Mobile Accessibility Practices

(Making mobile applications accessible and usable for persons with disabilities)

Table of Contents

1. Problem statement

The shift to digital governance and availability of assistive technologies have been both empowering as well as frustrating for persons with disabilities, who comprise approximately 150 million of the Indian population[1].

Government initiatives such as the Digital India campaign are increasingly delivering basic functions of governance through information technologies. In the past year, the government, private sector and the world at large have embraced mobile applications as a preferred medium for user interactions and transactions.

The Mobile Seva App Store hosts 790 government apps, which provide services including voter information, agricultural assistance, welfare scheme signups, and educational content provision[2].

In addition, the overall app market in India has also grown rapidly, with almost 5 times as many apps downloaded in 2015 compared to the previous year[3]. These include apps which let users access everyday services like transportation, communication and entertainment.

However, for persons with disabilities, many of these apps, and consequently the services they provide, are inaccessible and often impossible to use. Research in the past year that looked at several apps, both government and private, found that a majority of the apps are inaccessible and unusable, especially for persons with low vision and blindness.[4][5].

Some examples of inaccessible government and private apps are given below to aid a better appreciation of the situation.

Inaccessible Government Apps

The **ePathshala** app for instance, is not usable by persons with visual disabilities. The first screen that allows language selection is not labeled properly -- only the Hindi and English buttons are correctly announced. Many of the options that are available on the screen are not labeled with text, only graphics. In the absence of labels, a screen reader is unable to comprehend and describe an image to its user. Furthermore, the books

1 https://zeroproject.org/policy/india/

2 https://apps.mgov.gov.in/index.jsp

3 http://trak.in/tags/business/2016/01/07/indian-mobile-app-industry-interesting-highlights-2015/

4 http://retail.economictimes.indiatimes.com/news/e-commerce/e-tailing/most-popular-apps-inaccessible-to-millions-of-disabled-says-study/55533796

5 http://factordaily.com/tested-18-government-apps-citizens-found-accessibility-issues-disabled/

themselves are PDF or JPEG images, which cannot be read using a screen reader, The reading mode available for the books is also inaccessible.

The **MyGov** app, is a web based app, which is linked to a web site that is not completely accessible. The graphics in particular are poorly labeled. The animations used are also inaccessible, and the banner that scrolls the new announcement is completely unusable for screen reader users.

The Prime Minister’s app, the **Narendra Modi** app is highly inaccessible. The very first screen cannot be navigated by a visually impaired user. The controls on the rest of the app are labelled in all caps, which makes using the screen reader difficult. If one selects the feed option and follows a topic, the controls on the content screen are all simply labelled “Narendra Modi”, making it impossible for the user to make an informed choice. However, the text is presented using standard web controls, which means that once navigated to, a page can be accessed using a screen reader.

In the case of the **Aadhar** mobile app, The first screen lacks alternate text for any of the controls and graphics. Hence, it was impossible to test the app using a screen reader. The Swachh Bharat app is yet another example of an inaccessible app; The starting screen of this app is cluttered with controls and information, making it difficult to navigate. A few controls are labelled with text, but the majority are inaccessible.

A detailed study of the inaccessibility of the BHIM app and the steps that need to be taken to make it accessible, has been provided in Appendix A.

Inaccessible Private Apps

The situation gets no better when it comes to private Indian apps.

The **Swiggy** app, which is used to order food from nearby restaurants and big food chains is inaccessible to screen reader users. The first screen shows the discount / offers available using a graphics banner which is not marked with text making it completely inaccessible. Only a few buttons are labeled and it is not possible to focus on menu items. For instance, one can select the food category Soup but further selection of a particular soup is not possible as the focus simply stays on the main category, which makes it impossible to navigate the app and select items to order.

There are apps like **Myntra** and the **ICICI - Pocket app** to provide a host of banking facilities amongst ICICI as well as non ICICI customers which have absolutely no screen reader support and auditory feedback, rendering them completely inaccessible and unusable.

In the **Flipkart** app, the lack of labeled buttons makes it difficult to read and select options. It is also not user friendly.

In many of these cases, there is no alternate recourse for persons with disabilities to avail themselves of the services which these apps provide.

Impact

Hence, despite the availability of technologies such as text to speech to enable access to information independently, the failure to adhere to accessibility principles in India has had the effect of denying access to critical resources and information. While international apps such Amazon and Uber are completely accessible and user friendly, Indian apps remain a constant source of frustration and reminder of their disabilities for persons with disabilities. Instead of empowerment, there is a crippling effect on persons with different disabilities in multiple ways.

The Government of India has recognised that accessibility is a concern which it needs to address if it has to engage comprehensively and effectively with the public. The Guidelines for Indian Government Web sites (GIGW 2009[6]), the National Policy on Universal Electronics Accessibility (2013)[7] and most recently the Rights of Persons with Disabilities Act 2016[8] all require compliance with web accessibility standards and provision of public information and resources in accessible electronic format. The increasing adoption of mobile as an engagement platform hence necessitates the adoption of guidelines to ensure that applications are accesible to and usable for persons with disabilities.

2. International position on mobile app standards

Presently there is no single international standard on the accessibility of mobile apps. The World Wide Web Consortium (W3C) is working on a standard, which will take time to be published. In the meantime it has published some best practices, primarily based upon the (Web Content Accessibility Guidelines (WCAG) 2.0. There are also guidelines available from Android, iOS, BBC and some to be found within 508. However, none of these are comprehensive and cannot be adopted as is. Links to these guidelines are provided in the reference section of this submission.

3. Methodology and approach

These practices have been put together by a group of accessibility and technology experts from different organisations, most of whom are persons with disabilities themselves and have extensive experience in the domain of assistive technology. They draw upon the guidelines and best practices mentioned in the previous section, user surveys and interviews conducted amongst persons with different types of disabilities as well as any particular user experience insights gained by using Indian government/private apps.

6 http://web.guidelines.gov.in/

7 http://pib.nic.in/newsite/PrintRelease.aspx?relid=99845

8 http://www.disabilityaffairs.gov.in/upload/uploadfiles/files/RPWD%20ACT%202016.pdf

4. Purpose and objective

Given that apps have already permeated the sphere of everyday life and the likelihood of meeting one's daily needs without apps is becoming almost impossible, it is critical that this issue be addressed immediately. The purpose of this submission is hence to present a set of guidelines/ practices which will inform mobile app developers about how to create applications which will be accessible and usable to all persons, including persons with disabilities. These practices are not comprehensive and we recommend annual review so that these practices can be updated based on the feedback from users, developers and experts.

5. Introduction and explanatory note

The objective of these mobile accessibility practices is to help developers, designers and testers to create mobile apps that are universally accessible. An accessible application is one which is usable by everyone irrespective of their abilities. These mobile accessibility practices have been formulated after reviewing various globally accepted standards and guidelines, as mentioned in the section on the international position on Mobile app standards.

The Mobile Accessibility practices discussed below are not technology specific, but the examples are based on either Apple iOS or Google Android operating systems. The other mobile platforms are either not accessible or not used widely. The techniques to test or implement a specific practice may differ depending on the operating system.

Both the Android and iOS operating systems provide standardized mechanisms to communicate various attributes of a user interface element (UI Element) such as the label associated with a UI element, role of a UI element (such as whether it is a button or an edit control,) and state information (such as whether it is disabled, checked or pressed.) This mechanism is called Accessibility Application Programming Interface (API) and it provides reasonably good information for standard UI elements.

6. Mobile Practices

Mobile Practice 1: Support platform accessibility settings

Most mobile platforms provide accessibility settings such as contrast between background and foreground text, invert colors, large text, grayscale, mono audio etc. Users select the relevant setting as per their requirement and expect all the apps to behave accordingly. Review all the accessibility options in the device settings and make sure each accessibility feature behaves as intended. For example, if a user chooses invert color option, and the app is already showing black text on a white background then it should show white text on black background which is easy on the eyes for many users with photosensitive eyes. Many other users without any well-known eye condition also find this easier for prolonged reading.

Mobile Practice 2: Provide proper labels for UI elements

There must be an accessible label for each UI element, such as images, buttons and other controls. An accessible label is recognizable by assistive technology such as Voiceover or TalkBack. Avoid labels embedded into an image as they cannot be parsed by screen readers.
Consider the following key points while labeling UI elements:

1. **A label must be precise and clear:** Think about the purpose that the UI element serves. For example, label “Add to Cart” for adding an item to cart. Consider using action verbs that describe the purpose of the UI element in order to provide appropriate labels.

2. **Timely Update**: In case the functionality of the UI element changes, the label must be updated as well. For example, “Play” button must change to “Pause” and vice versa for media files. Updated labels make it easy for the users to interact with the app.

3. **Do not provide the role and state information as part of label**: This information is provided separately through Accessibility API (described in Practice 3). For instance, “Play” button to be labeled as “Play”, and not “Play button” because the button’s role will be indicated through accessibility API.

4. **Localize the label strings**: This is required for users using the applications in different languages.

WCAG 2.0 corresponding success criteria: 1.1.1, 1.3.1, 2.4.2, 3.3.2, and 4.1.2.

Mobile Practice 3: Provide role information for UI elements

Every UI element can be identified visually with its look and feel. As users with blindness cannot perceive visual information, the role for a UI element must be available programmatically so that assistive technology can report this either through speech or Braille. In order to do so, use platform specific roles or traits for standard UI elements. For example, a button is announced as “Button” along with the label for assistive technology users. In case of custom UI elements, use platform accessibility API to report the role information.

WCAG 2.0 corresponding success criteria: 1.3.1, 3.2.4 and 4.1.2.

Mobile Practice 4: Provide hints for active UI controls

A Hint is a brief, localized phrase that describes the results of an action on a UI control. It is like a tool tip that lets the user find out how to interact with the UI control. Hints are only required for UI controls that allow users some interaction, and are not required for UI elements such as labels or plain text. In case of custom UI controls, hints also report the screen reader gestures that users could perform to interact with the control. For example, in a shopping website that has a button “Add” that adds items to the cart, the button could have the hint as “Adds the item to the cart”. Similarly, for a drag and drop widget, the hint can be “Double tap and hold until you hear a sound, then drag your finger to move the element to the desired location and lift the finger”. The standard UI controls have hints supplied by the APIs, but those hints might have to be changed depending on the usage.

WCAG 2.0 corresponding success criteria: 3.3.2 and 3.3.5.

Mobile Practice 5: Provide state information for a UI control

In addition to the role of a UI control, assistive technologies must identify the current state of a UI control. For example, the state of checkbox checked/ unchecked, tab selected or not, a push button pressed or not etc. should be notified. This information must also be reported as soon as it is changed. The standard UI controls provide this information by default, but for custom controls, this information must be supplied by platform specific accessibility APIs. The changes of state must be dynamically updated and accurately available to the assistive technologies.
WCAG 2.0 corresponding success criterion: 4.1.2.

Mobile Practice 6: Group the related UI elements

Related UI elements such as song title and singer name for a song must be grouped together so that assistive technologies can present it as a single UI element, reducing the gestures for interaction. This also helps to increase the touch target (Explained in Practice 8) so that users with low vision, users having motor difficulties and users with big fingers can more easily interact with it.
The following points are important for grouping related elements:

1. A group must have only one actionable UI control.
2. Updating UI controls such as progress bar must not be grouped with any other control as users need only the updated information.

Mobile Practice 7: Design a simple interface and provide enough spacing

Make the UI clean and simple. Avoid vertical and horizontal scrolling. This allows users with low vision to zoom and interact with the controls with ease. Provide non-interactive space between actionable UI elements of at least one point for iOS or 1 DP for android. This allows users with low vision, users having motor difficulties and users with big fingers to avoid touching a wrong UI element.

Mobile Practice 8: Touch Target must be at least 9x9mm

Many users find it difficult to interact with small screen elements. It could be due to big or unsteady fingers or motor or visual difficulties. So, the touch targets must be at least 9x9mm regardless of screen size.

Mobile Practice 9: Bring focus to the active UI control

Since Mobile screens are small, all the UI elements cannot fit on the screen at a time. So UI elements such as buttons that take less space are used to bring up other UI elements such as dropdowns. For example, users would activate the “MM” button to bring up the month dropdown. In such scenarios, the dropdown should get the focus when the user activates the button. If the focus is not set properly, blind and low vision users may not be able to realize that the UI has changed. It sometimes takes many attempts to find out the new elements and if such interactions are time sensitive, a timeout could occur and the user would have to start all over again. Even without timeouts, new users could find it difficult to manage such interactions thus impacting the user experience.

WCAG 2.0 corresponding success criterion: 2.4.3.

Mobile Practice 10: Use custom actions for context specific UI controls

When a UI control has context specific menu items, users must be informed that such a menu is present and must be able to activate those menu items. A Custom Action is an effective technique to support such an interaction. Both Android and iOS provide Custom Actions that are available to assistive technology users. When an element with a custom action is focused, assistive technology lets the user know that such actions are available and then users can use well-known gestures to perform those actions. Alternatively, use the accessibility API to report to the user what new UI elements are available and where such elements are present on the screen. This way users can locate those elements. This technique should only be used if Custom Actions are not available.

WCAG 2.0 corresponding success criterion: 1.3.1.

Mobile Practice 11: Provide a logical and meaningful sequence

Screen reader mobile users rely on gestures to navigate and interact with the content and the UI controls. Content when navigated using the screen reader gestures, must form a meaningful sequence. The controls on the mobile screens and the interaction produced need to be logical.

WCAG 2.0 corresponding success criterion: 1.3.2, 2.4.3.

Mobile Practice 12: Handle screen orientation change consistently

Assistive technology users could lock screen orientation to avoid interference with their interaction with the device.
Pay attention to the following points while handling screen orientation:

1. **Screen orientation change is disabled**: If the user has turned on “Locked Orientation” option for iOS or disabled the Auto-rotate screen option for Android, then try not to change the screen orientation.
2. **Screen orientation change is not disabled**: Make sure that the screen orientation change is not disruptive and the focus does not move from the focused screen element.
3. **Report screen orientation change using accessibility API**: Report Screen orientation at the start if it is different from the default setting when screen orientation change is disabled. Otherwise, the change should be reported every time the orientation changes.

Mobile Practice 13: The content must be resizable

Users with low vision may need to increase the size of the UI elements to be able to see well. The app must resize its UI elements in accordance with device settings for text size.

WCAG 2.0 corresponding success criterion: 1.4.4.

Mobile Practice 14: Color contrast must be minimum 4.5:1

Users with low vision or users in poor lighting condition would find it difficult to see the UI elements on the screen if the foreground elements cannot be differentiated from the background. Therefore, suggested color contrast ratio between foreground text for up to 18 point font and background must be at least 4.5:1 as per WCAG 2.0 Level AA or 7:1 as per WCAG 2.0 level AAA.

WCAG 2.0 corresponding success criteria: 1.4.3 and 1.4.6

Mobile practice 15: Color or shape should not be the only way to communicate important information

Relying only on color or shape to communicate important information can be problematic for certain persons with disabilities such as users with color blindness or users with blindness.
The following considerations are critical:

1. **App designers must add text equivalence for color coded or shape dependent information**. For example, if an app has a required field, then it could provide the word (Required) if the space permits or use placeholders.
2. **The app must disable the button used to move the menu forward until the field is filled-out**. Just relying on the shape of a button to indicate the disabled state does not work for many users with disabilities.
3. **Apps must not use color-based references** such as Click on Red or Square button, instead have text references such as Click on Next button.
   [9].

WCAG 2.0 corresponding success criteria: 1.4.1, 1.3.1 and 1.3.3.

Mobile Practice 16: onscreen keyboard and hardware keyboard must be accessible

Mobile platforms provide support for both onscreen keyboard and hardware keyboard. App designers must ensure that both are accessible with assistive technology such as magnifier or a screen reader.
Note the following points while developing and testing the input interface:

1. **Do not automatically change focus**: If a user is entering data and the focus shifts automatically, the user would find it difficult to enter data. Focus must be changed only when the user activates a UI element that is designated for confirming an action such as the Submit button.
2. **Select the correct onscreen keyboard**: Ensure that the appropriate keyboard is invoked by the app depending on the type of field or the data that needs to be provided by the user. For example, the appropriate on-screen keyboard must be invoked for normal text, numerical data, email address or web address. This

9 http://webaim.org/articles/visual/colorblind#designing

recommendation is not only helpful for users with disabilities, it also enhances the comfort of other users.

3. **Apps must be compatible with hardware keyboard**: Though many users work with the onscreen keyboard, others still prefer using a hardware keyboard that comes built-in or is connected with mobile devices via Bluetooth. Therefore, apps must be tested with hardware keyboards as well.

WCAG 2.0 corresponding success criteria: 2.1.1, 2.1.2, 3.2.1, 3.2.2 and 3.2.5.

Mobile Practice 17: Keep the gestures simple

Avoid gestures that require 3 or more fingers to interact with UI elements. These complex gesture patterns make application usage difficult for those who do not have the use of all of their fingers, or use the device single-handedly. If such complex patterns cannot be avoided, provide an alternate to perform the same action or allow the user to create a custom gesture. For example, an added setting may be provided to customize gestures as per user requirements.

Mobile Practice 18: Provide enough time

Many users require extra time to be able to finish an action. So, avoid session timeouts. If a timeout cannot be avoided, then provide an option for users to extend the time limit before the timeout occurs. Also, make sure that the time extension element focus is properly set.

WCAG 2.0 corresponding success criterion: 2.2.1.

Mobile Practice 19: Provide captions for audio content and subtitles/transcripts for video content that is accompanied by audio

Many users who have hearing difficulties or who find the language in the audio difficult to understand would need captions or transcripts that help them to understand the text of the audio.

WCAG 2.0 corresponding success criteria: 1.2.2, 1.2.4, 1.2.6, 1.2.8 and 1.2.9.

Mobile Practice 20: Provide audio descriptions for video content

Users with blindness may find it difficult to understand important visual information which is not available in the audio format. If the application contains video that does not have an audio equivalent, provide audio description for the content that is crucial for blind users to understand the content. It is not required to provide audio for decorative and non-essential video content.

WCAG 2.0 corresponding success criteria: 1.2.1, 1.2.3, 1.2.5, 1.2.7, 1.2.8

Mobile Practice 21: No content must flash more than 3 times a second

Some users get seizures if any content flashes more than 3 times per second. Therefore, it is recommended that no content flashes more than 3 times a second.

WCAG 2.0 corresponding success criteria: 2.3.1 and 2.3.2.

Appendix A - Case Study : BHIM app

Background

Bharat Interface for Money (BHIM) is an initiative to enable fast, secure, reliable cashless payments through mobile phone. BHIM is interoperable with other Unified Payment Interface (UPI) applications, and bank accounts. BHIM is developed by the National Payment Corporation of India (NPCI).
A user can simply register his or her bank account with BHIM, and set a UPI PIN for the bank account. Users' mobile number is payment address (PA), and user can simply start transacting. Yes! It is that simple!
But that is not the case for many persons with disabilities, for whom the BHIM app is still an unfulfilled promise. For both Android and iOS, it is not accessible for users with disabilities, who are left out from the benefits of digital revolution.

Problems

In general, BHIM app has a few issues such as the inability to enter the desired information and get feedback about the user interface.
The following specific problems have been found with BHIM App for both Android and iOS app.

Android BHIM app issues

1. App uses own customized keyboard which is part of its user interface. Ideal to use standard keyboard. Or implement accessibility support for the custom keyboard.
2. On this keyboard, Clear and Enter keys are not spoken as they are unlabeled. This makes app usage very difficult to a Talkback user.
3. Customized Keyboard remains open which creates difficulty in the navigation for a Talkback user.
4. In BHIM app back button, notification button and drop down menus are unlabeled throughout the app.
5. One needs to use double tap to activate keyboard button instead of standard method of move and lift.
6. Maybe for security purpose app is using customized /virtual keyboards but then app should detect use of screen reader and accordingly provide accessibility.
7. In many places, hints and roles like edit boxes, auto fill in of OTP, moving to another screen, focus to next element, etc. are not spoken by Talkback.
8. For edit fields it opens same customized/web keyboard which creates navigation and data input barriers. E.g. for card number details field for MM and YY navigation is for each digit. After inputting data keyboard remains open.
9. Overall app response is very poor for Talkback user due to many customized elements.
10. All error messages should be in a pop ups with force focus for users but at present error notification pops at the bottom and disappears.
11. Font sizes are nonstandard, at few places big and few too small.

iOS BHIM app issues

1. Some user controls in BHIM app cannot be clicked with Voiceover running which is absolutely required for users to issue a command. Voiceover has a double tap gesture which is similar to single tap for users without Voiceover. For example, tapping on send option in transfer money does not do anything. Due to this issue, a voiceover user cannot use BHIM app. Similarly, clicking on verify in send money screen does not do anything.
2. For many controls, there is no indication about the type of control. Users need to know what type of user interface control they are interacting in order to understand how to interact with them. Voiceover provides different gestures to interact with dropdown vs checkbox. This problem is found all over the app but a few prominent examples are in 'Select your bank' and 'Choose language' screens.
3. There is no indication about the state of the user interface. For example, on selecting a bank, there is no indication whether a bank is selected. User gets confused whether any has been selected or not.
4. There are no back buttons in BHIM screens for users to be able to cancel an action.
5. In enter passcode screen, the option shows show passcode, but the passcode is still visible. This is a security hazard.

How to fix some of these issues

Both iOS and Android use accessibility Application Programming Interface (API) to get information from the User interface elements and present this information to assistive technology so that assistive technology such as voiceover or Talkback can present it for users with disabilities. If a user interface element does not do so, Talkback and Voiceover cannot facilitate interaction with these controls. The standard controls that these platforms provide have accessibility by default. If developers use standard controls, those controls should mostly be accessible. But if custom controls are used, some steps must be taken to make them accessible.

Preparing iOS for accessibility

For iOS, to make any custom control accessible, first thing that needs to be done is to set the property setIsAccessibilityElement to YES. Once this is done, other properties can be set to provide additional accessibility information.

Providing Labels

Many options in BHIM app are not labeled so users with blindness are not able to find out the purpose of the option. To understand how to define labels, refer to Mobile Accessibility practice 2. To provide labels for these controls, use the following techniques:

- For iOS, there are 2 ways to add custom labels. 1. Set the default label in interface builder and 2. Implement accessibilityLabel method in the view subclass. For more information refer to Making Your iOS App Accessible provided in references.

- For Android, also there are 2 ways to add the labels. 1. For static text, use android:contentDescription XML attribute, and 2. For dynamic elements, use setContentDescription() method. For more information, checkout Making Apps More Accessible for Android [provided in references].

Providing role and state information

Similar to labels, providing role and state information is very important for accessibility. The role helps users to know how to interact with the UI control. State lets the user know the current state of the control i.e. whether something is selected or not.

For iOS, role and state are defined with what is known as attributes. So to indicate that a UI control behaves as a button, you would set setAccessibilityTraits to UIAccessibilityTraitButton. Similarly, you could combine other traits to indicate that a control is selected etc.

For Android, role should be defined by the type of control specified. For example, if a “Next” button is required, then use <Button> and provide other appropriate attributes. to report any status change, use sendAccessibilityEvent method to indicate what has changed. For example, if an element is selected, sendAccessibilityEvent(TYPE_VIEW_SELECTED) could be called.

Conclusion

This brief case study shows what is the impact of the BHIM app not being accessible and how to solve couple of accessibility issues. As the examples demonstrate, fixing these issues is not difficult, but it needs attention. For more information, corresponding developer guides should be consulted.

References

Web Content Accessibility Guidelines (WCAG) 2.0
http://www.w3.org/TR/WCAG20/

Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile
http://www.w3.org/TR/mobile-accessibility-mapping/

BBC - Future Media Standards & Guidelines - Mobile Accessibility Guidelines v1.0
http://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile_access.shtml

Making Your iOS App Accessible
https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/iPhoneAccessibility/Making_Application_Accessible/Making_Application_Accessible.html#//apple_ref/doc/uid/TP40008785-CH102-SW5

Making Apps More Accessible | Android Developers
https://developer.android.com/guide/topics/ui/accessibility/apps.html

Contributors

Editorial	*Contributors*
Dinesh Kaushal NVDA Project Manager, Publicis.Sapient dineshkaushal@hotmail.com Dr. Nirmita Narasimhan Policy Director, Centre for Internet and Society nirmita@cis-india.org Prashant Ranjan Verma pverma@daisy.org Rakesh Paladugula Accessibility Engineer Adobe Systems, Founder Maxability.co.in. rakesh@maxability.co.in Srinivasu Chakravarthula srinivasu.c@serveominclusion.com Sujasree Kurapati Managing Director, Deque Software Private Limited sujasree.kurapati@deque.com	Dipendra Manocha President, National Association for the Blind, Delhi dipendra.manocha@gmail.com Manish Agrawal Director of Technology, Publicis.Sapient manish10@gmail.com Pranay Gadodia Disability Diversity Inclusion Expert pranaybg@yahoo.co.in Pranav Lal contact@security-writer.com Prashant Naik Talking ATM India pranaik@gmail.com Saidarshan Bhagat sai.bhagat@gmail.com