on the training data. The poor performance of OwlEye in dark mode may be related to the unbalanced ratio of dark mode screenshots in training data.

We investigated the consistency issues when used in light vs dark mode for four app development support tools – Accessibility Scanner, UIED, UI2code and OwlEye. We classified the type of consistency issues found for these four tools and analyzed the reason for these inconsistencies. We found that some consistency issues are related to the differences in dark mode UI display. Some other issues appear because dark mode features are not taken into consideration in these app development support tools.

4.4 Threats to Validity

4.4.1 Construct Validity. There are several ways to understand the issues developers encounter in dark-mode Android development. For example, interviewing and surveying users might be one way. In this article, we choose to study dark mode-related Stack Overflow posts instead. Both approaches have their own limitations. For example, with surveys, users may miss reporting on some instant feedback to dark mode, depending on their recollection. Nevertheless, the mining approach has its limitation as well. For example, the collected data cannot represent all issues in dark mode Android development. To avoid the possibility of missing dark mode issues type, we studied each post based on the question itself and the associated answers to identify the issues. Though posts are often precise, some complicated posts may contain multiple intentions. We assign each post with the most relevant label. We recorded all such multiple-label posts, and they account for no more than 4%. Besides, multi-label issues are often related and do not contradict with each other, so it does not influence the coding procedure. Therefore, we believed multiple labels do not influence much on the categorization results.

4.4.2 Internal Validity. We analyzed and manually labeled Stack Overflow posts. While manual labeling is a tedious and time-consuming task, the went through the manual analysis process to identify dark mode issue types and get detailed insights about the characteristics of the identified issue types. Further work can extend our study by providing approaches for identifying the raised issue type in posts and extend our work on a larger dataset.

In this article, we aimed to identify dark-mode posts and Android apps to perform our empirical study. Thus, our proposed approach is not fully automatic and involves much manual analysis effort. In the future, we plan to extend our study by proposing automated approaches.

We choose apps from Google Play to analyze dark mode features. However, there is no standard to evaluate the quality of dark mode design in real world Android apps. Although Google Play store allow users to rate apps, the rates are mainly based on the overall quality of Android apps.

4.4.3 External Validity. The generalizability of our results is a threat to validity. We have collected and identified the top 200 apps for 33 categories (6,600 apps in total) to analyze dark mode features in real-world Android applications. These real-world Android application are only a small part of the whole Google Play Store apps. The limited generalizability of the sample might introduce bias to our experiment results. We provided the first step towards this direction. We encourage future studies to explore these things in a deeper manner through several more in-depth and focused studies on various aspects.

A threat to validity is the sampling strategy. In this article, we collected category-balance numbers of the app, which may not represent the app distribution in real-world. To mitigate such a threat, we collected as many apps as possible (i.e., top 200) to cover most of the mainstream popular apps.

Besides, we also considered sub-categories provided by the Google Play Store to follow the app distribution in real-world.

5 DISCUSSION

Below we discuss the implications of our findings. We discuss actionable implications and future work for developers and researchers.

5.1 Implications for Developers

Developers should be aware of the potential risks in using unsuitable design schema in dark mode and develop good testing habits in developing dark mode apps. In RQ1, we found that widget design and layout design in dark mode are the most prevalent issues that developers care about. In RQ3, we observed that color contrast issues commonly appear in dark mode, which not only leads to unreadable text and images but also decreases the aesthetic quality of app UI. It can be difficult for developers to identify and correct the issues in a dark mode app. Instead, developers should be aware of the common issues in dark mode development and test these issues in the development process. For example, a developer may be confused as to how to reduce the accessibility issues in dark mode UI, since the fault may lie in ignoring the color contrast issues. However, with our results, the developer can know how to design a usable dark mode UI in practice so that he/she can find the solution with less trial and error.

Developers should carefully study official dark mode tutorials and dark mode development cases before developing dark mode applications. There are some official dark mode development tutorials and materials [4, 17]. Some errors may occur if developers develop dark mode applications without knowing these backgrounds. Although the official dark mode documentation highlights correct usages and incorrect usages, developers may not be aware of them. In RQ1, we found that developers often have trouble implementing Android dark mode designs. We found 44 posts discussing how to start Android dark mode design and 28 post related to implementation errors in dark mode. Developers should check the design examples in the official documentation to avoid misuse and be aware of the common issues. In our RQ3, we analyzed the common dark mode accessibility issues for developers to avoid common mistakes.

Developers should carefully deal with the relation between application theme and system theme settings. Since Android 10, Android officially supports a system-wide dark mode which means that many parts of the operating system have the ability to be switched to dark mode. For some third-part apps, users can switch the system-wide theme setting to change the theme in individual apps. In RQ1, we observed that the theme setting in some Android applications is inconsistent with the system theme. Among our 16 collected posts that are related to systematic dark mode, half of them discussed this issue. In this case, users have to manually switch the theme setting even though the system theme setting is changed. Developers should better tie the relation between dark mode application and system setting.

Developers should design specific icons for dark mode. In RQ2, we observed that icon design is an important part for dark mode design. Unlike other components such as text and background, icons may vary in both color and shape in light and dark mode. Based on the change of background and surrounding components, developers should consider to change the color of icons or even design a new icon.

5.2 Implications for Researchers

Future studies should provide easy-to-use dark mode design guidelines. Despite having access to the accessibility guideline released by Android [4] and iOS [17], designers and developers may not understand them very well due to too abstract concepts and the lack of real examples.

For example, it is not an easy task for designers to select color schema for not only highlighting the text. As we discussed in RQ2, there is no standard for color mapping between light and dark mode color schema. The same dark-mode color can be mapped into multiple light-mode colors, and the same light-mode color can also be mapped into multiple dark-mode colors. Generating contrast against white is challenging, and against darker colors even more. A key problem we found was that colors chosen to work against dark backgrounds were too strident. To help the development team better understand the accessibility issues, future works should help developers better understand the features of dark mode and issues.

Future research could investigate ways to automatically generate dark mode GUI for developers. Developers spend lots of time in design dark mode GUI. Designing dark patterns for GUI design is time-consuming and requires a large number of effort. Based on the analysis of RQ2, developers cannot just reuse colors and invert the shades. They have to ensure that dark mode GUI is delightful, balanced, and readable. Therefore, our research can inspire researchers to invent a model that can automatically modify the existing light mode UI design to dark mode, reducing the unnecessary design time and accelerating the development process. The model can be thought of as a professional designer who knows how to analyze the existing light mode mobile UI design interface and add required features dark mode. This task is an interdisciplinary topic that combines software engineering and industrial design.

Future research could investigate ways to automatically convert the applications to high quality dark mode for users. Some Android applications have no dark mode theme, which leads to a bad user experience for users. Based on the analysis in RQ2, only a small part of applications have dark mode design. It can be difficult for users to use these applications in low illuminance environments. Although Android Q provides force dark options for all applications [5], the results are not acceptable for many applications. Therefore, it would be important to provide support on converting light mode applications into dark mode for users.

Future studies are needed to automatically help developers detect or correct accessibility issues. Accessibility issues commonly appear in dark mode applications. In RQ3, we took a closer look at the occurrences of various types of accessibility issues in dark mode apps. We found that some dark mode issues are about color contrast such as text contrast and image contrast. Therefore, we believe that fixing the accessibility issues in dark mode deserves the attention of researchers. For example, it is not an easy task for designers to select a color scheme for not only highlighting the text but also improving visual comfort in dark mode. Therefore, the accessibility issue is an important problem for dark mode design. What is more, as discussed in RQ3, existing accessibility testing tools mainly focus on general accessibility issues without considering dark mode requirements. We call on researchers to develop automated techniques in this direction to facilitate the automated fix of accessibility issues of mobile dark mode apps.

Researchers should take dark mode features into consideration when developing app development support tools. Based on the analysis in RQ3, existing app development support tools mainly focus on the Android apps in light mode setting. Directly using these tools on dark mode setting leads to a worse performance. For instance, Accessibility Scanner cannot detect some dark mode specific accessibility issues. OwlEye cannot successfully detect some UI display issues in dark mode setting. Therefore, it is of importance to take dark mode features into consideration for app development support tools.

6 RELATED WORK

Application dark modes swap the default UI color scheme with a dark grey or black background and light grey or white text. The trend towards providing dark schemes is exemplified by Google's decision to include a global dark mode in Android Q, where the user can set the UI globally to a

dark theme and applications to their internal dark modes [3]. One study found that 80% of website content is white pixels [35]. The Chameleon browser [29] uses a proxy server to dynamically change the color of websites to produce darker content. However, its eventual implementation seems unlikely since it requires extensive proxy support.

Some research works can help dark mode UI implementation. Block-wise histogram-based features and image processing methods were used to recognize mobile UI elements from images [22, 48]. A neural machine translator was designed to transform mobile design images into a GUI skeleton [25], and deep learning techniques are also used in generating GUI codes [20, 31] and GUI designs [58].

Recently, many studies have started to investigate the accessibility issues in mobile apps and web applications. Some researchers conduct large-scale studies to analyze the general accessibility issues in the Android ecosystem [16, 32, 43, 44, 50]. Some research works focus on detecting accessibility issues [18, 26–28, 47]. Some research works aim to improve the accessibility of Android apps and websites by automatically generating or modifying some accessibility data with machine learning [15, 38–40, 57].

Some researchers have analyzed the effects of a dark mode on human eyes [30]. Deguang et al. focused on how to improve the effectiveness of OLED screens in modified devices interfaces [36]. Sheppard and Wolffsohn studied how the misused screen display strains the eye and leads to ocular disorders [46]. An-Hsiang et al. investigated how new display technologies improve the sustainability of applications on different screen types [53]. Coelho et al. manually evaluated four government mobile apps using W3C Accessibility Guidelines [12] and found that accessibility issues are extensive in these cases [45]. Milne et al. investigated the accessibility of mobile health sensors for blind users [42]. Walker et al. evaluated weather apps and found them not to be universally accessible [52]. Vendome et al. [50] performed an empirical study to understand Android apps' accessibility issues. These contrast to our study which focused on the accessibility issues in dark mode.

7 CONCLUSION

We presented the results of a large-scale empirical study aimed at understanding the dark mode in Android apps. We manually analyzed and tagged Stack Overflow discussions related to dark mode in Android. After this process, we built a taxonomy of the dark mode aspects discussed by developers. We analyzed the dark mode features in Android apps. We also investigated the consistency issues in four representative app development support tools. Our ultimate goal is to help catalyze advances in dark mode design by shedding light on the current state of affairs. Our findings can help practitioners by highlighting important skills to acquire. The findings also highlight opportunities for researchers to address the limitations of existing tools.

ACKNOWLEDGEMENTS

Grundy is supported by ARC Laureate Fellowship FL190100035.

REFERENCES

[1] 2021. Change your screen color at night on a Pixel phone. https://support.google.com/pixelphone/answer/7169926?hl=en Accessed Oct 10, 2021.
[2] 2021. Dark mode design: tips for creating dark theme websites and apps. https://99designs.hk/blog/web-digital/dark-mode/ Accessed Oct 10, 2021.
[3] 2021. Dark theme. https://developer.android.com/guide/topics/ui/look-and-feel/darktheme. Accessed Aug 18, 2021.
[4] 2021. Dark theme. https://material.io/design/color/dark-theme.html. Accessed Aug 18, 2021.
[5] 2021. Dark Theme. https://material.io/develop/android/theming/dark. Accessed Aug 18, 2021.
[6] 2021. Light-on-dark color scheme. https://en.wikipedia.org/wiki/Light-on-dark_color_scheme. Accessed Aug 18, 2021.

[7] 2021. Mobile App Growth and Success in 2021. https://clearbridgemobile.com/stats-for-mobile-app-growth-and-success/.

[8] 2021. Sylvain Boyer Designs the Friendlui Phone with Organic Elements. https://www.trendhunter.com/trends/friendlui. Accessed Oct 10, 2021.

[9] 2021. Teletext. https://en.wikipedia.org/wiki/Teletext. Accessed Oct 10, 2021.

[10] 2021. Use Dark Mode to Conserve Your Phone's Battery Power. https://www.designial.com/does-dark-mode-improve-the-user-experience/ Accessed Oct 10, 2021.

[11] 2021. Vector drawables overview. https://developer.android.com/guide/topics/graphics/vector-drawable-resources Accessed Oct 10, 2021.

[12] 2021. World Wide Web Consortium (W3C). https://www.w3.org. Accessed: 2020-10-18.

[13] 2021. Xbot. https://github.com/tjusenchen/Xbot. Accessed Aug 18, 2021.

[14] 2022. Dark UIs. The Good and the Bad. Dos and Don'ts. https://www.toptal.com/designers/ui/dark-ui Accessed April 10, 2022.

[15] Ali S Alotaibi, Paul T Chiou, and William GJ Halfond. 2021. Automated Repair of Size-Based Inaccessibility Issues in Mobile Applications. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 730–742.

[16] Abdulaziz Alshayban, Iftekhar Ahmed, and Sam Malek. 2020. Accessibility issues in android apps: state of affairs, sentiments, and ways forward. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 1323–1334.

[17] Apple. 2013. Introduction to Coding Guidelines for Cocoa Developing. https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CodingGuidelines/CodingGuidelines.html. Accessed: 2020-09-01.

[18] Mohammad Bajammal and Ali Mesbah. 2021. Semantic Web Accessibility Testing via Hierarchical Visual Analysis. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1610–1621.

[19] Patti Bao, Jeffrey Pierce, Stephen Whittaker, and Shumin Zhai. 2011. Smart phone use by non-mobile business users. In *Proceedings of the 13th international conference on human computer interaction with mobile devices and services*. 445–454.

[20] Tony Beltramelli. 2018. Pix2Code: Generating Code from a Graphical User Interface Screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Paris, France) *(EICS '18)*. ACM, New York, NY, USA, Article 3, 6 pages. https://doi.org/10.1145/3220134.3220135

[21] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[22] J. F. Canny. 1987. Readings in Computer Vision: Issues, Problems, Principles, and Paradigms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter A Computational Approach to Edge Detection, 184–203. http://dl.acm.org/citation.cfm?id=33517.33534

[23] Aaron Carroll, Gernot Heiser, et al. 2010. An analysis of power consumption in a smartphone.. In *USENIX annual technical conference*, Vol. 14. Boston, MA, 21–21.

[24] Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI design image to GUI skeleton: a neural machine translator to bootstrap mobile GUI implementation. In *Proceedings of the 40th International Conference on Software Engineering*. 665–676.

[25] Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI design image to GUI skeleton: a neural machine translator to bootstrap mobile GUI implementation. In *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. 665–676. https://doi.org/10.1145/3180155.3180240

[26] Sen Chen, Chunyang Chen, Lingling Fan, Mingming Fan, Xian Zhan, and Yang Liu. 2021. Accessible or Not An Empirical Investigation of Android App Accessibility. *IEEE Transactions on Software Engineering* (2021).

[27] Paul T Chiou, Ali S Alotaibi, and William GJ Halfond. 2021. Detecting and localizing keyboard accessibility failures in web applications. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 855–867.

[28] X Yu Daihua, Bambang Parmanto, Brad E Dicianno, and Gede Pramana. 2015. Accessibility of mHealth self-care apps for individuals with spina bifida. *Perspectives in health information management* 12, Spring (2015).

[29] Mian Dong and Lin Zhong. 2011. Chameleon: A color-adaptive web browser for mobile OLED displays. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*. 85–98.

[30] Austin Erickson, Kangsoo Kim, Gerd Bruder, and Gregory F Welch. 2020. Effects of Dark Mode Graphics on Visual Acuity and Fatigue with Virtual Reality Head-Mounted Displays. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 434–442.

[31] Sidong Feng, Suyu Ma, Jinzhong Yu, Chunyang Chen, Tingting Zhou, and Yankun Zhen. 2021. Auto-icon: An automated code generation tool for icon designs assisting in ui development. In *26th International Conference on Intelligent User*

Interfaces. 59–69.
[32] Raymond Fok, Mingyuan Zhong, Anne Spencer Ross, James Fogarty, and Jacob O Wobbrock. 2022. A Large-Scale Longitudinal Analysis of Missing Label Accessibility Failures in Android Apps. In *CHI Conference on Human Factors in Computing Systems*. 1–16.
[33] Google. 2019. Google developer documentation style guide. https://developers.google.com/style/code-samples. Accessed: 2020-09-01.
[34] Bernard J Jansen. 1998. The graphical user interface. *ACM SIGCHI Bulletin* 30, 2 (1998), 22–26.
[35] Antti Lääperi. 2009. Disruptive factors in the OLED business ecosystem. *Information Display* 25, 9 (2009), 8–13.
[36] Deguang Li, Bing Guo, Yan Shen, Junke Li, and Yanhui Huang. 2016. Making Image More Energy Efficient for OLED Smart Devices. *Mobile Information Systems* 2016 (2016).
[37] Zhe Liu, Chunyang Chen, Junjie Wang, Yuekai Huang, Jun Hu, and Qing Wang. 2020. Owl eyes: Spotting ui display issues via visual understanding. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 398–409.
[38] Suyu Ma, Chunyang Chen, Hourieh Khalajzadeh, and John Grundy. 2021. Latexify Math: Mathematical Formula Markup Revision to Assist Collaborative Editing in Math Q&A Sites. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–24.
[39] Suyu Ma, Zhenchang Xing, Chunyang Chen, Cheng Chen, Lizhen Qu, and Guoqiang Li. 2019. Easy-to-deploy API extraction by multi-level feature embedding and transfer learning. *IEEE Transactions on Software Engineering* 47, 10 (2019), 2296–2311.
[40] Forough Mehralian, Navid Salehnamadi, and Sam Malek. 2021. Data-driven accessibility repair revisited: on the effectiveness of generating labels for icons in Android apps. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 107–118.
[41] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[42] Lauren R Milne, Cynthia L Bennett, and Richard E Ladner. 2014. The accessibility of mobile health sensors for blind users. In *International Technology and Persons with Disabilities Conference Scientific/Research Proceedings (CSUN 2014)*. 166–175.
[43] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O Wobbrock. 2018. Examining image-based button labeling for accessibility in Android apps through large-scale analysis. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*. 119–130.
[44] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O Wobbrock. 2020. An epidemiology-inspired large-scale analysis of android app accessibility. *ACM Transactions on Accessible Computing (TACCESS)* 13, 1 (2020), 1–36.
[45] Leandro Coelho Serra, Lucas Pedroso Carvalho, Lucas Pereira Ferreira, Jorge Belimar Silva Vaz, and André Pimenta Freire. 2015. Accessibility evaluation of e-government mobile applications in Brazil. *Procedia Computer Science* 67 (2015), 348–357.
[46] Amy L Sheppard and James S Wolffsohn. 2018. Digital eye strain: prevalence, measurement and amelioration. *BMJ open ophthalmology* 3, 1 (2018), e000146.
[47] Camila Silva, Marcelo Medeiros Eler, and Gordon Fraser. 2018. A survey on the tool support for the automatic evaluation of mobile accessibility. In *Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*. 286–293.
[48] R. Smith. 2007. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. 629–633. https://doi.org/10.1109/ICDAR.2007.4376991
[49] Thomas Steiner. 2021. Let there be darkness! https://medium.com/dev-channel/let-there-be-darkness-maybe-9facd9c3023d Accessed Oct 10, 2021.
[50] Christopher Vendome, Diana Solano, Santiago Liñán, and Mario Linares-Vásquez. 2019. Can everyone use my app? an empirical study on accessibility in android apps. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 41–52.
[51] Fernanda B Viegas, Martin Wattenberg, and Jonathan Feinberg. 2009. Participatory visualization with wordle. *IEEE transactions on visualization and computer graphics* 15, 6 (2009), 1137–1144.
[52] Bruce N Walker, Brianna J Tomlinson, and Jonathan H Schuett. 2017. Universal design of mobile apps: Making weather information accessible. In *International Conference on Universal Access in Human-Computer Interaction*. Springer, 113–122.
[53] An-Hsiang Wang, Jia-Jen Fang, and Cheng-Hsun Chen. 2003. Effects of VDT leading-display design on visual performance of users in handling static and dynamic display information dual-tasks. *International journal of industrial ergonomics* 32, 2 (2003), 93–104.
[54] Chris Welch. 2021. Google confirms dark mode is a huge help for battery life on Android. https://www.theverge.com/2018/11/8/18076502/google-dark-mode-android-battery-life Accessed Oct 10, 2021.

[55] Terry Winograd. 1995. From programming environments to environments for designing. *Commun. ACM* 38, 6 (1995), 65–74.

[56] Mulong Xie, Sidong Feng, Zhenchang Xing, Jieshan Chen, and Chunyang Chen. 2020. UIED: a hybrid tool for GUI element detection. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1655–1659.

[57] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, et al. 2021. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[58] Tianming Zhao, Chunyang Chen, Yuanning Liu, and Xiaodong Zhu. 2021. Guigan: Learning to generate gui designs using generative adversarial networks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 748–760.