

# **Audio Journal Android Application**

A Major Qualifying Project  
submitted to the faculty of  
WORCESTER POLYTECHNIC INSTITUTE  
in partial fulfillment of the  
requirements for the degree of  
Bachelor of Science

by  
Anna Cherkinsky  
Vansh Patel  
Reily Siegel

Date: February 27th, 2023

Report Submitted to:

Professors Rodica Neamtu and Lane Harrison  
Worcester Polytechnic Institute

## Abstract

As advancements in technology play an increasing role in people's lives, news resources have a variety of platforms to circulate information. Many news sources disseminate information through written or printed material. People with blindness or visual impairments do not have easy access to local printed news. Audio Journal (*Audio Journal*, 2021) is a nonprofit radio reading service based in Worcester, MA that provides local news to blind and visually impaired people in the Central Massachusetts region. Over the past three years, three groups of students at Worcester Polytechnic Institute (WPI) developed and expanded upon an iOS application for Audio Journal to help the organization provide their listeners with a simple way to access live broadcasts and archived program recordings (Marion, 2020; Grigolia, 2021; Robinson, 2022).

Since the app was created for iOS users, Audio Journal listeners who use Android devices could not access the organization's services through an app. Over the span of 21 weeks, our project group set out to create a similar Audio Journal app for Android devices to provide easy access to Android users. We researched Android accessibility features, such as *TalkBack*, *Magnification*, and *Text Size*, that would help Audio Journal users navigate the app. We also conducted user tests to ensure the app experience performed smoothly. The app is currently available on the Google Play Store to download [[Link](#)].

## **Acknowledgements**

We would like to thank the Executive Director of Audio Journal, Harry Duchesne, for all his guidance and help. We had an amazing experience working with him and greatly appreciate his time. We would also like to extend our gratitude to Jean Theurkauf, at Audio Journal, for providing us with technical support and helping us resolve API issues, and Julie Standrowicz, Audio Journal project manager, for giving us feedback on the app. We extend our gratitude to our advisor Professor Rodica Neamtu, who provided excellent mentorship for our project work. Her guidance and feedback helped us succeed in our project goals. We are grateful to all current and previous Audio Journal listeners and those who participated in testing our app. Finally, we would like to thank Emmanuel Ola, an iOS Audio Journal app student developer, for helping us get started with the project and its organization, as well as providing a helping hand in the development process during A-term (the first 7 weeks).

## **Authorship**

All team members worked on and contributed to both the project and report in an equal manner. The report sections were divided equally among members of the group to write individually. After each section was written, the remaining team members edited the writing as necessary. Once each member individually edited a section, the entire team edited the writing together, to make each section a representation of the team's collective work.

## Table of Contents

<b>Abstract</b> .....	2
<b>Acknowledgements</b> .....	3
<b>Authorship</b> .....	4
<b>Table of Contents</b> .....	5
<b>Table of Figures</b> .....	6
<b>1.Introduction</b> .....	7
<b>2.Background</b> .....	9
<b>2.1 VIPs and Access to Print Materials</b> .....	9
<b>2.2 Audio Journal</b> .....	11
<b>2.3 iOS Audio Journal App</b> .....	12
<b>2.4 iOS Accessibility Guidelines</b> .....	15
<b>2.5 Accessibility Guide from Android</b> .....	15
<b>2.6 Android Audio Journal App Features</b> .....	18
<b>3.Methodology</b> .....	20
<b>4.Implementation</b> .....	23
<b>5.User Study</b> .....	34
<b>5.1 Our Audience</b> .....	34
<b>5.2 Procedure</b> .....	35
<b>5.3 User Testing</b> .....	35
<b>5.4 Analysis</b> .....	36
<b>6.Future Work</b> .....	42
<b>7.Conclusion</b> .....	45
<b>8.Works Cited</b> .....	46
<b>Appendix A: IRB Consent Form</b> .....	49
<b>Appendix B: Questionnaire</b> .....	53
<b>Appendix C: Code Documentation</b> .....	57

## Table of Figures

<b>Table 1:</b> iOS and Android Accessibility Features.....	8
<b>Figure 1:</b> Accessibility color pallets suggested by Apple.....	12
<b>Figure 2:</b> Audio Journal iOS App Color Palettes.....	13
<b>Figure 3:</b> Audio Journal App Logo Icon.....	13
<b>Figure 4:</b> Development Timeline Snippet.....	21
<b>Figure 5:</b> Tasks for the milestones in Jira.....	22
<b>Figure 6:</b> Audio Journal Android App Home Screen.....	25
<b>Figure 7:</b> Audio Journal Android App Search Function.....	27
<b>Figure 8:</b> Audio Journal Android App Broadcast Episode Player.....	29
<b>Figures 9:</b> Program Page not Favorited.....	31
<b>Figure 10:</b> Program Page Favorited.....	31
<b>Figure 11:</b> Bar Graph - Feedback Navigating the Android App.....	37
<b>Figure 12:</b> Bar Graph - Android App is Similar to iOS.....	38
<b>Figure 13:</b> Bar Graph - FontSize Rating.....	38
<b>Figure 14:</b> Bar Graph: Rating App Easy to Use.....	38
<b>Figure 15:</b> Bar Graph - TalkBack Rating.....	38
<b>Figure 16:</b> Pie-Chart Feedback.....	39
<b>Table 2:</b> Average Ratings from the Questionnaire per Question.....	40

## 1. Introduction

Printed material is one of the main methods of sharing information. Newspapers, magazines, and online articles can reach a wide audience. Printed material can spread information about current events and politics, as well as education and entertainment. Local printed material can be used to reach local communities. Unfortunately, visually impaired people (VIPs) face challenges retrieving such information. It is essential for information transmitted through print material to be accessible for people with visual impairments to provide them with opportunities to connect to their community and be aware of significant events.

Technological advancement has allowed people with disabilities to enhance their experiences. Digital devices such as phones allow people to access endless amounts of information and media through the internet and apps. The function of such apps is customizable for people with disabilities through accessibility features.

Audio Journal is a nonprofit organization that provides visually impaired people in the Central Massachusetts area with audio resources for local printed news, information, and entertainment. The Audio Journal team has over 100 volunteers to read the print material live. Currently, the Audio Journal organization provides an iOS app to its listeners which was continuously developed by WPI students through IQP and MQP projects.

The features provided by the Audio Journal iOS app include:

- Listen to live broadcasts
- Listen to archived recordings
- Search programs by keyboard or voice

- Compatibility with VoiceOver and Audio Description features
- Rewinding or skipping recordings by 15, 30, or 60 seconds
- Speeding up the recordings by 1.5 or 2 times
- Favoriting programs
- Resuming the last recording played
- Viewing the daily broadcast schedule for a certain day

The iOS app had great success. Audio Journal received feedback showing interest in creating the Audio Journal app for Android devices. 38 percent of all smartphone users in the United States are Android users according to 2021 data (Business of Apps, 2022). Since the iOS app sufficiently served VIPs in its design and function, the Android app must display equivalent functionality to maintain consistency between the apps. Based on the recommendation of the 2021-2022 MQP team, the Android version of the Audio Journal app is implemented such that it can incorporate Android's accessibility features as well as match and expand upon the features offered by the Audio Journal iOS app (Robinson, 2022).

<b>iOS Accessibility Feature</b>	<b>Comparable Android Accessibility Feature</b>	<b>Function</b>
Zoom and Magnifier	Magnification	Zooms into a part of the screen
Audio Descriptors and VoiceOver	TalkBack	The screen items are read aloud and users may navigate through taps and voice control
Dynamic Typing	Text Size	Increasing text size of screen contents

**Table 1:** The table which describes iOS and comparable Android accessibility features.



## **2. Background**

In the following section, we will discuss the experience of visually impaired people (VIPs) related to media and print material, background on the Audio Journal streaming service organization, previous Audio Journal app development projects, and accessibility guidelines for iOS and Android.

### **2.1 VIPs and Access to Print Materials**

Large amounts of news and media in current times are spread through print form, whether on paper or digitally. News articles, novels, scientific journals, encyclopedias, and other publications are mostly all documented in text form. Most individuals can easily access and read these publications, but it poses a challenge to visually impaired people (VIPs) and print-impaired people. In Massachusetts, this causes a problem for 3% of the population, who are blind or visually impaired (National Federation of the Blind, 2019).

Blindness and visual impairment come in several different forms. Visual impairment is defined as a decrease in vision causing problems that are not correctable by common aids, such as glasses or other vision augmentation devices. Blindness is the inability to see due to injury, disease, or genetic conditions. Vision impairment and blindness are classified under four categories: partially sighted, low vision, legally blind, and totally blind. Partially sighted refers to someone with partial vision either in one or both eyes. An individual with low vision has a severe visual impairment, in which their better-seeing eye has a visual acuity of 20/70 or poorer and cannot improve with glasses or contacts. Individuals classified as legally blind are unable to correct their vision to 20/200 or better in their better-seeing eye, even after using visual aids.

Lastly, totally blind individuals have a complete loss of sight in both eyes (Industries for the Blind and Visually Impaired, 2021). Globally, at least 2.2 billion people have a near or distance vision impairment, with the leading causes being cataracts, unaddressed refractive error, and glaucoma (WHO, 2021).

Visually impaired and blind people struggle with accessing and reading print media, which makes the information less accessible to them. Although there are resources available to help overcome some of the disadvantages of print disabilities and visual impairments, they require ownership of expensive devices such as a computer or braille display. Many visually impaired people may not be able to afford these devices. The prevalence of vision impairment is estimated to be four times higher in low to middle-income regions compared to high-income regions, leaving visually impaired people with a lack of options (WHO, 2021). These impairments also cause disparities in the workforce, as visually impaired people have higher rates of unemployment and earn less than those without disabilities. In 2017, the employment of the working-age population (18-64) who were blind or visually impaired was 44.2%, compared to 79.4% for those without any disabilities. For the same year, the average wage for blind and visually impaired people of working age with any education level was \$37,195, while the average wage for those without any disabilities was \$50,337 (AFB, 2017).

Some groups and organizations have attempted to alleviate these issues by creating audio-based resources, such as audiobooks or public broadcasts. However, smaller local news companies reporting for small towns may not have the resources and funding available to reach these listeners. One organization trying to help solve the issue is Audio Journal, a non-profit organization based in Worcester, Massachusetts. Audio Journal records news and programs from

local newspapers and broadcasts to visually impaired people in Worcester County and Central Massachusetts.

## **2.2 Audio Journal**

Even in the digital world, a portion of media comes from printed sources that carry useful information. People with visual impairments find it difficult to use printed sources. Audio Journal is a non-profit organization located in Worcester, MA. Audio Journal serves Worcester County, providing people with visual impairments and other disabilities access to information distributed by print through live broadcasts and recordings. Hence, Audio Journal has been working to develop a website and apps geared towards low or no-vision people. Audio Journal indicates that the goal of the organization is, “to connect individuals with a visual impairment, or an inability to access print material, to their communities through broadcasting of local news, information, and entertainment, with exclusive programs and content” (*Audio Journal*, 2021). Audio Journal has loyal listeners, volunteers, and employees. The organization provides information relevant to its audience. A study conducted to understand what information visually impaired people want to receive concluded that visually impaired users may prefer to access information about visual disabilities and ways to navigate around life (Rayini, 2017). Audio Journal provides their listeners with such information, in addition to local news and events.

Audio Journal was started in the Worcester Public Library in 1987 and transferred to an office in 2003. The office incorporates accessibility for the visually impaired. Audio Journal recruits volunteers, usually among retirees, to record their broadcasts. Initially, volunteers would come into the office to cut up sections of printed media and record the text. Since the COVID-19 pandemic, many volunteers pre-record the text and send it in for a later broadcast. Alternatively,

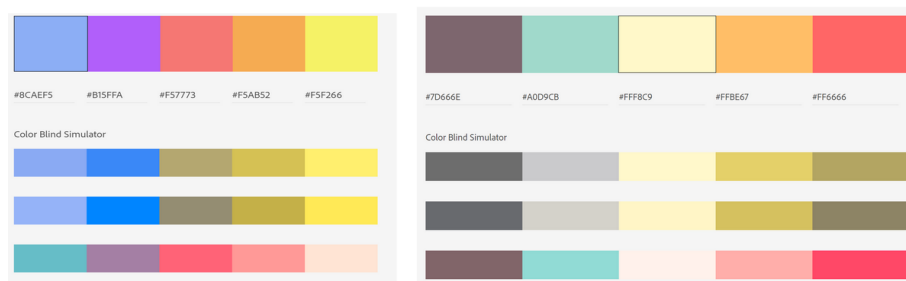
some volunteers come into the office and record live. Audio Journal has a team with diverse experience. The organization recruits interns and collaborates with Worcester Polytechnic Institute (WPI) for outreach and development. In 2019, the Audio Journal team reached out to the Worcester Community Project Center to achieve greater outreach for their broadcasting (Doyle et al., 2020). Since WPI communicated with the Worcester Community Project Center to recruit students for Interactive Qualifying Projects (IQP) and Major Qualifying Projects (MQP), several WPI student teams helped Audio Journal develop their app.

### 2.3 iOS Audio Journal App

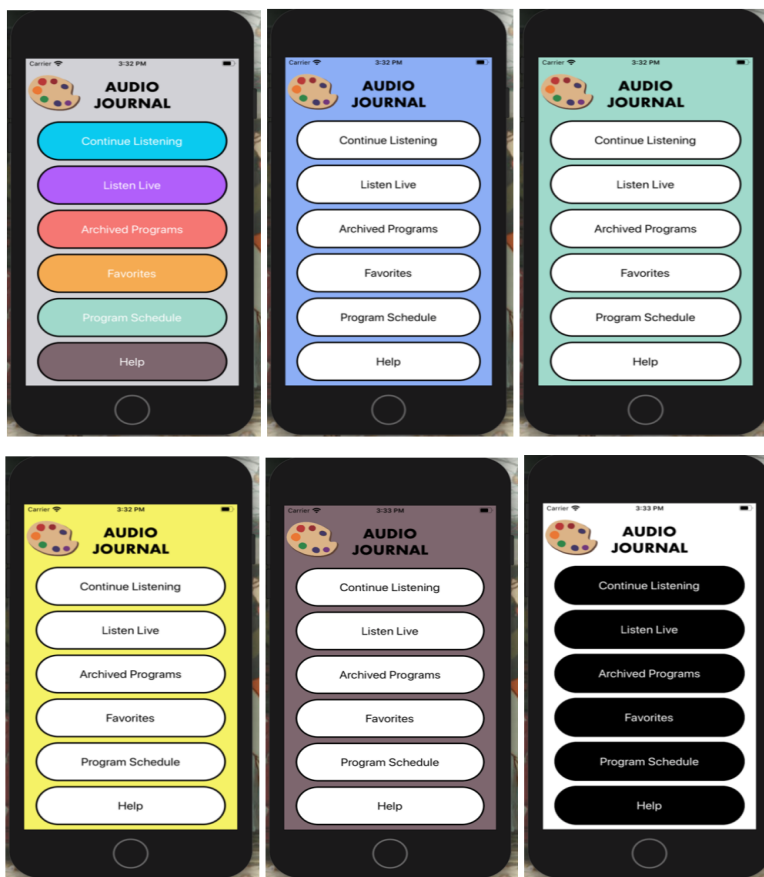
In the past three years, WPI students have been designing and developing the Audio Journal App. In early 2020, an Interactive Qualifying Project (IQP) group conducted research and made design recommendations for the Audio Journal App (Marion, 2020). The students conducted a formal user study with 21 visually impaired individuals to learn about the features most preferred for the app. The students identified features, including accessibility features such as *VoiceOver*, *Zoom*, and *Dynamic Type*, as well as color palettes and layouts for the app which accommodate the preferences of VIPs.

#### Accessible Color Palette Options

More rainbow-like palettes, similar the original Adobe XD button colors in vibrancy



**Figure 1.** Accessibility color palettes suggested by Apple.



**Figure 2.** Color palettes and layout in the Audio Journal iOS app.

In the 2020-2021 academic year, three students worked on a Major Qualifying Project (MQP) to work on implementing the Audio Journal App (Grigolia, 2021). In 2021, the “Audio Journal App” was published in the Apple App Store.

The following year, a different team of three students expanded and enriched the user experience by implementing additional features for the Audio Journal App. The group implemented a different Listen Live screen, altered the playback speed feature, added longer skip times and airtime, fixed bugs, and improved



**Figure 3:** Audio Journal App icon used for iOS and Android apps

speech and voice search as well as program schedule features (Robinson, 2022).

### *iOS App Features*

The iOS Audio Journal App accommodates visually impaired users through its features and incorporates accessibility. The Audio Journal App supports accessibility features for iOS devices. This means if the user selects any accessibility settings, the corresponding changes will affect the views, displays, and navigational function of the app accordingly. The app supports *Dynamic Type* (increasing text size), *VoiceOver*, *Bold Text*, *Larger Text*, *Zoom*, and *Magnifier*. *VoiceOver* is a feature that allows users to navigate their phones by voice, taps, and swipes. *Zoom* and *Magnifier* enlarge certain parts of the screen and let the user navigate to other parts of the screen by using three fingers. Window zoom creates a window on the screen, which displays specific screen contents. The app also implements distinct color palettes that users can switch to in order to accommodate people with color blindness. The color palettes were selected through research related to color blindness (Grigolia, 2021). The app has features such as “Listen Live”, listening to archived broadcasts, “Resume Last Broadcast”, “Favorite Programs”, “Program Schedule”, and “Help” on how to use the app. Users can search the archived programs by voice. The search function recognizes keywords and produces related results. With the media player, users may skip the audio by 15, 30, or 60 seconds, or speed up the audio by 1x, 1.5x, or 2x. People with visual impairments can use the “Favorite Programs” feature to quickly access the archived programs they like without having to navigate through the archived programs. The “Resume Last Played” feature also simplifies the process of finding the program users were previously listening to. The search function can recognize synonymical keywords to find programs, which may help completely blind individuals to search for programs by voice. The

discrete options for speeding up the audio and skipping through the audio allow visually impaired individuals to quickly rewind or speed up if they need to. Overall, the Audio Journal app succeeds in accommodating the users to get the media resources they want.

## **2.4 iOS Accessibility Guidelines**

Apple has five basic accessibility guidelines: “Design with accessibility in mind”, “Simplicity”, “Perceivability”, “Support personalization”, and “Audit and test the app or game for accessibility” (inc. A., n.d.). The guidelines explain that designing with accessibility in mind involves thoroughly brainstorming any accessibility needs which may arise to support inclusivity while keeping the design simple. Simplicity requires developers to integrate familiar and consistent interactions. To integrate perceivability, developers should make sure that the content can be perceived by sight, hearing, or touch. The app should allow the user to personalize their experience and accessibility features. Lastly, developers must test the app for accessibility, for example, by running tests with accessibility features enabled (inc. A., n.d.).

## **2.5 Accessibility Guide from Android**

Android also has guidelines for incorporating accessibility in apps which are similar to the iOS guidelines. In addition to guidelines, Android has accessibility requirements for its developers.

### **Design Guidelines**

#### Make navigation intuitive

The Android team recommends that apps follow intuitive and straightforward design guidelines, especially for major functions. Any user should be able to perform the basic

tasks in a way they are used to, meaning there should be trans-app consistency.

Accessibility features should be easy to use on the major functions. (*Accessibility*, n.d)

#### Use recommended touch target sizes

Developers should follow element size guides provided by Android. It is recommended for touch targets to be 48dp minimum. The current iOS app meets this recommendation. Visually impaired users, as well as users with manual dexterity challenges should be able to use the app, and the components should accommodate their needs. (*Accessibility*, n.d)

#### Label visual UI elements meaningfully

It is recommended to label functional UI components. Such components may include buttons, icons, tabs with icons, etc. Symbols must be made clear to all users.

(*Accessibility*, n.d)

#### Provide alternatives to affordances that time out

Android guidelines note that some controls or icons may be implemented to disappear for an amount of time on some apps. For example, playback controls may disappear five seconds after a video starts. The guidelines note that *TalkBack* feature will not read aloud limited-time controls. It is recommended to not rely on the disappearance of app components to integrate the apps with accessibility features. (*Accessibility*, n.d)

#### Use standard framework controls or enable *TalkBack* for custom controls

The guidelines recommend keeping the existing Android accessibility features in mind while developing. Developers should consider how such settings may work or affect the app experience. The guidelines indicate that such features can be turned on in settings.



For implementing custom controls, Android provides developer tools which follow the provided guidelines. The Android guidelines recommend testing the app with the accessibility features switched on. (*Accessibility*, n.d)

## **Development Requirements**

### Describe user interface controls

Android requires developers to provide descriptions for user interface components without visible text. This should be done for *ImageButtons*, *ImageView*, and *CheckBoxes*. Developers are asked to use the *android:contentDescription* XML layout attribute or *setContentDescription*. The description will be necessary for the *TalkBack* feature to describe the components to the users. (*Accessibility developer checklist*, n.d.)

### Enable focus-based navigation

Android requires developers to ensure users can navigate screens with hardware-based or software directional controls. Some examples listed are D-pads, trackballs, keyboards, and navigation gestures. *Android:focusable* may need to be used to allow a view to take focus either during *Magnification* or *TalkBack*. Focusable parameter allows views to be interacted with through keyboard, clicks, or taps. (*Accessibility developer checklist*, n.d.)

### Custom view controls

Accessibility interfaces should be implemented with custom interface controls built. Support Library will help implement the latest accessibility features.

### No audio-only feedback

The requirements state that audio feedback must always have visual corresponding feedback to accommodate users with hearing disabilities. An example provided by the guidelines is that sound alerts for messages must have notifications. (*Accessibility developer checklist*, n.d.)

### Test

Developers are required to test their apps by using accessibility features. Some examples include directional controls and eye-free navigation. (*Accessibility developer checklist*, n.d.)

## **2.6 Android Audio Journal App Features**

### **General Features**

The Audio Journal app requires a navigation interface and a media player interface. Android development tools provide these features. The *MediaPlayer* class and *ExoPlayer* library can be used to control audio or video playback as well as streams, although it is not thread safe (*MediaPlayer : android developers*, n.b.). Android development tools allow us to create a mirrored version of the iOS Audio Journal app.

## **Accessibility**

Android provides users with several accessibility features which correspond to the iOS ones described above. The alternative features for corresponding iOS accessibility features are *TalkBack* for Apple *VoiceOver*, *Magnification* for *Zoom*, and *TextSize* for *DynamicType*.

### *TalkBack*

*TalkBack* allows users to control their device by having components read aloud to them. Users may swipe left or right to navigate screens. On each new screen, *TalkBack* will announce the components and where the focus is. Users can double tap to select an option. (*Move around your home screen with Talkback - Android accessibility help*. Google., n.d)

### *Magnification*

Like iOS *Zoom*, the user can tap anywhere on the screen except the keyboard or navigation bar to magnify the screen or screen components through *Magnification*. The user can use two fingers to move around the screen and pinch to adjust the magnification. Users can use magnification for the entire screen to touch and hold anywhere on the screen or partial to only keep the zoom on a particular part of the screen. (*Use magnification on your screen - android accessibility help*, n.d. 3)

### *Text Size*

Like iOS *Dynamic Type* accessibility, users can increase or decrease text size. The Android accessibility settings provide users with a slider to adjust text size. The screen components adjust accordingly. Users can also make the text bold and adjust text and

background colors. They can select the dark theme or invert the colors on the screen.

Users can remove animations, use a large mouse pointer, use high text contrast, or make the screen extra dim.

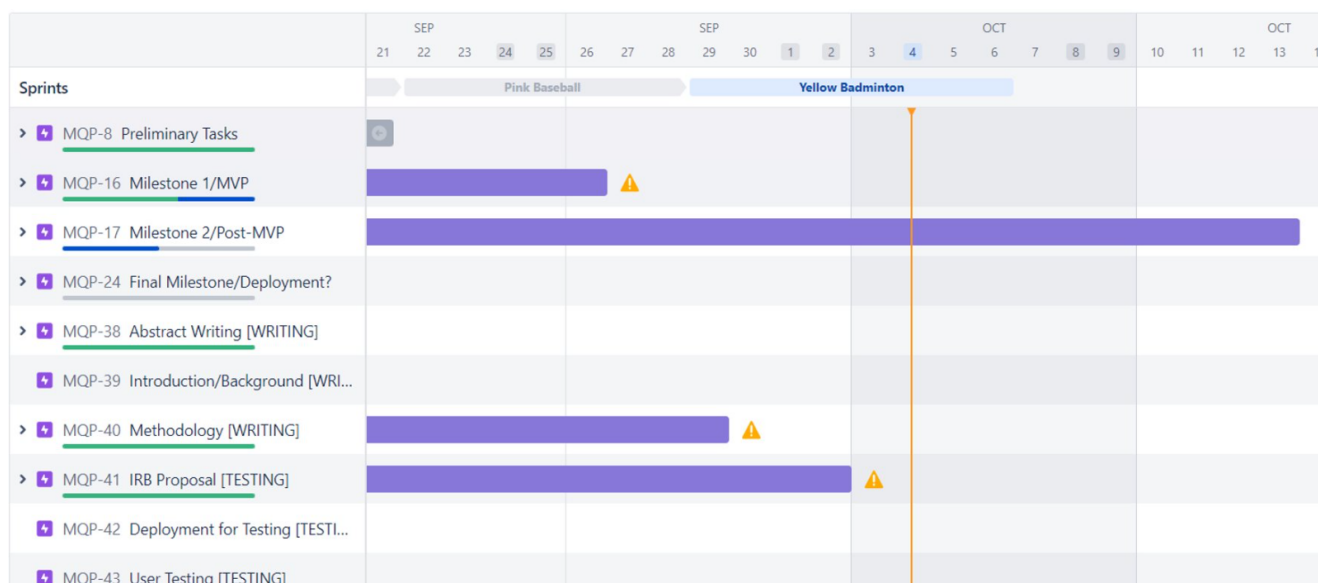
Android provides other accessibility features. For example, for Android 10 and up, users can set up multi finger support. This will allow users to have a braille keyboard in *TalkBack* settings (*Use the talkback Braille Keyboard - Android accessibility help*, n.d. 4). Android also provides Accessibility Suite, which is an app with other accessibility options, such as vibration response. Android will allow the users to place input by voice if Gboard is installed. (*Type with your voice - android - gboard help.*, n.d. 2)

### **3. Methodology**

In this section, we first discuss our development process.

Agile development is a set of practices that better enable a software engineering team to iterate quickly and respond to changing design requirements (*What is Agile Software Development?*, n.d.). In agile development, all requirements are added as tasks to a backlog. Tasks from the backlog are assigned to a sprint, which lasts one week. At the beginning of the sprint, each task is assigned one, two, or three points, which indicates the expected difficulty of completing the task. At the end of the week, any uncompleted tasks roll over to the next sprint. This structure encourages tasks to be split into small subtasks which can easily be completed. This one-week limit also forces continuous consideration of task prioritization.

The core communication component of our agile development methodology is our daily stand-up, where most of our scheduled communication occurs. The purpose of the stand-up is to have a brief meeting, at most 15 minutes, to discuss the progress made in the previous day, and the next task to accomplish. This meeting is not meant to be in-depth but merely serves as a check-in to ensure all team members are on track. All in-depth discussion happens outside the stand-up, usually on an ad-hoc basis.



**Figure 4:** Snippet of our timeline. Each portion of development is broken down into milestones with assigned dates. The yellow warning signs show us that some features we are working on may go beyond the date of the milestone. At the top bar, you can see the sprints.

The image displays two screenshots of Jira milestones. The left screenshot is titled "Final Milestone/Deployment?" and shows a list of child issues under the heading "Child issues" with a progress bar at 0% Done. The issues are: MQP-62 Help Menu, MQP-52 Change Color Palette, MQP-29 Dark Mode, MQP-54 Favorites, MQP-55 Search, and MQP-56 Resume Last Played. The right screenshot is titled "Milestone 2/Post-MVP" and also shows a list of child issues under the heading "Child issues" with a progress bar at 0% Done. The issues are: MQP-59 Playback Speed, MQP-53 Browse Past Programs, MQP-27 Voice Control, MQP-57 Weekly Broadcast Sch..., MQP-58 Media Controls, and MQP-61 Archived Program Pa...

**Figure 5:** Tasks for the milestones in Jira.

To facilitate iterative development, our development team uses Jira (Atlassian, n.d.), a tool commonly used in the industry to track development progress. Jira maintains our backlog of tasks, as well as the assignment of tasks to sprints. Using this tool, we can track how many points we complete in an average sprint, as well as what percentage of tasks assigned to a sprint are completed. In addition to Jira for tracking tasks, we use GitHub to track our code. GitHub is a version control system that has sophisticated tools for reviewing code changes, such as creating a pull request and asking users to rebase their code by working through conflicts of different code versions before merging the code into a single code base (*Let's build from here*, n.d). As a policy, our team requires each code change to have the approval of at least two other team members. This helps reduce programming errors and is made manageable by the atomic changes agile development encourages.

## 4. Implementation

We used Jetpack Compose (*Jetpack Compose UI app development toolkit*, n.d.) for development, which is a Kotlin (*Kotlin programming language*, n.d.) based Android toolkit to build native UI. Kotlin is a programming language interoperable with Java (*What is Java technology and why do I need it?*, n.d.) and used for Android app development. While Kotlin is object-oriented based and mostly used for back-end development, Jetpack Compose uses composable functions for visuals, merging backend and frontend development into a similar coding pattern. Each Composable function may operate as an object, which in turn can be called in other functions to reuse pieces of UI.

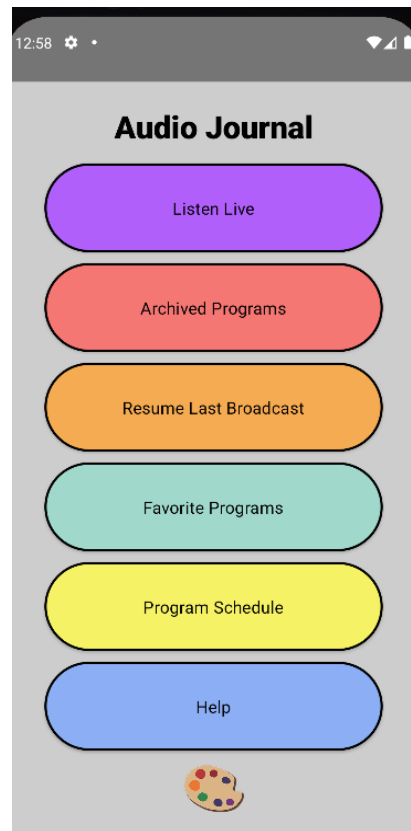
To organize our development process, we divided our implementation into three main milestones. During the first milestone, we worked on the minimum viable product (MVP) which included the home screen, compatibility with Android accessibility features, and Listen Live feature. For the second milestone, we worked on implementing the program schedule, archive program menus, and the archive program player. For the final milestone, we added a search function for archived programs, the ability for users to add a program to their favorite list of programs, the “Resume Last Played” broadcast feature, the help menu, Bluetooth connection, and the color changer feature.

Our initial goal was to complete all three milestones by the end of A-term (the first seven weeks of the timeline), but our priorities adjusted as we used more time to learn about Android development. So, we emphasized completing the first two milestones instead. By the end of the term, we were able to complete the first milestone. This included solving the hurdle of pulling data from Audio Journal’s API into the app to display the appropriate data and have the Listen Live feature work accordingly. We also accomplished the second milestone, as we implemented

features based on data from the API including displaying the program schedule, archive program menus, and the archive program player. Our goal for B-term (the second seven weeks of the project timeline) was to complete the final milestone and conduct user testing to use feedback for app improvement. We were able to implement the tasks in the final milestone before the testing session, and we used utilized feedback to improve details in our app accordingly.



## Home Screen, Archived Programs, and Menus



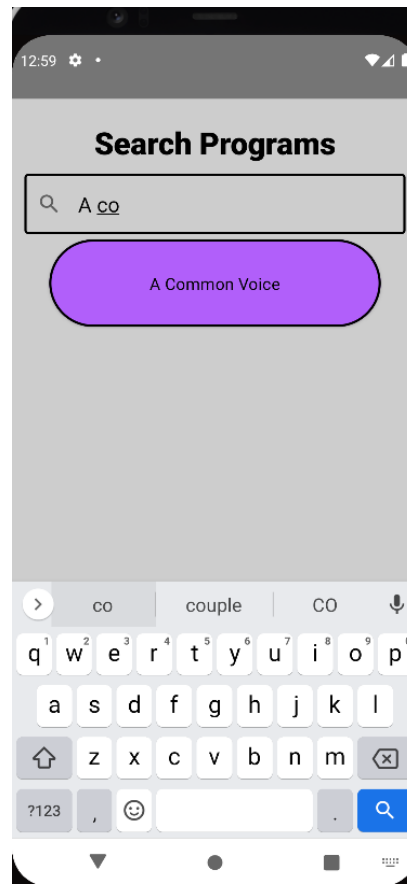
**Figure 6:** Audio Journal Android App home screen

The home screen served as a template for other menu screens. We worked to visually match the menus and their layout to those of the iOS Audio Journal app. We ended up using abstractions for different components of the screen. We have a composable function for a page that takes on the color theme, a header, and the composable unit containing content for the screen. We reuse this abstracted component for all pages. For the menus, we create a list of menu items that we use to make buttons from the Menu composable. Each menu item also carries a universal resource identifier (URI) link for navigation from one screen to another. Each button navigates the user to an appropriate page. We adapted the design of the app to fit the native Android functionality, which led to some discrepancies between the iOS and Android versions.

For example, since Android has a back button, we removed the implementation of the back button in the top left corner of the screen that is seen in the iOS app. For the home screen, specifically, we also pass in a button composable for changing the color palette.

For the archived program menus, we had to obtain information regarding the program categories, archived program options, and program episode information. To achieve this, we were provided application programming interface (API) endpoints from our sponsors which contained JavaScript Object Notation (JSON) (*Introducing json*, n.d.) strings for the current archived program information. For each menu, we implemented networking workflows (using the Retrofit library) which take in information from the API via GET requests (*HTTP request methods - http: MDN*, n.d.). These requests are handled in view models, which contain the app's logic separate from UI code, and the information is saved into Kotlin objects. From the Kotlin objects, we create menu items that reflect the current archived program data that is available to the organization. The intention is that if there are any alterations to these programs or their categories, the Audio Journal organization may edit their API data, and the app will reflect the changes accordingly. Since it may take a few milliseconds to process the calls, we added loading indicators that display while data is loading. The current archive program navigation flow begins with program categories, which leads to a list of programs in the category, which then lists information for that category (description) and a list of episodes. Episode buttons lead to the media player containing the specific archived program episode selected.

## Search Function



**Figure 7:** Audio Journal Android App search function

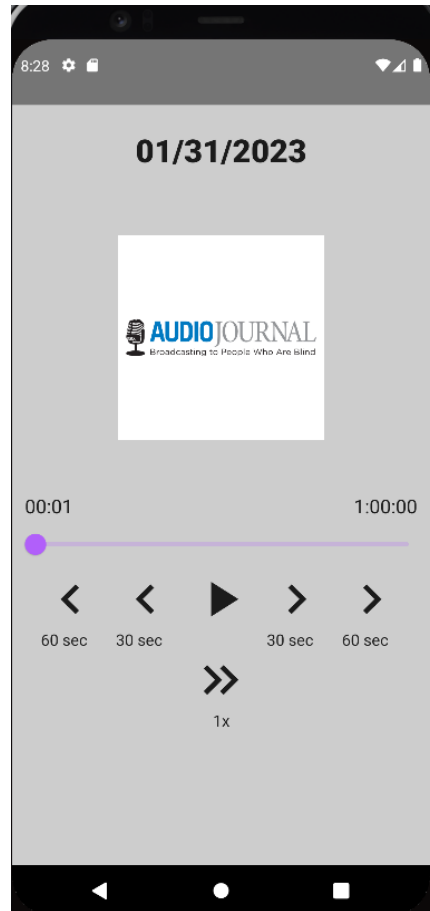
Searching for archived programs can be done through the program categories screen. A button to search programs navigates the user to a screen with a search bar. The user can input the name of the program and programs closest by relevancy will appear on the screen. The search bar is implemented from scratch using composable functions which accessed the keyboard. State objects track the user input as it is entered live. We then use a full program list to select and sort the programs by the relevancy of input. Since the devices we worked with had access to Gboard (*Gboard - the google keyboard - apps on Google Play*, n.d.), the users may also enter the name of

the program by pressing the microphone which appears on the keyboard. The user may speak the program name as input into the text field to search for a program.

### **Search by Voice**

Our implementation of the Audio Journal app uses the Google Assistant API, which allows users to search for a program using their device's built-in voice assistant functionality. This approach has several benefits for accessibility. Most Android phones have the assistant mapped to a physical button, enabling VIPs to access the voice search feature. Furthermore, this search functionality works even when the Audio Journal app is not opened on the phone. Together, this presents a much more streamlined search interface than hierarchical navigation with TalkBack. Users can ask Google Assistant to open an Audio Journal program using a keyword. If the keyword is unique to the name of a certain program the program information page will open in the app, where the user can access episodes. If the keyword matches multiple programs, it is automatically typed into the Audio Journal search function and lists the related programs that users can choose from.

## Listen Live, Archived Broadcasts, and Media Player



**Figure 8:** Audio Journal Android App broadcast episode player

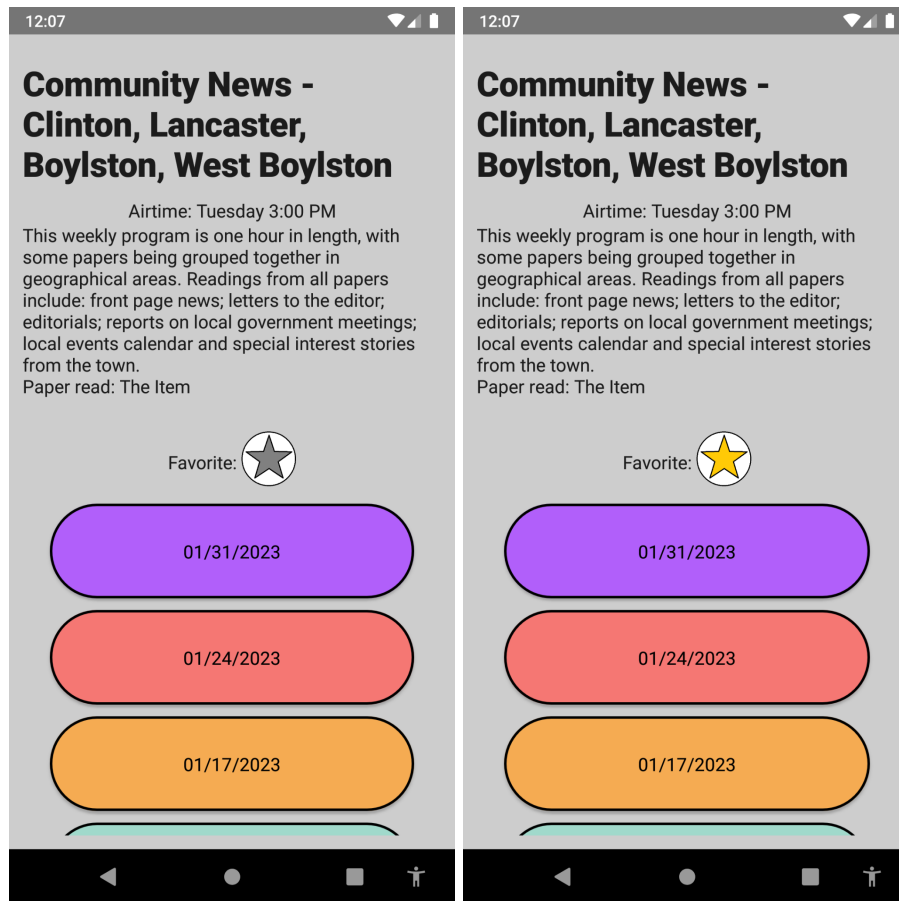
We used ExoPlayer, a third-party Android library, to implement the “Listen Live” and “Archived Programs” features. The ExoPlayer API uses a URI to download audio data. This URI is provided in the Audio Journal API. The player features are implemented through composable functions. Each control is an icon button that reads information from the Exoplayer instance. For example, the skip buttons read the live timestamp of the recording to skip a certain number of seconds. The media player we implemented for archived recordings contains features for pausing and playing the recording, skipping 30 or 60 seconds forward and backward, speeding up the audio by 1.5 or 2 times, and slowing down the audio by 0.5 times. These are the features present

in the iOS Audio Journal app. Similar to the iOS app, the “Listen Live” player only contains the pause and play buttons.

### **Resume Last Played**

We implemented a feature that allows the user to easily return to the broadcast they were previously listening to. When a user navigates away from an archived program recording, the recording and timestamp they left at are saved in memory. When a user clicks the Resume Last Played button located on the Home Screen, the app navigates to the media player and plays the last recording listened to, starting from the timestamp the program was stopped at.

## Favoriting Archived Programs



**Figures 9 and 10:** The favorites selection in the program screen.

On the program screen, the user can choose to favorite a program by clicking the Favorites button, which will turn yellow when the user adds the program. The user can then navigate to the “Favorite Programs” page from the home screen and will see a menu of buttons containing the programs they added to their list of favorite programs. When they click on a button, they will be directed to the appropriate program. When a user clicks on a favorites button that contains a yellow star, the specific program will be removed from the list of favorite programs and will be removed from the favorites page, with the star turning gray.

## **Color Palette Changer**

The color palette changer was implemented by tracking the local theme, which is derived from color scheme objects containing colors for the background, buttons, and text. We included all the color palettes implemented in the iOS app and extracted the hexadecimal codes from iOS Audio Journal app reports.

## **Data Persistence**

“Resume Last Broadcast”, “Favorite Programs”, and the color palette changer all rely on data persistence, permitting the user to reload the app while allowing data related to those features to persist. If the user adds a program to their list of favorite programs, the program will still appear in their list of favorite programs after they reload the app. Similarly, the resume last played will play the previously played broadcast, and the color palette settings will persist. This was achieved using Jetpack DataStore (*App Architecture: Data layer - datastore*, n.d.), which is a data storage solution allowing the storage of key-value pairs to add and read from inside of an application. For the “Resume Last Broadcast” feature, two key-value pairs are stored. One pair contains the last played broadcast name stored as a string, and the second pair contains the timestamp the user left off on. For the “Favorite Programs” feature, each program that is favorited is stored as a key-value pair, with the name of the program as the key and its URI as the value. For the color palette feature, one pair is saved with the value representing the index location of the color choice in the array of palette options.

## **Program Schedule**

The program schedule is a feature that lists the airtimes of broadcasts for a given day of the week. The user will first navigate to a menu containing options for each day of the week. They can choose a day of the week and see a listing of all upcoming programs for that day and



the time they will be broadcasted at. The schedule information was retrieved from the Audio Journal production API, similar to the archived program information and the broadcasts. Since the schedule retrieved data from the organization directly, the schedule information will be updated accordingly.

### **Help Menu**

The “Help” menu contains information about the features listed above. The user can utilize the “Help” menu to learn more about each feature and how to use the app. Each menu item leads to a page with a description of the feature. The feature information was hardcoded into a hash table.

### **Bluetooth Connection**

If an Android device is connected to a TV or speaker device through Bluetooth, Audio Journal broadcasts will be played on the connected audio device.

### **Composable Functions and Navigation**

Our app implementation uses Jetpack Compose’s navigation component to create a smooth and seamless navigation experience for users. A navigation graph is the most important piece to implementing navigation, which uses a *NavController* to keep track of the screen a user is on. The navigation graph contains routes based on the page to be navigated. These routes call the appropriate composable function with arguments passed in to display UI. Some routes use arguments based on parameters, such as the name of an archived program or the day of the week which a user wants to view the schedule for, allowing for more flexibility in navigating the app.

### **Accessibility Compatibility**

During implementation, we ensured that our app was compatible with Android accessibility features which are potentially very important for the visually impaired. We included

content descriptors for each user interface (UI) object which would be read out loud if the user turns on the TalkBack feature. The sizes of our objects and text use scalable pixels which can be manipulated when the size is altered using the Display and Text Size accessibility features in the device settings. We also tested our app with the Magnification accessibility feature, and it works accordingly.

## **5. User Study**

We conducted a user study to test the implementation of the Audio Journal Android App. The 10 users were recruited for testing with the help of our sponsors at Audio Journal. The testers with minimal visual impairments were recruited on the WPI campus by the student development team.

### **5.1 Our Audience**

The targeted users for the Audio Journal Android App are visually impaired people with Android devices. Much of Audio Journal's audience is visually impaired. Our sponsors helped us contact 10 potential users to conduct this user study. We also tested the app on people with minimal visual impairment (i.e., no visual impairment or visual impairments which could be resolved with glasses or contact lenses) as a control group to determine if there is a difference in the quality of user experience. We recruited some of the iOS Audio Journal app testers from previous Audio Journal projects to compare the usability of our app to the iOS app. The feedback from our users provided us with a guide for any improvements or recommendations for future projects.

## 5.2 Procedure

To start testing, we needed the user study forms to be approved by the IRB (Institutional Review Board) at WPI. We submitted the consent form for IRB approval (see Appendix A). To conduct the user study, we created an app bundle using Android Studio. Then, we deployed the temporary testing bundle onto an Android device. The link to the test version of the app was sent to the testers, which allowed them to access the app on their device. (*Build and test your android app bundle* : *android developers*, n.d.) We conducted user testing in-person on WPI's campus. The testers who could not meet with us tested their app on their own time. We provided the asynchronous testers with our contact information, so they could get in touch if they ran into app-related issues. At the end of each testing session, we distributed a questionnaire in verbal form for feedback (found in Appendix B). We conducted the questionnaire over a phone call with asynchronous testers and recorded the answers as they were given to us.

## 5.3 User Testing

We organized in-person and asynchronous user testing sessions. During the in-person testing session, testers came to WPI's campus to test our app. The in-person session contained four volunteers, two of which were completely blind, one with a visual impairment, and one with no visual impairments. We started by reading the consent form to our testers and obtaining their consent. The volunteers with complete blindness used the TalkBack accessibility feature to test our app. Each of our team members supervised the testers by reading them the instructions, providing any help if needed, and answering their questions. At the end of the testing phase, we verbally asked the survey questions to each tester and filled out the questionnaire for them. The questionnaires were published using Qualtrics, a survey tool used by WPI.

We also had six asynchronous testers, none of whom had visual impairments. We sent the link to an app APK file which allowed users to download the app onto their Android devices. We provided them with testing instructions and app installation instructions. A link to the online questionnaire was sent for them to fill out on their own time. At-home testers were given three days to test and answer the questionnaire. We also provided testers with our contact information to reach out for help or to ask questions. The testers received the consent form to sign electronically, as well.

## **5.4 Analysis**

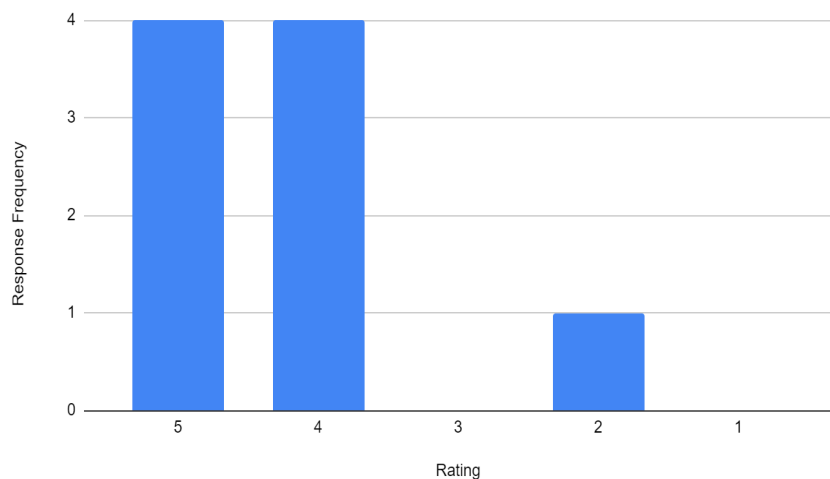
### *Feedback*

Testers were instructed to complete a questionnaire after the testing session to provide feedback for the Android app. The first few questions asked users about their background, such as if they are Audio Journal listeners or if they use accessibility features on their devices. In regards to app feedback, testers filled out a section with different prompts regarding their app experience and provided a response using a slider numbered 1 to 5, where 5 meant “strongly agree” and 1 meant “strongly disagree” in response to the statement. This feedback helped us quantify the user experience. There is also an open response section where users could answer questions in more detail and provide any additional feedback they may have. We conducted a verbal questionnaire for testers with visual impairments and filled out their responses accordingly. We have a total of 10 responses, with some responses being incomplete.

Four out of 10 testers were Audio Journal listeners, three of whom used a TV or smart speaker for listening to broadcasts. Three testers have previously used the iOS Audio Journal App to access the broadcasts. Only one participant had previously participated in the testing of

the iOS Audio Journal app. Three testers used TalkBack while testing the app, and one tester used the Display/Text size feature.

"Navigating the app was easy and intuitive" - 1= Strongly disagree 5= Strongly agree/Response Frequency



**Figure 11: Bar graph demonstrating the rating feedback for navigating the app.**

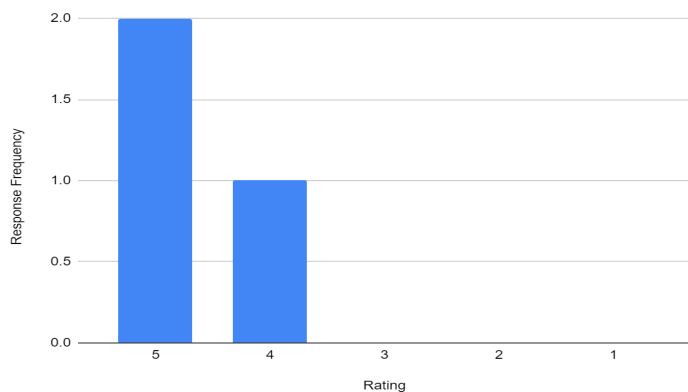
Portrayed in the bar graph to the left are the responses and the number of respondents per option. For app navigation, only one response was below 4 (agree), which was 2 (disagree), meaning navigation was not clear for one tester. The response for the prompt regarding the app responding as

expected had “strongly agree” and “agree” as responses. Another bar graph on the following page shows how the testers rated

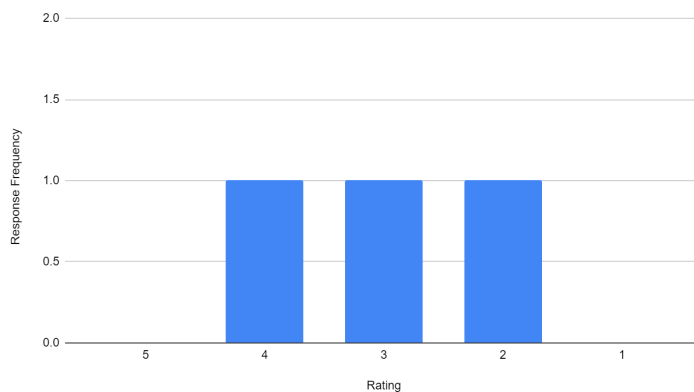
the performance of different accessibility features while using the app.

No user used magnification or voice input. We received a lower score for the font size of the app. One user responded neutrally to using TalkBack. For other questions (see figures 12-16) users responded with high ratings, except for the question asking if using the app was easier than most apps, where one respondent said they disagreed. In terms of general app use the ratings were quite high, with a few issues to address.

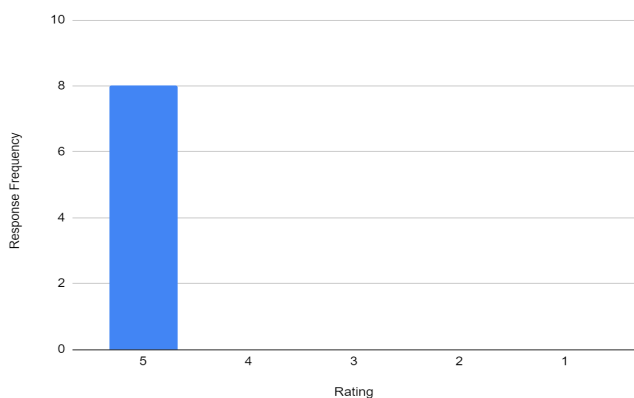
If used or tested iOS App: The app is very similar to the iOS Audio Journal app - 1= Strongly disagree 5= Strongly agree



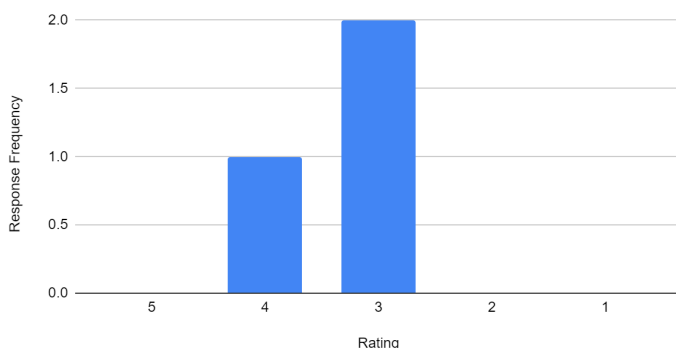
If you used accessibility features: Accessibility features were easy to use with the app - Font size: 1= Strongly disagree 5= Strongly agree



I was easily able to listen to Audio Journal's live broadcast - 1= Strongly disagree 5= Strongly agree



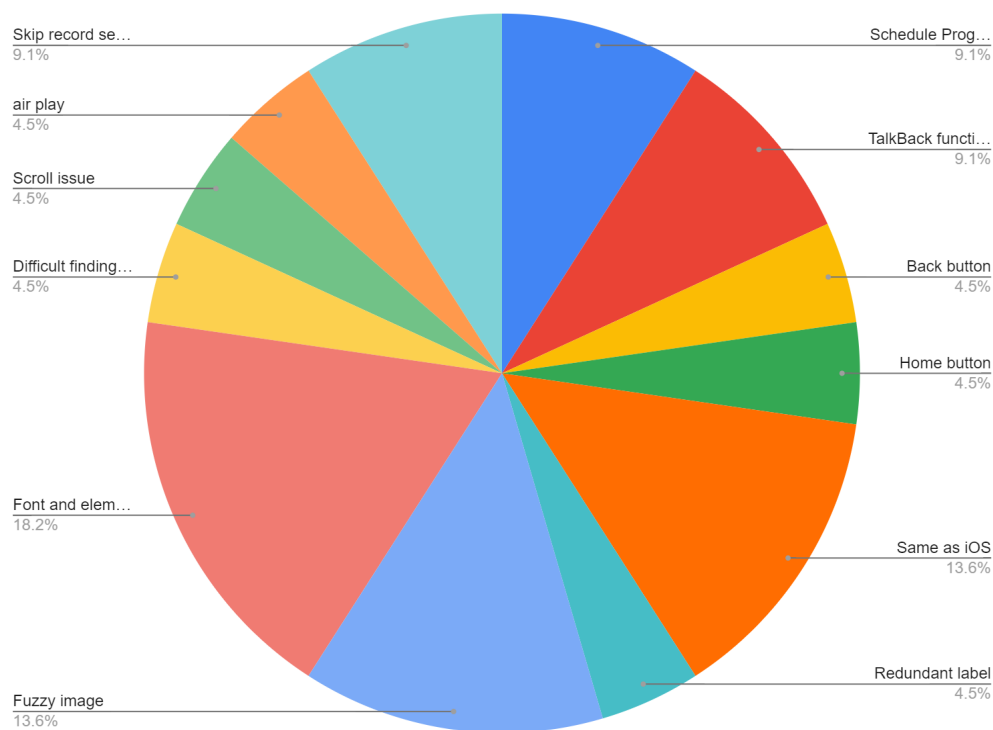
If you used accessibility features: Accessibility features were easy to use with the app - TalkBack: 1= Strongly disagree 5=...



**Figures 12, 13, 14, 15:** Bar graphs depicting the response frequencies for different ratings of the following questions. The responses which gave ratings of ‘2’ came from the same tester.

We also received some feedback in the open response section. The pie charts below illustrate the frequencies of keywords that appeared in written feedback. The font size was mentioned by 3 testers. We received some positive feedback regarding our app being very similar to the iOS version, and that using TalkBack on our app was quite easy for one of the testers who uses the iOS Audio Journal app with the VoiceOver feature.

Frequency of Feedback Terms



**Figure 16:** Pie chart representing frequency of themes in the open response section of user feedback

The table below shows average response ratings between visually impaired testers and testers without visual impairments. For the most part, the feedback from visually impaired testers was positive. The testers noted that the app seemed to be very similar to the iOS version. One critical issue we discovered along with a visually impaired tester is that some screen objects had extraneous screen descriptions, which caused some word repetitions and redundancy while the TalkBack feature was in use. The testers without visual impairments also provided positive responses, as well as some criticism, which discussed blurry images and the small text size and size of visual elements.

Question/Statement	Average Rating for visually impaired testers	Average Rating testers with no visual impairments
Navigating the app was easy and intuitive	4.5	4.14
The app performed in a way that I was expecting.	4.5	4.8
If you used accessibility features: Accessibility features were easy to use with the app - TalkBack:	3.5	N/A
I was able to easily find the color palette which fits my usability preferences and needs	3	4.4
I was easily able to listen to Audio Journal's live broadcast	5	5
I was easily able to find past broadcasts that I wanted to listen to	4.5	5
I was able to easily favorite broadcasts I liked, and listen to them later	4	5
I could receive the broadcast schedule information for a specific day of the week	5	5
Using this app was easier than using most other apps	5	4
If used or tested iOS App: The app is very similar to the iOS Audio Journal App	5	4.5

**Table 2:** Displays the statement from the questionnaire and the average ratings given by the tester, depending on whether the tester has visual impairments or not.



### *Criticisms*

We received a few points of constructive criticism regarding our app and areas to fix. First and foremost, the font size for components was too small even for someone with no visual impairments. We addressed this issue by setting a larger font size for devices with the default text size setting so that the text could still be altered through the Font Display accessibility setting. We also made our interactive components (buttons and text size) relative to the screen size of the device. The size relation of the components provides a more visually intuitive experience to the user and prevents components from appearing too small relative to the screen. We also received feedback that the back button appeared to be missing, which is present on the iOS app. Earlier in the development process, we decided to remove the back button we created, since most Android devices have a built-in back button, and we did not want to confuse the user. We decided against adding the app-specific back button since Android users would be used to the functionality of the native back button on their devices. We received feedback regarding blurry images used for the splash screen and media player screens, which we updated with higher-quality images. We also made the mistake of putting a description for every screen item, which confused testers who used TalkBack, since some text and icon objects had the same description. For instance, the Favorites button had a text “Favorites” label as well as the button icon label, which confused the testers. This was adjusted by combining the text and button item into one TalkBack item so that when a user scrolls on the favorites button, only one description is read. The descriptor has also been adjusted to tell the user whether or not the specific program is part of the list of favorite programs or not, and what clicking on the button would do. Descriptors on images that are not necessary are now removed. One of the testers noted that the app does not have a similar feature implemented to AirPlay on iOS which will allow users to connect the app to smart objects (such

as a TV or Speaker). We considered implementing this feature but decided to not do so because of time constraints and how complex the codebase would become if the feature was added.

### *Difficulties*

One of our major difficulties was that we could only test with 10 people, three of whom had visual impairments. Scheduling testing sessions was also a small roadblock due to the need to manage everyone's (as well as the testers') schedule. Since our testing group and the number of targeted users were small, we may not have received thorough feedback and could have missed potential issues.

## **6. Future Work**

The success of Audio Journal's app has inspired several accessibility projects for the Worcester area. Another WPI student team worked concurrently alongside our team to implement the Worcester Art Museum accessibility app which was based on the Audio Journal iOS app. The student team created the app, which is available on both iOS and Android devices. Developing similar apps for Worcester-based organizations that focus on accessibility can help Worcester residents greatly. We are making three main recommendations regarding the Audio Journal-inspired accessibility apps, as well as general recommendations for Audio Journal about the app's future. We want to discuss recommendations for creating the EcoTarium accessibility app, creating a general Worcester accessibility app, and recommendations for Audio Journal and future development teams concerning app maintenance and development.

### *EcoTarium*

The EcoTarium is the museum of natural sciences located in Worcester, MA. Similar to the Worcester Art Museum, the EcoTarium is looking to make the museum experience more accessible to people with visual impairments by creating an app that can be used to access information about items on display at the museum.

For app development, we recommend following a model similar to the development of the Worcester Art Museum App, considering the app was implemented successfully. We recommend creating the app in React Native for the app to be available on iOS and Android devices.

### *Worcester App*

As more organizations in Worcester seek to develop accessibility apps for the visually impaired, we recommend creating a single app that contains all of the accessibility apps integrated for the Worcester area. The app would allow the user to choose whether to receive information from the Worcester Art Museum, the EcoTarium, or Audio Journal. The app should be implemented in React for consistency. If other facilities wanted to implement similar apps to accommodate VIPs, the new app can be added to this general accessibility app package.

### *Recommendations for Audio Journal*

With the current state of the app, the major development process has been completed. There are several minor features we would recommend for Audio Journal to implement into the iOS and Android apps. First, we recommend providing a home button on the schedule, program, and help screens. Currently, the only way to get back to the home screen is by clicking the back

button several times. Based on our feedback, we also recommend, if the resources are available, dividing the broadcasts into chapters and allowing the user to skip to a certain chapter in the recording. Finally, we recommend reviewing and maintaining the similarity between iOS and Android apps. The Android app has the ability for the user to open a particular program in the Audio Journal app by voice using Google Assistant. The iOS app does not currently have a corresponding feature implemented.

Since there are no essential features that need to be added to the Audio Journal app, the Audio Journal organization is now focusing on app maintenance and bug handling. We were notified that Audio Journal will be working with a software maintenance company, WireReady, to maintain the app codebase. Our recommendation is to provide an email or a phone number to app users to help them contact the WireReady team to report bugs. The Audio Journal organization can hire an assistant to gather and keep records of bugs which would be submitted to the WireReady team. We have provided an email and a phone number in the app to report bugs to Audio Journal.

#### *Recommendations to future student developers*

We are providing several recommendations to future accessibility app development teams. Our major challenge was recruiting testers. We had a total of 10 testers, three of which were visually impaired. Since our accessibility app was created to serve people with visual impairments, our testing pool for the target audience was small in comparison to our testing goal. We recommend future accessibility app teams determine the testing period during the development process. Developers are recommended to schedule a longer testing period, which may last a month or more. Recruitment for testing should begin a few weeks before the testing

period, even if the app is still in development. To recruit testers who would be part of the targeted audience of the app, we recommend using several recruitment sources. The developers should recruit testers with the help of their sponsor, as well as reach out to local facilities, even paying them an in-person visit, if necessary.

## **7. Conclusion**

Overall, we were able to successfully develop the Audio Journal App for Android devices. While we had some complications with tester recruitment, we received enough feedback to improve the app before deployment. The Audio Journal App is now publicly available on the Google Play Store to download [[Link](#)]. While the majority of visually impaired Audio Journal listeners have iOS devices, according to our sponsor at Audio Journal, there are several people experiencing loss of sight who use Android devices. Expanding the platform that provides access to Audio Journal's service will help visually impaired Android users to have access to news and journal material.

Audio Journal's overall success has expanded its impact on the Central Massachusetts area VIPs. With the Worcester Art Museum implementing a similar accessibility app, and the EcoTarium looking to do the same, Audio Journal has influenced the Worcester community to provide solutions to the problems visually impaired individuals face in an increasingly digital society.

## 8. Works Cited

- Accessibility developer checklist*. Android Developers. (n.d.). Retrieved September 19, 2022, from <https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/ui/accessibility/checklist.html>
- Accessibility*. Android Developers. (n.d.). Retrieved September 19, 2022, from <https://stuff.mit.edu/afs/sipb/project/android/docs/design/patterns/accessibility.html>
- American Foundation for the Blind. (2017). Expanding possibilities for people with vision loss. <https://www.afb.org/>
- App Architecture: Data layer - datastore*. Android Developers. (n.d.). Retrieved December 12, 2022, from [https://developer.android.com/topic/libraries/architecture/datastore?gclid=Cj0KCQiAnNacBhDvARIsABnDa6\\_rm8SbQy1f-HQLkZmEKFPhaQ3xO5zSQshQzoHw3hjSLGRWSqQ8IfsaAqJNEALw\\_wcB&gclsrc=aw.ds](https://developer.android.com/topic/libraries/architecture/datastore?gclid=Cj0KCQiAnNacBhDvARIsABnDa6_rm8SbQy1f-HQLkZmEKFPhaQ3xO5zSQshQzoHw3hjSLGRWSqQ8IfsaAqJNEALw_wcB&gclsrc=aw.ds)
- Atlassian. (n.d.). *Jira: Issue & project tracking software*. Atlassian. Retrieved October 6, 2022, from [https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=136973856930&campaign=18440774103&creative=624486809641&device=c&keyword=jira&matchtype=e&network=g&placement=&ds\\_kids=p73335831609&ds\\_e=GOOGLE&ds\\_eid=700000001558501&ds\\_e1=GOOGLE&gclid=Cj0KCQjw-fmZBhDtARIsAH6H8qhLmfsL26pPFo5kx1fuy\\_s\\_HPXdB1RpslzPCCYOiArAjeDXet7qA\\_oaAmWGEALw\\_wcB&gclsrc=aw.ds](https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=136973856930&campaign=18440774103&creative=624486809641&device=c&keyword=jira&matchtype=e&network=g&placement=&ds_kids=p73335831609&ds_e=GOOGLE&ds_eid=700000001558501&ds_e1=GOOGLE&gclid=Cj0KCQjw-fmZBhDtARIsAH6H8qhLmfsL26pPFo5kx1fuy_s_HPXdB1RpslzPCCYOiArAjeDXet7qA_oaAmWGEALw_wcB&gclsrc=aw.ds)
- Audio Journal*. AudioJournal. (2021, September 29). Retrieved October 5, 2022, from <https://audiojournal.org/>
- Blindness statistics | National Federation of the blind*. National Federation of the Blind. (2019, January). Retrieved October 3, 2022, from <https://stage.nfb.org/resources/blindness-statistics>
- Blind vs. visually impaired: What's the difference?* IBVI. (2020, April 2). Retrieved October 3, 2022, from <https://ibvi.org/blog/blind-vs-visually-impaired-whats-the-difference/>
- Build and test your android app bundle : android developers*. Android Developers. (n.d.). Retrieved September 30, 2022, from <https://developer.android.com/guide/app-bundle/test>
- Curry, David. "Android Statistics (2022)." *Business of Apps*, 11 Jan. 2022, <https://www.businessofapps.com/data/android-statistics/>.

- Google. (n.d.). *Gboard - the google keyboard - apps on Google Play*. Google. Retrieved December 12, 2022, from [https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin&hl=en\\_US&gl=US](https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin&hl=en_US&gl=US)
- Google. (n.d.). *Move around your home screen with Talkback - Android accessibility help*. Google. Retrieved September 19, 2022, from [https://support.google.com/accessibility/android/answer/6283678?hl=en&ref\\_topic=10601570](https://support.google.com/accessibility/android/answer/6283678?hl=en&ref_topic=10601570)
- Google. (n.d.). *Type with your voice - android - gboard help*. Google. Retrieved September 19, 2022, from <https://support.google.com/gboard/answer/2781851?hl=en&co=GENIE.Platform%3DAndroid>
- Google. (n.d.). *Use magnification on your screen - android accessibility help*. Google. Retrieved September 19, 2022, from <https://support.google.com/accessibility/android/answer/6006949?hl=en>
- Google. (n.d.). *Use the talkback Braille Keyboard - Android accessibility help*. Google. Retrieved September 19, 2022, from <https://support.google.com/accessibility/android/answer/9728765?hl=en>
- Grigolia, I., & Nelson, K. (2021). *App for Audio Journal*. : Worcester Polytechnic Institute.
- HTTP request methods - http: MDN*. HTTP | MDN. (n.d.). Retrieved December 12, 2022, from <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
- Inc., A. (n.d.). *Accessibility*. Accessibility - Foundations - Human Interface Guidelines - Design - Apple Developer. Retrieved September 19, 2022, from <https://developer.apple.com/design/human-interface-guidelines/foundations/accessibility/>
- Industries for the Blind and Visually Impaired. "Blind vs. Visually Impaired: What's the Difference?" IBVI, 2 Apr. 2020, <https://ibvi.org/blog/blind-vs-visually-impaired-whats-the-difference/>.
- Introducing json*. JSON. (n.d.). Retrieved December 12, 2022, from <https://www.json.org/json-en.html>
- Jetpack Compose UI app development toolkit*. Android Developers. (n.d.). Retrieved December 11, 2022, from <https://developer.android.com/jetpack/compose>
- Kotlin programming language*. Kotlin. (n.d.). Retrieved December 12, 2022, from <https://kotlinlang.org/>
- Let's build from here*. GitHub. (n.d.). Retrieved February 22, 2023, from <https://github.com/>

Marion, B. A., Grigolia, I., & Doyle, R. B. (2020). Improving Media Access for Audio Journal's Print Disabled Listeners. Retrieved from <https://digitalcommons.wpi.edu/iqp-all/5736>

*MediaPlayer* : *android developers*. Android Developers. (n.d.). Retrieved September 19, 2022, from <https://developer.android.com/reference/android/media/MediaPlayer>

National Federation of the Blind. "Blindness Statistics." National Federation of the Blind, <https://stage.nfb.org/resources/blindness-statistics>.

Rayini, J. (2017). LIBRARY AND INFORMATION SERVICES TO THE VISUALLY IMPAIRED PERSONS. 14.

*Reviewing the Disability Employment Research on People who are Blind or Visually Impaired: Key Takeaways*. The American Foundation for the Blind. (n.d.). Retrieved October 3, 2022, from <https://www.afb.org/research-and-initiatives/employment/reviewing-disability-employment-research-people-blind-visually>

Robinson, B., Farwell, C., & Orozco, A. (2022). *Audio Journal App Continuation*. : Worcester Polytechnic Institute.

*What is Agile Software Development?* Agile Alliance |. (2022, May 26). Retrieved October 6, 2022, from <https://www.agilealliance.org/agile101/>

*What is Java technology and why do I need it?* Java.com. (n.d.). Retrieved December 12, 2022, from [https://www.java.com/en/download/help/whatis\\_java.html](https://www.java.com/en/download/help/whatis_java.html)

World Health Organization. (2021, October 14). *Blindness and vision impairment*. World Health Organization. Retrieved October 3, 2022, from <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>



## **Appendix A: IRB Consent Form**

### **Informed Consent Agreement for Participation in Research Study**

**Student Investigators:** Anna Cherkinsky, Vansh Patel, Reily Siegel

**WPI Faculty Advisors:** Rodica Neamtu, Lane T. Harrison

#### **Contact information:**

##### **Student Investigators:**

Anna Cherkinsky, Email: [acherkinsky@wpi.edu](mailto:acherkinsky@wpi.edu)

Vansh Patel, Email: [vspatel@wpi.edu](mailto:vspatel@wpi.edu)

Reily Siegel, Email: [rsiegel@wpi.edu](mailto:rsiegel@wpi.edu)

##### **Faculty Advisors:**

Rodica Neamtu, Email: [rneamtu@wpi.edu](mailto:rneamtu@wpi.edu)

Lane T. Harrison, Email: [lharrison@wpi.edu](mailto:lharrison@wpi.edu)

**Title of Research Study:** Android App for Audio Journal

**Sponsor:** Audio Journal

#### **Introduction:**

You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents

information about the study so that you may make a fully informed decision regarding your participation.

**Purpose of the Study:**

Audio Journal is a radio reading service in Worcester, MA that broadcasts information found in printed material for blind and visually impaired people. We are a student team which is working with Audio Journal to implement the Android version of the Audio Journal app, which is used to access the broadcast material. Audio Journal already has an Apple iOS app. The purpose of the study is to test the Audio Journal Android App. Our overall goal is for users to navigate the app effectively, and that all the features work as intended. Visually impaired people are the focus of this study, but we would also like people with no or minimal visual impairments to participate in the study. We will be writing a report and analysis of our findings.

**Procedures to be followed:**

- We will send the study materials to the participants who will participate asynchronously and those with Android devices over email.
- We will provide Android devices to use to in-person participants who do not have access to an android device.
- We will instruct you (in person or by email) on how to open the test version of the app (bundle).
- You will be asked to person some tasks regarding navigating the app.
- If you participate asynchronously, you will be provided a week to do the testing.
- At the end, you will be asked to answer a set of questions.
- Your identity and responses to individual questions will be kept confidential.

**Risks to study participants:**

Standard risks associated with using smartphones

**Benefits to research participants and others:**

You will be the first to have access to the Android Audio Journal App. Your participation will help us to improve access to Audio Journal's content, giving listeners more control over how they listen.

**Your participation in this research is voluntary.**

Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

**For more information about this research or about the rights of research participants, or in case of research-related injury, contact:**

WPI IRB Manager: Ruth McKeogh, Tel. 508 831-6699, Email: [irb@wpi.edu](mailto:irb@wpi.edu) or

WPI Human Protections Administrator: Gabriel Johnson, Tel. 508 831-4989, Email:

[gjohnson@wpi.edu](mailto:gjohnson@wpi.edu)

**By signing below**, you acknowledge that you have been informed about the study, and consent to be a participant in the study described above. If you are under the age of 18, your parent or legal guardian must also sign this form. Make sure that all of your questions are

answered before signing this form. You are entitled to retain a copy of this consent agreement.

**Printed Name** \_\_\_\_\_

**Study Participant Signature:** \_\_\_\_\_ **Date:**

**(if participant is under 18 years of age)**

**Printed Name of Parent/Guardian:** \_\_\_\_\_

**Parent/Guardian Signature:** \_\_\_\_\_ **Date:**

**(for investigator use only)**

**Investigator Signature:** \_\_\_\_\_ **Date:**

## Appendix B: Questionnaire

Which accessibility options did you use when navigating the app?

Check all that apply

- TalkBack
- Display/Text Size
- Magnifier
- Audio Descriptions
- Voice Control
- No Accessibility Settings

Are you a current or former Audio Journal listener?

If Yes:

What is/was your main way of listening to broadcasts? Check all that apply

- Smart Speaker
- Smart phone
- Website link
- Landline phone
- Cable tv

Receiver

Have you used the Audio Journal website to access previous broadcasts?

Yes

No

Have you used the Audio Journal app on Apple devices to access previous broadcasts?

Yes

No

Have you previously participated in testing the Audio Journal app?

Yes

No

### **Opinion Based Numerical Questions:**

Answer the following questions with a scale from 1 to 5, with 1 meaning you strongly disagree, and 5 meaning you strongly agree.

1 - 2 - 3 - 4 - 5  
Strongly Disagree Strongly Agree

Navigating the app was easy and intuitive

The app performed in a way that I was expecting

If you used accessibility features: Accessibility features were easy to use with the app

Font size

TalkBack

Magnification

Voice control

I was able to easily find the color palette which fits my usability preferences and needs

I was easily able to listen to Audio Journal's live broadcast

I was easily able to find past broadcasts that I wanted to listen to

I was able to easily favorite broadcasts I liked, and listen to them later

I could receive the broadcast schedule information for a specific day of the week

Using this app was easier than using most other apps

If used or tested iOS App: The app is very similar to the iOS Audio Journal app

**Opinion Based Qualitative Questions:**

Did the app features work like you expected? If not, which features did not work like you expected?

Did you have any difficulty using the app? If yes, what was difficult?

If you had experience with the iOS app, how does the Android app compare?

(Only answer the next question if you used TalkBack when testing the app) When navigating the app with TalkBack, did all of the buttons have descriptive labels? If not, which labels would you change?

Are there any features you feel are missing? If so, what would you add?

Will you use the app in the future?



## Appendix C: Code Documentation

Link to GitHub Repository: <https://github.com/eoola/Audio-Journal-Android>

### MainActivity

File that loads the rest of the pages in the app (inside of the *onCreate()* function).

The Composable function *GetAirtimeMap* computes the airtime for each program based on the program schedule, while the function *SetupNavGraph* sets up the navigation of the app.

### StoreData

File that leverages Android's use of DataStore, which uses key-value pairs to store user data locally to remember for the next instance of opening the app. For the app, the key value pairs stored are as followed:

- Favorite Programs: For each favorited program, the name of the program as a key with its URI as the value (program navigation route) is stored, and mapped out to display the corresponding buttons
- Resume Last Broadcast: 3 key value pairs are stored.
  - The first is the key called "title", with the value being the name of the program last played.
  - The second key is called "playTime", which stores a long value with the last timestamp the user played the broadcast at.
  - The third key is called "programURL", which stores a string containing the route to the specific episode.

- Color scheme: 1 key value pair is stored, which is a key called “colorIndex” containing an integer corresponding to the color scheme stored in memory.

## Navigation (NavGraph.kt)

*SetupNavGraph* – The main function which sets up navigation for the app, and contains *NavHost*, which allows you to specify the first screen to navigate to upon loading the app

Each composable in the *NavGraph* file allows you to specify a page to go to by calling the composable function that renders its’ screen. Routes are specified inside of the parentheses of the composable, which can also accept arguments.

*MenuItem* – Allows for the creation of a singular menu item (Ex. Buttons on Home Screen)

```
composable( route: "home") { it: NavBackStackEntry
    HomeView(
        menuItems = listOf(
            MenuItem( title: "Listen Live", uri: "listen-live"),
            MenuItem( title: "Archived Programs", uri: "archived-programs"),
            MenuItem( title: "Resume Last Broadcast", uri: "resume-last-broadcast"),
            MenuItem( title: "Favorite Programs", uri: "favorite-programs"),
            MenuItem( title: "Program Schedule", uri: "program-schedule"),
            MenuItem( title: "Help", uri: "help")
        ),
        navController = navController,
        setColorScheme = setColorScheme
    )
}
```

Ex. The composable is named “home” and calls HomeView to render the MenuItem on the main page.

**Subfolder Overview:**

- **Models:** Contains files for objects that parse data from Audio Journal's API and puts them in a structured format to use (Ex. *Category*)
- **Navigation:** Contains files relating to the use of navigation from one screen to another in the app, with *NavGraph* being the main file to handle navigation
- **Repositories:** Contains only the *AudioJournalService* file, which is used to directly call API routes and parse them into data. Uses the Retrofit library.
- **UI:** Contains two subfolders to separate reused UI components, the "component" and "theme" folders. The component folder contains files for UI components related to the data and navigation of the app, while the theme folder contains files strictly related to the appearance of the app (color, text appearance, etc.).
- **View:** Contains the code for the views, which are the pages the user will see when using the app. Most views make use of the *Menu* composable, which makes it easier to display MenuItem buttons in the style seen in the app.
- **ViewModels:** Contain the logic of the app regarding parsing data from the *AudioJournalService* functions to the view. Essentially, they call the appropriate function and parse its data to be sent to the view so that it can be displayed correctly. One note is that since the favorites feature does not rely on the use of the API at all, it

only contains code to get data from *StoreData* to perform the appropriate task (adding/removing a program from favorites, resume last broadcast, color scheme)

```
@GET("schedule")
suspend fun getSchedule(): Response<Map<String, Map<String, String>>>
```

Ex (*AudioJournalService*). The “schedule” route is called from the Audio Journal API using a GET request, and its data is parsed into the function *getSchedule()*

## Gradle Scripts

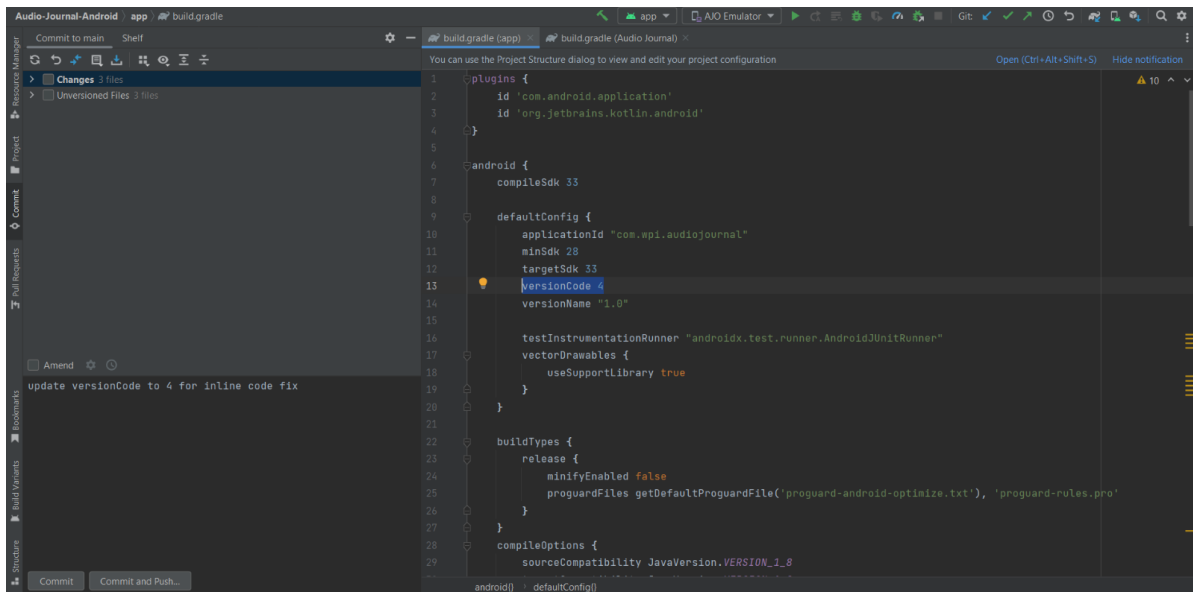
At the bottom of the project directory in Android Studio, you will see a dropdown named “Gradle Scripts”. Two files are important: *build.gradle (Project: Audio\_Journal)* and *build.gradle (Module:app)*. These two files help manage the build of the app and dependencies (library versions, language versions, etc).

- *build.gradle (Project: Audio\_Journal)*: This file allows you to update the Jetpack Compose version (based on the latest stable version) located in “ext”, along with other general Android configurations.
- *build.gradle (Module:app)*: This file allows you to define important information including the targeted/minimum Android SDK version of the app, the versionCode of the app (see section: Updating the app), and the dependencies in the app (external libraries the app uses).

## Updating the app

### Step 1: On Android Studio/IDE

After adding/removing code as needed, make sure to go to the *build.gradle (Module:app)* file. In the field named “android”, look for the “defaultConfig” field. There should be a value named “versionCode”. Make sure to change that value, preferably to the current value plus one (Ex. 4 -> 5). For every new app update, the version code must be unique. The versionName can also be updated but is not necessary.



```

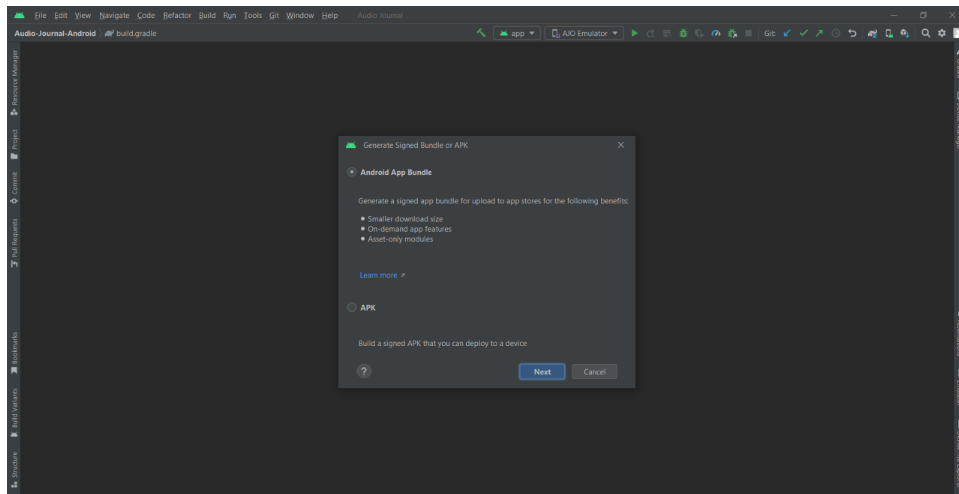
1  plugins {
2      id 'com.android.application'
3      id 'org.jetbrains.kotlin.android'
4  }
5
6  android {
7      compileSdk 33
8
9      defaultConfig {
10         applicationId "com.wpi.audiojournal"
11         minSdk 28
12         targetSdk 33
13         versionCode 4
14         versionName "1.0"
15
16         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
17         vectorDrawables {
18             useSupportLibrary true
19         }
20     }
21
22     buildTypes {
23         release {
24             minifyEnabled false
25             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
26         }
27     }
28     compileOptions {
29         sourceCompatibility JavaVersion.VERSION_1_8

```

Ex. Updating the version code. It is highlighted in blue above.

If using Git, make sure to Commit and Push all changes to Git to make sure the main branch is updated appropriately. Then, in the top menu bar, click “Build”, then “Generate Signed Bundle / APK...”. Make sure “Android App Bundle” is selected and click “Next”. Then, make sure the

key store path is correct, along with key store password and key alias/password. Next, in the build variants, make sure “release” is highlighted in blue. Click create, and now the app bundle should be created after a few minutes. A prompt in the bottom right-hand corner should show up saying the app bundle was created. Click the blue “locate” link to open the app bundle in your file directory.

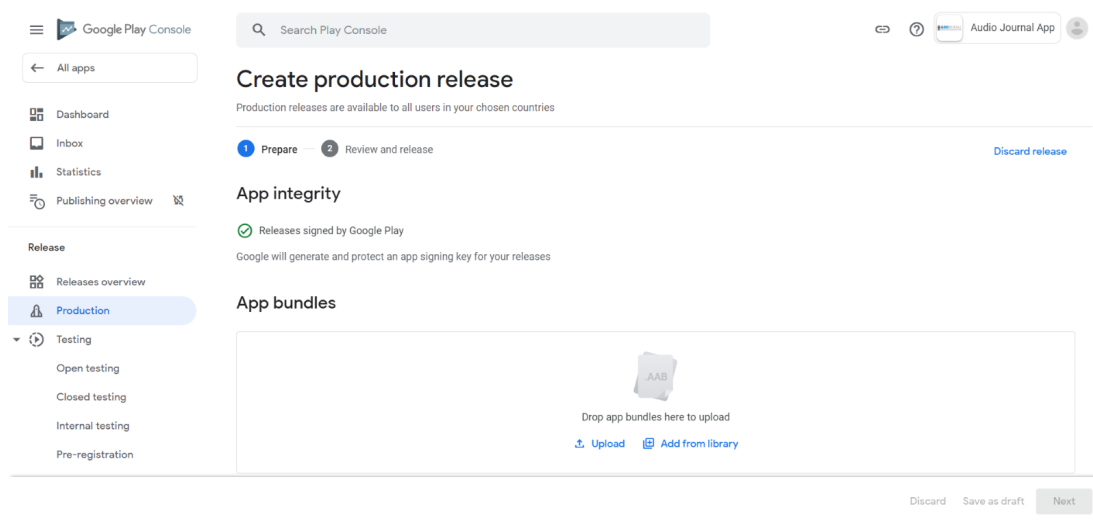


Ex. After clicking “Generate Signed Bundle / APK...”, you should see this popup. Make sure “Android App Bundle” is selected.

## Step 2: On Google Play Console

Log into the [Google Play Console](#) (click the blue “Go to Play Console” button and login with appropriate account) under the Audio Journal developer account. Click on the Audio Journal app under “All Apps”. In the left sidebar, click the “Production” button (located in the Release subsection). Then, in the top right, click the blue “Create new release” button. Copy and paste or

drag and drop the app bundle from your file directory we created in Android Studio into the “App bundles” box. Give the release a name (Ex. Audio Journal v1.0) and add any release notes in between the tags if needed. Then, click the blue “Next” button in the bottom right-hand corner. Make sure to resolve any errors if described. If there are no errors, click the blue “Start rollout to Production” button to send the update to Google to review. This process should take anywhere from a few hours to a few days to complete. Once done, the [store listing](#) should update with the new version of the app.



Ex. You should be on this page after clicking the “Create new release” button.