Braille Code Reader

Android Graphics Module

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By

Dipto Barman, 121289

Under the supervision of

Suman Saha

To

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat,

Solan-173234, Himachal Pradesh

CERTIFICATE

I hereby declare that the work presented in this report entitled **"Braille Code Reader"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2015 to December 2015 under the supervision of **Mr Suman Saha, Assistant Professor (Grade-II)**, Computer Science & Engineering, JUIT

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Dipto Barman,
121289

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Mr. Suman Saha

Designation: Assistant Professor (Grade-II)

Department name: Computer Science & Engineering

Dated:

ACKNOWLEGDEMENT

This may seem long but the task of my project work both theoretically and practically may not have been completely without the help, guidance and mental support of the following person. Firstly, I would like to thank my guide, **Mr Suman Saha,** Assistant Professor, Department of Computer Science and Engineering, Japyee University of information Technology, Waknaghat, Solan (H.P), who provided me the related material and the idea of the project proposal. He indeed guide me to do the task for my project in such a way that it seems to be research work encouraged me a lot for doing my project in a very smooth manner.

Secondly, I would like to thank my **Parents** who have always been with me for inspiring me that I can do the good task with hard work. Their instigation always helped me to grow my mind focused towards the hard work for implementation of project with having the research work in the mind.

Date..... Signature.....

Name..Dipto Barrman.....

Table of Content

LIST OF FIGURES

LIST OF TABLES AND GRAPHS

ABSTRACT

I am creating this application to cater to the needs of a visually impaired individual. My main motivation for this application was to make accessibility of a power platform like android that are available on every device to every user.

The integration of voice commands, voice actions and voice feedbacks would help the user to understand what action is being taken by the device. Another feature, that is the tutor feature would allow the different users to learn braille if they wish to. This application would also all the user to control their whole device, with just some possible braille key options. This would not only increase the efficiency but also the performance of the device.

CHAPTER 1
BRAILLE CODE READER
INTRODUCTION

1.1 Braille

Braille is a tactile and systematic writing system used by visually challenged and low vison people.

Named after French Citizen Louis Braille, who lost his eyes in a childhood due to some accident**.** At the age of 15 years, in 1824 he developed a writing code in French to facilitate night writing.

Braille character are small rectangular blocks called cells that contain tiny palpable bumps called raised dots. In standard Braille code there are six dots to represent a braille character.

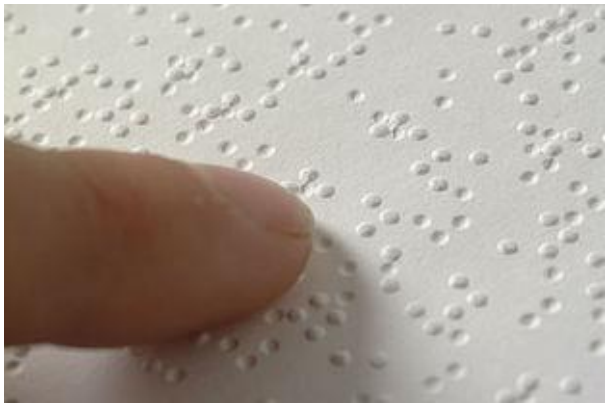
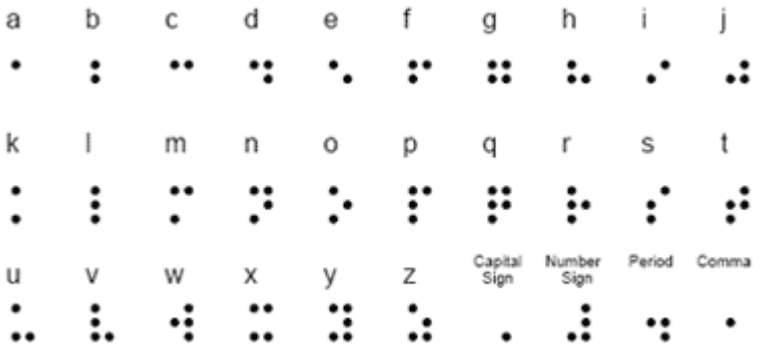


Figure 1

The Braille characters originated as Symbolic codes of existing writing languages, so the mappings or the Braille representation varies from language to language.

1.2 What Does Braille Look Like?

Braille symbols are composed of units of space known as braille cells. A braille cell consists of six raised dots arranged in two parallel rows each having three dots. The dot positions are identified by numbers from one through six. Sixty-four combinations are possible using one or more of these six dots. A single cell can be used to represent an alphabet letter, number, punctuation mark, or even a whole word. This braille alphabet and numbers page illustrates what a cell looks like and how each dot is numbered.



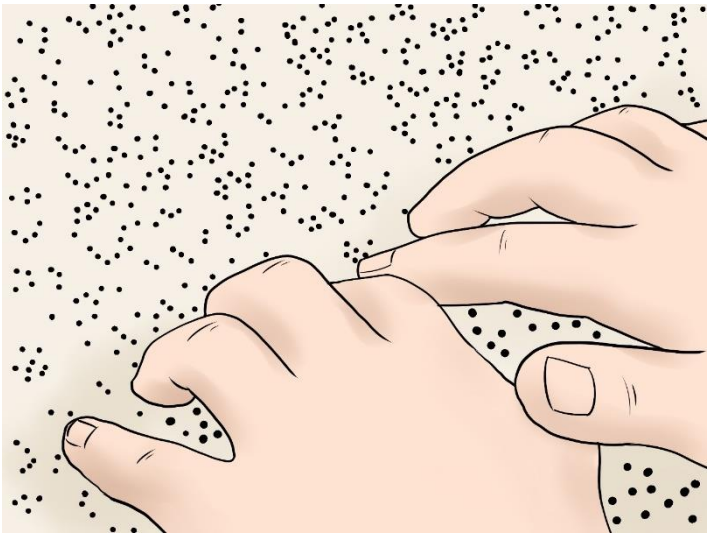


Figure 2

1.3 How Was Braille Invented?

Louis Braille was born in Coupvray, France, on January 4, 1809. He attended the National Institute for Blind Youth in Paris, France, as a student. At that time, books were created using raised print which was laborious to produce, hard to read, and difficult for individuals to write. While attending the Institute, Braille yearned for more books to read. He experimented with ways to create an alphabet that was easy to read with the fingertips. The writing system he invented, at age fifteen, evolved from the tactile "Ecriture Nocturne" (night writing) code invented by Charles Barbier for sending military messages that could be read on the battlefield at night, without light. Learn more about the creation of the braille code by exploring AFB's Louis Braille Online Museum.

1.4 How Is Braille Written?

When every letter of every word is expressed in braille, it is referred to as uncontracted braille. Some books for young children are written in uncontracted braille although it is less widely used for reading material meant for adults. However, many newly blinded adults find uncontracted braille useful for labelling personal or kitchen items when they are first learning braille.

The standard system used for reproducing most textbooks and publications is known as contracted braille. In this system cells are used individually or in combination with others to form a variety of contractions or whole words.

There are 180 different letter contractions used in contracted braille (including 75 shortform words like "him", which are simple abbreviations). These "short cuts" are used to reduce the volume of paper needed for reproducing books in braille and to make the reading process easier. Most children learn contracted braille from kindergarten on, and contracted braille is considered the standard in the United States, used on signs in public places and in general reading material.

Just as printed matter can be produced with a paper and pencil, typewriter, or printer, braille can also be written in several ways. The braille equivalent of paper and

pencil is the slate and stylus. This consists of a slate or template with evenly spaced depressions for the dots of braille cells, and a stylus for creating the individual braille dots. With paper placed in the slate, tactile dots are made by pushing the pointed end of the stylus into the paper over the depressions. The paper bulges on its reverse side forming dots. Because of they are inexpensive and portable, the slate and stylus are especially helpful for carrying to jot quick notes and for labelling such things as file folders.

Braille can also produce by a machine known as a braillewriter. Unlike a typewriter which has more than fifty keys, the braillewriter has only six keys, a space bar, a line spacer, and a backspace. The six main keys are numbered to correspond with the six dots of a braille cell. Because most braille symbols contain more than a single dot, combinations of the braillewriter keys can be pushed at the same time.

Figure 3

Technological developments in the computer industry have provided and continue to expand additional avenues of literacy for braille users. Software programs and portable electronic braille devices allow users to save and edit their writing, have it displayed back to them either verbally or tactually, and produce a hard copy via a desktop computer-driven braille embosser. Because the use of computers is so common in school, children learn both the braille contractions and also how to spell words out letter for letter so they can spell and write using a keyboard.

Figure 4

Since its development in France by Louis Braille in the latter part of the nineteenth century, braille has become not only an effective means of communication, but also an essential avenue for achieving and enhancing literacy for people who are blind or have significant vision loss.

1.5 Android (Operating System) - Revolution in Mobile Technology

Android's mobile operating system is based on the Linux kernel and it is a software stack for mobile devices. This operating system is one of the world's best-selling Smartphone platform.

Android involves many developers writing applications that helps in extended the functionality of the devices. There are currently over 1, 50,000 applications available for Android. Android Market is the online application store run by Google, though applications can also be downloaded from third-party sites. Developers write in the Java language.

The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 80 hardware, software, and telecom companies devoted to advancing open standards for mobile devices. Most of the Android code is released by Google under the Apache License.

The Android open-source software stack consists of Java applications running on a Java-based, object-oriented application framework on top of Java core libraries. Libraries written in C include SQLite relational database management system, WebKit layout engine, SGL graphics engine, SSL. The Android operating system, including the Linux kernel, consists of roughly 12million lines of code including 3million lines of XML, 2.8million lines of C, 2.1million lines of Java, and 1.75million lines of C++.

1.6 Version History

After original release, many updates of androids have been seen. These updates focus on fixing bugs as well as adding new features. Each new version is developed under a code name based on a dessert item.

The most recent released versions of Android are:

- 2.0/2.1 (Eclair), which introduced HTML5 and Exchange ActiveSync 2.5 support
- 2.2 (Froyo), which introduced speed improvements with JIT optimization , Wi-Fi and Adobe Flash support
- 2.3 (Gingerbread), which introduced the soft keyboard and copy/paste features, and added support for Near Field Communication
- 3.0 (Honeycomb), which supports larger screen devices and introduces many new user interface features, and supports multicore processors and hardware acceleration for graphics.

1.7 Hardware Running Android

The main supported platform for Android is the ARM (Advanced Risc Machines) architecture. ARM is one of the most licensed and thus widespread processor cores in the

world. It is used especially in portable devices due to low power consumption and reasonable performance. Now a day's cell phones, note books and tablets, including the Dell Streak, Samsung Galaxy Tab, TV and other devices can use the functionality of Android. HTC Dream was the first phone to run android, released on 22 October 2008.

1.8 Features

Current features and specifications:

- Bluetooth, edge, 3G, Wi-Fi support.
- Camera GPS, accelerometer support.
- GSM telephony support.
- Integrated browser based on the open source webkit engine
- Media support for common audio video, still image formats.

1.9 Problem statement

The problem that we are planning to tackle is to create an application that will cater to the basics needs to a visually challenged individual. We intend to make this application in which a way that the visually impaired individuals can use this android powered device so as to attend all the functionality that this device can provide.

Since, all the braille devices are all based upon refreshable screen and brallie keyboards, which are not convenient enough to carry all around. But in today's generation a smart phone or a smart device can fit in anyone pocket. So we want to empower our smart devices with capabilities such a braille code writing, learning, interaction etc., so that it eliminates the need for other mediums which are not handy in nature and provides easy accessibility for the visually challenged Individual.

1.10 Objective

Our main objective is to make use of the powerful android platform so as to develop such an application which will cater to the needs of a visually challenged individual.

Nowadays, Smart devices are abundant in our society. They have enabled mankind to make huge stride in the field of technological development. They can be used to as:

1. A source of Entertainment
2. Calling
3. Texting
4. Social media

And many more.

But these devices all work on the idea of a feedback i.e. if a user gives a command, they give the output. But in case of a visually impaired individual, it is impossible to use such a device. Since these devices (mainly smartphones, tablets) are screen based devices, they would not be able to give the visually impaired individual the feedback they require.

We intend to make an application that would work on any android based device so that these individuals can make use of these devices so as to access their smart phones, for easy typing or get a voice feedback on what they are typing and so forth.

The Different modules that we are going to implements are:

- Text Keyboard module
- Calling and accessibility module
- Learning Module.

CHAPTER 2

LITERATURE REVIEW

For this section, we have reviewed some of existing android application which use android graphics as their base. Some of the applications are stated below:

2.1 Art Flow: Paint Draw Sketchbook

This is a basic application that uses android graphics so as to convert an android powered device whether be it a tablet or a phone into a digital platform so as to sketch and paint a canvas or screen surface. It gives the use a fast intitutive painting application that allows to use his or hers complete imagination to paint and display the painting on the screen of the device.

Some of features that it provides are as listed below:

1. It provides a high performance painting Engine.
2. It has about 70 or more different types of tools to paint.
3. Million combination of colours
4. Mistakes can be corrected with just a click of a button.
5. It can work on high resolution screens, 1024*1080, 2K and 4K displays.
6. It has multi-touch feature that works on multi-touch android devices.
7. Support for different geometrical shapes.

And many more.

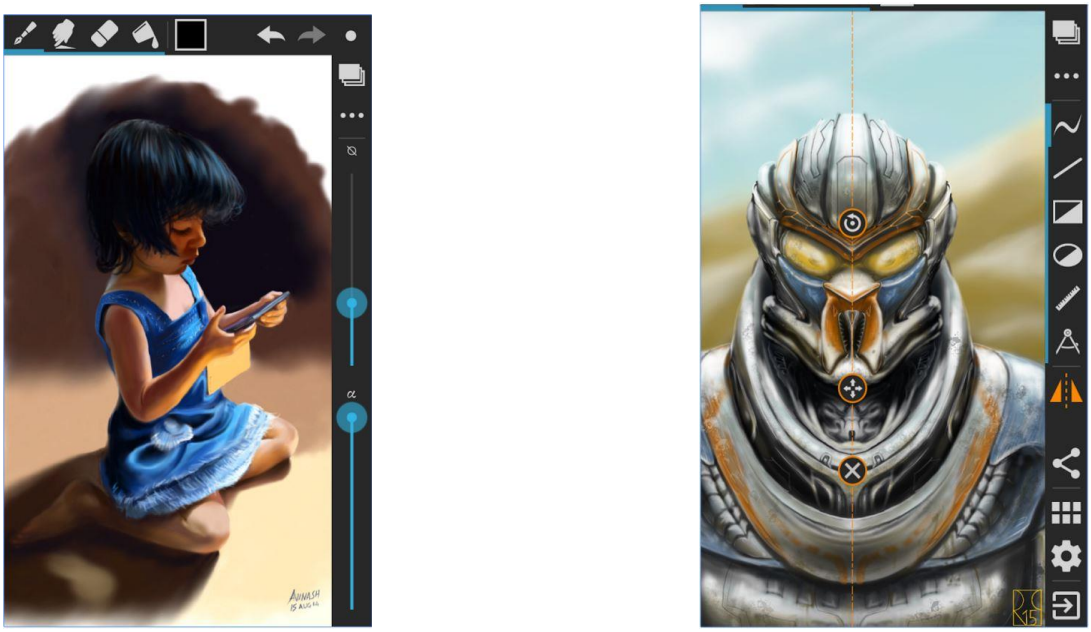


Figure 5

2.2 Refreshable braille display

This is display that basically an electro-mechanical device that is used for displaying or showing braille characters, which are usually displayed by the use of round-tipped pins that are raised through roles in the flat surfaces. As our application is based on braille reading and writing, it is important to understand how a braille terminal works. So those users who are blind, and cannot use a computer screen to read a text output, they can use this electro-mechnical device which can give feedback to the users as such.

It can also be stated that speech synthesizers can also be used for this similar task, and give the blind user to switch between any two systems or use both of them in the same time interval depending the use of the application.

The inputs that are given to the refreshable braille which are formed by two sets of three keys as well as adding a space bar. The output is given by a refreshable braille display which consist of a row of electro-mechanical character cells, each of which can give a combination of six pins or dots. These combination of round-tipped pins will give an output to the user leading to understanding of what the combination the tips mean.

This mechanism that gives the user feedback is based on the use of piezo effect of some crystals, which expand when a particular voltage is applied onto them. This crystal is then connected a lever, which give the different combinations of the dots. Therefore there are eight crystals for each dot of the display.

The software of the display is called as screen reader. It uses the contents of the screen and sends it to the operating system, which further converts into the braille language, which gets analysed and then sends it back to the display. These screen readers are basically very complex in nature because the different graphical components like the crossbars or window panes are to be interpreted and written in the text forms. The newer OS have now Application Programming interface which help the screen reader to obtain the needed information.

Some of the examples of this software are:

- UI Automation for Microsoft Windows.
- VoiceOver for OS X and IOS
- AT-SPI for GNOME.

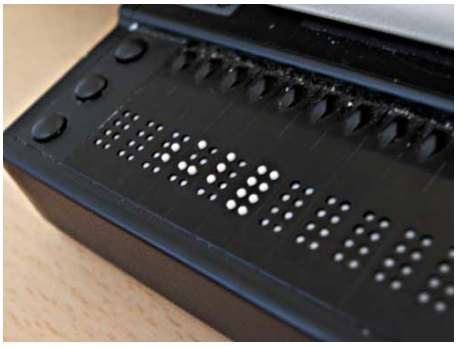


Figure 6

2.3 Google Talkback

This is an application that acts as an accessibility service that would help the visually-impaired or challenged users to interact with their android devices.

It has a three main components of our importance that are being listed below:

1. Spoken: This aspect of this application gives the user the vocal feedback as to which the user is supposed to get when he or she presses the screen of the android device.
2. Audible: This aspect mainly focus on hearing what the user says and act upon it.
3. Vibration: Different vibrations schemes are being used to give the user a certain kind of feedback for different actions are taken up by the devices.

This application is going to help our application to actually act as a moderator b tween our application and the user. Since we require all the three components for our working of the application we plan to make, this application is quite important for us to study.

It is a system application for most of the android devices, and are now already installed on the devices.

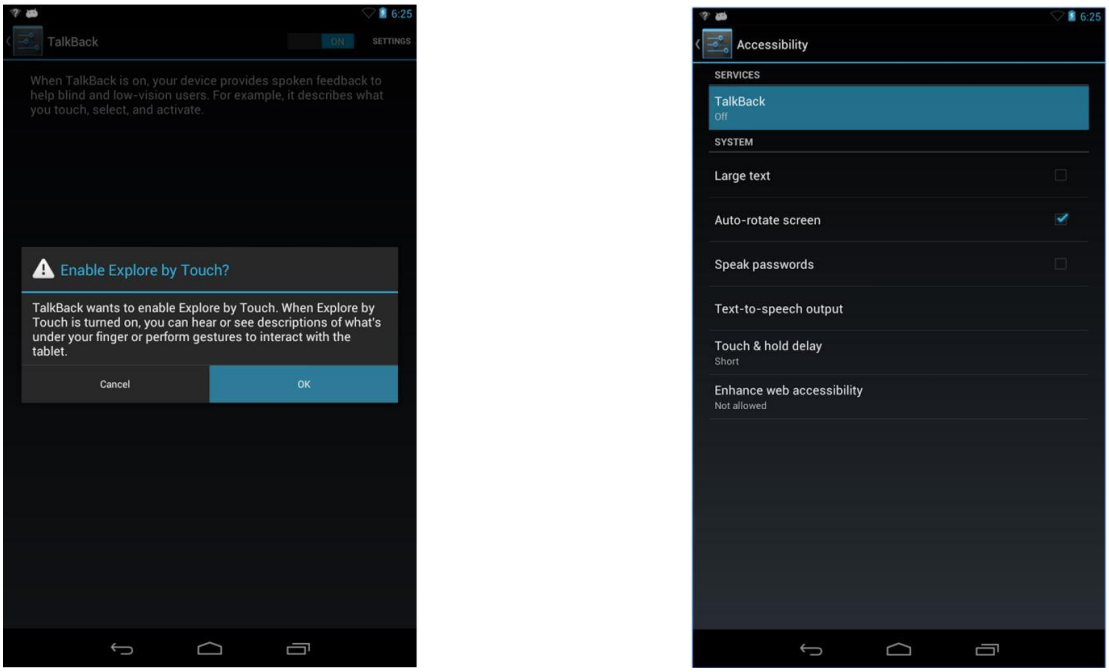


Figure 7

2.4 Start

This application is used for the pattern recognition that we want to incorporate in our application. This application is basically a lock screen application. It gives the user a nine point pattern choice. The user then selects a pattern and the applications saves this. So as to open the device, that particular pattern has to be provided by the user. This feature

uses the pattern recognition that we require. Since braille characters is a combination of six dots, therefore we want to use this pattern recognition in our application by studying about start application.

It has further applications such as:

1. Helps in security and maintain privacy.
2. Provides quick access to important application that user requires.
3. Allows beautiful themes and wallpapers.
4. Helps to personalise the device to the most elaborated details.
5. Also provides updates on quiz etc.
6. News feed
7. Weather updates
8. Smart Search.

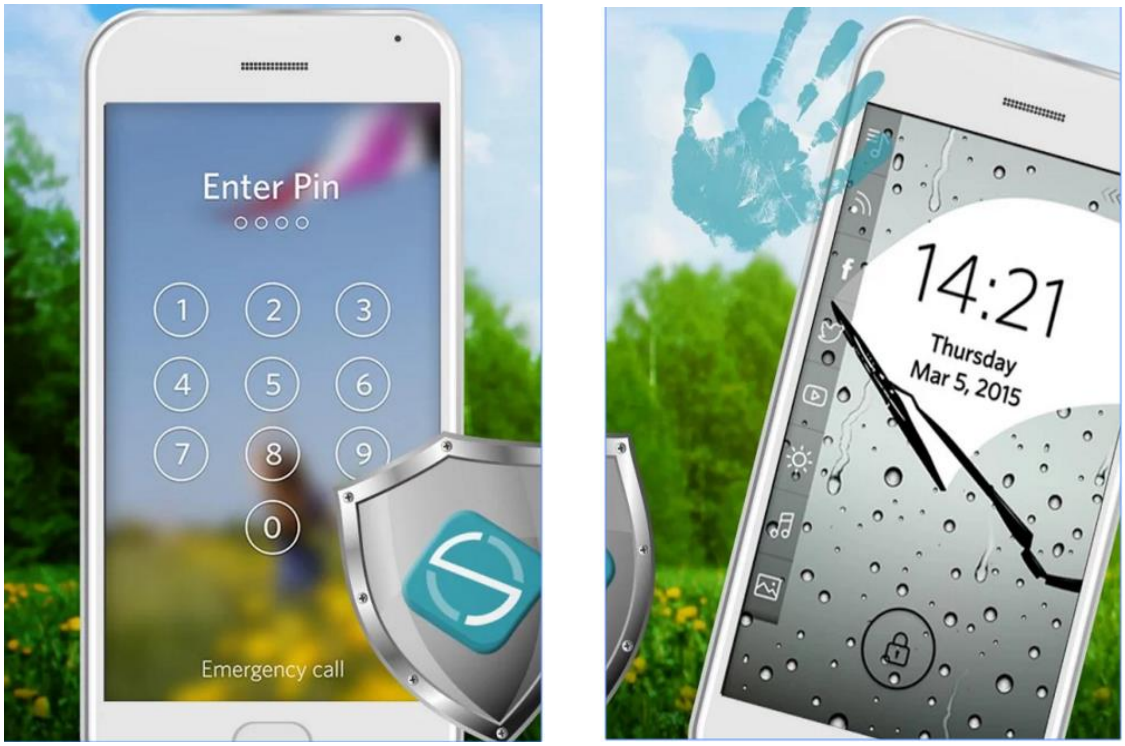


Figure 8

2.5 BrailleBack

This is an accessibility service that is provided by google to the help the visually challenged individuals to make use of braille devices. It works with the talkback application as mentioned above so as to give a combined braille and speech experience. This applications lets the user connect a supported refreshable braille display to your android supported device via Bluetooth. All the screen contents are then presented on the refreshable braille display and it can also be used to navigate and then interact with the android device using the keys on the display. The braille keyboard can also be used to take input from the user.

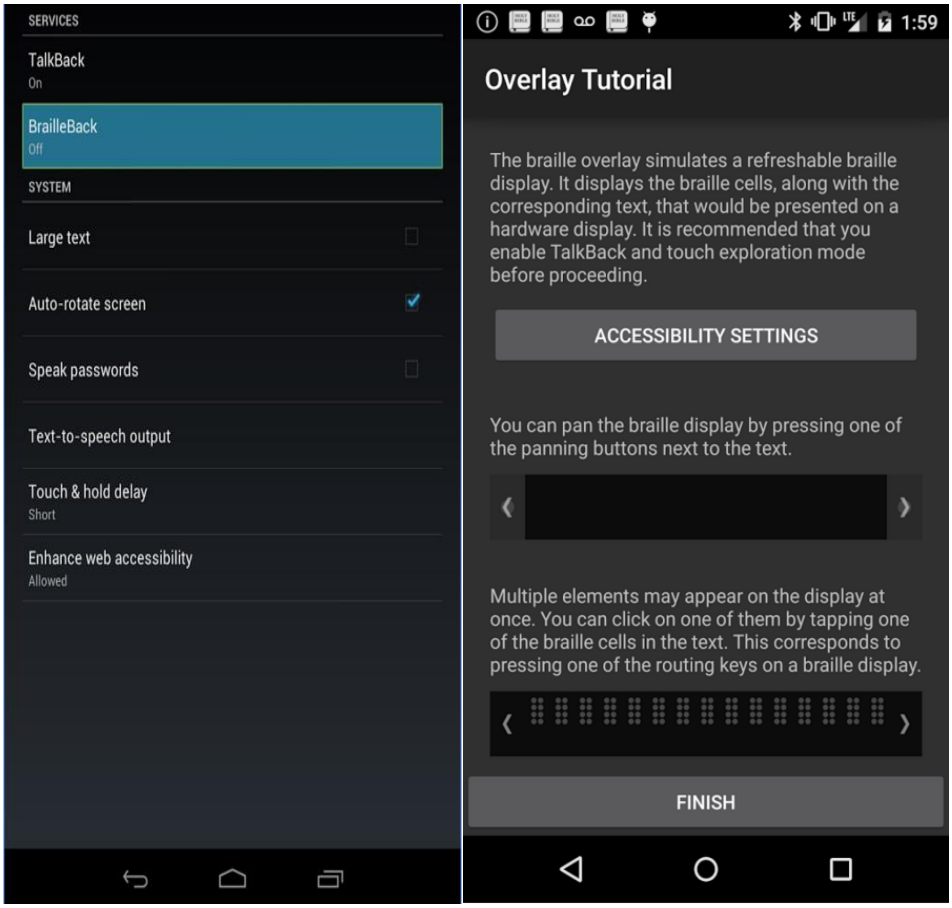


Figure 9

CHAPTER 3

ALGORITHM (BRAILLE CODE READER)

(1) Calculation of touch area and division.

Calculating the screen width and length.

Dividing Screen length by 3

Division Area Length=Length/3;

Division Area Width=Width/2;

So in total whole screen would be divided in six rectangular areas. These are our Touch areas.

Touch Areas named as 1 to 6. (Increasing downwards, Left to right).

(2) A Timer of 0.6 to 0.8 sec for detecting any input sequence. Once screen detects a touch input, timer starts and the touched regions would be detected and added to the queue.

Two queues will be used:

- **Buffer Queue**: storing all the typed words

-**Input Queue**: second for storing a validating the input regions.

(3) A handler running in background to detect touch inputs, and detecting valid Combinations.

When even any touch event is occurred, our handler would start adding the multi touched/ Single touched areas to the input queue until the current timer time expires.

After expiration, the input queue contents would be checked for valid combinations and if a valid character is found, the character would be added to the buffer queue.

The input queue will be flushed regardless of whether input was valid or not.

(4) The sequence of any input does not matters. i.e., for example Alphabet ‘b’ has Braille code in regions 1, 2.

(5) Now if the user presses region 2 then region 1 or vice versa, the end result should be the same. So that's why there is a need for a standard format of input for checking the valid characters.

After every time out the input queue would be flushed and the system would be ready for accepting next input.

The user will have the option to playback the written content in the current session by using the short cut- Swipe-Left. The entire content of the Buffer Queue would be send to speak () method of TTS.

(6) The contents of the input queue would be sorted first, and then duplicates if any would be removed. Now since the combinations are stored in the Hash Map with Braille sequences as the unique key, the sorted string should now point to an entry in the Hash Map. If the hit on the hash map is successful, then the value retrieved by the hash map (Alphabet in English) would be added to the Queue.

3.1 For The learner module

User will give speech input and then TTS would synthesize the input and parse it to an English Alphabet, and then its Sequence would be used to draw on the canvas and also give as speech.

The retrieved sequence would be incrementally broken into individual digits and then using canvas those digits would be decoded to Cell No1 and that cell would be illuminated.

3.2 For the Call Module

The number entered by the user would be given to the calling intent to call the recipient.

For Permanent Storage of Braille Code Sequences:

Since data stored would be lost when the app restarts, so we need a more reliable method to store the data and even app restarts, we will build our Table again with minimum overhead.

There are two Approaches:

(1) JSON Serialization (2.) SQLite

1. **JSON Serialization:** For this we need to add GSON library in our project.
    - Wrap the map to a class.
    - Serialize Hash Map Object to JSON
    - Store it in Shared Preferences.
    - On Restart, retrieve the string and DE serialize.

2. Using SQLite:

SQLite has a Cache mechanism and SQLite was specially designed for storage and retrieval of large amounts of data. SQLite will also cache some of most used fragments of data. We will have to store sequences in the database only for the first time.

Whenever the App starts, we will have to create a connection to the database and use the same connection until the App is closed down. Then the database connection will also close.

3.3 Android Libraries

The major android components used in the application are:

- Gesture Detector.
- Android TTS engine.
- SQLite.

3.3.1 Gesture Detector

This class detects various gestures and events using the supplied MotionEvents. The GestureDetector.OnGestureListener callbacks are used to notify users when a particular motion/touch event on the screen has occurred. This class should only be used with MotionEvents reported via touch (don't use for trackball events).

Android provides various touch screen events such as double tap, pinch, long presses and flinch. These are all known as gestures.

To use gestures, we need to:

- Create an instance of the Gesture Detector for the View.
- In the on Touch Event (Motion Event) method you have to call on Touch Event (Motion Event). The methods defined in your callback will be executed when the defined events occur.
- If listening for on Context Click (Motion Event) you must call on Generic Motion Event (Motion Event) in on Generic Motion Event(Motion Event).

Various classes and interfaces, provided by it are as follows:

- **OnContextClickListener**: This listener is used to notify when a context click occurs.
- **OnDoubleTapListener:** This listener is used to notify when a double-tap or a confirmed single tap occurs.
- **OnGestureListener:** This listener interface is used to notify when gestures occur.
- **SimpleOnGestureListener:** A class which should be extended only when listening for a subsets of all the gestures.

3.3.2 Android TTS (Text To Speech)

TTS is a Voice engine which enables a user to convert Text into Speech output and creating audio files from text. This is used in various applications like alarms, text to speech conversion applications (Like Android Talk Back), games, for creating audio files etc.

For using TTS, we have a first instantiate a TTS object and specifying the Listener (onInitListener).

```
private EditText write;
ttobj=new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
   @Override
   public void onInit(int status) {
   }
}
);
```

In listener, we have the choice of specifying the languages. For any language to be used, there must be voice data installed on the device for that language.

After initializing a TTS object we have the option of setting various desired properties, but the major Flags are:

1. Language
2. Speech Rate
3. Pitch

To set the language, we use set Language **()** which requires a Locale object for specifying Language.

```
ttobj.setLanguage(Locale.UK);
```

There are various Locale Available are:

- UK (Female)
- CHINA
- CANADA_FRENCH
- GERMANY
- ITALY
- .JAPAN
- US (Male)

After setting the language, it is very important for the host device to support that language. We should first check if the Language or Voice data for the particular language is available or not.

Now for converting text to speech output, we use speak () to for calling speech conversion.

TTS_Object speak(String , Queue Flags , Utterance_ID);

String: The text sequence to be converted in speech

Queue Flags: For specifying the Queue behaviours for playback. There are two main Flags available:

QUEUE_ADD:

Using this flag, every entry will be added to the end of the playback queue.

Constant Value= 1

QUEUE_FLUSH:

In this queue mode, all entries in the playback (text to be played) is replaced by the incoming new entries. Queue will be flushed with respect to the inputs processed

Entries from other calls are not discarded in the Queue.

Constant Value: 0

Setting QUEUE_ADD flag would keep on adding inputs to the queue, whether previous requests have been performed or not. It is a sync with the queue callbacks as it does not waits for the queue to return anything.

For e.g., we input braille code of a, five times and before letting previous requests to be processed. Then the Android TTS would output "a" five times.

Setting QUEUE_FLUSH flag would keep on clearing the previous or the pending requests if they have not been started processing.

For ex, In above scenario, where user enters "a" 5 times, before even getting outputs of previous inputs then we would get only current inputs as all the previous input requests are flushed and only the current request is stored on to the Queue.

3.3.3 SQLite

SQLite is the default built in database implementation in Android. SQLite is an open-source database that helps to store data on to the device in the form of text files. It supports

all features of Relational Database. It provides methods of create, delete and execute SQL commands and other database related work.

For create table and storing data, we need to create a class which extends **SqliteOpenHelper.**

To create a database, we need to call **openOrCreateDatabase().**

Insertion:

We can insert data into the table by storing query in a string and passing it to execSQL method.

```
SQLiteDatabse mydatabase = openOrCreateDatabase("your database name",MODE_PRIVATE,null);
```

Retrieval:

To retrieve data from the database, we can use Object of the Cursor class. We can execute a query using **rawQuery**. This will return a **resultset** with cursor pointing to the table. We can fetch data row by row using cursor by moving it forward till all the rows have been retrieved.

```
Cursor resultSet = mydatbase.rawQuery("Select * from TutorialsPoint",null);
resultSet.moveToFirst();
String username = resultSet.getString(1);
String password = resultSet.getString(2);
```

We can also use other methods for more information like:

- **getColumnCount():** Returns the number of columns in the rows retrieved.
- **getColumnIndex(String name) :** It returns the index of the column named as "name".
- **getColumnName(int index) :** This is the revere of the previous function. It returns the column name of the index passed as parameter.
- **getColumnNames():** This returns names of all the columns in the table. We have to use array to store the result.

- **getCount():** It will return the number of rows in the result.
- **getPosition():** This returns the current position of the cursor variable.

3.4 BRAILLE CODE READER (BRAILLED)

3.4.1 APPLICATION DESIGN

Braille code reader provides a way for the visually challenged people to type and communicate with each other, help people learn Braille and even help to call some of the emergency contact. At all activities, there will be audio assistance available for the user for swift and smooth interaction. Since there will be some special function keys required, we will be hard coding some of the special keys and notify the user through speech interactions.

At any point of navigation, the user will have the option to call help using a special mapping for the Speech assistance of Special function keys, as well as other functions.

Braille uses a Standardized mapping of characters which are uniform throughout the world and even the nomenclature of cells is standardized, which will help to interact with user unambiguously. Braille code is a combination of any of the six cells and every combination is unique. Cells are numbered from 1 to 6 (2 columns with 3 cells each) top-down and left to right manner as shown in figure

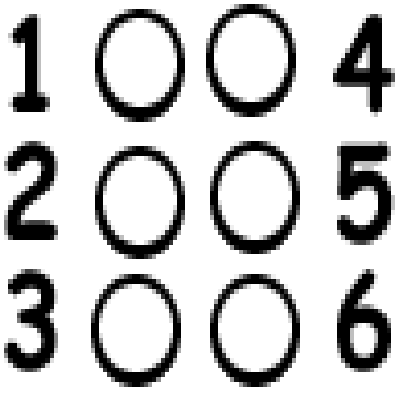


Figure 10

Since the numbering is same, throughout the world, special commands can be coded easily and would be understood uniformly anywhere, anytime.

There would be many special mappings like Space character, switching Type mode, Help and Options Mode. There would be also special keys for sending the message and calling a contact.

BrailleD will have 3 main modes:

- **Texting Mode**
- **Calling Mode**
- **Learning Mode**

Shortcuts for facilitating typing:

- **Swipe Right**- Sending the Typed input to a contact or it can be used for browsing (in Future Prospects).If there is no typed input , Then a voice output will be given to prompt the user to type, otherwise user would be prompted to select saved contacts (out of the 9 Contacts) or type the recipient number.

- **Swipe Left**- For playing the audio feedback of the typed inputs in the current session. As of now, the entire input stream would be played which can be hectic in case user wants to check only last few words out the very big document. This functionality can be extended by using another auxiliary Queue and setting its capacity to a fixed number of inputs. In any time when new inputs are added and capacity would be overflowed, the oldest inputs would be Flushed (Deque) and new inputs would be pushed (Enque) to the queue to maintain a call back of fixed last input recall.

Texting Mode

Texting mode is the main dash board for the user. This is the Mode which will help users to type Braille text and send it as a message. While this functionality will be extended

to more options like Emailing, Web surfing and using it Sync with Android TalkBack for audio feedback of search result content.

As the App starts, this will be default Screen shown to the user. Voice output would be given that the BrailleD is ready. User can now start typing and scanned inputs would be added to the queue. Whenever user touches certain areas, when touch even is recorded and an input sequence is generated. If this sequence matches to any of the existing commands, then the Input would be added to the queue and speech output would be given back to the user. Some special combinations which don't match with any existing sequences. User can use the special Input for sending the text through options mode or can use the Shortcuts for calling option mode and sending it or switching to other functionalities. This activity would have no View elements and UI except a small TextBox in order to provide maximum area for the visually challenged user to give touch inputs. As almost whole of the Activity screen is available to the user, there will also be less mistakes while typing.

TextBox provided will show the current input scanned and will help the developer for the testing purposes also.

Learning Mode

This is a very simple functionality to help people who want to learn Braille. In this context, the functionality would be reversed, instead of providing speech output, we would we taking speech inputs. Whenever the learning mode activity starts, out app would be ready to synthesize speech inputs and will parse the inputs and give audio feedback of the asked character sequence. Using canvas, the pattern would also be drawn on the screen. In this activity, App would no longer accept Braille inputs except for special inputs like switching mode and Help function.

The module will provide learning for all the characters, numbers and alphabet. The output will be in form of sequence, for ex:

User speaks "B", and gets output "1 2" and also a pattern is drawn on the screen.

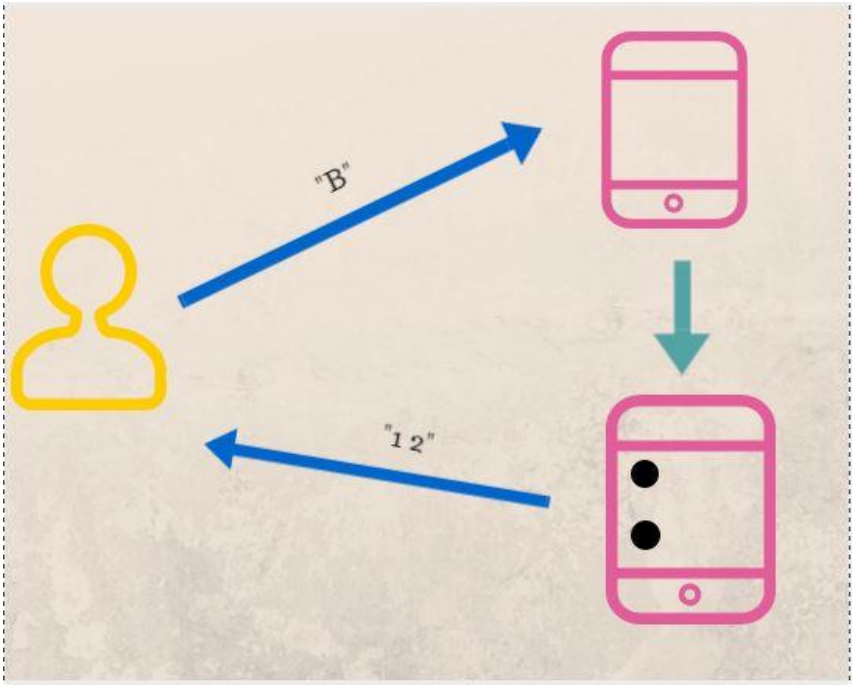


Figure 11

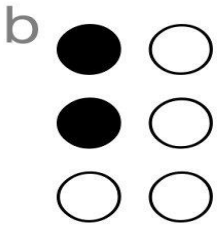


Figure 12

Which is the braille code of "b".

User can any time switch to Options mode , using a sequence "1 6" (Diagonal two corners of the screen).

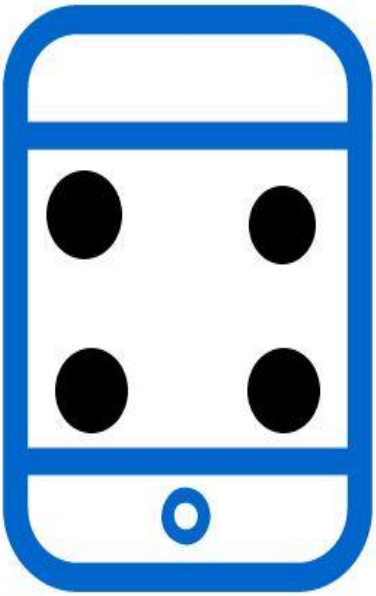


Figure 13

This will open up a menu with audio sync, to narrate the options.

Figure 14

Option 1- Text Board: To switch to the default typing screen.

Option 2- Fast Key On/Off Toggle: This is for further improving accessibility. Fast key mode would count the number of pointers active on the screen. Pointers in android mean the number of touch points. Since these making these options touchable would be meaningless for a visually challenged person, user given be given speech narration of various options. Also when this mode is on, user can also give speech input and app would be ready for both touch inputs and listening to speech inputs for selecting options.

If fast key mode is off, then user will have input the corresponding Option number like, for selecting option 4 - to exit the app, the user will have to type the Braille code for number 4.

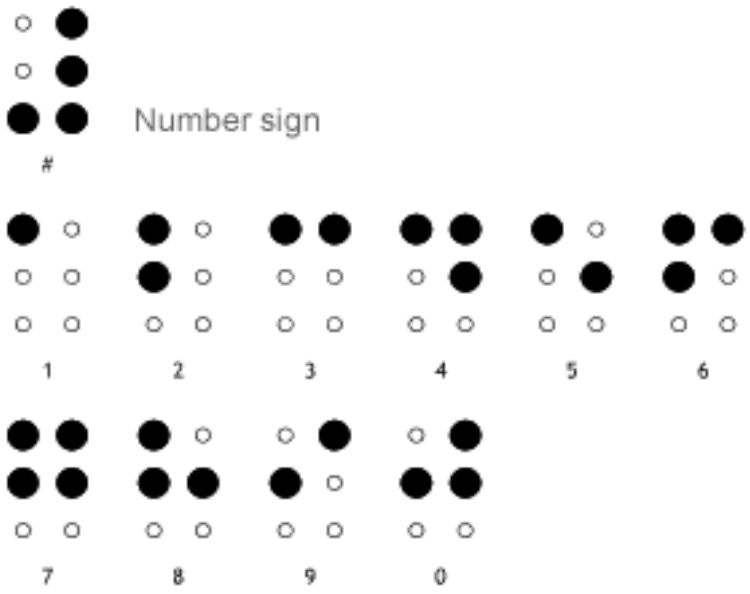


Figure 15

If the Fast Key mode is on, then for selecting option 4, the user will have to touch on four points anywhere on the screen, and the count of active pointers can be used for selecting options. This feature can be used for the situations when user can't use both hands on all times. Since the number of options are limited to 5 only, one hand can be used for at least navigating the options. This fast key Mode would be accompanied by voice options also.

The user can also give option number in speech input and then the voice commands would be synthesized and corresponding option would be selected.

Calling Mode

This will enable a user to call a contact. This can be either a saved a contact or a user can dial a number. There will be an option for adding new contacts and choosing a contact for calling (speed dial). At any time, there is option for saving 5 contacts.

For calling a contact, user would have to first user special command to navigate to option menu, from there he can choose "Change mode" and then choose call option.

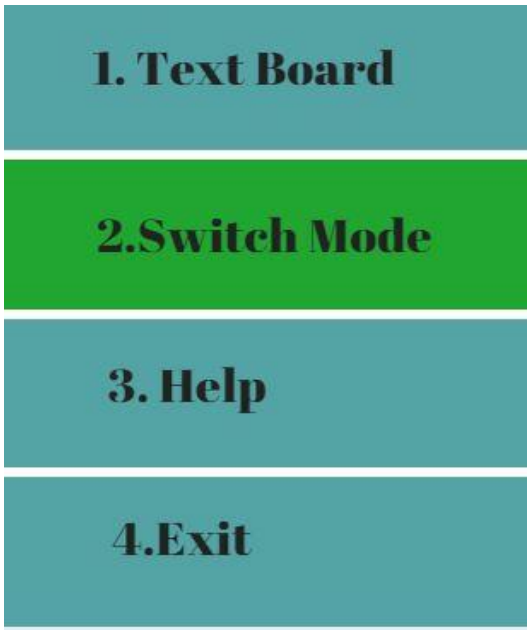


Figure 16

After choosing "Switch Mode" option, the new option menu will appear.

Figure 17

After selecting "Call Someone", we will navigate to another new, where another option will appear. They will help user to either call one of the saved 5 contacts, or dial a new number. Since we will provide Audio replay of the contacts, so that is why we have limited the number of saved contacts because it will become too hectic and time wasting to give entire contact list in speech results. Further, there is a possibility of using a contact search using audio callback or entering Braille code of name to be searched and the search results will be notified to the user in speech output.

Figure 18

Choosing Saved contacts will start speech library to utter all the contacts saved and their position. For getting the speech result, the user will have to SWIPE LEFT. User will be notified of the contacts with their respective position. The user will have to type the Braille code of that position. If the user enters a number more than the available contacts or greater than 5(since the maximum capacity is 5 contacts), then the user will get an error message audio, and the user will be prompted to again type the input.

After entering a correct contact position, we user will have 3 options,

1.) Call Contact

2.) Edit Contact

3.) Back

Dial Number Screen

Figure 19

The user can enter the braille code and Swipe left when done entering number to call the given contact. The number would be parsed and given to calling intent for connecting the call.

Contact List

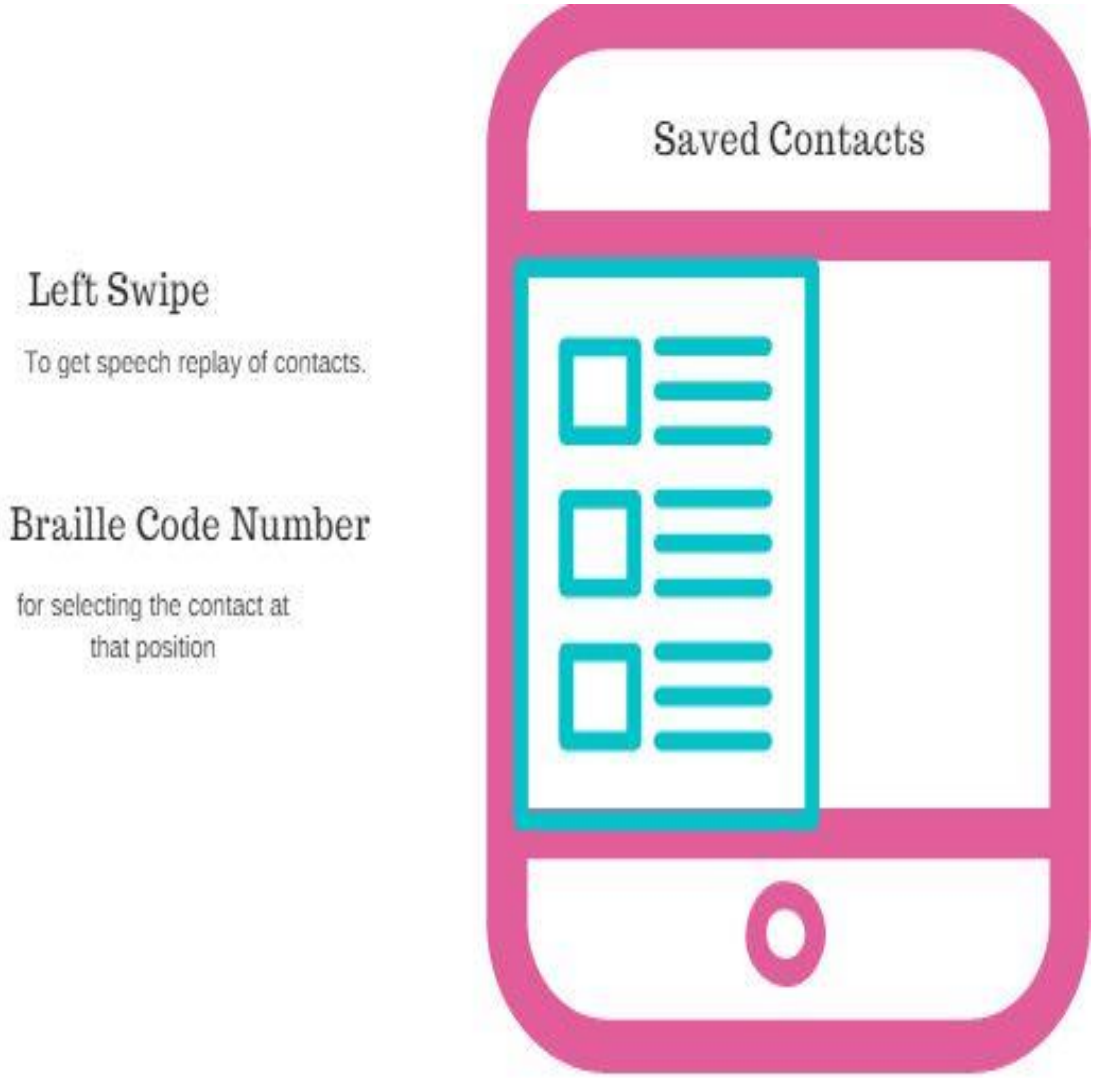


Figure 20

After selecting the contact list, the user would be given speech output of the stored contact

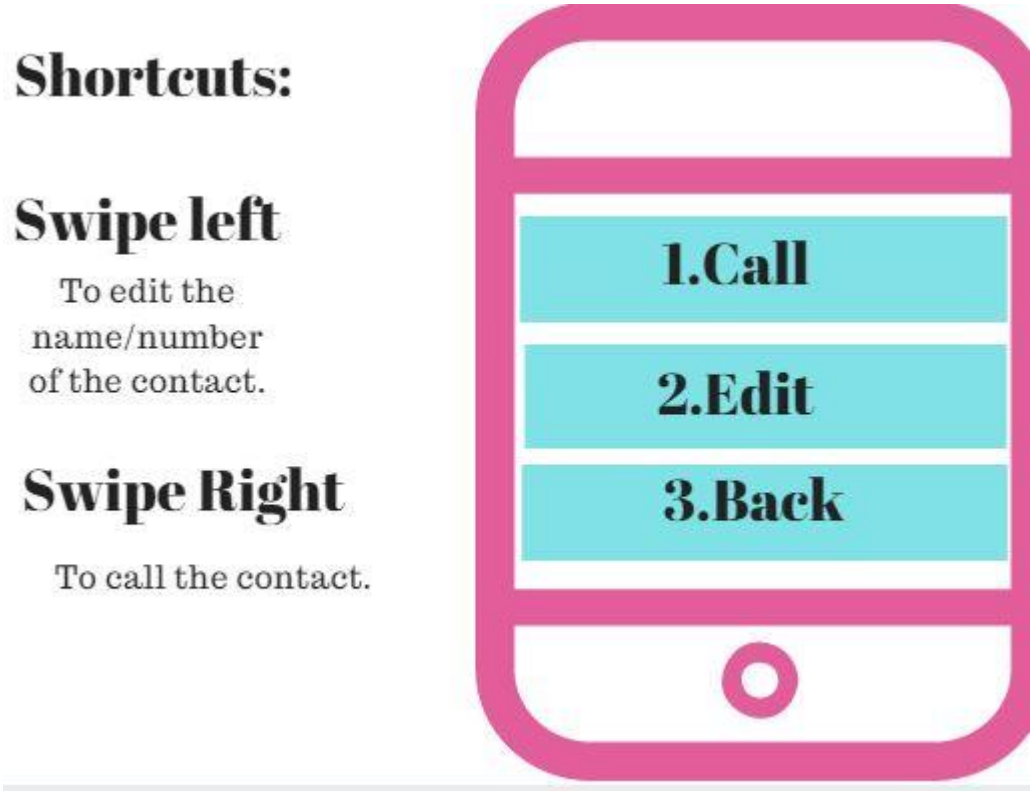


Figure 21

For Mapping of space Character, we have special Code (3 6) for leaving one Blank.

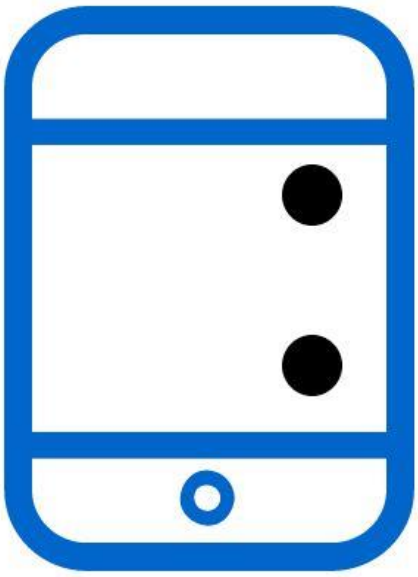


Figure 22

Braille Sequences

The braille codes are converted into their respective regions and then it produces a unique sequence. The following are the Sequences when are standard Braille characters (Alphabets and Numbers) and some are special Mappings.

Braille Character	Sequence	Braille Character	Sequence
a	1	I	2 4
b	1 2	j	2 4 5
c	1 4	k	1 3
d	1 4 5	l	1 2 3
e	1 5	m	1 3 4
f	1 2 4	n	1 3 4 5
g	1 2 4 5	o	1 3 5
h	1 2 5	p	1 2 3 4
q	1 2 3 4 5	#(Number Sign)	3 4 5 6
r	1 2 3 5	1	1
s	2 3 4	2	1 2
t	2 3 4 5	3	1 4
u	1 3 6	4	1 4 5
v	1 2 3 6	5	1 5
w	2 4 5 6	6	1 2 4
x	1 3 4 6	7	1 2 4 5

y	1 3 4 5 6	8	1 2 5
z	1 3 5 6	9	2 4
		0	2 4 5

Table 1

For Special Sequences

Braille Characters	Braille Sequences
Space	3 6
Option key	1 6
.(period)	2 5 6
,(comma)	2

Table 2

CHAPTER 4

PERFORMANCE ANALYSIS

4.1 HashMaps

Since, braille codes are saved in form of key-value pairs with sequences as the key and corresponding key or characters, the size of hash-map would become very large.

Since every object in java has a hashCode() which is used to map it in Java collections, we can use hashing techniques like taking modulus of sequence with a large prime number to reduce the size of hashmap.

In Hash maps, in case of collisions the multiple keys would be mapped to single index and then multiple nodes would be linked in the form of linked list as shown in figure:

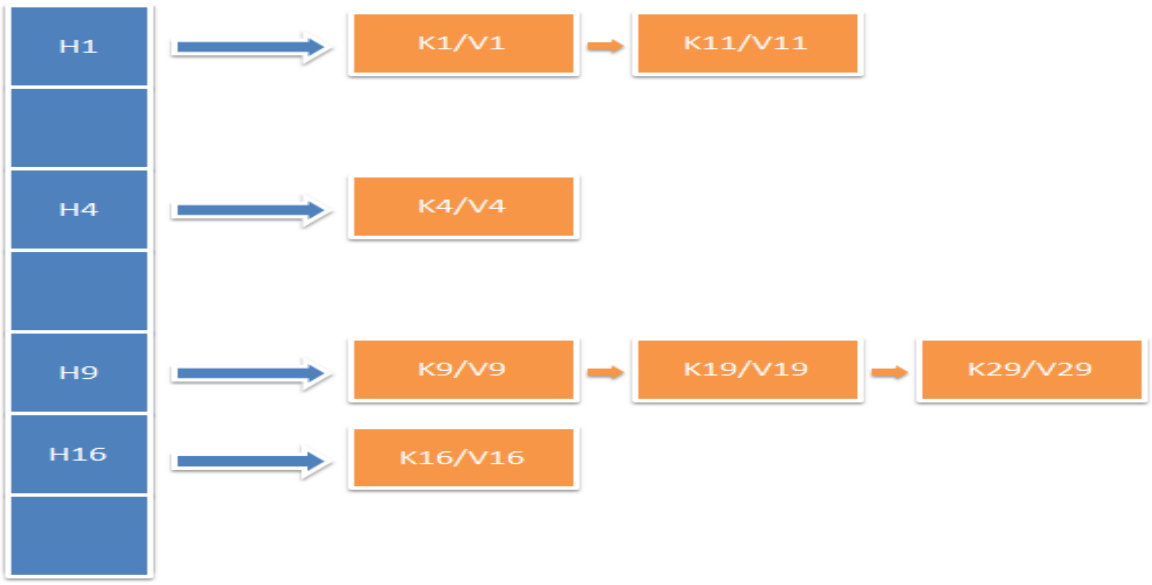


Figure 23

In this example, at index 9 (H9), there are multiple nodes which point to the same hash codes. The nodes are then linked in the form of linked list. When searching is performed,

then the hash code is calculated and then checked if the key as well as the hash code must be matched and all the nodes are traversed using links of the Linked List.

After the application is deployed, then will analyze the size as the hash map grows and the time to return value from the hash map.

We can use load testing to analyze the application performance with respect to number of inputs given and number of outputs given by the Android TTS.

4.2 SQLite Performance Analysis

Other option to store Braille codes is in the form of table in SQLite provided by android.

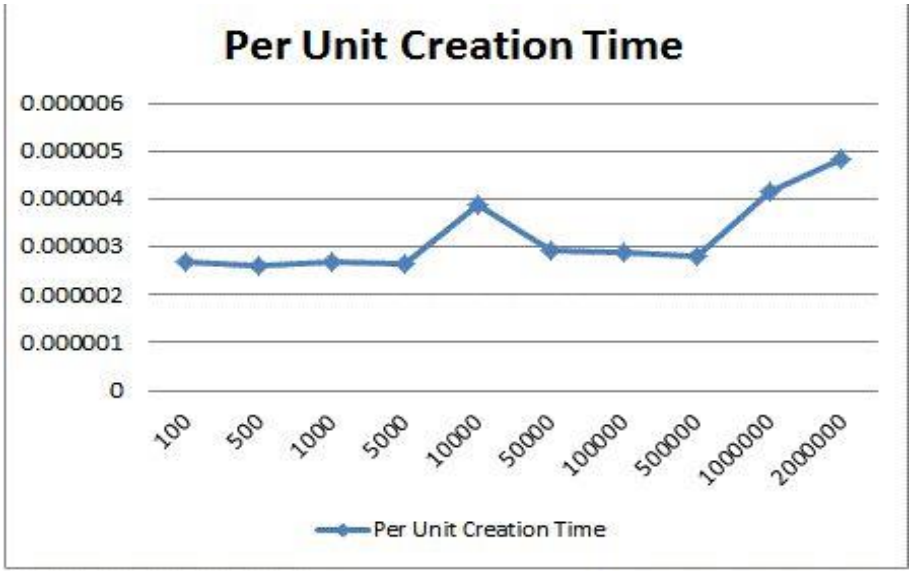
We have used a performance analysis done by a web site to get idea of the time for execution query as the size of database hits increases.

Machine Specs Used: CPU: Intel Core 2 Duo @ 2.33 GHz, 4 GB RAM, 4 Mb cache.

4.2.1 Per Unit Creation

TableSize	Creation Time	Per Unit Creation Time
100	0.000267029	2.67029E-06
500	0.001297	0.000002594
1000	0.00267196	2.67196E-06
5000	0.013272	2.6544E-06
10000	0.038739	3.8739E-06
50000	0.145373	2.90746E-0
100000	0.289102	2.89102E-06
500000	1.40898	2.81796E-06
1000000	4.16353	4.16353E-06
2000000	9.66753	4.83377E-06

Table 3

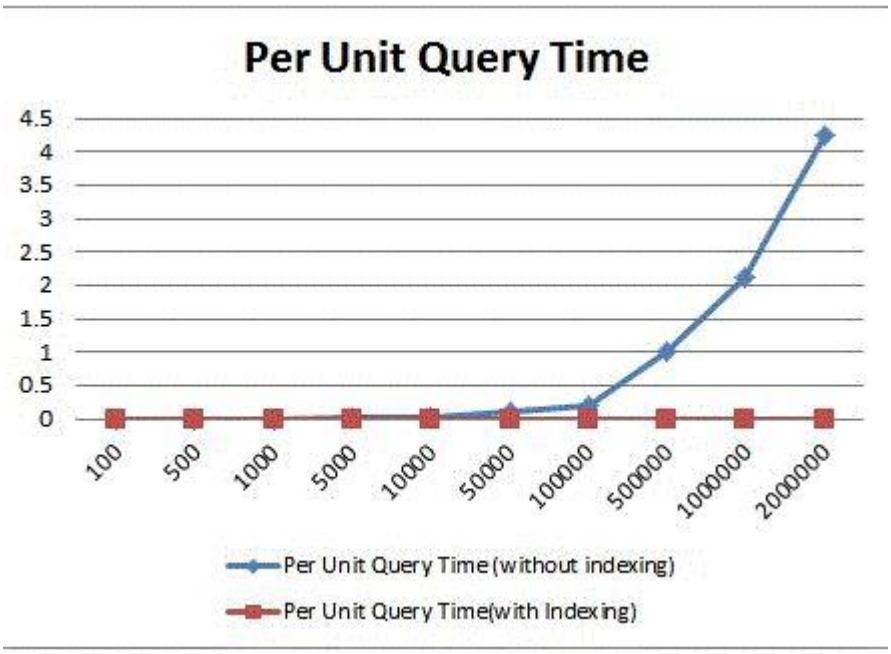


Graph 1

4.2.2 Per Unit Query Time (without indexing)

TableSize	Query Time Without Indexing(150 Queries)	Query Time/TableSize	Per Unit Query Time (without indexing)
100	0.0577531	0.0005775	0.000385
500	0.211629	0.0004233	0.001411
1000	0.409095	0.0004091	0.002727
5000	2.75264	0.0005505	0.018351
10000	5.05658	0.0005057	0.033711
50000	17.102	0.000342	0.114013
100000	32.2116	0.0003221	0.214744
500000	153.639	0.0003073	1.02426
1000000	317.506	0.0003175	2.116707

Table 4

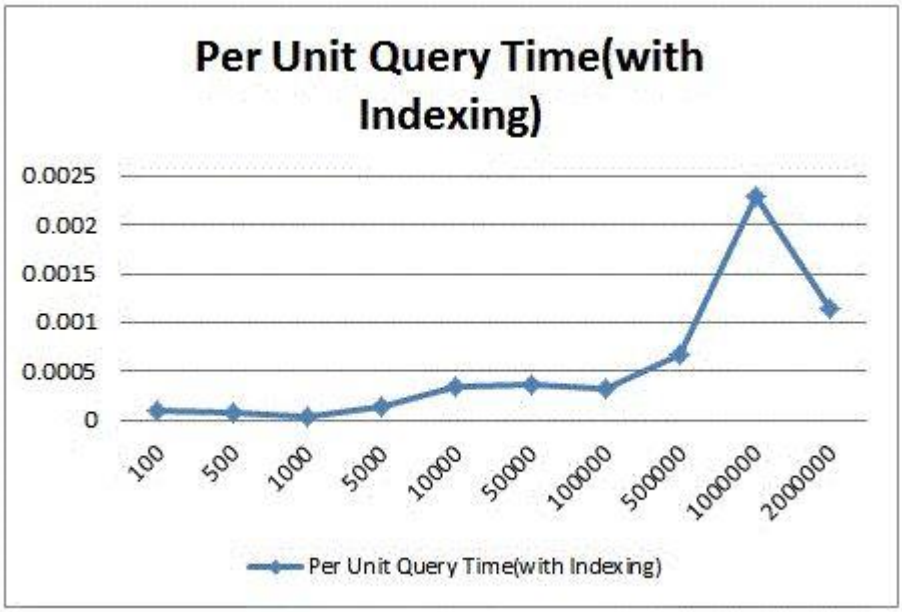


Graph 2

4.2.3 Per Unit Query Time (with Indexing)

TableSize	Indexing Time	Query Time With Indexing(150 Queries)	Per Unit Query Time(with Indexing)
100	0.00176001	0.014441	9.63E-05
500	0.0136271	0.012418	8.28E-05
1000	0.0286882	0.005842	3.89E-05
5000	0.157743	0.0193071	0.000129
10000	0.338535	0.050318	0.000335
50000	2.17958	0.0541899	0.000361
100000	5.44195	0.046885	0.000313
500000	35.41	0.100924	0.000673
1000000	94.695	0.344231	0.002295
2000000	193.37	0.170012	0.001133

Table 5


Per Unit Query Time(with Indexing)
0.0025
0.002
0.0015
0.001
0.0005
0
100
500
1000
5000
10000
50000
100000
500000
1000000
2000000
Per Unit Query Time(with Indexing)

Graph 3

CHAPTER 5

CONCLUSION

Our main objective of creating this application is to cater to the needs of a visually impaired individual. Our main motivation for this application was to make accessibility of a power platform like android that are available on every device to every user.

The integration of voice commands, voice actions and voice feedbacks would help the user to understand what action is being taken by the device. Another feature, that is the tutor feature would allow the different users to learn braille if they wish to. This application would also all the user to control their whole device, with just some possible braille key options. This would not only increase the efficiency but also the performance of the device.

Our calling Feature acts as a quick calling method to the frequently called individual in their phone. It also allows the visually challenged individual to always have a contact in their hands without evening opening the contact list. This makes accessibility much easy for the users.

Our teaching mode would allow any user who wants to learn braille. A perfect key cheat will be displayed on the screen and so forth can be used to learn braille key code.

Advantages

(1) Standard Unified Representation (Done in two rounds of unification in 1878 and 1951).

(2) Only a Touch Screen Phone is required. Now with the vast variety of smart phones blooming into the market, Mobile phone has become accessible to everyone and at a very affordable cost. The enhanced APIs and Larger screen display only add to the larger benefit to make phones capable to provide multi-functionality support.

(3)Only six dots are required to represent any Braille character in any language. Most of the languages having common alphabets have same representation of alphabets and add specific literals according to the platform.

(4)Multi-Language support are available in many platforms , for example Google TTS(Text To Speech) engine supports various major languages English (United States), English (UK), German, Hindi, Indonesian, Italian, Korean, Polish, Portuguese, Russian, Spanish (Spain), Spanish (US), Thai (Thailand) and Turkish (Turkey).

(5) Accessibility of smart phones makes it a fair better choice than to carry Braille Writers.

(6)Text-to-Speech engines may be used by apps such as Books on Google Play Store, for voice translation and pronunciation of written content, audio feedback accessibility applications like Google Talkback and many other third party apps. For this, the users must install voice data for each language.

Future Scope

Following are the modules that we want to incorporate in our existing modules:

1. Security Encryption: Security Encryption can also be done using this application. A certain message can be encoded into braille code, and then transmitted. This braille code can be similar to the existing braille characters or it could be entirely different to the similar. This would allow hard encryption and only decrypted if the cheat key is available onto the receiver's end.
2. Contact searching: Using the braille code, we wish to use it as searching mechanism to search a contact. This would not allow the visually impaired individual to search his whole contact list, but also it would be an efficient manner to go through the whole contacts.
3. We also wish to integrate the braille refreshable displays and be compatible with our application that runs on android devices.
4. Multiple language support: Currently, we are working on English alphabets for the braille reader. But after the due release of the application, we also want to provide multiple language support so that we carter to not only the English speaking community but also to a larger audience.

References

1. https://developer.android.com/reference/android/speech/tts/TextToSpeech.html
2. https://en.wikipedia.org/wiki/Braille
3. http://www.androidhive.info/2014/07/android-speech-to-text-tutorial/
4. https://developer.android.com/reference/android/speech/package-summary.html
5. https://pixlr.com/editor/
6. https://developer.android.com/about/start.html
7. https://play.google.com/store/apps/details?id=com.celltick.lockscreen&hl=en
8. https://play.google.com/store/apps/details?id=com.bytestorm.artflow&hl=en
9. https://play.google.com/store/apps/details?id=com.googlecode.eyesfree.brailleback
10. http://www.tutorialspoint.com/android/android_multitouch.htm
11. http://developer.android.com/training/gestures/multi.html
12. http://android-er.blogspot.in/2011/12/detect-multi-touch-for-10-pointers.html