**Chapter 1**

**Introduction**

Hearing impairment is a broad term that refers to hearing losses of varying degrees from hard-of-hearing to total deafness. The major challenge facing students with hearing impairments is communication. Hearing-impaired students vary widely in their communication skills. Among the conditions that affect the development of communication skills of persons with hearing impairments are personality, intelligence, nature and degree of deafness, degree and type of residual hearing, degree of benefit derived from amplification by hearing aid, family environment, and age of onset. Age of onset plays a crucial role in the development of language. Persons with prelingual hearing loss (present at birth or occurring before the acquisition of language and the development of speech patterns) are more functionally disabled than those who lose some degree of hearing after the development of language and speech.

Since much learning is acquired aurally, many students with hearing problems have both experiential and language deficiencies. Because they do not hear environmental noises and day-to-day conversations, hearing-impaired children miss a great deal of crucial information usually learned incidentally by non-hearing-impaired children. Although students can overcome some of these problems to varying degrees through great investments of time, energy, and effort by parents and educators, such deficiencies continue to be fairly common within the hearing-impaired population.

Most students with hearing impairments use a variety of communication methods. The most frequently used method is a combination of speech reading (lipreading) and residual hearing, which is often amplified by hearing aids. It is important to note, however, that speech reading is only a partial solution, since experts estimate that only about 30 to 40 percent of spoken English is distinguishable on the lips even by the best speech readers under the most favorable conditions.

Many students with hearing impairments can and do speak. Most deaf students have normal speech organs and have learned to use them through speech therapy. Some deaf students cannot monitor or automatically control the tone and volume of their speech, so their speech may be initially difficult to understand. Understanding improves as one becomes more familiar with the deaf student’s speech pattern.

Hearing-impaired students who communicate with speech and speech reading, as opposed to communicating manually with sign language, are referred to as “oral.” The incidence of oral and manual communication varies with regional philosophical differences on the issue. The population of hearing-impaired students at Allegheny is relatively small. Most of the hearing-impaired students at Allegheny are oral.

Most hearing-impaired students use note takers in class because it is difficult to speech read and take notes at the same time. Some hearing-impaired students may have language and vocabulary deficiencies.

Assumptions should not automatically be made about a hearing-impaired student’s ability to participate in certain types of classes. Hearing-impaired students may be able to learn much about music styles, techniques, and rhythms by observing a visual display of the music on an oscilloscope or similar apparatus or by feeling the vibrations of music. Some hearing-impaired students will have enough residual hearing so that amplification through hearing aids, earphones, public address systems, or personal FM transmitter/receiver units will allow participation. It is always best to discuss with the student the requirements of a class and to determine if there are ways that the materials can be modified so that the student can participate in what may become an exciting learning experience for all concerned.

A number of research studies have quite clearly demonstrated that children with a hearing loss have many disadvantages particularly if left unrecognized, untreated or under treated.  
  
**Hearing loss disadvantages and quality of life issues**

Research conducted by Hearing Review in late 2007 found there are a number the quality of life issues faced by deaf children. A mild loss creates a personal nuisance for the child and their family but a profound loss is a major disability which affects all aspect of life.   
  
  
Kochkin et al as well as other researchers have found that hearing impaired children had issues in a number of areas  
(1) Social interaction  
(2) Language and Communication  
(3) Education  
(4) Behavioral Problems  
(5) Mental Health  
(6) Safety  
  
  
**Social Interaction**

• Social skills were poor or under developed and may have a negative impact on social development.  
• Self-esteem and confidence were low and this affected their relationship with peers and family.  
• They find it hard to hear in group situations, therefore group participation is lower than that for their peers and this affects their leisure/play activities.  
• They could not easily use a phone, something their hearing peers could, and this made them different.  
• They do not always hear, so others have to yell at them causing embarrassment.  
• If the child doesn’t answer, peers think they are being ignored causing misunderstanding.  
• Many withdraw from social situations because of embarrassment and this contributes to isolation.  
• They often have poor speech meaning others cannot understand them so they become self-conscious.  
  
**Language & Communication**

Language and communication develop within the first two years of life.   
  
Kochin et al found Hearing Impaired children   
• Often have delays in, as well as poor, speech and language development hindering their ability to communicate effectively.  
• Have poor mental acuity due to poor speech and language development.  
• Do not have the skills to ask the questions to get help because they do not know what they do not hear and parents have to guess whether the child is hearing well enough to cope with their hearing loss because the child doesn’t have the experience to know and explain where they need help.  
• Need others to repeat themselves and frequently people show frustration at having to do this resulting in embarrassment.  
• Many need sign language but it is imperative their family and close friends learn it as well so the child is exposed to as much language as possible.  
  
**Education**

If a child is to attend a main stream school then oral and written communication are necessary. However, a deaf child may not have the skills to allow them to keep up with their peers.   
• Davis et al 1984 found that “hearing loss of any degree appeared to affect psychoeducational development adversely, leading to the conclusion that even minimal hearing loss places children at risk for language and learning problems.”  
• 71% of hearing impaired children had difficulty maintaining attention compared to 9% of normal hearing children.  
• Children with a mild to profound loss were behind in education achievement and their grades were lower than their hearing peers.  
• Reading development will be delayed. Average reading age of profoundly deaf 16 year olds was that of hearing 9 year olds.  
• 50% of the children researched could not read at all.  
• The child needs personal attention especially in school. 1 in 3 children with a hearing loss need help in the classroom to interpret the teacher’s instructions.   
• Children need to learn coping mechanisms such as sitting at the front of the class or positioning themselves on the best hearing side or learning lip reading. One third of hearing impaired children receive preferential seating (ie at the front of the class).  
• Lip reading is only possible if they already know the language.   
• May need a note taker or translator.  
• Many drop out early because they cannot cope resulting in a poor education and this means fewer job opportunities.  
  
**Behavioral problems**

Misunderstandings causes confusion and results in behavioral problems.   
• Behavioral problems are caused because of the frustration the child feels especially with peer group.  
• Teachers report behavioral problems for one third of their hearing-impaired students particularly impulsivity and inattention.  
• Hearing impaired children may listen to music or TV too loud disturbing the rest of the family.  
  
**Mental Health**

Children especially, do not like to be different and if they feel different or ‘lacking’ their emotional health can be affected.  
• Because the child must rely on others to be their ears they lack a sense of independence.  
• Frustration at not making themselves understood nor being able to understand.  
• Frustration and anger because they cannot perform as well as their peers.  
• Lack of self-esteem and confidence.  
  
**Safety**

• Both parents of as well as deaf children lack a sense of safety.  
• Not hearing in noisy situations can mean a child misses safety cues.  
Intervention issues  
• Parents are often told the child cannot be helped although this is not necessarily the case.  
• Parents in denial (will grow out of it!) or see it as nothing terrible and easily dealt with.  
• Children are not comfortable with a hearing aid and parents fear they will lose it.  
• There is a stigma in wearing a hearing aid. Hearing aids are for old people, makes the child feel different. They are ugly and don’t work. Other kids make fun of them.  
  
**Recommended intervention**

When the recommended intervention was implemented Kochkin found that:  
(1) 61% of children’s grades improved  
(2) 46% had improved social skills  
(3) 51% had improved classroom behavior  
(4) 49% had improved self-esteem  
(5) If children under 6 months of age wear hearing aids then some of the effects on communication development can be prevented or minimized.  
  
Most of the issues deaf children face; social interaction difficulties, behavioral issues, language and communication difficulties, poor education and so on, come about because these children have poor linguistic skills.

A reliable source of communication medium for individuals with hearing loss or complete deafness is sign language or an oral interpreter. However not everyone has the means or the capability to possess these resources. Thus when it comes to education these users face tremendous problems. With them as our prospective users our application aims at providing a platform based on android with various modules that will help reduce the complexity of learning. Student management system, Volunteer management system and Donation are the modules that will incorporate various sub modules developed mainly using Android studio and JavaScript.

The student module has the distinctive feature of text-to-speech conversion. Completely deaf or hard hearing users can communicate to other individuals by inputting their text message which is converted to audio. This simple feature saves the effort to invest in an oral interpreter or learning of sign language. Students using this application will be largely benefited by giving tests. It has customized tests in Maths, English and logic. One user per test is the user to test ratio. Users have access to in depth study materials in form of videos and PDFs uploaded by the volunteers. In addition to this users have educational games to broaden their logical skills. Apart from educational progress hearing impaired users who are prospective job seekers can acquire them via the classified section in the job module of the application. The resources and skill training for these jobs will be monitored by volunteers who can upload the essentials keeping the requirement of the hiring company in view.

The volunteer module works along with student module. The study material required for giving the tests are uploaded by the volunteer. The volunteer are also required to go to the Center where they teach students using this application. So, only when the volunteers reach the desired location can they logon to the app and this message of login activity is immediately sent to the administrator so that they can monitor volunteer movement. Volunteers also get the progress report of the tests given by the students. This progress report is unique for each individual student and thus providing volunteer with the academic record of the student. Another distinctive feature of the application is a attendance record of all the students are generated. Admin shall be provided with the access to the entire system, the admin shall be able to register new students and grant them access to all the uploaded educational material.

The application comprises of a module for external users. Wherein by downloading the application they can donate money, books and used android mobile phones. With an operating version of 4.0 ice cream sandwich which is compatible for running the application.

**Chapter 2**

**Literature Survey**

**2.1 Android Studio**

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA .On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

* A flexible Gradle-based build system
* A fast and feature-rich emulator
* A unified environment where you can develop for all Android devices
* Instant Run to push changes to your running app without building a new APK
* Code templates and GitHub integration to help you build common app features and import sample code
* Extensive testing tools and frameworks
* Lint tools to catch performance, usability, version compatibility, and other problems
* C++ and NDK support

Built-in support for [Google Cloud Platform](http://developers.google.com/cloud/devtools/android_studio_templates/), making it easy to integrate Google Cloud Messaging and App Engine

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

* Android app modules
* Library modules
* Google App Engine modules

By default, Android Studio displays your project files in the Android project view. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

* **manifests**: Contains the AndroidManifest.xml file.
* **java**: Contains the Java source code files, including JUnit test code.
* **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown.

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

1. The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.
3. The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
5. The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

### Tool Windows

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

* To expand or collapse a tool window, click the tool’s name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.
* To return to the current default tool window layout, click **Window > Restore Default Layout** or customize your default layout by clicking **Window > Store Current Layout as Default**.
* To show or hide the entire tool window bar, click the window icon  in the bottom left-hand corner of the Android Studio window.
* To locate a specific tool window, hover over the window icon and select the tool window from the menu.

You can also use keyboard shortcuts to open tool windows. Table 2.1 lists the shortcuts for the most common windows.

**Table 2.1** Keyboard shortcuts for some useful tool windows.

|  |  |  |
| --- | --- | --- |
| Tool Window | Windows and Linux | Mac |
| Project | **Alt+1** | **Command+1** |
| Version Control | **Alt+9** | **Command+9** |
| Run | **Shift+F10** | **Control+R** |
| Debug | **Shift+F9** | **Control+D** |
| Android Monitor | **Alt+6** | **Command+6** |
| Return to Editor | **Esc** | **Esc** |
| Hide All Tool Windows | **Control+Shift+F12** | **Command+Shift+F12** |

If you want to hide all toolbars, tool windows, and editor tabs, click **View > Enter Distraction Free Mode**. This enables Distraction Free Mode. To exit Distraction Free Mode, click **View > Exit Distraction Free Mode**.

You can use Speed Search to search and filter within most tool windows in Android Studio. To use Speed Search, select the tool window and then type your search query.

### Code Completion

Android Studio has three types of code completion, which you can access using keyboard shortcuts.

**Table 2.2.** Keyboard shortcuts for code completion.

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Description | Windows and Linux | Mac |
| Basic Completion | Displays basic suggestions for variables, types, methods, expressions, and so on. If you call basic completion twice in a row, you see more results, including private members and non-imported static members. | **Control+Space** | **Control+Space** |
| Smart Completion | Displays relevant options based on the context. Smart completion is aware of the expected type and data flows. If you call Smart Completion twice in a row, you see more results, including chains. | **Control+Shift+Space** | **Control+Shift+Space** |
| Statement Completion | Completes the current statement for you, adding missing parentheses, brackets, braces, formatting, etc. | **Control+Shift+Enter** | **Shift+Command+Enter** |

You can also perform quick fixes and show intention actions by pressing **Alt+Enter**.

### Navigation

Here are some tips to help you move around Android Studio.

* Switch between your recently accessed files using the Recent Files action. Press **Control+E** (**Command+E** on a Mac) to bring up the Recent Files action. By default, the last accessed file is selected. You can also access any tool window through the left column in this action.
* View the structure of the current file using the File Structure action. Bring up the File Structure action by pressing **Control+F12** (**Command+F12** on a Mac). Using this action, you can quickly navigate to any part of your current file.
* Search for and navigate to a specific class in your project using the Navigate to Class action. Bring up the action by pressing **Control+N**(**Command+O** on a Mac). Navigate to Class supports sophisticated expressions, including camel humps, paths, line navigate to, middle name matching, and many more. If you call it twice in a row, it shows you the results out of the project classes.
* Navigate to a file or folder using the Navigate to File action. Bring up the Navigate to File action by pressing **Control+Shift+N** (**Command+Shift+O** on a Mac). To search for folders rather than files, add a / at the end of your expression.
* Navigate to a method or field by name using the Navigate to Symbol action. Bring up the Navigate to Symbol action by pressing **Control+Shift+Alt+N**(**Command+Shift+Alt+O** on a Mac).
* Find all the pieces of code referencing the class, method, field, parameter, or statement at the current cursor position by pressing **Alt+F7**.

### Style and Formatting

As you edit, Android Studio automatically applies formatting and styles as specified in your code style settings. You can customize the code style settings by programming language, including specifying conventions for tabs and indents, spaces, wrapping and braces, and blank lines. To customize your code style settings, click **File > Settings > Editor > Code Style** (**Android Studio > Preferences > Editor > Code Style** on a Mac.)

Although the IDE automatically applies formatting as you work, you can also explicitly call the Reformat Code action by pressing **Control+Alt+L**(**Opt+Command+L** on a Mac), or auto-indent all lines by pressing **Control+Alt+I** (**Alt+Option+I** on a Mac).

### Version Control Basics

Android Studio supports a variety of version control systems (VCS’s), including Git, GitHub, CVS, Mercurial, Subversion, and Google Cloud Source Repositories.

After importing your app into Android Studio, use the Android Studio VCS menu options to enable VCS support for the desired version control system, create a repository, import the new files into version control, and perform other version control operations:

1. From the Android Studio **VCS** menu, click **Enable Version Control Integration**.
2. From the drop-down menu, select a version control system to associate with the project root, and then click **OK**.

The VCS menu now displays a number of version control options based on the system you selected.

**Note:** You can also use the **File > Settings > Version Control** menu option to set up and modify the version control settings.

## **Gradle Build System**

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android plugin for Gradle. This build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

* Customize, configure, and extend the build process.
* Create multiple APKs for your app, with different features using the same project and modules.
* Reuse code and resources across sourcesets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files. Android Studio build files are named build.gradle. They are plain text files that use Groovy syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

### Build Variants

The build system can help you create different versions of the same application from a single project. This is useful when you have both a free version and a paid version of your app, or if you want to distribute multiple APKs for different device configurations on Google Play.

### Multiple APK Support

Multiple APK support allows you to efficiently create multiple APKs based on screen density or ABI. For example, you can create separate APKs of an app for the hdpi and mdpi screen densities, while still considering them a single variant and allowing them to share test APK, javac, dx, and ProGuard settings.

**2.2 Firebase**

Firebase is a mobile and web application platform with tools and infrastructure designed to help developers build high-quality apps. Firebase is made up of complementary features that developers can mix-and-match to fit their needs. The team is based in San Francisco and Mountain View, California. The company was founded in 2011 by Andrew Lee and James Tamplin. Firebase's initial product was a real time database, which provides an API that allows developers to store and sync data across multiple clients. Over time, it has expanded its product line to become a full suite for app development. The company was acquired by Google in October 2014 and a significant number of new features were featured in May 2016 at Google I/O.

Firebase is a Backend-as-a-Service — BaaS — that started as a YC11 startup and grew up into a next-generation app-development platform on Google Cloud Platform. Firebase frees developers to focus crafting fantastic user experiences. You don’t need to manage servers. You don’t need to write APIs. Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs. Yeah, you’ll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can’t be everything to everybody. But it gets pretty close.

#### **It’s a Real time Database**

Real-time data is the way of the future. Nothing compares to it.

Most databases require you to make HTTP calls to get and sync your data. Most databases give you data only when you ask for it.

When you connect your app to Firebase, you’re not connecting through normal HTTP. You’re connecting through a WebSocket. WebSockets are much, much faster than HTTP. You don’t have to make individual WebSocket calls, because one socket connection is plenty. All of your data syncs automagically through that single WebSocket as fast as your client’s network can carry it.

Firebase sends you new data as soon as it’s updated. When your client saves a change to the data, all connected clients receive the updated data almost instantly.

#### **It’s File Storage**

Firebase Storage provides a simple way to save binary files — most often images, but it could be anything — to Google Cloud Storage **directly from the client!**

Firebase Storage has it’s own system of security rules to protect your GCloud bucket from the masses, while granting detailed write privileges to your authenticated clients.

#### **It’s Authentication**

Firebase auth has a built in email/password authentication system. It also supports OAuth2 for Google, Facebook, Twitter and GitHub. We’ll focus on email/password authentication for the most part. Firebase’s OAuth2 system is well-documented and mostly copy/paste.

If you’ve ever written an authentication system, let’s commiserate for a moment. Custom authentication is terrible. I will never write an auth system again for as long as I live. I fell in love with Firebase Auth at first sight, and the flame has never wavered. Sometimes I get frustrated. Sometimes we fight. But I never forget the cold, dark abyss of a custom auth system. I count my blessings.

Oh, and Firebase Auth integrates directly into Firebase Database, so you can use it to control access to your data. I’m writing this as if it’s an afterthought. It’s not. It’s the second reason that you will love Firebase Auth.

#### **It’s Hosting**

Firebase includes an easy-to-use hosting service for all of your static files. It serves them from a global CDN with HTTP/2.

And to make your development particularly painless, Firebase hosting utilizes Superstatic, which you can run locally for all of your testing. I run Superstatic as BrowserSync middleware. The following implementation uses Gulp, but Gulp is purely optional.

Quick and dirty dev server

The BrowserSync + Superstatic development environment is slick. BrowserSync handles reloading your development app across all connected devices and Superstatic replicates Firebase hosting locally in such a way that you can deploy straight to Firebase for production use.

#### **It’s a Fully-Featured App Platform**

The Firebase team has integrated a bunch of new and existing Google products with Firebase. I don’t plan to cover these features in detail quite yet…

A bunch of these features apply to iOS and Android but not to web.

* Remote Config
* Test Lab
* Crash
* Notifications
* Dynamic Links
* AdMob

### Firebase Pros & Cons

It’s not all roses.

I mean, it’s mostly roses, but watch the thorns.

#### **Pros**

* Email & password, Google, Facebook, and Github authentication
* Realtime data
* Ready-made api
* Built in security at the data node level
* File storage backed by Google Cloud Storage
* Static file hosting
* Treat data as streams to build highly scalable applications
* Don’t worry about your infrastructure!

#### **Cons**

* Limited query abilities due to Firebase’s data stream model
* Traditional relational data models are not applicable to NoSQL; therefore, your SQL chops will not transfer
* No on-premise installation

**Chapter 3**

**Scope**

This application is mainly for the end users who are hearing impaired. This application will focus on their education and progress; also provide them with job opportunities. The volunteers/teaching staff will be provided with a login of their own to keep a track of the student’s performance. Also, uploading educational materials (video lectures and pdf’s) on the application.

Admin shall be provided with an overall access to the entire system, the admin shall be able to register new students and grant them access to all the educational material. External Viewers of our application will be able to donate to the organization in terms of Money, Study Materials, Mobile Phones and other electronic devices.

The students will be provided with an interactive interface to make their learning process more interesting with interactive learning modules and games.

The volunteers/teaching staff will be provided with a login of their own to keep a track of the student’s performance. Also, uploading educational materials (video lectures and pdf’s) on the application. Admin shall be provided with an overall access to the entire system, the admin shall be able to register new students and grant them access to all the educational material.

External Viewers of our application will be able to donate to the organization in terms of Money, Study Materials, Mobile Phones/ other electronic devices.

Our system is an android application implementing the Client-Server Model. Unmute provides a user friendly environment for our users to access various services provided by the organization.

**3.1 Project Deliverables**

**3.1.1. Student Management System:**

The students who are registered by the administrator will have access to a large pool of study material, tests and job opportunities. Students will also be able to view their progress. To make learning more interactive, we shall add various educational games that’ll promote writing skills, mathematical skills etc.

**3.1.2. Volunteer Management System:**

It will allow the volunteers to share their study materials with students as well as other volunteers. The volunteers will also be able to mark the attendance of the students. Additionally volunteers attendance will be marked by administrator when he reaches his/her allocated center.

**3.1.3. Donation:**

External users can donate money by filling a form and thus they will be registered with the NGO who will facilitate the donations in cash or cheque. Donors can also donate mobiles or books using this module.

**3.1.4. Job Opportunities:**

Students looking for jobs will be informed about vacancies in companies willing to employ hearing impaired people. Progress Analysis: The student shall be able to view his/her progress in the tests provided by the organization.

**Chapter 4**

**Proposed System**

**4.1 Student Management System**

The students who are registered by the administrator will have access to a large pool of study material, tests and job opportunities. Students will also be able to view their progress. To make learning more interactive, we shall add various educational games that’ll promote writing skills, mathematical skills etc.

**4.2 Volunteer Management System**

It will be a location based service which will allow administrator to verify the location of the volunteers when they login to the system after reaching their specified destination. It will also allow the volunteers to share their study materials with students as well as other volunteers. The volunteers will also be able to mark the attendance of the students.

**4.3 Donation**

External users can donate money by Bank Transfers or Credit/Debit card payments through a payment gate-way. Donors can also donate mobiles or books using this module.

**4.4 Job Opportunities**

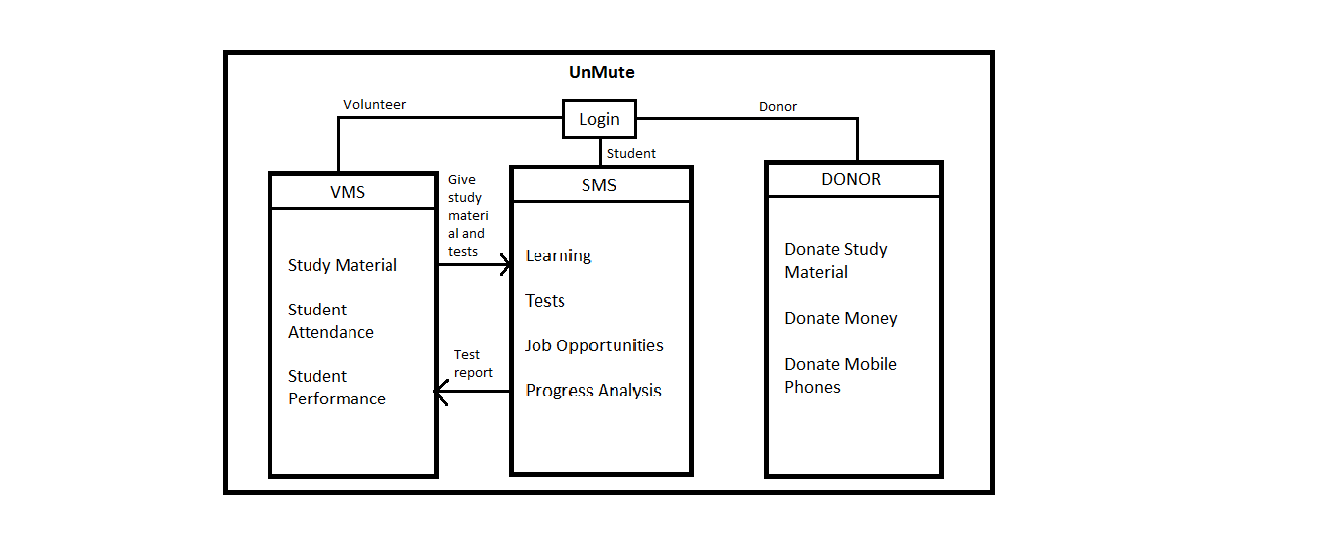
Job Opportunities: Students looking for jobs will be informed about vacancies in companies willing to employ hearing impaired people. Progress Analysis: The student shall be able to view his/her progress in the tests provided by the organization.

**Chapter 5**

**System Design**

**5.1 BLOCK DIAGRAM**

The Block Diagram of our application consists of three major modules, namely, Student, Volunteer and Donor management system. The student management system consists of four sub-modules: Learning, Test module, Job opportunities and progress analysis. The volunteer module consists of four sub-modules: Location, study material, attendance and student performance. Finally the donor module consists of donation of money, phones and study material.



**Figure 5.1: Block Diagram of the System**

System is divided into following parts:

**5.1.1 Student Management System**

Learning: Registered students once logged in can access the uploaded material

Test Module: Based on the topic that the students have learned, regular tests shall be conducted and their performance can be monitored.

**5.1.2 Volunteer Management System**

Location Login: Making use of the Location Services of the Volunteer’s phone, the location of the volunteer shall be updated whenever he/she logs into the system.

Sharing of Study Material: Volunteers will be able to upload their study materials in the form of video lectures as well as pdf’s to a common server which shall be made accessible to students as well as other volunteers.

Attendance: Whenever a volunteer logs into the system, he/she can mark the attendance of students which will be recorded for future reference.

Student Performance: The volunteers teaching a particular group of students shall be able to monitor the performance of those students.

**5.1.3 Donor Management System**

Donate Money: Those who wish to donate money to the organization can do so by accessing our application and selecting the donate money option.

Donate Phones: Those who wish to donate old mobile phones or any other electronic items can do so by selecting the donate phones option.

Donate Books: Those who wish to donate books can do so by accessing our application and selecting the donate books option.

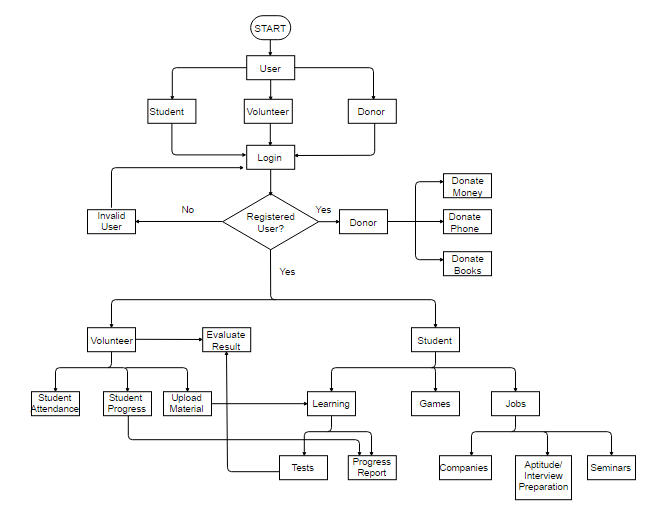
**5.2 FLOWCHART**

These components work in a particular sequence. This sequence can be illustrated by the following flow chart of the system. The following chart shows the complete flow of the system and gives a clear picture of the working of the entire system.

The user on the home screen will be requested to select the type of user he/she is. Once the type is selected the user will be asked to enter the login credentials.

On entering valid credentials the user will be directed to the appropriate page as per the type selected by the user.

If login credentials are invalid, a message will be dis-played informing the user about the incorrect credential values and will be redirected back to the login page.

**Figure 5.2: Flowchart of the System**

If the User is a Student then:

1. The user will be allowed to access the study material posted by Volunteers.

2. The user can give tests and play interactive educational games.

3. The user will be allowed to view his/her progress report, which is given by the volunteer.

If the user is a Volunteer then:

1. The volunteer’s location on logging into the sys-tem will be recorded in the database.

2. The Volunteer can upload study material.

3. The Volunteer can Mark Student attendance.

4. The Volunteer can assess the tests given by students and create progress report.

If the User is a Donor then:

1. The user will be asked about what the user wants to donate.

2. If user selects Money, then user will be directed to a payment gateway. If user selects Books or Phone, then user will be notified about the successful entry of his/her donation request into the database.

**5.3 USE CASE DIAGRAM**

The use case diagram below shows the functionality of the tool from the perspective of the users using it. Types of users using this application:

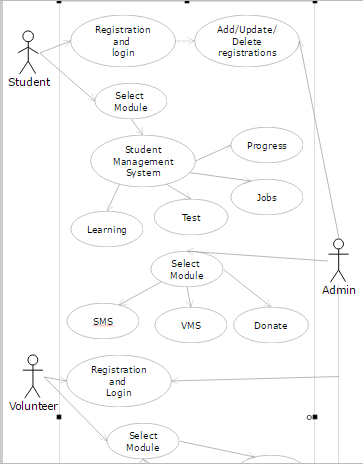
⦁ Administrator: The Administrator will register new students into the system, maintain records of students and volunteers, keep a track of volunteer’s activities and make

changes to the system.

⦁ Volunteer: The Volunteer shall keep a track of the attendance, performance and

progress of the students.

⦁ Students: Students must be familiar with Android and should have at least basic

knowledge about using Android apps. They must have basic understanding of English language.

**Figure 5.3: Use Case Diagram**

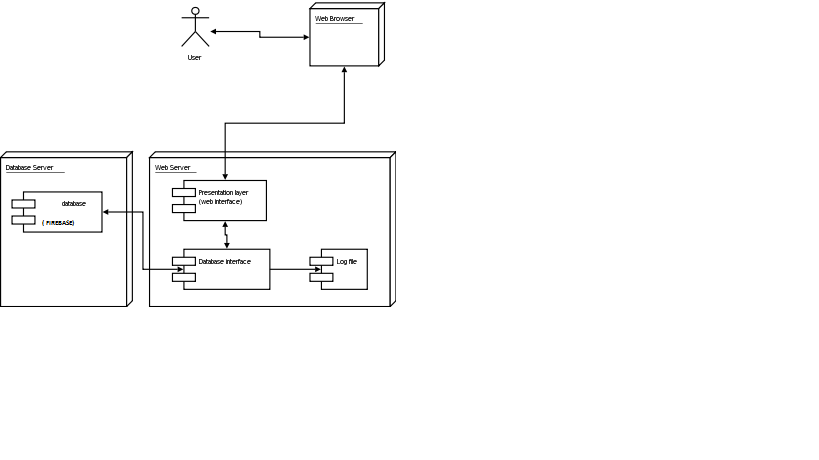
**5.4 COMPONENT DIAGRAM**

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executables, libraries etc.

So the purpose of this diagram is different, Component diagrams are used during the implementation phase of an application. But it is prepared well in advance to visualize the implementation details.

Initially the system is designed using different UML diagrams and then when the artifacts are ready component diagrams are used to get an idea of the implementation.

This diagram is very important because without it the application cannot be implemented efficiently. A well prepared component diagram is also important for other aspects like application performance, maintenance etc.



**Figure 5.4: Component Diagram**

**5.5 CLASS DIAGRAM**

The class diagram consists of classes for user, admin, student management system, volunteer management system, donor management system. The user class has three types of users they are the student, volunteer and donor.

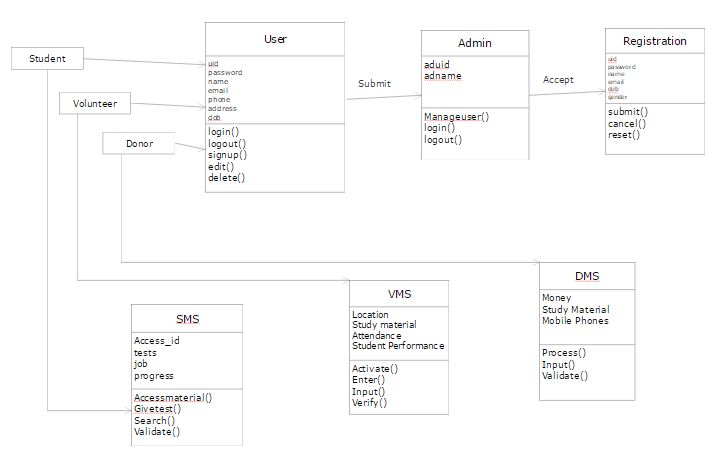
User: The user has his primary key as user id, and then various other attributes like email, phone and DOB. His actions are login, signup, edit and delete.

Admin: It is responsible for managing the user processes and contains actions like login, logout and accepting and editing registration of various users.

Student management system: This class is responsible for accessing the valid access id with the tests, job and progress report for the use by student user.

Volunteer management system: It consists of location, study material, attendance and student performance. The actions the volunteer can perform are enter student details as input and verify them.

Donor management system: This class is responsible for performing processing of donation.

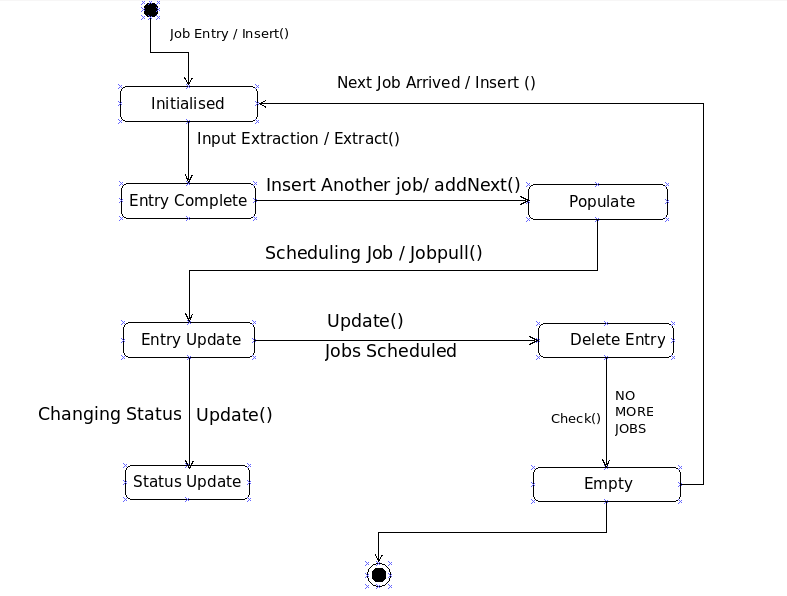


**Figure 5.5: Class Diagram**

**5.6 STATE CHART DIAGRAM**

Statechart diagram defines the states of a component and these state changes are dynamic in nature. So its specific purpose is to define state changes triggered by events. Events are internal or external factors influencing the system.

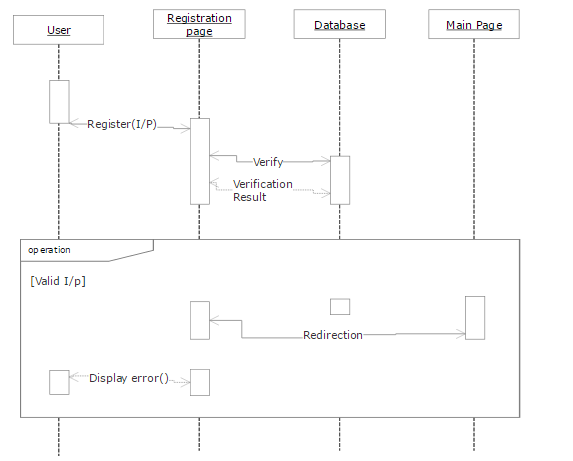
Statechart diagrams are used to model states and also events operating on the system. When implementing a system it is very important to clarify different states of an object during its life time and statechart diagrams are used for this purpose. When these states and events are identified they are used to model it and these models are used during implementation of the system.

****

**Figure 5.6: State Chart Diagram**

**5.7 SEQUENCE DIAGRAM**

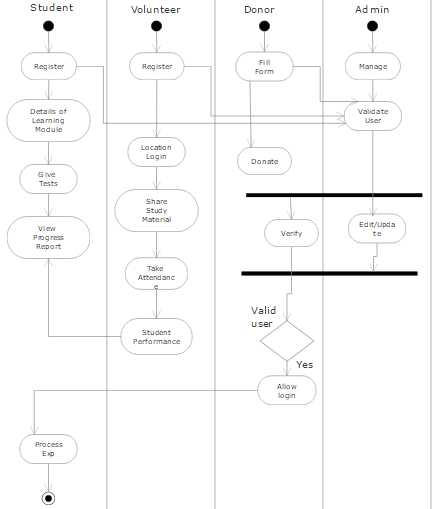
The sequence diagram consists of sequence of activities which will allow all the types of users to access the various modules. The first step is to register the user which will led him to the registration page and subsequently the admin validates the old user and the data of the new user gets stored in the database. This validated data will be shown to the user and the user can perform the various related activities of accessing the various modules.



**Figure 5.7: Sequence Diagram**

**5.8 ACTIVITY DIAGRAM**

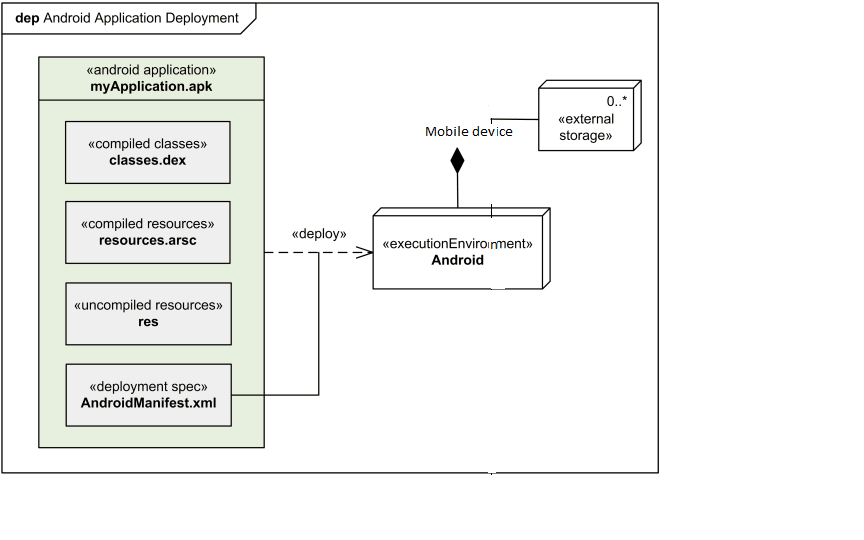
The activity diagram comprises of four users student, volunteer, donor and admin. All the users register and access the details. For e.g. the student can access the learning module, then give tests and view progress report. Similarly the volunteer can give his location update, share study material and then take attendance. The donor has to fill a form and then perform donation. The admin has to manage all the users validate them and if found any discrepancy send an error report to the user, he can also edit and verify all the activities and then finally the process completes.



**Figure 5.8: Activity Diagram**

**5.9 DEPLOYMENT DIAGRAM**

Our Deployment diagram is used for describing the hardware components where software components are deployed. The Android application (myApplication.apk) consists of various compiled classes, compiled and uncompiled resources and deployment specifications which are encompassed into the execution environment. The mobile device will have this execution environment installed and will have access to external storage through internet to perform the intended functionality efficiently.



**Figure 5.9: Deployment Diagram**

**Chapter 6**

**Implementation**

**6.1 Implementation Details**

This project is an implementation of Hadoop framework with Apache Pig for processing huge amount of data so as to create a tool that can be used to create ad hoc reports.

**6.1.1 Hardware Requirements**

The hardware components used in this project are as follows:

1. RAM 2GB minimum, 8 GB RAM recommended

2. Available disk space of 2GB minimum,

3. Recommended 4GB (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)

4.1280 x 800 minimum screen resolution

5. For accelerated emulator: 64-bit operating system and Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality.

6. An Android Phone with Android 4.0 and above.

7. GPS Device

8. High Speed Internet Connectivity

**6.1.2 Software Requirements**

The software components used in this project are as follows:

1. Microsoft® Windows® 7/8/10 (32- or 64-bit)

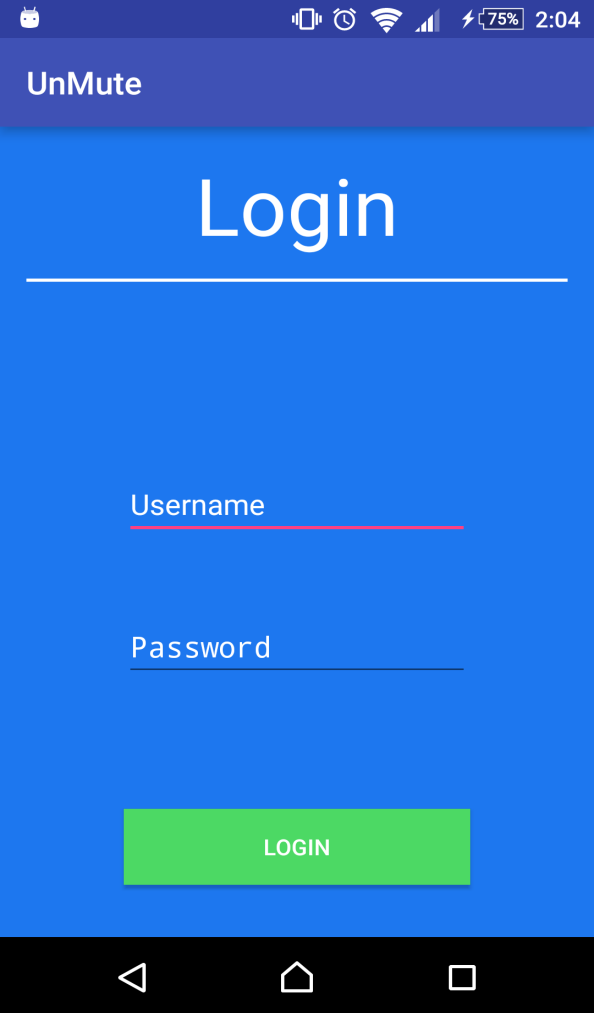
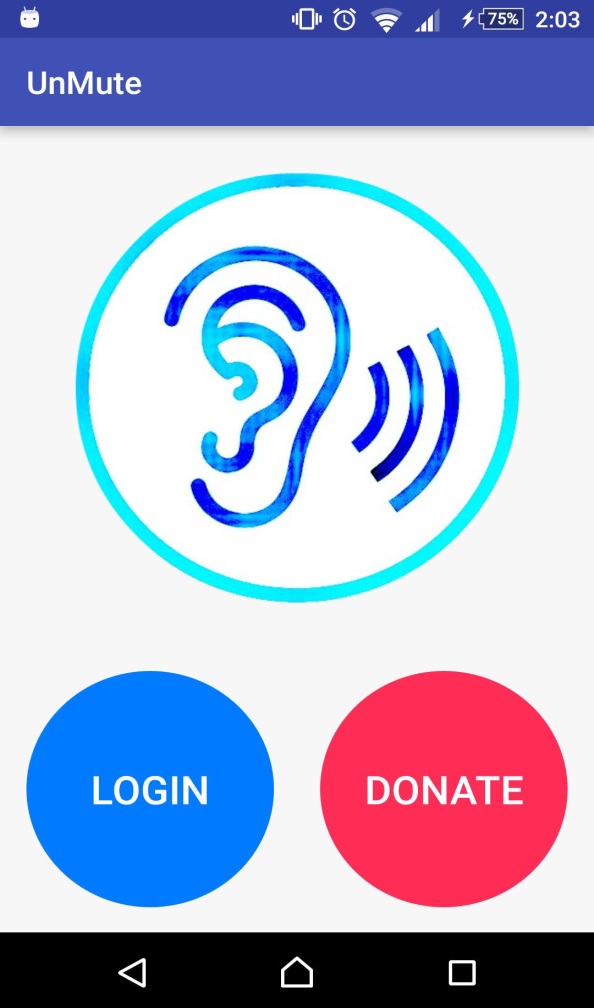
2. Java Development Kit (JDK) 8

3. Android 4.0 or above

**6.2 Screenshots of the application**

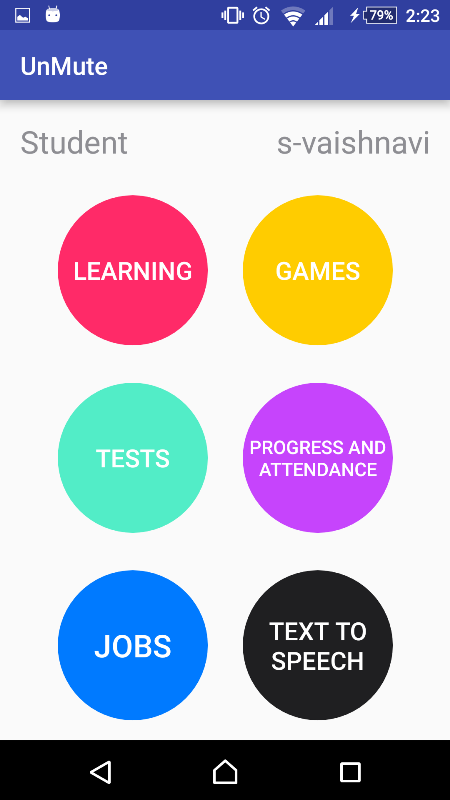
**6.2.1 Login Screen**

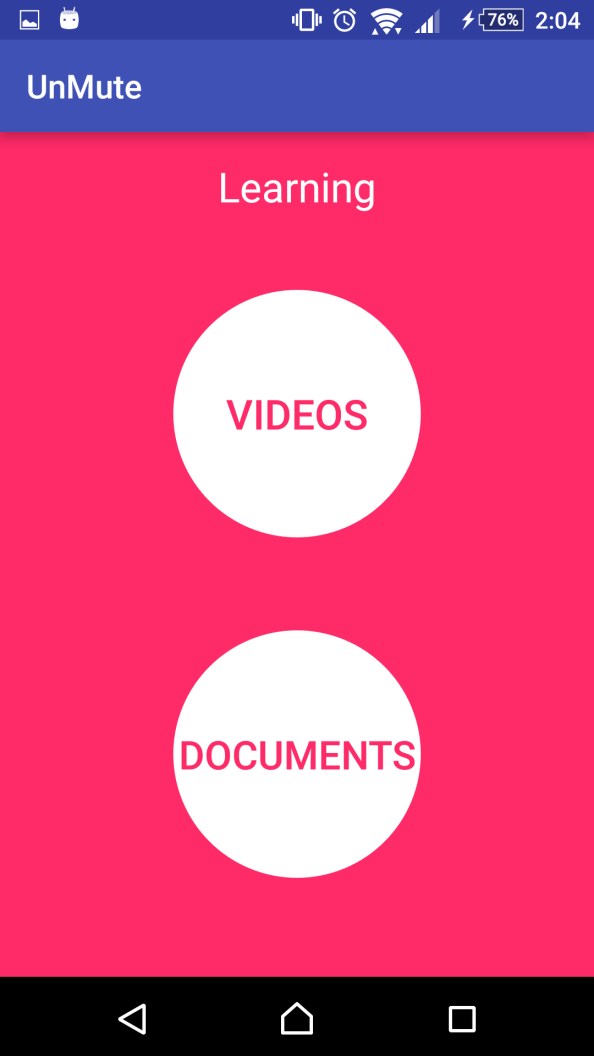
The start screen of the tool has a login so that authenticated users can log in and use the app.



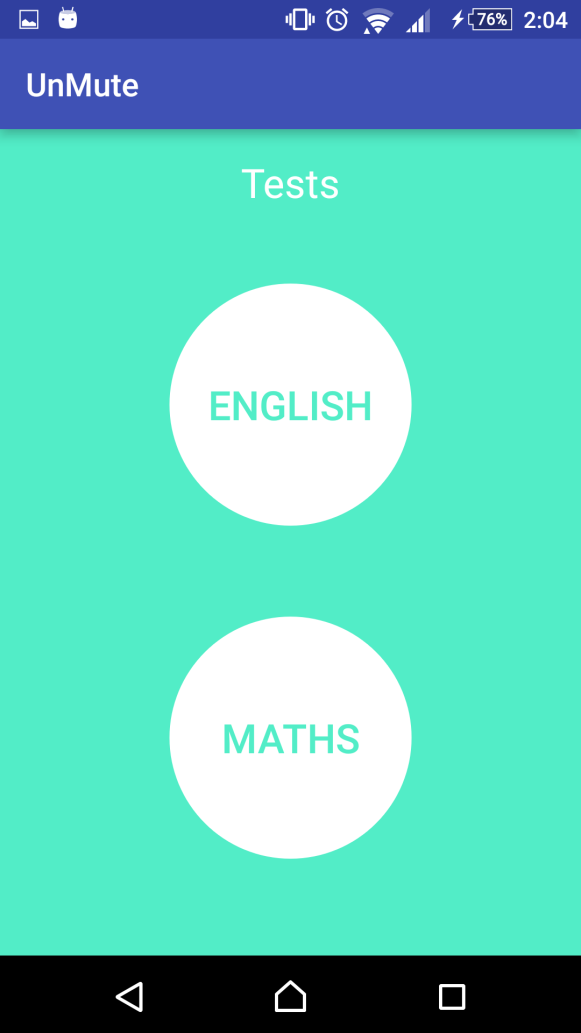
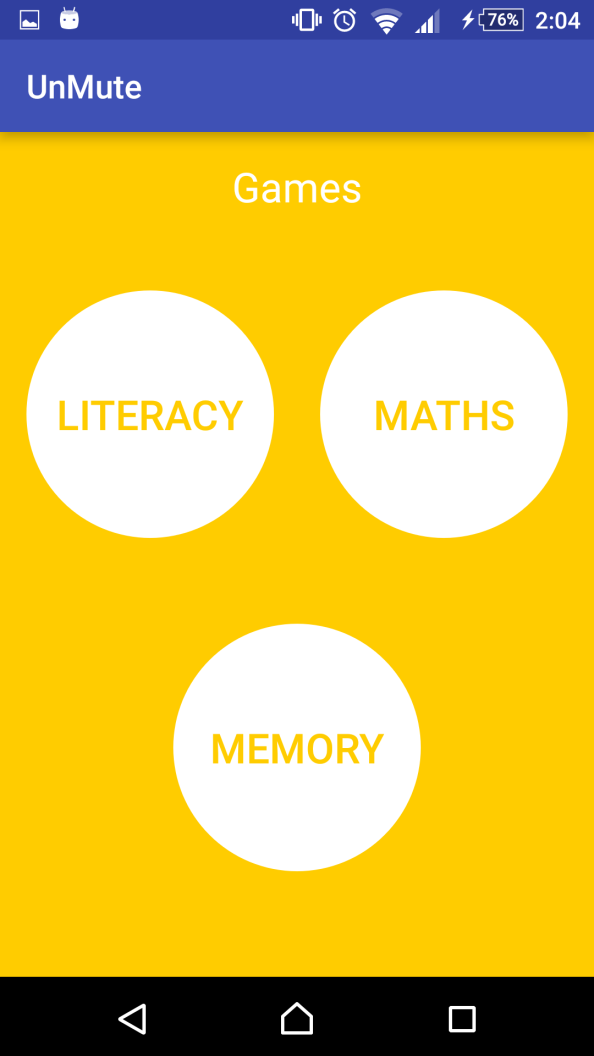
**Figure 6.2.1: Login Screen**

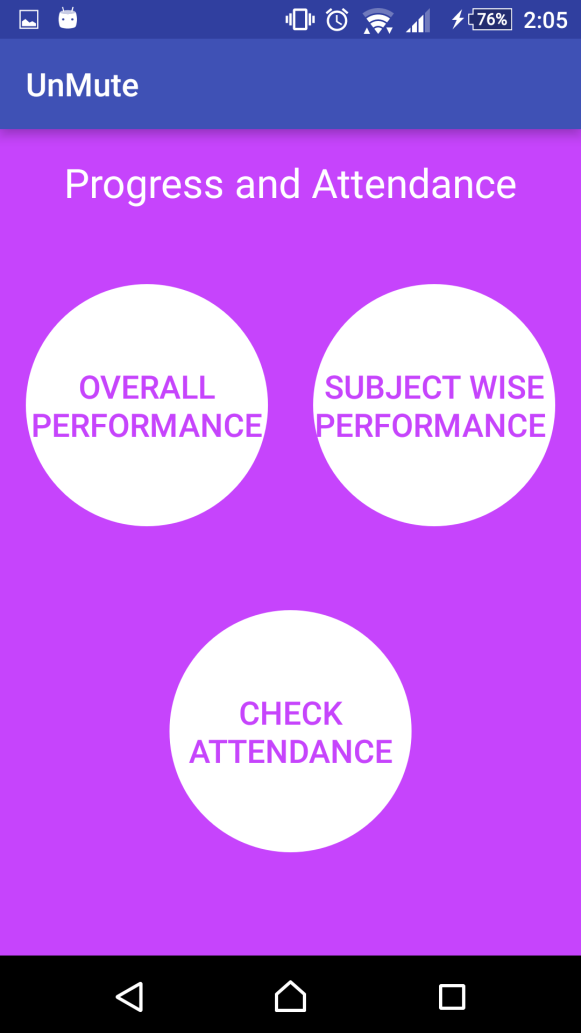
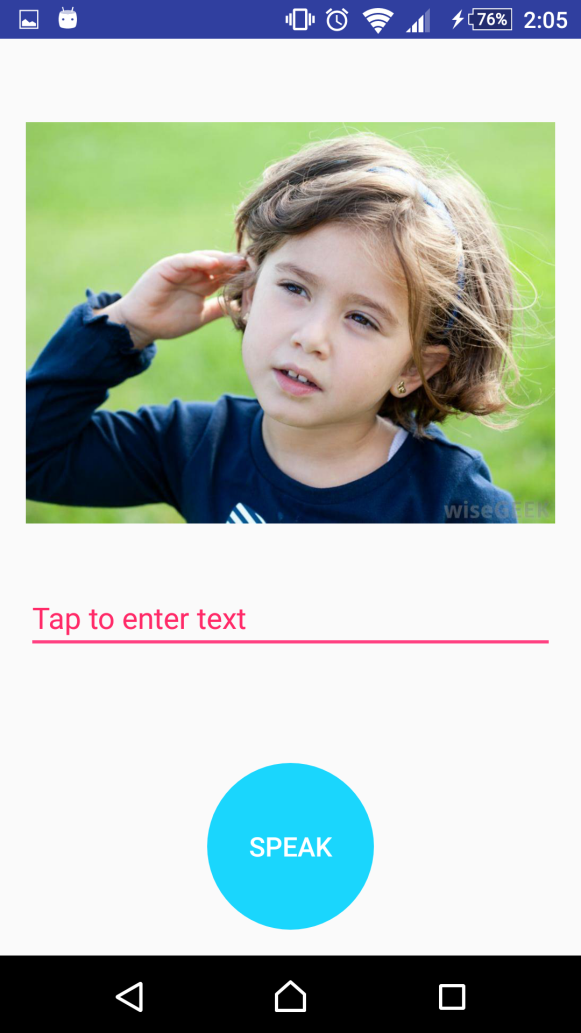
**6.2.2 Student Module**

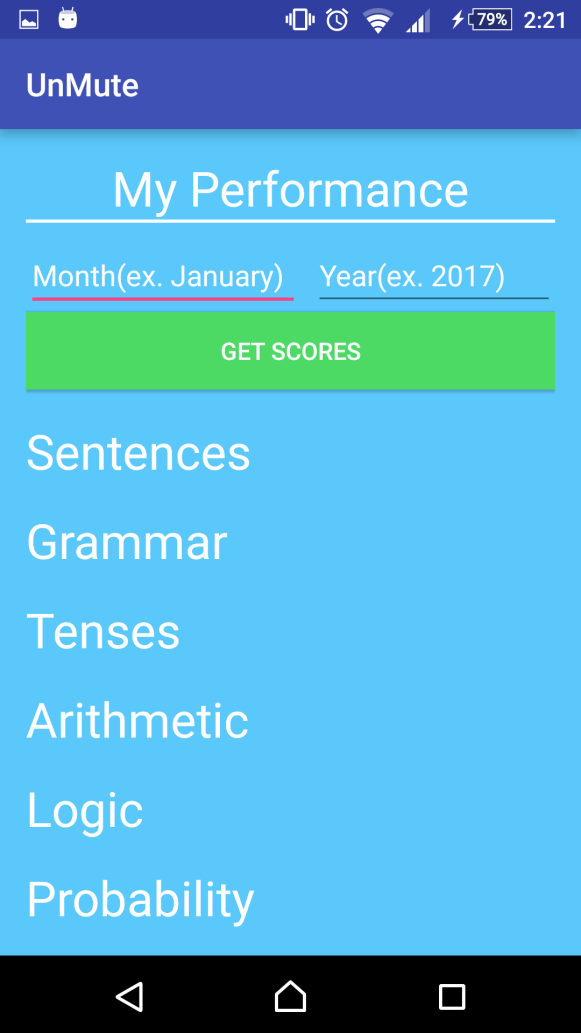




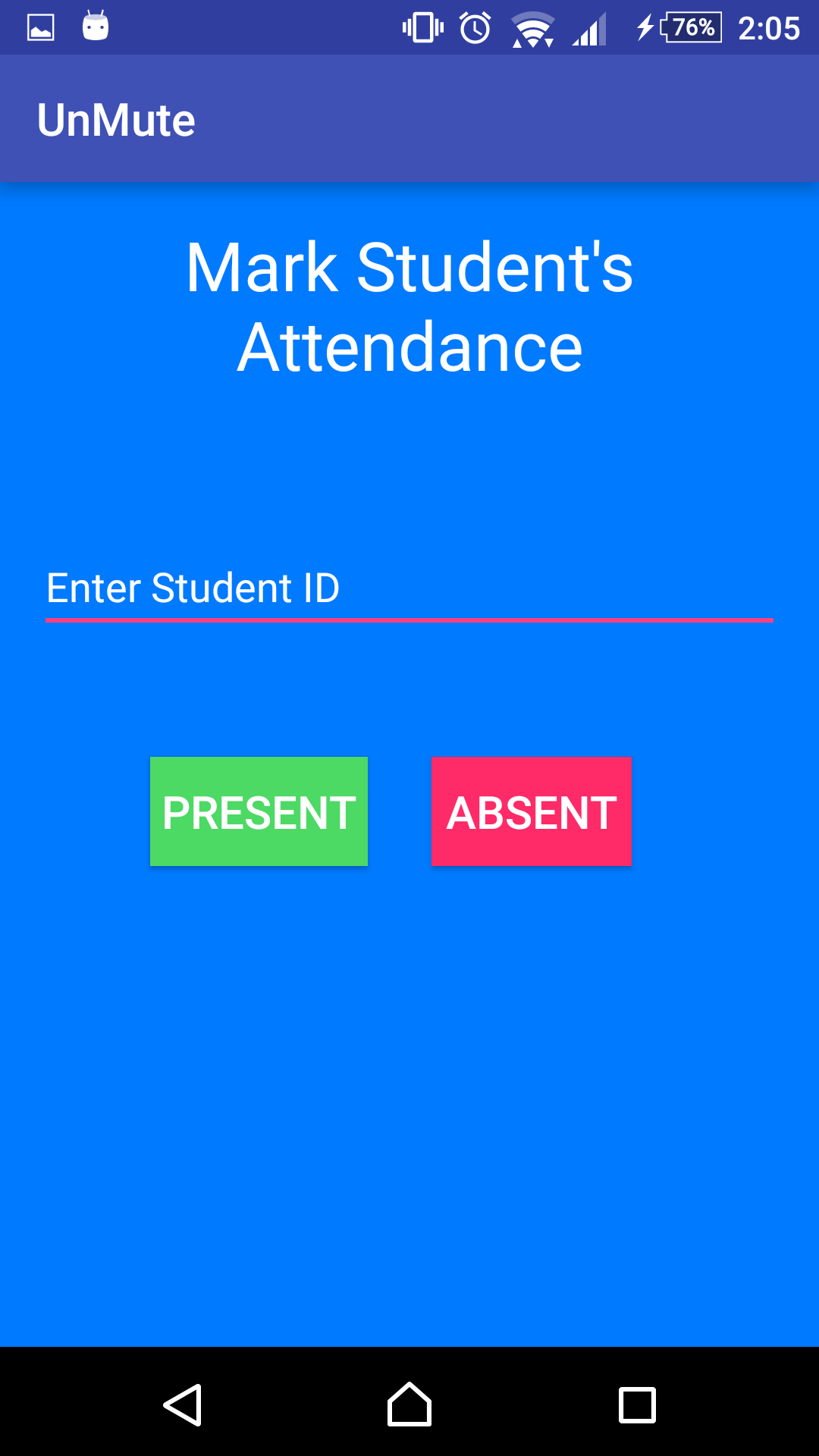
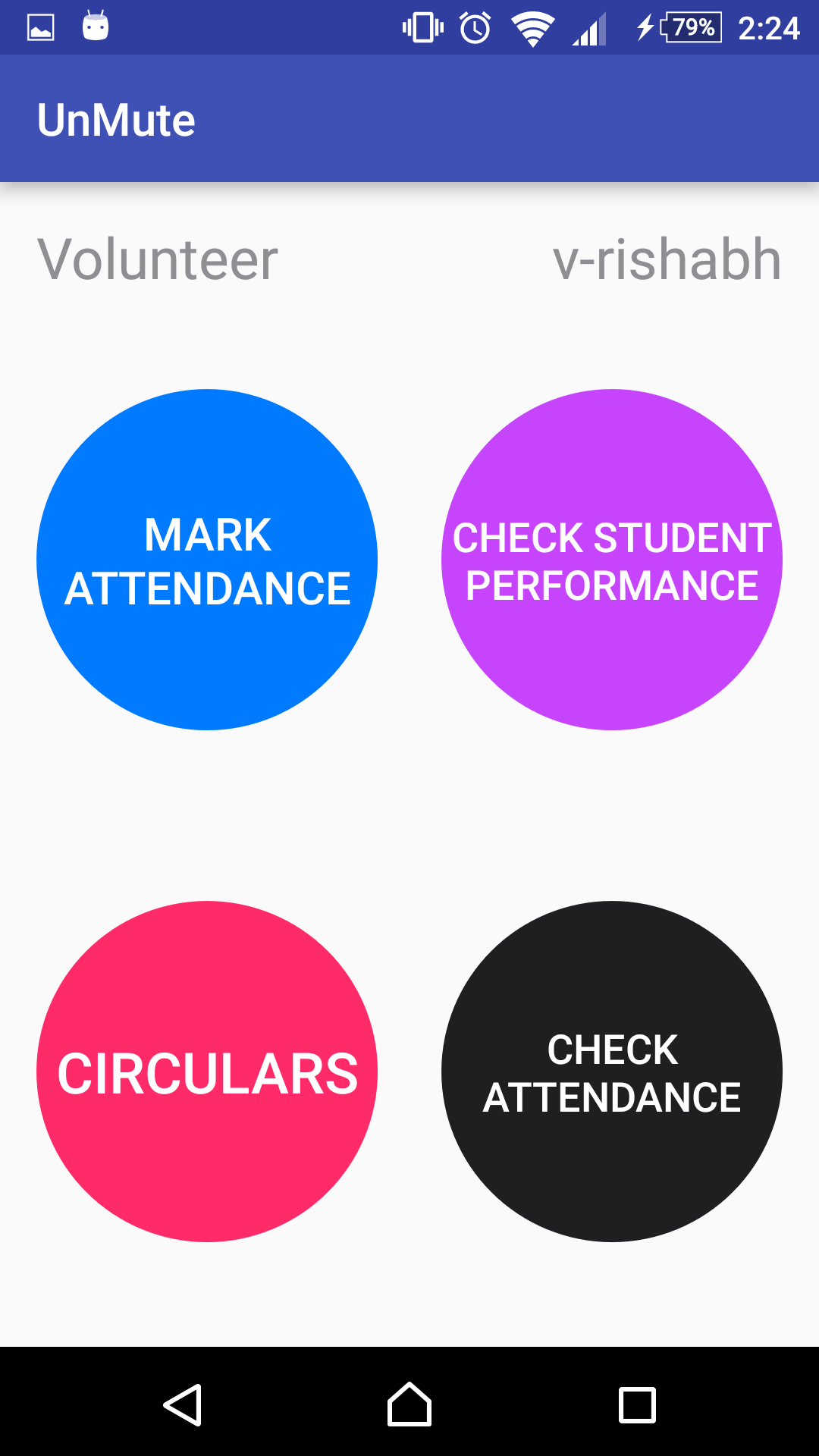
**Figure 6.2.2 Student Module**



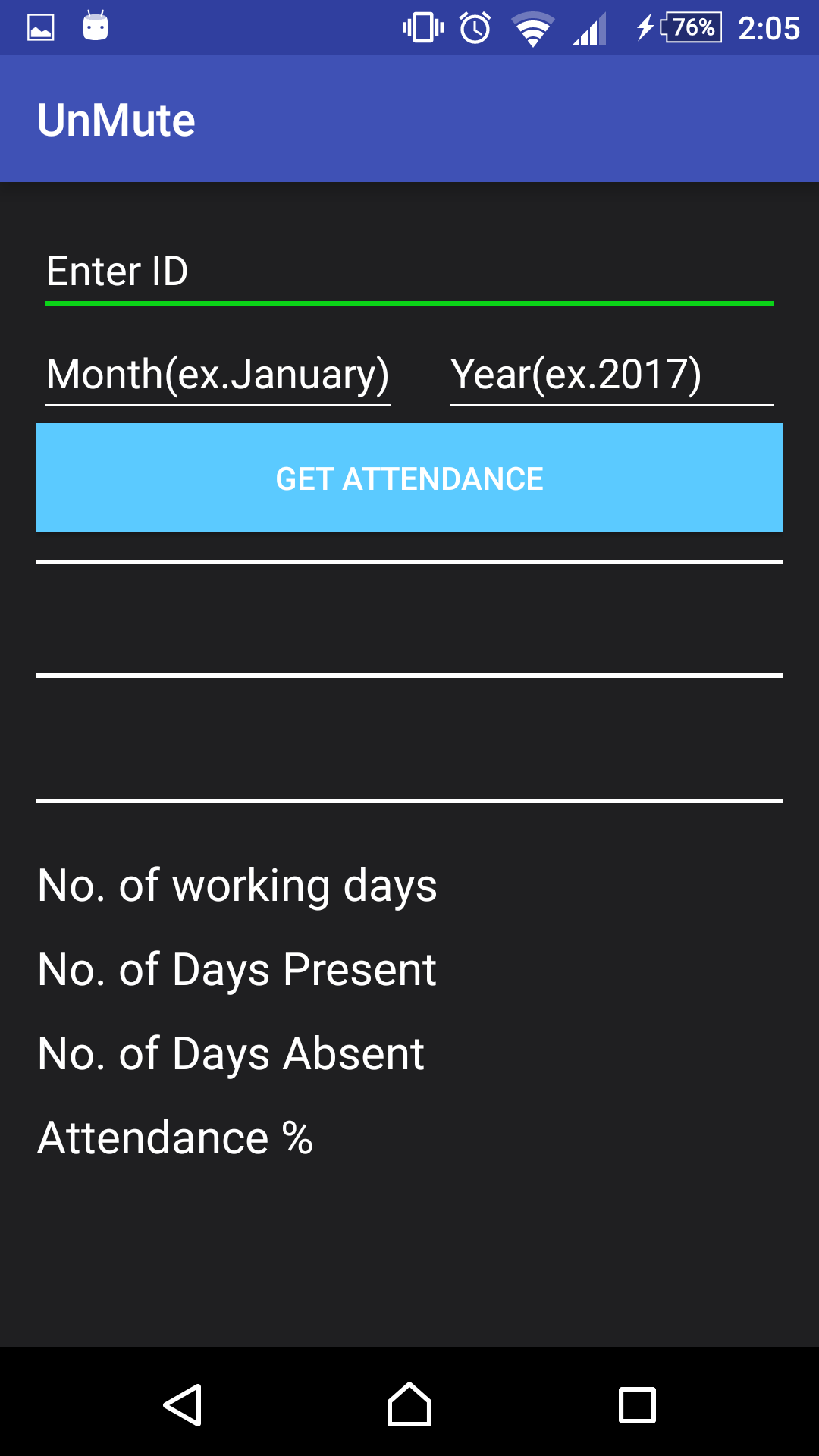
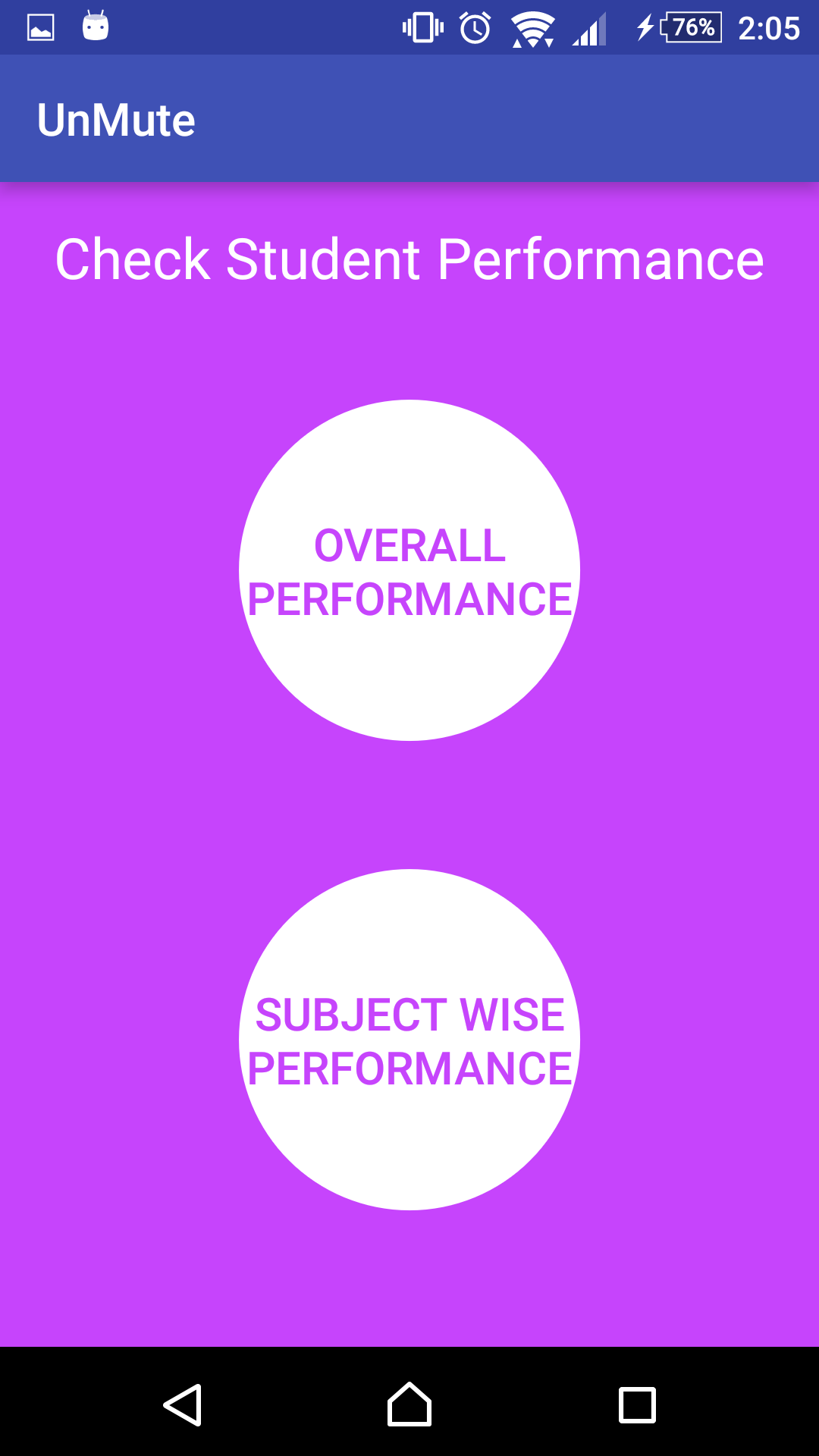
 

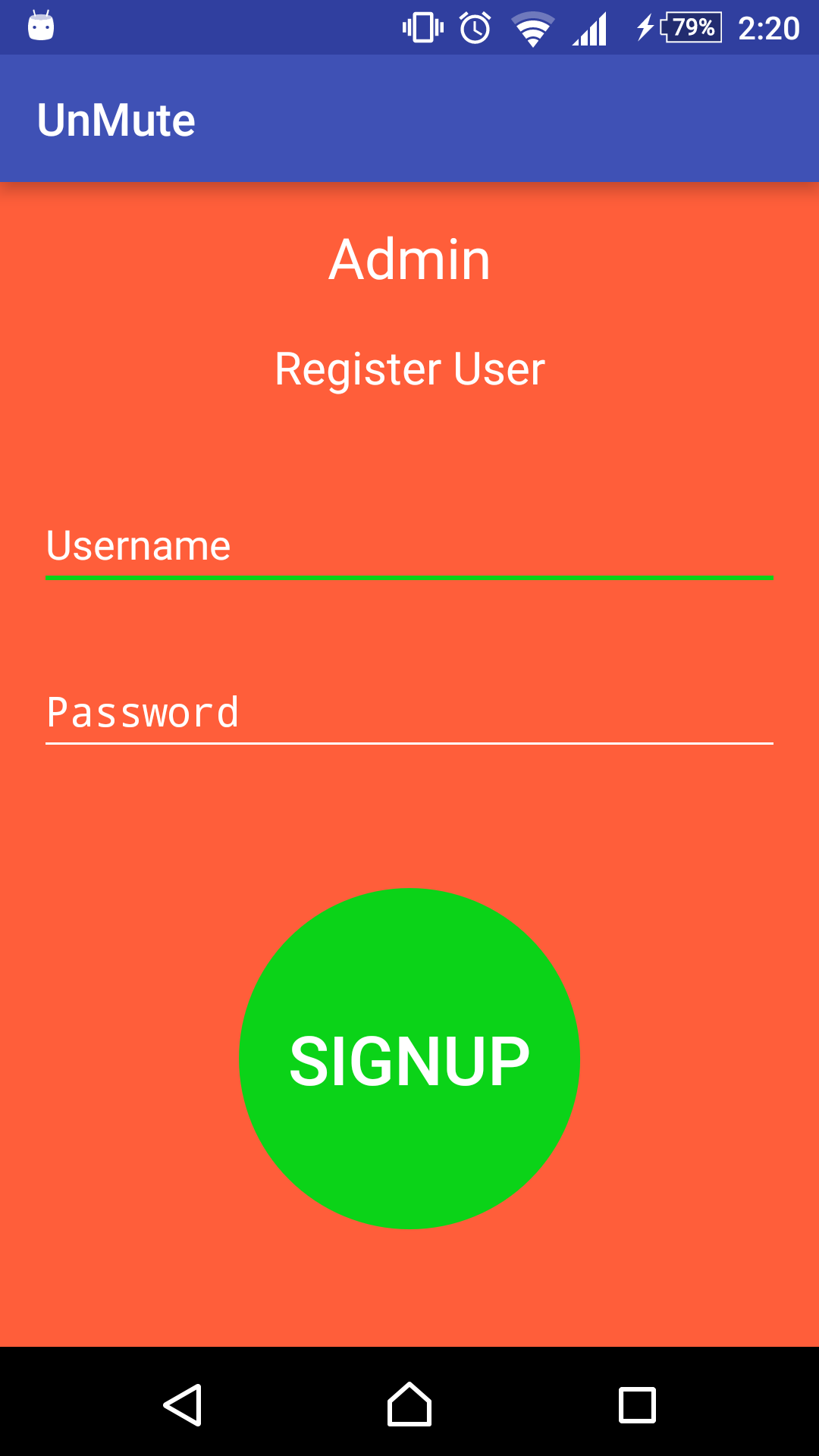
 

**6.2.3 Volunteer Module**



**Figure 6.2.3 Volunteer Module**

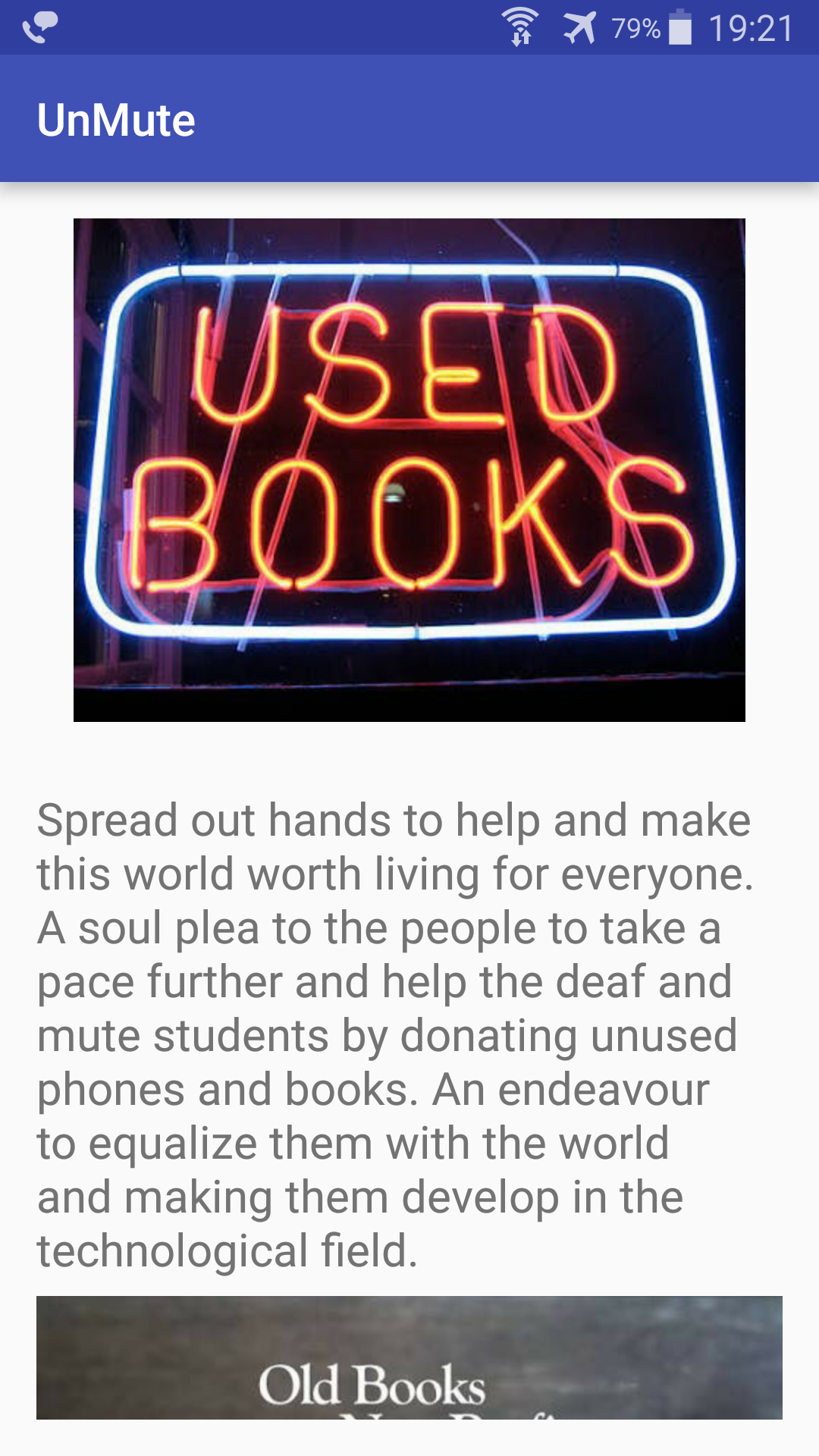


**6.2.4 Admin Module**



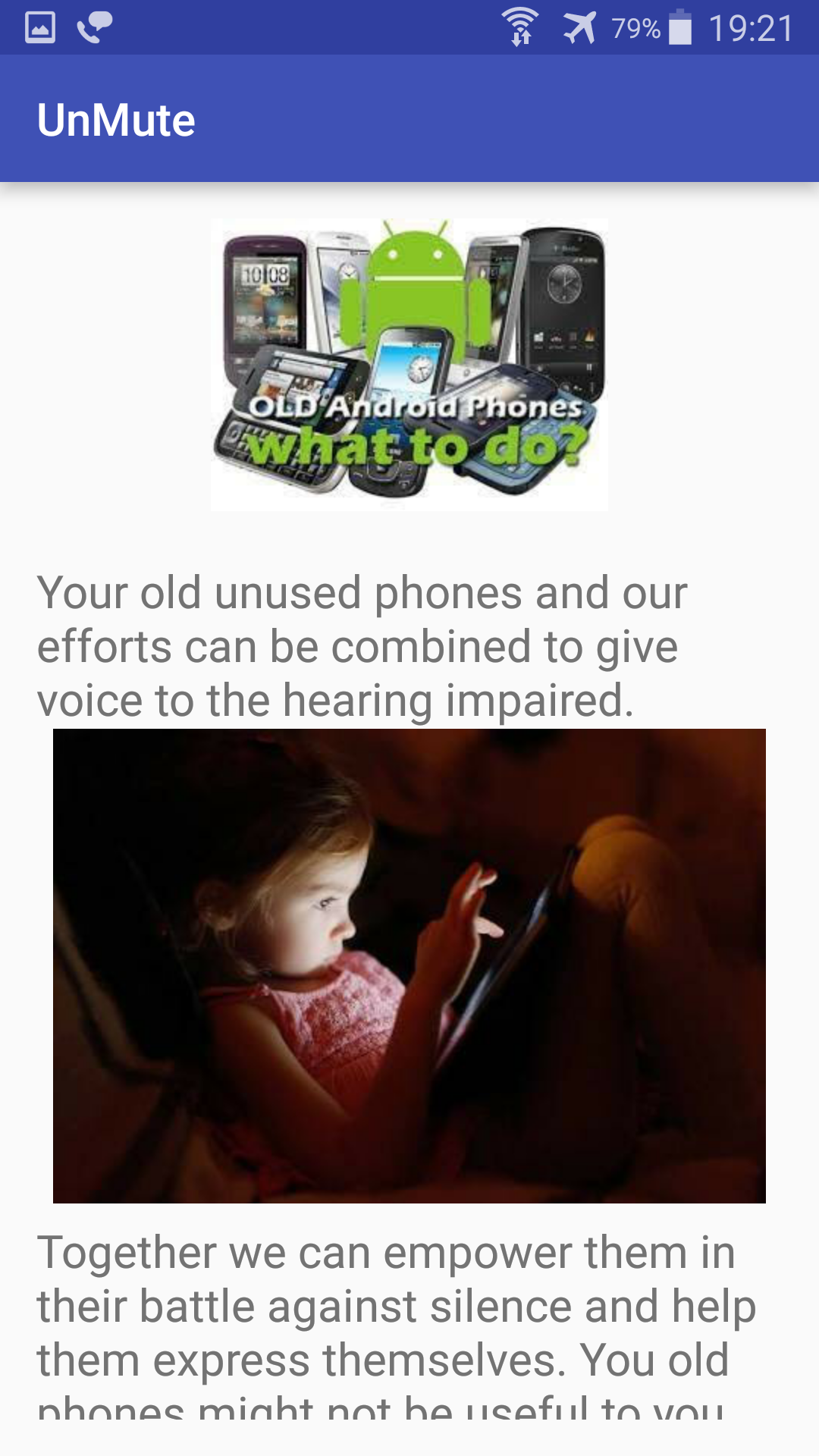
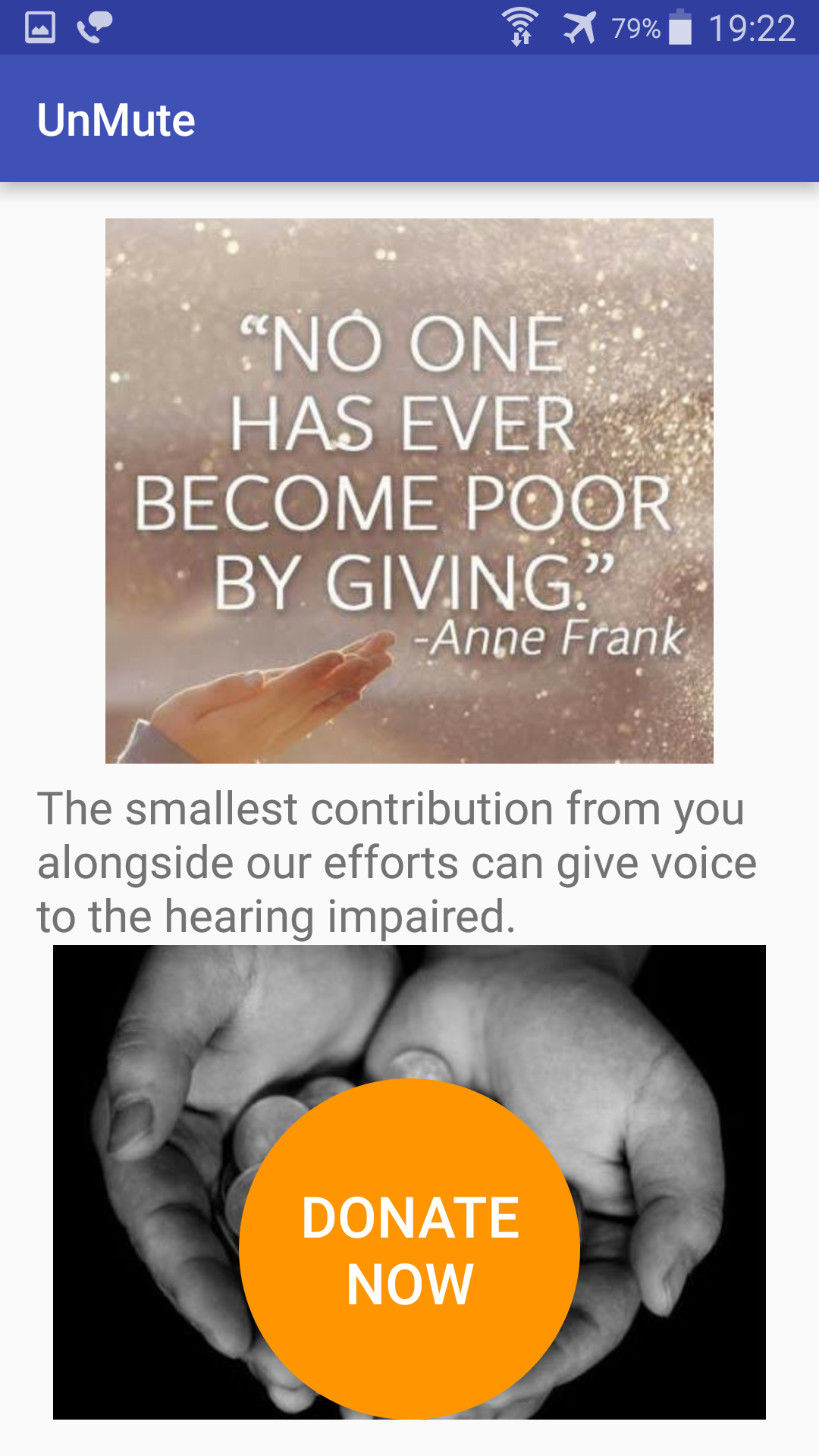
**Figure 6.2.4 Admin Module**

**6.2.5 Donor Module**

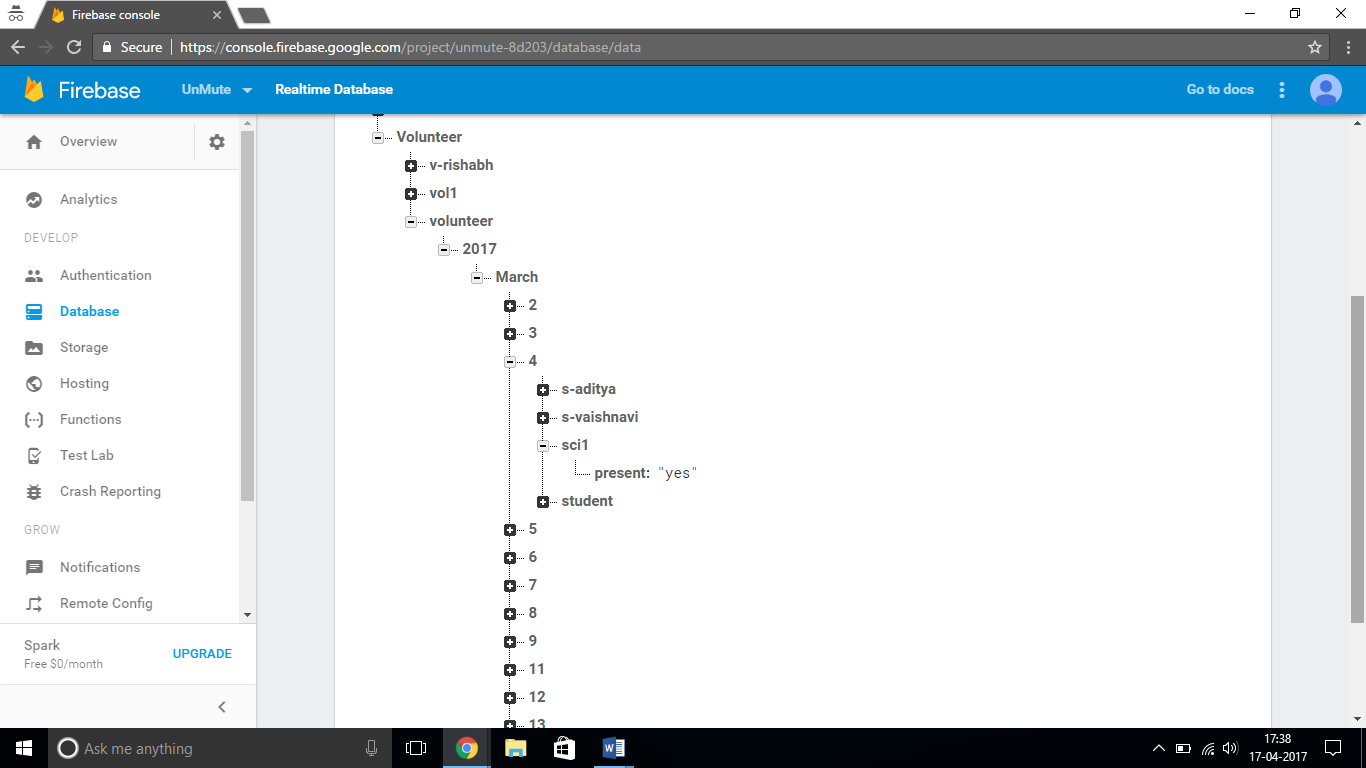
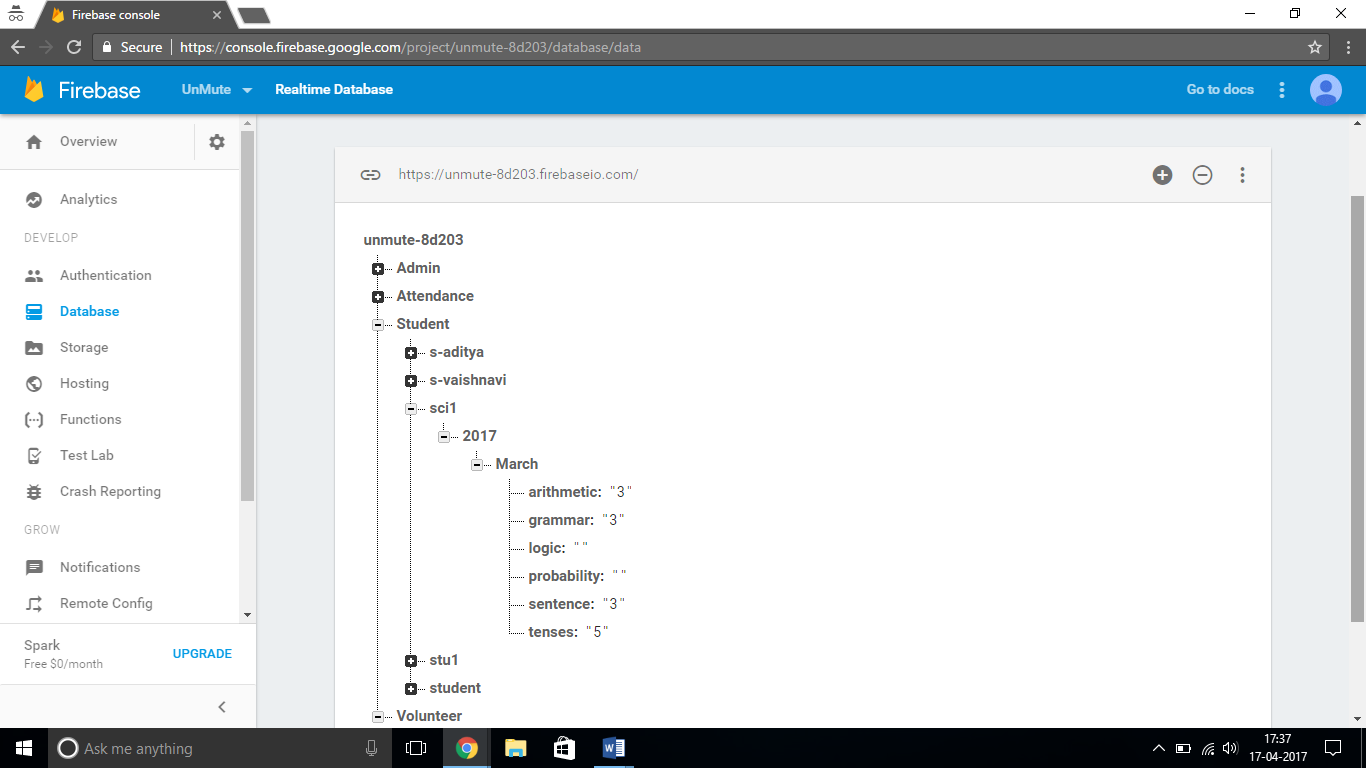
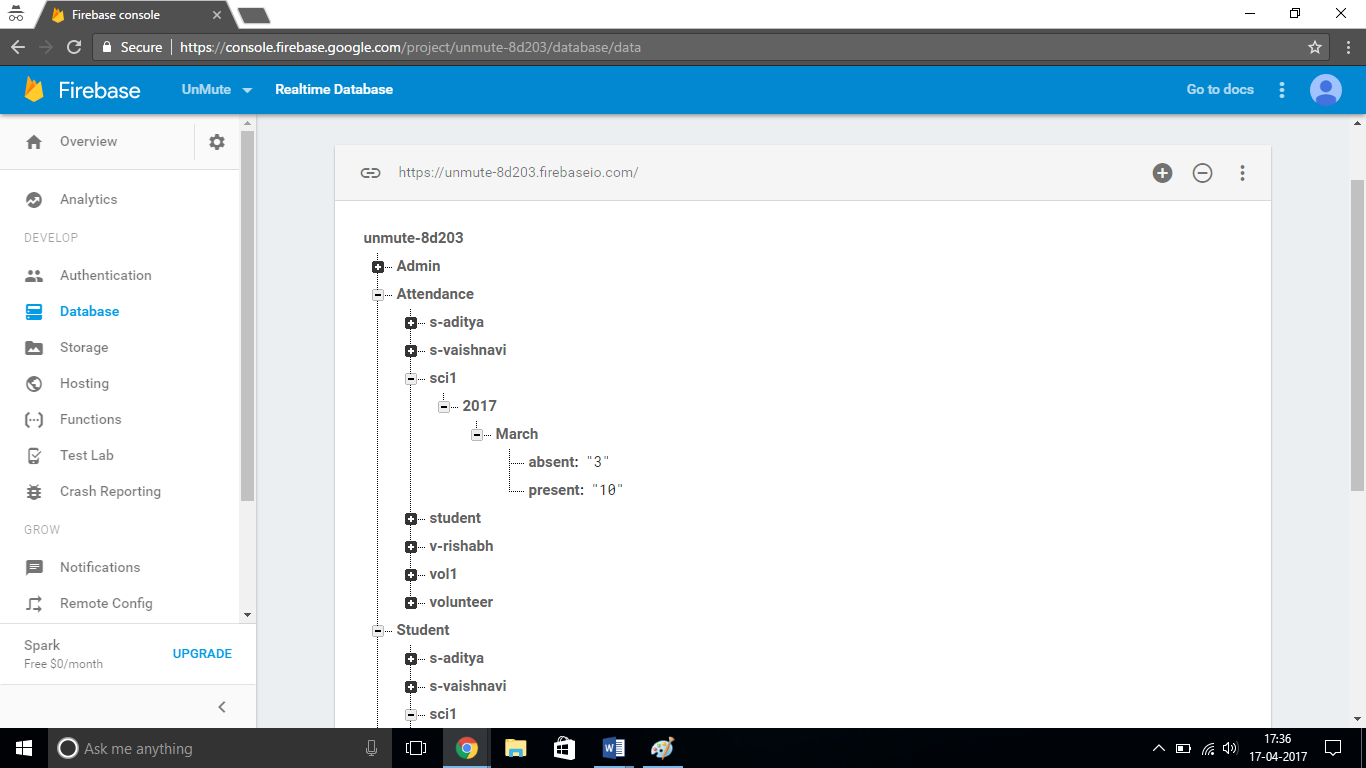
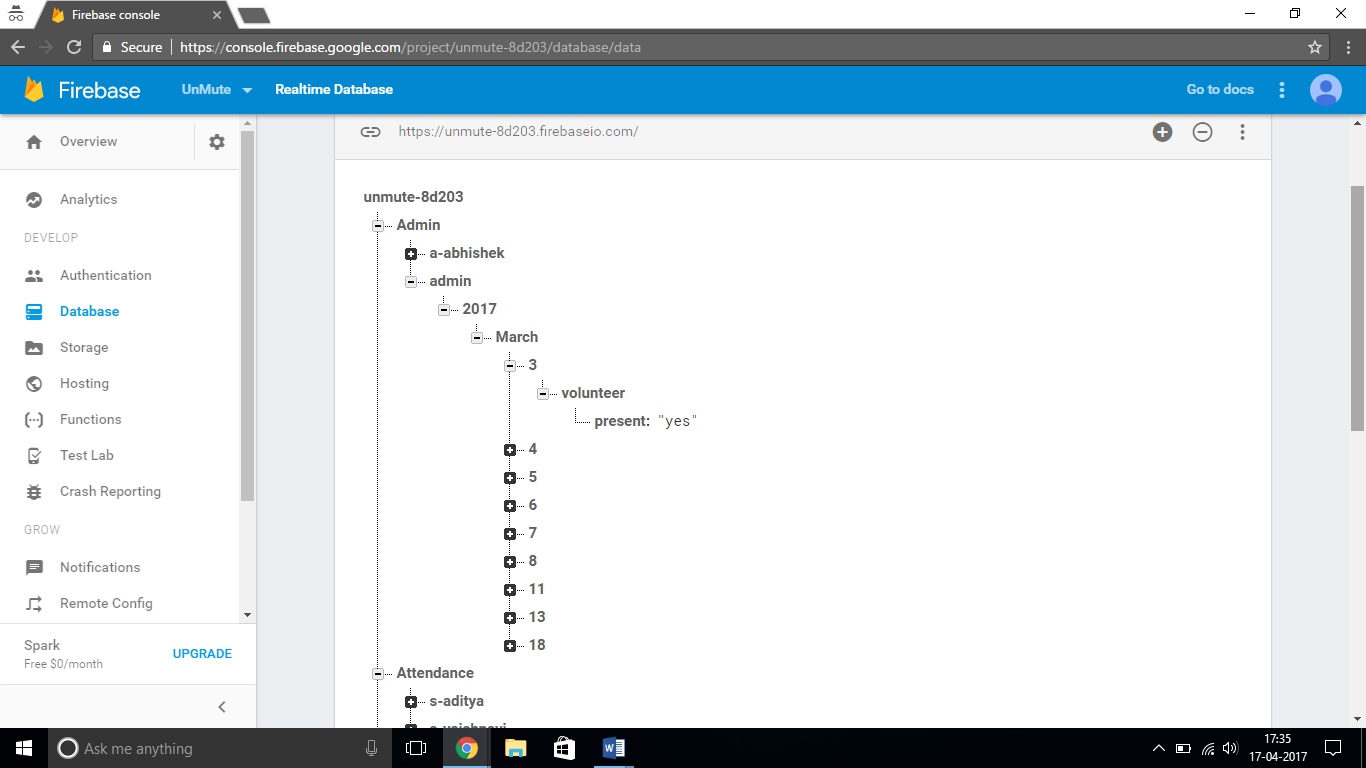




**Figure 6.2.5 Donor Module**



**6.2.6 Firebase Database**

****

**Chapter 7**

**Testing**

**Mobile Application Testing**

Like web application testing, Mobile application testing is also based on same test strategy and methodology. The difference could be in the tools used for testing, some common tools used for mobile application testing are Sikuli, TestComplete, FoneMonkey, Robotium, etc.

Mobile application types are categorized into three section

* Web Application- It is accessed by users over a network like internet or an intranet
* Native Application- It is developed for specific platform and installed on a computing device
* Hybrid Application- It combines elements of both Web and native. Eg Facebook.

For most of the mobile platform, you can use simple CSS, HTML, JS, etc.

**What to test for Mobile Application Testing**

A complete mobile testing application strategy includes device and network infrastructure, selection of target devices, and an effective combination of manual and automated testing tools to cover both non-functional and functional testing.

For mobile application, things to be tested are

* Installation
  + OTA
  + Wi-Fi
  + Data Cable
  + Bluetooth
* Uninstallation
* Application logo
* Splash
* Low Memory
* Visual Feedback
* Exit Application
* Start/Restart of Application

**Mobile Testing Challenges**

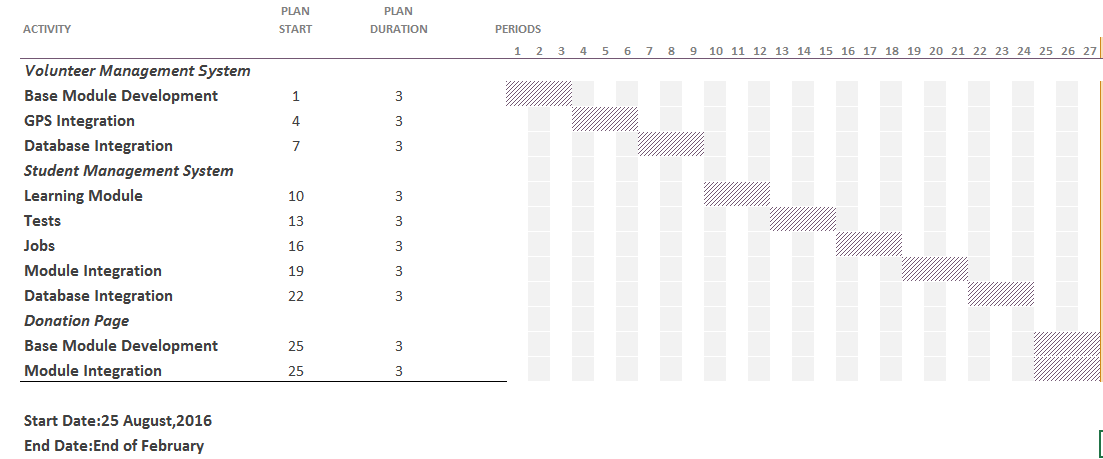
With increased number of mobile user and devices, testing mobile app becomes more and more complex. Testing a mobile application is significantly different from that of a desktop-based web application. The common challenges faced during mobile testing are

* Comprehensive test coverage
* Managing fragmentation ( different O.S version, processor, memory)
* Lack of test plan
* Time Pressure
* Lack of Physical devices
* Diversity in platform and OS

**Chapter 8**

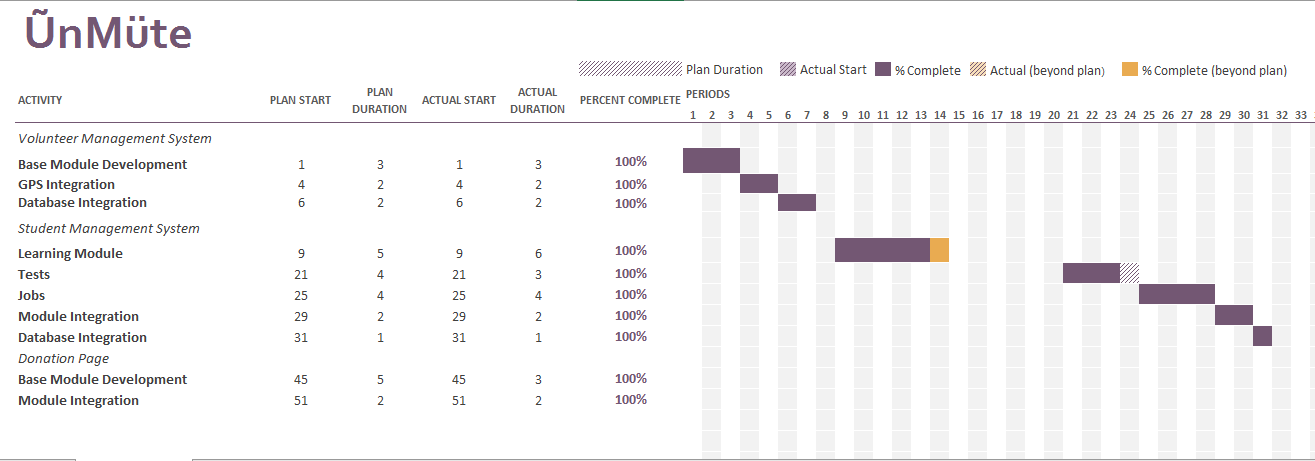
**Timeline Charts**

**8.1 WORK COMPLETED IN SEMESTER VII**



**Figure 8.1: Timeline Chart 1**

**8.2 WORK COMPLETED IN SEMESTER VIII**



**Figure 8.2: Timeline Chart 2**

**Chapter 9**

**Conclusion and Future Scope**

We found that this project has provided us with a new experience to evolve ourselves with the working of a professional world. On completing the project we all are proud with the end result and have found it to be one fulfilling endeavour. Learning android studio integrating with the Firebase database, testing it and rerunning it a million times made it a long journey but the experience was a wonderful one.

This project will be used by the organization for helping the deaf and hard hearing students cope up with difficulties of schooling in a classroom environment. Through the application students will be effectively at par with the normal students.

**APPENDIX I**

**Android Pseudo Code of the Project**

Arithmetic Java code

package com.unmuteapp4gmail.unmute;

import android.content.Intent;

import android.icu.util.Calendar;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

public class Arithmatic extends AppCompatActivity {

public final static String EXTRA\_MESSAGE = "com.example.learn.MESSAGE";

private ArithmaticQuestionLibrary mQuestionLibrary = new ArithmaticQuestionLibrary();

private TextView mScoreView;

private TextView mQuestionView;

private Button mButtonChoice1;

private Button mButtonChoice2;

private Button mButtonChoice3;

private String mAnswer;

private int mScore = 0;

private int mQuestionNumber = 0;

private DatabaseReference databaseReference;

private FirebaseAuth auth;

String userName, month, year;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_arithmatic);

Calendar calendar = Calendar.getInstance();

int thisYear = calendar.get(Calendar.YEAR);

year = Integer.toString(thisYear);

int thisDay = calendar.get(Calendar.DAY\_OF\_MONTH);

int thisMonth = calendar.get(Calendar.MONTH);

if(thisMonth == 0){

month = "January";

}

else if(thisMonth == 1){

month = "February";

}

else if(thisMonth == 2){

month = "March";

}

else if(thisMonth == 3){

month = "April";

}

else if(thisMonth == 4){

month = "May";

}

else if(thisMonth == 5){

month = "June";

}

else if(thisMonth == 6){

month = "July";

}

else if(thisMonth == 7){

month = "August";

}

else if(thisMonth == 8){

month = "September";

}

else if(thisMonth == 9){

month = "October";

}

else if(thisMonth == 10){

month = "November";

}

else if(thisMonth == 11){

month = "December";

}

mScoreView = (TextView)findViewById(R.id.score);

mQuestionView = (TextView)findViewById(R.id.question);

mButtonChoice1 = (Button)findViewById(R.id.choice1);

mButtonChoice2 = (Button)findViewById(R.id.choice2);

mButtonChoice3 = (Button)findViewById(R.id.choice3);

updateQuestion();

databaseReference = FirebaseDatabase.getInstance().getReference().child("Student");

//Start of Button Listener for Button1

mButtonChoice1.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(View view){

//My logic for Button goes in here

if(mQuestionNumber>10){

Intent intent = new Intent(Arithmatic.this, EndArithmatic.class);

String message = "Your final Score = "+Integer.toString(mScore)+"/"+Integer.toString(mQuestionNumber-1);

intent.putExtra(EXTRA\_MESSAGE, message);

startActivity(intent);

auth = FirebaseAuth.getInstance();

userName = auth.getCurrentUser().getEmail();

String userName1 = userName.replaceAll("@unmute.com","");

Students obj = new Students();

obj.setArithmetic(Integer.toString(mScore));

databaseReference.child(userName1).child(year).child(month).child("arithmetic").setValue(Integer.toString(mScore));

}

else {

if (mButtonChoice1.getText() == mAnswer){

mScore = mScore + 1;

updateScore(mScore);

updateQuestion();

//This line of code is optiona

//Toast.makeText(QuizActivity.this, "correct", Toast.LENGTH\_SHORT).show();

}else {

// Toast.makeText(QuizActivity.this, "wrong", Toast.LENGTH\_SHORT).show();

updateQuestion();

}}

}

});

//End of Button Listener for Button1

//Start of Button Listener for Button2

mButtonChoice2.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(View view){

//My logic for Button goes in here

if(mQuestionNumber>10){

Intent intent = new Intent(Arithmatic.this, EndArithmatic.class);

String message = "Your final Score = "+Integer.toString(mScore)+"/"+Integer.toString(mQuestionNumber-1);

intent.putExtra(EXTRA\_MESSAGE, message);

startActivity(intent);

auth = FirebaseAuth.getInstance();

userName = auth.getCurrentUser().getEmail();

String userName1 = userName.replaceAll("@unmute.com","");

Students obj = new Students();

obj.setArithmetic(Integer.toString(mScore));

databaseReference.child(userName1).child(year).child(month).child("arithmetic").setValue(Integer.toString(mScore));

}

else {

if (mButtonChoice2.getText() == mAnswer){

mScore = mScore + 1;

updateScore(mScore);

updateQuestion();

//This line of code is optiona

// Toast.makeText(QuizActivity.this, "correct", Toast.LENGTH\_SHORT).show();

}else {

// Toast.makeText(QuizActivity.this, "wrong", Toast.LENGTH\_SHORT).show();

updateQuestion();

}}

}

});

//End of Button Listener for Button2

//Start of Button Listener for Button3

mButtonChoice3.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(View view){

//My logic for Button goes in here

if(mQuestionNumber>10){

Intent intent = new Intent(Arithmatic.this, EndArithmatic.class);

String message = "Your final Score = "+Integer.toString(mScore)+"/"+Integer.toString(mQuestionNumber-1);

intent.putExtra(EXTRA\_MESSAGE, message);

startActivity(intent);

auth = FirebaseAuth.getInstance();

userName = auth.getCurrentUser().getEmail();

String userName1 = userName.replaceAll("@unmute.com","");

Students obj = new Students();

obj.setArithmetic(Integer.toString(mScore));

databaseReference.child(userName1).child(year).child(month).child("arithmetic").setValue(Integer.toString(mScore));

}

else {

if (mButtonChoice3.getText() == mAnswer){

mScore = mScore + 1;

updateScore(mScore);

updateQuestion();

//This line of code is optiona

//Toast.makeText(QuizActivity.this, "correct", Toast.LENGTH\_SHORT).show();

}else {

//Toast.makeText(QuizActivity.this, "wrong", Toast.LENGTH\_SHORT).show();

updateQuestion();

}}

}

});

//End of Button Listener for Button3

}

private void updateQuestion(){

if(mQuestionNumber<10) {

mQuestionView.setText(mQuestionLibrary.getQuestion(mQuestionNumber));

mButtonChoice1.setText(mQuestionLibrary.getChoice1(mQuestionNumber));

mButtonChoice2.setText(mQuestionLibrary.getChoice2(mQuestionNumber));

mButtonChoice3.setText(mQuestionLibrary.getChoice3(mQuestionNumber));

mAnswer = mQuestionLibrary.getCorrectAnswer(mQuestionNumber);

}

mQuestionNumber++;

}

private void updateScore(int point) {

mScoreView.setText("" + mScore);

}

public void exitTest(View view){

Intent intent = new Intent(this,Tests.class);

startActivity(intent);

}

@Override

public void onBackPressed() {

Intent intent = new Intent(Arithmatic.this,Tests.class);

startActivity(intent);

}

}

Arithmetic Question Library

package com.unmuteapp4gmail.unmute;

public class ArithmaticQuestionLibrary {

private String mQuestions [] = {

"(1) 2.75 + .003 + .158 = ?",

"(2) 7.86 × 4.6 = ?",

"(3) 7/20 = ?",

"(4) Which of the following is the least?",

"(5) All of the following are ways to write 25 percent of N EXCEPT",

"(6) Which of the following is closest to 27.8 × 9.6?",

"(7) A soccer team played 160 games and won 65 percent of them. How many games did it win?",

"(8) Three people who work full-time are to work together on a project, but their total time on the project is to be equivalent to that of only one person working full-time. If one of the people is budgeted for one-half of his time to the project and a second person for one-third of her time, what part of the third worker’s time should be budgeted to this project?",

"(9) 32 is 40 percent of what number?",

"(10) 313 – 225 = ?"

};

private String mChoices [][] = {

{"4.36", "2.911", "0.436"},

{"36.156", "36.216", "351.56"},

{"0.035", "0.858", "0.35"},

{"0.105", "0.501", "0.015"},

{"0.25N", "1/4N", "25N"},

{"280", "300", "2800"},

{"94", "104", "114"},

{"13.3%", "35.2%", "16.7%"},

{"12.8", "128", "80"},

{"112", "115", "88"}

};

private String mCorrectAnswers[] = {"2.911", "36.156", "0.35", "0.015", "25N", "280", "104", "16.7%", "80", "88"};

public String getQuestion(int a) {

String question = mQuestions[a];

return question;

}

public String getChoice1(int a) {

String choice0 = mChoices[a][0];

return choice0;

}

public String getChoice2(int a) {

String choice1 = mChoices[a][1];

return choice1;

}

public String getChoice3(int a) {

String choice2 = mChoices[a][2];

return choice2;

}

public String getCorrectAnswer(int a) {

String answer = mCorrectAnswers[a];

return answer;

}

}

Arithmetic XML code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:orientation="vertical"

tools:context="com.unmuteapp4gmail.unmute.Arithmatic">

<RelativeLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:padding="8dp"

android:layout\_marginBottom="0dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Score"

android:textSize="15sp"

android:layout\_alignParentLeft="true"

android:id="@+id/score\_text"/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/score"

android:layout\_alignParentRight="true"

android:text="0"

android:textSize="15sp"/>

</RelativeLayout>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="85dp"

android:text=""

android:textSize="15sp"

android:padding="8dp"

android:layout\_marginBottom="0dp"

android:id="@+id/question"

android:layout\_weight="0.2" />

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text=""

android:background="#0091EA"

android:textColor="#fff"

android:padding="8dp"

android:layout\_marginBottom="8dp"

android:id="@+id/choice1"/>

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text=""

android:background="#0091EA"

android:textColor="#fff"

android:padding="8dp"

android:layout\_marginBottom="8dp"

android:id="@+id/choice2"/>

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text=""

android:background="#0091EA"

android:textColor="#fff"

android:padding="8dp"

android:layout\_marginBottom="24dp"

android:id="@+id/choice3"/>

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Quit"

android:background="#B71C1C"

android:textColor="#fff"

android:padding="8dp"

android:layout\_marginBottom="8dp"

android:id="@+id/quit"

android:onClick="exitTest"

/>

</LinearLayout>

Check Attendance Java code

package com.unmuteapp4gmail.unmute;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

public class CheckAttendance extends AppCompatActivity {

EditText name, umonth, uyear;

Button getAttendance;

private FirebaseAuth auth;

TextView name1, month1, year1, nwd, ndp, nda, ap;

String userName, userMonth, userYear, curruser;

int pre, abs, total;

float perc, div;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_check\_attendance);

FirebaseDatabase database = FirebaseDatabase.getInstance();

final DatabaseReference ref = database.getReferenceFromUrl("https://unmute-8d203.firebaseio.com/Attendance");

name = (EditText) findViewById(R.id.editText);

umonth = (EditText) findViewById(R.id.editText2);

uyear = (EditText) findViewById(R.id.editTextYear);

getAttendance = (Button) findViewById(R.id.buttonFind);

name1 = (TextView) findViewById(R.id.textViewName);

month1 = (TextView) findViewById(R.id.textViewMonth);

year1 = (TextView) findViewById(R.id.textViewYear);

nwd = (TextView) findViewById(R.id.textView4);

ndp = (TextView) findViewById(R.id.textView6);

nda = (TextView) findViewById(R.id.textView10);

ap = (TextView) findViewById(R.id.textView12);

getAttendance.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

userName = name.getText().toString().trim();

userMonth = umonth.getText().toString().trim();

userYear = uyear.getText().toString().trim();

auth = FirebaseAuth.getInstance();

curruser = auth.getCurrentUser().getEmail();

final String curruser1 = curruser.replaceAll("@unmute.com","");

char firstchar;

firstchar = curruser1.charAt(0);

if(firstchar == 's') {

if(TextUtils.isEmpty(userName)){

Toast.makeText(CheckAttendance.this, "Please Enter the ID!", Toast.LENGTH\_LONG).show();

name1.setText("");

month1.setText("");

year1.setText("");

nwd.setText("");

ndp.setText("");

nda.setText("");

ap.setText("");

return;

}

else {

if (!userName.equals(curruser1)) {

userName = curruser1;

Toast.makeText(CheckAttendance.this, "You can check only your own attendance", Toast.LENGTH\_LONG).show();

}

}

}

if(firstchar == 'v') {

if(TextUtils.isEmpty(userName)){

Toast.makeText(CheckAttendance.this, "Please Enter the ID!", Toast.LENGTH\_LONG).show();

name1.setText("");

month1.setText("");

year1.setText("");

nwd.setText("");

ndp.setText("");

nda.setText("");

ap.setText("");

return;

}

else {

char firstchar1;

firstchar1 = userName.charAt(0);

if (firstchar1 == 'v') {

if (!userName.equals(curruser1)) {

userName = curruser1;

Toast.makeText(CheckAttendance.this, "You cannot check attendance of other volunteers", Toast.LENGTH\_LONG).show();

}

}

}

}

if(TextUtils.isEmpty(userName)){

Toast.makeText(CheckAttendance.this, "Please Enter the ID!", Toast.LENGTH\_LONG).show();

name1.setText("");

month1.setText("");

year1.setText("");

nwd.setText("");

ndp.setText("");

nda.setText("");

ap.setText("");

return;

}

if(TextUtils.isEmpty(userMonth)){

Toast.makeText(CheckAttendance.this, "Please Enter the Month!", Toast.LENGTH\_LONG).show();

name1.setText("");

month1.setText("");

year1.setText("");

nwd.setText("");

ndp.setText("");

nda.setText("");

ap.setText("");

return;

}

if (TextUtils.isEmpty(userYear)) {

Toast.makeText(CheckAttendance.this, "Please Enter the Year!", Toast.LENGTH\_LONG).show();

name1.setText("");

month1.setText("");

year1.setText("");

nwd.setText("");

ndp.setText("");

nda.setText("");

ap.setText("");

return;

}

ref.addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

if(dataSnapshot.hasChild(userName)) {

if(dataSnapshot.child(userName).hasChild(userYear)){

if(dataSnapshot.child(userName).child(userYear).hasChild(userMonth)){

ref.child(userName).child(userYear).child(userMonth).addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

StuAttendance obj = dataSnapshot.getValue(StuAttendance.class);

if(obj.getPresent().equals("") && !obj.getAbsent().equals("")) {

pre = 0;

abs = Integer.parseInt(obj.getAbsent());

total = pre + abs;

perc = (pre / total) \* 100;

}

if(obj.getAbsent().equals("") && !obj.getPresent().equals("")) {

abs = 0;

pre = Integer.parseInt(obj.getPresent());

total = pre + abs;

perc = (pre / total) \* 100;

}

if(!obj.getPresent().equals("") && !obj.getAbsent().equals("")) {

pre = Integer.parseInt(obj.getPresent());

abs = Integer.parseInt(obj.getAbsent());

total = pre + abs;

div = (float)pre/total;

perc = div \* 100;

}

if(obj.getPresent().equals("") && obj.getAbsent().equals("")){

pre = 0;

abs = 0;

total = 0;

perc = 0;

}

name1.setText(userName);

month1.setText(userMonth);

year1.setText(userYear);

nwd.setText(Integer.toString(total));

ndp.setText(Integer.toString(pre));

nda.setText(Integer.toString(abs));

ap.setText(String.valueOf(perc));

Toast.makeText(CheckAttendance.this, "Data Found!", Toast.LENGTH\_SHORT).show();

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

else{

Toast.makeText(CheckAttendance.this, "Invalid Month", Toast.LENGTH\_SHORT).show();

name1.setText("");

month1.setText("");

year1.setText("");

nwd.setText("");

ndp.setText("");

nda.setText("");

ap.setText("");

}

}

else{

Toast.makeText(CheckAttendance.this, "Invalid Year", Toast.LENGTH\_SHORT).show();

name1.setText("");

month1.setText("");

year1.setText("");

nwd.setText("");

ndp.setText("");

nda.setText("");

ap.setText("");

}

}

else{

Toast.makeText(CheckAttendance.this, "Please Enter Valid ID!", Toast.LENGTH\_SHORT).show();

name1.setText("");

month1.setText("");

year1.setText("");

nwd.setText("");

ndp.setText("");

nda.setText("");

ap.setText("");

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

});

}

}

Check Attendance XML code

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/activity\_check\_attendance"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#1f1f21"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context="com.unmuteapp4gmail.unmute.CheckAttendance">

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="textPersonName"

android:hint="Enter ID"

android:ems="10"

android:layout\_alignParentTop="true"

android:theme="@style/MyEditText"

android:textColorHint="#ffffff"

android:textColor="#ffffff"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:id="@+id/editText" />

<View

android:layout\_width="match\_parent"

android:layout\_height="2dp"

android:layout\_marginTop="200dp"

android:background="#ffffff"

android:id="@+id/ViewTop"

/>

<View

android:layout\_width="match\_parent"

android:layout\_height="2dp"

android:layout\_marginTop="255dp"

android:background="#ffffff"

/>

<View

android:layout\_width="match\_parent"

android:layout\_height="2dp"

android:layout\_marginTop="150dp"

android:background="#ffffff"

/>

<Button

android:text=" Get Attendance "

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="#5bcaff"

android:textColor="#ffffff"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="90dp"

android:id="@+id/buttonFind" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_alignTop="@+id/textView2"

android:layout\_alignParentRight="true"

android:id="@+id/textView4" />

<TextView

android:text="No. of Days Present"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/textView2"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:layout\_marginTop="10dp"

android:id="@+id/textView5" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignTop="@+id/textView5"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_alignRight="@+id/textView4"

android:layout\_alignEnd="@+id/textView4"

android:id="@+id/textView6" />

<TextView

android:text="No. of Days Absent"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/textView5"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_marginTop="10dp"

android:id="@+id/textView7" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignTop="@+id/textView7"

android:layout\_alignRight="@+id/textView6"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_alignEnd="@+id/textView6"

android:id="@+id/textView10" />

<TextView

android:text="Attendance %"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/textView7"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_marginTop="10dp"

android:id="@+id/textView11" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignTop="@+id/textView11"

android:layout\_alignRight="@+id/textView10"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_alignEnd="@+id/textView10"

android:id="@+id/textView12" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/textView11"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_marginTop="5dp"

android:id="@+id/textView13" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignTop="@+id/textView13"

android:layout\_alignParentRight="true"

android:textColor="#ffffff"

android:textSize="20dp"

android:id="@+id/textView14" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignTop="@+id/textView15"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_alignParentRight="true"

android:id="@+id/textView16" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/buttonFind"

android:layout\_centerHorizontal="true"

android:textColor="#ffffff"

android:textSize="19dp"

android:layout\_marginTop="20dp"

android:id="@+id/textView"

android:textAlignment="center" />

<TextView

android:text="No. of working days"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/textView2"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_marginTop="95dp"

android:layout\_below="@+id/textView"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true" />

<TextView

android:layout\_width="160dp"

android:layout\_height="wrap\_content"

android:id="@+id/textView15"

android:textColor="#ffffff"

android:textSize="20dp"

android:layout\_alignParentBottom="true"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true" />

<EditText

android:layout\_width="160dp"

android:layout\_height="wrap\_content"

android:inputType="textPersonName"

android:hint="Month(ex.January)"

android:ems="10"

android:textColorHint="#ffffff"

android:theme="@style/MyEditText"

android:textColor="#ffffff"

android:layout\_below="@+id/editText"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true"

android:id="@+id/editText2" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textAlignment="center"

android:textColor="#ffffff"

android:layout\_alignBottom="@+id/ViewTop"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="18dp"

android:id="@+id/textViewName" />

<EditText

android:layout\_width="150dp"

android:layout\_height="wrap\_content"

android:inputType="textPersonName"

android:hint="Year(ex.2017)"

android:textColorHint="#ffffff"

android:theme="@style/MyEditText"

android:textColor="#ffffff"

android:ems="10"

android:layout\_below="@+id/editText"

android:layout\_alignParentRight="true"

android:id="@+id/editTextYear" />

<TextView

android:layout\_width="170dp"

android:layout\_height="wrap\_content"

android:textAlignment="center"

android:textColor="#ffffff"

android:layout\_marginTop="18dp"

android:id="@+id/textViewMonth"

android:layout\_below="@+id/ViewTop"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true" />

<TextView

android:layout\_width="140dp"

android:layout\_height="wrap\_content"

android:layout\_alignBottom="@+id/textViewMonth"

android:layout\_alignParentRight="true"

android:textColor="#ffffff"

android:textAlignment="center"

android:layout\_alignParentEnd="true"

android:id="@+id/textViewYear" />

</RelativeLayout>

**REFERENCES**

|  |  |
| --- | --- |
| [1] | A. Sujith Kumar, Shahabaz Begum (2014) 'Sign Mobiles: An Android App for Specially Able People', International Journal of Advanced Research in Computer Science and Software Engineering, 4(9), pp. [Online]. Available at: <https://www.ijarcsse.com/docs/papers/Volume_4/9_September2014/V4I9-0204.pdf> |
| [2] | Miralkumar Surati , Mahesh T. Kolte (2015) 'Comprehensive Regarding Hearing Impairment using Smart Android Phone ', International Journal of Advanced Research in Computer and Communication Engineering, 4(7), pp. [Online]. Available at: http://www.ijarcce.com/upload/2015/july-15/IJARCCE%2019.pdf |
| [3] | Apeksha Khilari, Manasi Marathe, Aishwarya Parab, Manita Rajput (2016) ‘Android based aid for the deaf', International Journal of Technical Research and Applications, (40), pp. [Online]. Available at: http://www.ijtra.com/special-issue-view/android-based-aid-for-the-deaf.pdf |
| [4] | Nadene Eisner (n.d.) Engaging Deaf and Hard of Hearing Students in the School Library: A Handbook for Teacher-Librarians. [Online]. Available at: http://groups.lis.illinois.edu/k-12program/Eisner\_CAS\_Project.pdf |
| [5] | http://sites.allegheny.edu/disabilityservices/students-who-are-deaf-or-hard-of-hearing/ |
| [6] | http://www.bellaonline.com/articles/art46258.asp |
| [7] | https://developer.android.com/studio/intro/index.html |
| [8] | https://en.wikipedia.org/wiki/Firebase |
| [9] | https://howtofirebase.com/what-is-firebase-fcb8614ba442 |
| [10] | http://www.guru99.com/application-testing.html |

**PUBLICATIONS**

[1] Aditya Vadhavkar , Abhishek Sengupta, Rishabh Patil, Vaishnavi Patil, in “Android Application as Schooling Medium for the Hearing Impaired” in International Journal for Scientific Research and development(ISSN: 2321-0613 ), Volume 4, Issue 8, December 2016.

**ACKNOWLEDGEMENT**

We would like to thank our Principal,**Dr. S.M. Khot** for giving us the time and resources for completion of the synopsis report for our final year project.

​ We would also like to thank **Dr. Lata Ragha**, HOD of Computer Department and Mrs. **Smita Dange**, coordinator of Computer Department for their support.

​ We convey our sincere gratitude to our guide, **Mr. Rahul Jadhav** for his constant help and support. Without his valuable input and feedback this report would not have met the required standards. Our special thanks to our mentor from Sounds of Silence- **Ms. Shruti Singhal**for her valuable guidance and the precious time.

​ We would like to thank all the staff and non-staff members for their help and support without which we would never be able to complete our work before the deadlines.We would like to extend our thanks to all the people whose valuable work helped us study and understand all the concepts pertaining to Android and its related technologies.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Abhishek Sengupta (101354)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Aditya Vadhavkar (101361)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Rishabh Patil (101350)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Vaishnavi Patil (101345)

Date: