# Final Design

## App

The following images represent the final design and are referenced in the following sections for element descriptions and verification.

|  |  |  |  |
| --- | --- | --- | --- |
| Icon  Description automatically generated  (a) Welcome page | Graphical user interface  Description automatically generated  (b) BT device info | Icon  Description automatically generated  (c) BT connection pending | Icon  Description automatically generated  (d) BT connection successful |
| Icon  Description automatically generated  (e) BT connection failed | Graphical user interface, text, application, chat or text message  Description automatically generated  (f) Default mode home screen | Graphical user interface, text, application, chat or text message  Description automatically generated  (g) Training mode home screen | Text  Description automatically generated  (h) Info mode |
| Icon  Description automatically generated  (i) Retrieving results | Icon  Description automatically generated  (j) Default mode result | Icon  Description automatically generated  (k) Training mode question | Graphical user interface, application, icon  Description automatically generated  (l) Training mode correction |
| Icon  Description automatically generated  (m) View scan results | A screenshot of a red phone  Description automatically generated with low confidence  (n) Accuracy warning | (o) Contaminants warning |  |

**Figure 1: AmIGarbage User Interface App Screens**

## Data Transfer

Data transfer between the app and the Jetson NANO are all performed over Bluetooth with JSON formatted strings. \*\*\*\*\*\*

## Data Storage

All data, apart from the name of the Bluetooth connection to the Jetson NANO device, is stored in an XSV file on linux, that can be easily parsed. The current implementation allows for easy data import into SQL servers for future further development while still satisfying the current product minimum requirements. The following table notes all data stored on the Jetson NANO device. Note that this table would have a unique primary key in a SQL database, and is emptied after each new scan. The logic for the app needs only to scan the most recent line; as such, file needs only one line, and saves on storage.

**Table 1: Sample Data File Stored on Jetson NANO Available for Parsing and Sending to App**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| [string] | [int] | [boolean] | [int] | [boolean] | [int] | [int] |

The following are descriptions of each data type:

**wasteType** – a string of either “plastic”, “paper”, “glass”, “metal”, or “garbage”

**totalScans –** an int of total scans since shipping or most recent software update

**incorrect** – a Boolean of weather the scan was accurate. Only overwritten in training mode

**totalIncorrect** – an int of the total number of incorrect scans. Only overwritten in training mode

**glass** – a Boolean of weather the scan was *first* identified as glass. Only overwritten during scan

**totalGlass** – an int of the total number of scans identified as glass

**incorrectGlass** – an int of the total number of scans misidentified as glass

Since the logic of the app parses the last line, the shipped XSV file has the following information:

**Table 2: Shipped Dynamic XSV File Values**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| Null | 0 | False | 0 | False | 0 | 0 |

A static XSV file is also needed. This will only be overwritten during software updates. This table also has a primary key.

**Table 3: Static Data Table for Accuracy Calculations**

|  |  |
| --- | --- |
| **shippedAccruacy** | **shippedInaccuracy** |
| [double] | [double] |

The following 4 paths are demonstrated are all of the possibilities that can update the dynamic data table.

### Happy Path – Default Mode

The scan identifies an object as “glass”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| Null | 0 | False | 0 | False | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| glass | 0 | False | 0 | False | 0 | 0 |

### Happy Path – Training Mode

The scan identifies an object as “glass”. User confirms the object was “glass”.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| Null | 0 | False | 0 | False | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| glass | 1 | False | 0 | True | 1 | 0 |

### Incorrect Scan (Non-glass) – Training Mode

The scan identifies an object as “paper”. User confirms the object was “plastic”. Note this can also be used when “glass” is first identified as another type, but not the other way around.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| Null | 0 | False | 0 | False | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| plastic | 1 | True | 1 | False | 0 | 0 |

### Incorrect Scan (Glass related) – Training Mode

The scan identifies an object as “glass”. User confirms the object was “garbage”.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| Null | 0 | False | 0 | False | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wasteType** | **totalScans** | **incorrect** | **totalIncorrect** | **glass** | **totalGlass** | **incorrectGlass** |
| garbage | 1 | True | 1 | True | 1 | 1 |

# Constraint – Must be at least 80% accurate at sorting waste + Must be at most 10% inaccurate at sorting Styrofoam and glass when shipped

From the controls testing, the system is at the following verified accuracies for sorting:

**Table 4: Accuracy of All Waste Types from Controls Testing**

|  |  |
| --- | --- |
| **Waste Type** | **Accuracy (%)** |
| Paper |  |
| Plastic |  |
| Metal |  |
| Glass |  |
| Garbage |  |
| *Average* |  |

The average shipped safety accuracy is therefore expected to be 95%, and the inaccuracy of distinguishing glass vs Styrofoam is 5%.

To maintain these accuracies, users can use *Training Mode* in the app. The following logic is tested and is used in *Training Mode*. To measure the current accuracy, the following equations are used:

For example, if the system is shipped (or has had a software update) at a rated 95% accuracy, and the system has scanned 10 items with 2 items that were misidentified, the current accuracy is:

The same logic is used to calculate the current inaccuracy of identifying glass vs Styrofoam, and is shown below:

Fr example, if the system is shipped (or has had a software update) at a rated 5% inaccuracy for identifying Styrofoam, and 2 Styrofoam items were identified as glass out of a scan of 10 glass items, then the current inaccuracy is:

# Constrain – Must Identify + Sort at Least 6 Items / Minute

The assumptions for this constraint are:

1. This identification process is done on default mode (with no user feedback)
2. This identification process is measured only by scan and app performance, with no expectation on the user’s end to press the ‘view scan’ button
3. This is assuming the Bluetooth connection has been established and is consistently connected throughout the process.

For verification of design, the prototype would normally be tested through performance integration tests. This general process is noted below. Note that 100 tests are performed to give an accurate depiction of the performance of the process.

Diagram

Description automatically generated

**Figure 2: Performance Integration Test Logic for Bluetooth Communication**

Since the software team and the control team are not located in the same vicinity, and they are limited by inability to test the application across platforms (the controls team does not have access to a computer platform that can run an ios app), some basic calculations are used instead to estimate the performance of the Bluetooth communications. These calculations are below. The size of the data transfer is the max string size, which is “garbage”. This equates to 7 bytes. As such, a buffer of 1 byte is included to ensure bit alignment. The Bluetooth adapter used is rated at 3Mbps [1]. The download to Jetson NANO is one request, which can be a Boolean of 1 byte. Additionally, the app can make use of Swift’s inherent *SlowDataStore* class which works with asynchronous data. This is noted to take 0.5 – 2 seconds and is the slowest possible call to the UI [2].

Since the total time maximally takes 2 seconds, and provides for a safety factor of approximately 3, this criterion is met.

# Constraint – Must have an Assembled Size of less than 0.05m^3

This constraint is not applicable to software.

# Constraint - Meets Canadian Safety Standards

In order to meet Canadian safety standards for the user interface, all accessibility requirements and recommendations outlined by the Apple Developer community were met for the development of the iPhone SE compatible app (ios 14.4.1). The following is a summary of the 14 accessibility requirements and recommendations, along with verification that they were met.

## All controls and interactive elements have a hit target of at least 44x44 pt [3].

All interactive element types are summarized in the table below. All occurences of the same type of element are noted in the column ‘Location’ and refer to Figure 1.

**Table 5: Interactive Element Sizes in AmIGarbage? App**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Image** | **Width [pt]** | **Height [pt]** | **Location** |
| Navigation button | A picture containing text  Description automatically generated | 251 | 51 | a,b,d,e,j,l,n |
| Text bar | Shape, rectangle  Description automatically generated | 250 | 44 | b |
| Main action button | A close-up of a logo  Description automatically generated with medium confidence | 247 | 114 | f,g |
| Tab button | A picture containing player, light  Description automatically generated | 59 | 52 | f,g,h |
| Confirmation button | A picture containing text, clipart  Description automatically generated | 112 | 51 | k |

## Don’t override the platform gestures [3].

Apple gestures are summarized in the table below for ios 14.4.1.

**Table 6: IOS 14.4.1 Platform Gestures** [3]

|  |  |
| --- | --- |
| An illustration of iPhone with an arrow that indicates swiping up from the bottom. | **Go home.** |
| An illustration of iPhone with an arrow that indicates swiping down from the top-right corner. | **Quickly access controls.** |
| An illustration of iPhone. A line ending with a dot indicates a drag up from the bottom of the screen, then a pause. | **Open the App Switcher.** |
| An illustration of iPhone. A two-headed arrow indicates swiping left or right across the bottom edge of the screen. | **Switch between open apps.** |
| An illustration of iPhone with an arrow pointing to the side button on the upper right. | **Ask Siri.** |
| An illustration of iPhone with two arrows indicating double-clicking the side button on the upper right. | **Use Apple Pay.** |
| An illustration of iPhone with three arrows indicating triple-clicking the side button on the upper right. | **Use Accessibility Shortcut.** |
| An illustration of iPhone with arrows pointing to the side button on the upper right and the volume up button on the upper left. | **Take a screenshot.** |
| An illustration of iPhone with arrows pointing to the side button on the upper right and a volume button on the upper left. | **Use Emergency SOS.** |
| An illustration of iPhone with an arrow that indicates triple-clicking the side button on the upper right. | **Use Emergency SOS.** |
| An illustration of iPhone with arrows pointing to the side button on the upper right and a volume button on the upper left. | **Turn off.** |
| An illustration of iPhone with arrows pointing to the side button on the upper right and the volume up and volume down buttons on the upper left. | **Force restart.** |

The only gestures used in the *AmIGarbage?* app are tap and slider gestures.

## Use simple gestures for interaction [3].

The only gestures used in the *AmIGarbage?* app are tap and slider gestures.

## Use system-define haptics [3].

Haptics are used in the *AmIGarbage?* app and are summarized in the following table. The for every user interaction, there is a corresponding haptic. All haptics are used with the imported *UIKit* library.

**Table 7: Haptics in Used in AmIGarbage? App**

|  |  |  |  |
| --- | --- | --- | --- |
| **Haptic Category** | **Class** | **Haptic Type** | **User / App Action** |
| **Notification** | [UIImpactFeedbackGenerator](https://developer.apple.com/documentation/uikit/uiimpactfeedbackgenerator) | Success | Bluetooth paired properly. |
| Failure | Bluetooth connection failed. |
| Warning | Training mode accuracy drops below 80%. |
| Impact | [UINotificationFeedbackGenerator](https://developer.apple.com/documentation/uikit/uinotificationfeedbackgenerator) | Medium | Scan result comes into view. |
| Selection | [UISelectionFeedbackGenerator](https://developer.apple.com/documentation/uikit/uinotificationfeedbackgenerator) | Selection | User chooses appropriate waste category in training mode. |
| Selection | User chooses weather scan was accurate in training mode |

## Use a consistent style hierarchy to communicate the relative importance of buttons [3].

## All buttons in the *AmIGarbage?* app have the following structure:

Navigation buttons all same size and location.

Scan and Results button same size in center of page.

Secondary tab buttons are bottom and lighter colour.

## Use typical navigation frameworks [3].

There are three typical navigation types that apps made for ios use. Their layout is documented in the figure below. In a hierarchical navigation, the user makes one choice per screen until a destination is reached. To reach another destination, the user must start from the beginning. In flat navigation, users choose between multiple content categories. In an experience-driven navigation, users move freely from node to node through the content.

|  |
| --- |
| Diagram that shows nine circles that are connected by single-headed arrows. Some circles connect to exactly one other circle and other circles connect to more than one other circle. The diagram suggests a random path through a set of nine locations.  (a) experience-driven navigation |
| Diagram that shows seven squares connected in a hierarchy. The top square connects to three squares shown on a row below it. The leftmost square on the second row connects to the three remaining squares shown on a third row.  (b) hierarchical navigation |
| Diagram that shows eight squares, with four squares in one row and the remaining four squares in a row below it. The first square in the first row connects to the second square in the first row with a double-headed arrow. In the same way, the second square connects to the third square and the third square connects to the fourth square. Each square in the first row uses a single-headed arrow to connect to the square below it in the second row.  (c) flat navigation |

**Figure 3 IOS Defined Navigation Types**

For the *AmIGarbage?* app, the Bluetooth setup functionality (for first-time users) employs an experience-driven navigation where each page is treated as a node, and some nodes lie in a cyclical pattern. This experience-driven navigation style allows the user to enter Bluetooth information as many times as needed until the Bluetooth connection is successfully established. The navigation style is documented in the figure below.

Diagram

Description automatically generated

**Figure 4: Experience-Driven Navigation Style of Bluetooth Connection Setup for First-Time Users**

The main functionalities use a hybrid flat-navigation and hierarchical experience. The flat navigation is used to move seamlessly between *default mode*, *training mode*, and *information mode*. The flat navigation is useful because users can easily move between any mode in one-click, and intuitively sees that these modes are the main functions of the app.

Both default mode and training mode each have an embedded hierarchical navigation because users must interact with the app to view the intended information or make the intended corrections to the scan results.

This hybrid layout is documented in the figure below. Note that the pages outlined by a dotted line indicates the pages involved in the flat navigation layout. The other pages use a hierarchical layout.

Diagram

Description automatically generated

**Figure 5: Hyrbid Hierarchical and Flat Navigation of Main Functionalities of AmIGarbage? App**

## Ensure that the ios ‘VoiceOver’ functionality allows users to navigate to every element [3].

The VoiceOver functionality inherent to ios allows users to navigate to each element. This is is achieved with the accessibility classes available in Swift. See the example below for a code snippet of accessible elements.



**Figure 6: Example of Accessibility Tags for IOS**

A description of all elements, tags, and corresponding descriptions are documented in the table below. Note that the page locations refer to Figure 1.

**Table 8: Element Tags and Role Description for Accessible Voice Over Use**

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Label** | **Role Description** | **Page Location** |
| App Name | Title | The name of the app, AmIGarbage? | a |
| Logo | Logo | A green recycling triangle with a gear, a camera, and a PCB. |
| Get started button | Button | Press to get started. |
| Page progress indicator | Page Progress | Shows the progress of setting of initial Bluetooth setup. 1/3 pages complete. |
| Bluetooth instructions | Text | Please enter the name of your Jetson NANO device. | b |
| Bluetooth name input | User Input Text | Type Bluetooth name here. |
| Confirm button | Button | Confirm and attempt to connect Bluetooth. |
| Page progress indicator | Page Progress | Shows the progress of setting of initial Bluetooth setup. 2/3 pages complete. |
| Bluetooth status text | Text | Attempting to connect via Bluetooth. | c |
| Progress circle | Process Progress | Bluetooth connection trying to be established. |
| Bluetooth status text | Text | Bluetooth connection successful. | d |
| Green check image | Image | Green check indicating Bluetooth connection is successful. |
| Continue button | Button | Press to continue to main part of the app. |
| Bluetooth status text | Text | Bluetooth connection failed. | e |
| Red X image | Image | Red x indicating Bluetooth connection has failed. |
| Try again button | Button | Press to try to connect to Bluetooth again. |
| View scan button | Button | Press to view scan result. | f,g,h |
| Default mode button | Button | Default mode button in tab bar. Press to enter non-feedback mode. |
| Training mode button | Button | Training mode button in tab bar. Press to enter feedback mode. |
| Info mode button | Button | Info mode button in tab bar. Press to show app instructions. |
| View stats button | Button | Press to view scan results history. | g |
| App instructions | Text | How to scan: Place one item of waste into machine. Press view scan and wait for results. Default mode: Users can scan waste and view the results. Training mode: Users can scan waste and view the results. They can also give feedback on whether the scan was correct. | h |
| Retrieving result text | Text | Retrieving result. | i |
| Progress circle | Process Progress | Trying to retrieve result. |
| Result certainty text | Text | I am [x]% certain that the item was: | j,k,m |
| Result icon | Image | [paper]. The scan result is [paper]. |
| Continue button | Button | Press to continue back to the home page. | j,l,m |
| Did I get it right text | Text | Did I get it right? | k |
| Yes button | Button | Yes. Press if the scan result was correct. |
| No button | Button | No. Press if the scan result was incorrect. |
| What was the item text | Text | What was the item supposed to be? | l |
| Waste type icon | Image | A picture of [paper]. |
| Waste type toggle | Toggle | Toggle if the item was supposed to be [paper]. |
| Accuracy warning text | Text | The accuracy of the system is below 80%. We recommend you install a software update to your Jetson NANO. | n |
| Ok button | Button | Press to continue to homescreen. |

## Use dynamic type such that the app layout is adaptable to all font sizes, avoid full text justification, and use regular or heavy font weights in your app [3].

All font is typed at least at 14 point was per WACG 2.0 requirements. All headings had bold. All other text have regular; both comply with the contrast ratio regulations. The table below details all locations of text and their size, justification and font weight. Note the page location refers to the image from Figure 1.

**Table 9: Text Size, Justification and Weight**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Size** | **Justification** | **Weight** | **Page Location** |
| App Name | 32 | Left | Bold | a |
| Get started button | 28 | Left | Regular |
| Bluetooth instructions | 28 | Left | Regular | b |
| Bluetooth name input | 20 | Left | Regular |
| Confirm button | 28 | Left | Regular |
| Bluetooth connection text | 28 | Left | Regular | c |
| Bluetooth succeeded | 28 | Left | Regular | d |
| Continue button | 28 | Left | Regular |
| Bluetooth failed | 28 | Left | Regular | e |
| Try again button | 28 | Left | Regular |
| View scan button | 44 | Left | Bold | f,g,h |
| Default mode button | 18 | Left | Regular |
| Training mode button | 18 | Left | Regular |
| Info mode button | 18 | Left | Regular |
| View stats button | 44 | Left | Bold | g |
| App instructions | 20 | Left | Regular | h |
| Retrieving result text | 28 | Left | Regular | i |
| Result certainty text | 28 | Left | Regular | j,k,m |
| Continue button | 28 | Left | Regular | j,l,m |
| Did I get it right text | 28 | Left | Regular | k |
| Yes button | 28 | Left | Regular |
| No button | 28 | Left | Regular |
| What was the item text | 28 | Left | Regular | l |
| Accuracy warning text | 28 | Left | Regular | n |
| Ok button | 28 | Left | Regular |

## Prefer system colours for text, and respond correctly to inverse colours [3].

For all colour requirements, the *AmIGarbage?* app meets all Web Content Accessibilty Guildelines (WACG) 2.0 AA standards, which makes “content accessible to a wider range of people with disabilities, including blindness and low vision, deafness and hearing loss, learning disabilities, cognitive limitations, limited movement, speech disabilities, photosensitivity and combinations of these”, and is the universal standard for accessibility for web and application development [4]. The requirements are:

1. Up to 17pt with any text weight must have a minimum contrast ratio of 4.5:1
2. 18pt and large text with any weight must have a minimum contrast ratio of 3:1
3. Any text that is bold must have a minimum contrast ratio of 3:1

The process to determine the contrast ratio between any two colours with a 6-digit hexadecimal code is as follows.

Luminance:

Where:

And, for a hexcode ‘XXXXXX’:

*Shape

Description automatically generated with medium confidence*Shape

Description automatically generated with medium confidence

For example, take white who’s hex code is FFFFFF:

Then,

Finally,

All colours in the *AmIGarbage?* app are characterized in the table below.

**Table 10: Colours Used in App and Corresponding Hex Codes and Relative Luminance**

|  |  |  |
| --- | --- | --- |
| **Colour** | **Hex Code** | **Luminance** |
| Blue | 0079CE | 0.28278336145840 |
| Grey | D9D9D9 | 0.36291794320712 |
| Green | 1D8400 | 0.18265721405286 |
| Red | DF2D31 | 0.65687527738189 |
| Purple | B400FF | 0.08651636397971 |
| White | FFFFFF | 1.00000000000000 |
| Black | 000000 | 0.00000000000000 |
| Light red | C04945 | 0.21635292201714 |
| Light green | A5FF9E | 0.72652881702560 |

To calculate contrast ratio, where L1 is the lighter colour’s relative luminance and L2 is the darker colour’s relative luminance. Convert colours from hex to numbers.

For example, to calculate the contrast ratio between black (L=0) and white (L=1), the following equation can be performed:

WACG 2.0 requirements require at minimum 4.5:1 ratio for any text less that 18pt, and 3:1 for all other text or foreground/background. All colour schemes in the app meet this criteria and are summarized in the table below.

**Table 11: Contrast Ratios for all Colours in App**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature Example** | **18pt** | **Colour 1** | **Colour 2** | **Contrast Ratio** |
| Text  Description automatically generated with low confidence  White and blue | Yes | FFFFFF | 0079CE | 4.53:1 |
| White and green | Yes | FFFFFF | 1D8400 | 4.82:1 |
| White and purple | Yes | FFFFFF | B400FF | 4.78:1 |
| A red and white logo  Description automatically generated with low confidence  White and red | Yes | FFFFFF | DF2D31 | 4.60:1 |
| Text  Description automatically generated  Black and grey | Yes | 000000 | D9D9D9 | 14.87:1 |
| Icon  Description automatically generated  Light green and blue | No | A5FF9E | 0079CE | 3.76:1 |
| Graphical user interface, text, application, chat or text message  Description automatically generated with medium confidence  White and dark grey | No | FFFFFF | 949494 | 3.03:1 |
| Text  Description automatically generated  White and light red | Yes | FFFFFF | C04945 | 4.90:1 |
| Purple and grey | No | B400FF | D9D9D9 | 3.39:1 |

# Constraint – All parts must be available within Canada

All software is available internationally, since it is posted to GitHub. The Bluetooth adapter is also available to ship to Canada.

# Constraint – Must accept items of up to 0.03m^3 \*\*\*\*\*????

Not applicable to software.

# Constraint – Must fit standard size garbage bin / recycling bin

Not applicable to software.

# Bibliography

|  |  |
| --- | --- |
| [1] | D. Diener, "Best USB Bluetooth Adapters that are Linux-compatible," 29 January 2020. [Online]. Available: https://www.addictivetips.com/ubuntu-linux-tips/best-usb-bluetooth-adapters/#:~:text=%20Best%20USB%20Bluetooth%20Adapters%20that%20are%20Linux-compatible,The%20ZEXMTE%20Bluetooth%20adapter%20is%20a...%20More. |
| [2] | A. FORSSELL, "Dynamic Lists with Asynchronous Data Loading in SwiftUI," April 2020. [Online]. Available: https://callistaenterprise.se/blogg/teknik/2020/04/20/swiftui-lists/. |
| [3] | Apple Inc., "Human Interface Guidelines," [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/accessibility/overview/user-interaction/. [Accessed March 2021]. |
| [4] | MIT, ERCIM, Keio, "Web Content Accessibility Guidelines (WCAG) 2.0," December 2008. [Online]. Available: https://www.w3.org/TR/WCAG20/. [Accessed March 2021]. |