Apps for Everyone: Mobile Accessibility Learning Modules

By Yasmine N. El-Glaly, Anthony Peruma, Daniel E. Krutz, and J. Scott Hawker,
Rochester Institute of Technology

Mobile applications (apps) should be accessible to everyone, yet many of even the most popular are not. To address the lack of accessibility problem, we created a set of educational modules. These modules may be used to teach students and developers about proper methods of creating accessible apps, and on the importance of accessibility guidelines. Each module contains a well-defined accessibility problem, provides details about the accessibility issue, and simulates the effects of the accessibility barrier. Information is provided on how to fix the accessibility issue. Additionally, each module includes augmenting educational materials (slides, instructional videos, etc.), and example apps.

Mobile devices consume a substantial portion of digital media in the U.S. as well as a significant portion of their users' time. Unfortunately, many mobile apps are not fully accessible to individuals with disabilities. This is a significant problem since one in five Americans have a disability which could affect their ability to use a computing device [4]. Hence, it becomes a necessity to teach developers how to create accessible software. In 2017, ABET, the Accreditation Board for Engineering and Technology, updated the definition of Engineering Design to include accessibility as one of the criteria to consider in design [1]. Pedagogical researchers in computing have examined numerous accessibility and universal access topics in computing courses [10,12], and both accessibility advocates and industry support teaching accessibility in undergraduate programs as well [6,11]. Two vital components are needed to engage engineering students on the topic of accessibility. First, students must be engaged in practical real-world applications, such as developing an accessible mobile app. Second, students must gain an empathy for people with disabilities so that the accessibility guidelines will be meaningful to them. Our Mobile Inclusive Learning Kit (MILK) modules address these goals with real-world examples and an emulation feature designed to allow empathy for users with disabilities.

MILK
MOBILE INCLUSIVE LEARNING KIT

Developers create inaccessible apps for a variety of reasons:

- ignorance of accessibility guidelines e.g. WCAG 2.0 guidelines [13], and Android accessibility guidelines [3],
- inadequate technical abilities, and
- a lack of understanding of the importance of creating accessible software [10].

Unfortunately, due to resource constraints, many institutions lack the ability to add accessibility-related activities to their curriculum, resulting in a gap in mobile accessibility education [7]. The freely available MILK training modules are derived from real-world mobile accessibility problems and are systematically designed to cover principles that are fundamental in creating accessible mobile applications.

Our project has two main learning objectives.

1. To inform developers about why they should create accessible software. Creating inaccessible software has numerous negative ethical, legal and monetary implications. For example, Section 508, an amendment to the United States Workforce Rehabilitation Act of 1973, mandates that all electronic and information technology developed, or used by the federal government be accessible to people with disabilities [2]. Violating Section 508 can lead to filing lawsuits against the software company [8]. An initial step in getting developers to create accessible software is to show them the importance of creating it.
2. To inform developers how they can create accessible software.

Two vital components are needed to engage engineering students on the topic of accessibility. First, students must be engaged in practical real-world applications, such as developing an accessible mobile app. Second, students must gain an empathy for people with disabilities so that the accessibility guidelines will be meaningful to them.

> We would like developers of all experience levels to have at least a fundamental understanding of good accessibility practices and methods of creating software for everyone.

There were two primary concerns which led us to the creation of these modules.

1. Necessary Skillsets: Instructing students and developers how to create accessible software is a difficult task. All computing instructors cannot be expected to be experts in accessibility and are therefore often not prepared to create a robust set of accessibility modules.
2. Creation Effort: Designing and developing interesting and informative accessibility exercises such as ours is not easy, frequently requiring substantial amounts of time and other resources. Unfortunately, instructors at most institutions do not have these luxuries and thus are unable to offer robust mobile accessibility exercises in their curriculum.

Our project has several goals.

- Simplicity of Adoption: Even the most informative modules will not see widespread use if they are not easily adoptable. Modules must require modest set-up time and should be as easy to use as possible.
- Applicability: Creating relevant, real-world accessibility exercises will not only allow students to learn about relevant accessibility issues but will keep them interested in the activities due to their real-world nature.
- Addressing a Range of Skill Levels: Building accessible apps is important for all developers, regardless of experience level. While many of the modules are designed to address more complicated accessibility challenges, others are created to be simple enough for novice students and developers.

Thus far, we have created ten mobile accessibility exercises which may be found on our project website [9]. All exercises are implemented in Java for the Android platform. As many students learn Java in their undergraduate program, it will be easy for instructors to adopt the MILK modules in their courses. This set of modules is expected to grow not only from our efforts, but from contributions from other external sources. We invite other instructors, students, and developers to become involved in the MILK project and not only create new educational material, but to enhance existing modules as well.

Although there are numerous resources containing best practice information for creating accessible mobile software [3,13], to our knowledge our work is the first of its kind in that all educational modules

- are encapsulated into easily adoptable materials,
- include a simulation to the interaction barrier it addresses,
- have been reviewed by accessibility experts to ensure quality, and
- are evaluated to ensure that they meet the intended educational objectives.

The MILK modules are developed for use at any institution in a variety of courses including mobile computing, human-computer interaction (HCI), software engineering, and usability engineering.

ACTIVITIES

Although our set of activities is growing, we presently have ten accessibility exercises ranging from vision to speech-based solutions. The process outline for each activity is shown in Figure 1.

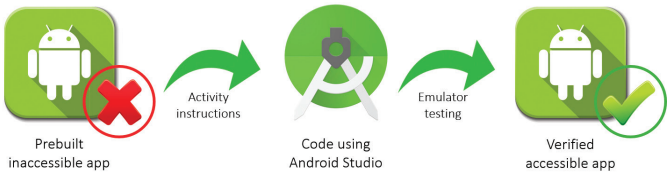

Figure 1: App Repair Process

Each of the modules contains:

1. mobile apps which contain well-defined accessibility issues;
2. documentation on the detrimental effects of the accessibility issue and how they can affect people with disabilities; and
3. step-by-step documentation on how to repair the accessibility issue, along with instructions and methods to demonstrate that the accessibility issue has been repaired.

Each activity begins with providing background information on the accessibility issue along with when, why, and how the accessibility issue occurs. When possible, students are provided with real-world examples of the accessibility problem occurring in real apps. All activities include at least one app, created specifically for each module, that contains an example of the accessibility issue. Using the steps outlined in the modules, students can recreate and experience the accessibility problem first-hand.

One challenge for the implementation of our modules was

how to emulate the experiences of an individual with different accessibility needs for a user without these needs. When possible, example apps contain a 'toggle' to allow the average user to mimic the experiences of a person with a disability as closely as possible while using the app. As an example, to demonstrate the experience of a blind user, we will add a black overlay on the top of the user interface and will activate the screen reader. The screen reader, known as TalkBack on Android devices, is an assistive technology that reads aloud the content of the screen provided that the contents are accessible. Developers are expected to implement the accessibility guidelines when creating the software so that user interface elements will be accessible to the screen reader. For example, if software displays a button that does not have a description (alternate text), the screen reader will recognize the existence of the button but will not be able to provide the blind users any details about the purpose of this button. Figure 2 demonstrates alternating what a color-blind (deuteranopia) and non-color-blind user would experience (toggle feature surrounded by red box). The feature is designed to not only create empathy for users with disabilities, but to demonstrate the importance of creating accessible apps.

Once students experience the accessibility challenge, they are provided with steps to repair the accessibility problem, including code or structural changes that are required. At the end of each module, students attempt to recreate the accessibility issue, frequently using the accessibility toggle feature. If they completed the activity correctly, they should no longer experience this accessibility challenge. To verify that the accessibility defect has been addressed properly, students can test the app using Google Accessibility Scanner [5]. It is a free tool by Google that highlights accessibility defects in an app. This will not only demonstrate that the issue has been properly addressed, but more importantly provide a sense of accomplishment to the student.

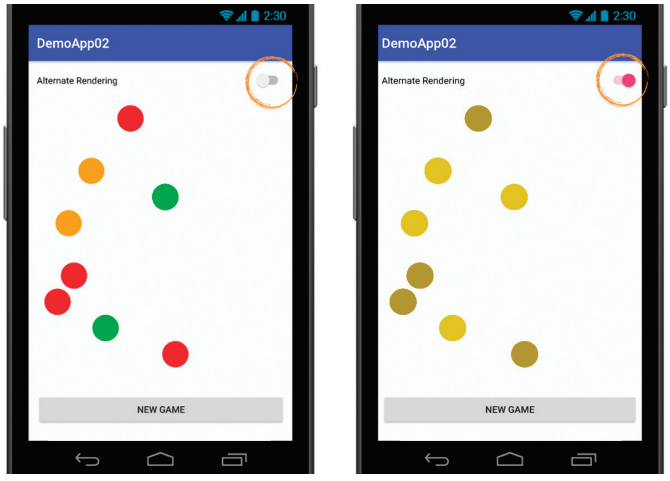

Figure 2: Sample App Showing Accessibility Toggle

The following are some of the existing modules.

- Keyboard Navigation (beginner): In this activity, developers will understand how to ensure that User Interface (UI) elements contained in the app can be accessed by a directional pad (D-pad) instead of touch. The activity addresses another key form accessibility best practice—automatically focusing on the first field in the form when the screen loads.
- Speech-Based Accessibility (beginner): Through this simple activity, developers learn how to associate descriptive text with UI elements.
- UI Element Sizing & Spacing (beginner): This activity involves small, moving buttons to simulate the behavior of a user with hand tremors.
- Video Captions (beginner): One of the accessibility features provided by the Android operating system is the ability to render captions for videos. Among other settings, users can set the default font size and colors of the captions. This activity involves associating a pre-created captions file with a pre-created video file.
- Text-to-Speech (intermediate): This activity incorporates text-to-speech functionality to provide vision-impaired users with speech-based feedback when the user interacts with the UI elements or when the app performs an action.
- Speech Based Accessibility (intermediate): The purpose of this activity is to demonstrate that Android's speech-based accessibility features are not limited to only providing static text for UI elements that are spoken when touched.
- Color-Blind Safe Accessibility (intermediate): This activity demonstrates the challenge faced by colorblind users using a simple game. In this activity, colored circles move across the screen, and a colorblind impaired user will not be able to differentiate the colors of the circles. An example of this activity is shown in Figure 2, and further details are below.
- Navigation Standards (advanced): Navigating within an app can be challenging for users if the developer utilizes a custom navigation approach. Android Studio includes a project template that provides developers with the ability to build apps that utilize a persistent "navigation drawer." This UI element provides developers with the ability to present a standard navigation mechanism for users to navigate within the app.
- Non-audio Notifications (advanced): Deaf and hard-of-hearing users are challenged when utilizing apps that have pure sound-based notifications. This activity involves the use of Android's vibration and LED API's to provide alerts to the user when a specific action/event occurs in the app.

EXAMPLE MODULE: COLOR-BLINDNESS

Color blindness or color vision deficiency (CVD) is the inability to differentiate colors that normal color observers can distinguish. There are three types of color blindness—monochromatism (or achromatopsia), dichromatism, and anomalous trichromatism. Studies show that 8% of Caucasian males, 5% of Asiatic males, 3% of African-American, and 3% of Native-American males are color deficient. The most common form of CVD is the red-green color blindness. Color is a pervasive component in Graphical User Interfaces (GUIs) and is mainly used to enhance the visual appeal of the software and to encode information (red, for in-

stance, can mean high-priority). If the designer chose a combination of colors that CVD viewers cannot differentiate, then this interface will not be accessible to a variety of different users. Colors should not be the primary means of conveying information. They can be used as an additional medium of information, but not the only one [13]. An example app is shown in Figure 2.

CONCLUSION

We have created a publicly available set of Android apps designed to not only instruct students on how to properly create accessible apps, but to demonstrate the importance and necessity of creating such apps. The modules are designed for easy adoption inside a variety of classrooms and are adaptable enough to be used in other settings such as outreach events. All activity-related material may be found on our project website [9]. ❖

Acknowledgements
Elements of this work are sponsored by a SIGCSE Special Projects Grant.

References

1. ABET definition of Engineering Design; http://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2016-2017/. Accessed 2017 December 13.
2. The ADA and Section 508; https://508compliantdocumentconversion.com/americans-with-disabilities-act/. Accessed 2018 January 13.
3. Android accessibility guidelines; https://developer.android.com/guide/topics/ui/accessibility/index.html. Accessed 2017 December 13.
4. Census bureau reports; https://www.census.gov/newsroom/releases/archives/miscellaneous/cb12- 134.html. Accessed 2017 August 1.
5. Google accessibility scanner; https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor. Accessed 2018 January 13.
6. Ko, A. and Ladner, R. E. 2016. AccessComputing promotes teaching accessibility. ACM Inroads 7, 4 (2016), 65–68.
7. Lewthwaite, S., & Sloan, D. Exploring pedagogical culture for accessibility education in Computing Science. *Proceedings of the 13th Web for All Conference*, 3 (ACM, 2016).
8. Marmarelli, T., & Ringle, M. 2009. The Reed College Kindle study; https://www.reed.edu/cis/about/kindle_pilot/Reed_Kindle_report.pdf. Accessed 2017 June 30.
9. Mobile Inclusive Learning Kit; https://milk-modules.github.io. Accessed 2017 August 1.
10. Putnam, C., Dahman, M., Rose, E., Cheng, J., and Bradford G. Best practices for teaching accessibility in university classrooms: Cultivating awareness, understanding, and appreciation for diverse users. *ACM Transactions on Accessible Computing (TACCESS)*, 8, 4 (2016),13.
11. Teach Access initiative; http://teachaccess.org/. Accessed 2017 December 13.
12. Waller A., Hanson, V. L., and Sloan, D. Including accessibility within and beyond undergraduate computing courses. *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, 155–162. (ACM, 2009).
13. Web Content Guidelines; Web Content Accessibility Guidelines (WCAG) 2.0; https://www.w3.org/TR/WCAG20/. Accessed 2017 December 13.

Yasmine N. El-Glaly
Department of Software Engineering
Rochester Institute of Technology, Rochester, NY, USA
yasmine@se.rit.edu

Anthony Peruma
Department of Software Engineering
Rochester Institute of Technology, Rochester, NY, USA
axp6201@rit.edu

Daniel E. Krutz
Department of Software Engineering
Rochester Institute of Technology, Rochester, NY, USA
dxkvse@rit.edu

J. Scott Hawker
Department of Software Engineering
Rochester Institute of Technology, Rochester, NY, USA
hawker@mail.rit.edu

DOI: 10.1145/3182184