VOCAL COMBO ANDROID APPLICATION

by

SRAVYA MATHUKUMALLI

B.Tech., Andhra University, 2014

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Approved by:

Major Professor
Dr. Mitchell L. Neilsen

Abstract

Now-a-days people from various backgrounds need different information on demand. People are relying on web as a source of information they need. But to get connected to internet all the time with a computer system is not possible. Android, being open source has already made its mark in the mobile application development. The highest smart phone user base and the ease of developing the applications in Android is an added advantage for the users as well as the Android developers.

The vocal combo is an Android application which provides the required functionality on Android supported smart phone or a tablet. This provides the flexibility of accessing information at the users' fingertips. This application is built using Android SDK, which makes the application easy to deploy on any Android powered device. Vocal Combo is a combination of voice based applications. It includes a Text-To-Voice convertor and Voice-To-Text convertor. This application helps the user to learn the pronunciation of various words. At the same time the user can also check his/her pronunciation skills. This application also provides the functionality of meaning check where the user can check the meaning of the words he types in or speaks out. At any point of time, the user can check the history of the words for which he has checked the meaning or pronunciation for. The application also provides the support to the user on how to use this application.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to extend my sincere thanks to my academic advisor, Dr. Mitchell L Neilsen, who encouraged me in developing this application and provided me his valuable suggestions and feedback when I took up the mobile application development domain as my subject for my project which boosted my confidence in Android development.

I take immense pleasure in expressing my sincere gratitude to my committee members, Dr. Daniel Andresen and Dr. Torben Amtoft, who despite of their busy schedule made time to serve on my committee and for providing their invaluable feedback on the project.

This project would not be possible without the help of the Academic staff of the Computing and Information Sciences who are always there to help students by providing the required resources for their work.

This would be incomplete without acknowledging the love and support of my parents and friends who are there to help me out in my odds, thereby indirectly responsible for providing me a peaceful environment to complete this project.

Chapter 1 - Project Description

1.1 Introduction

Vocal Combo is an Android application which serves as a voice dictionary for the users. Now-a-days people from different countries, majorly non-native speakers of English are planning to study abroad which requires them to communicate in English. This application is targeted for such users. This application provides the functionality of checking the pronunciation of the word or sentence the user types in. This helps the user to know the correct pronunciation of the word before-hand. The user can also test his pronunciation skills using this application. In addition to pronunciation check, vocal combo provides the meaning check for any word the user types in or speaks out. The user can check the history of the words for which he has checked the pronunciation or meaning. This helps the user to practice or test his pronunciation skills for the words which he has searched for previously.

1.2 Motivation

My motivation to develop this application is from my very own experiences. I am a non-native speaker of English who have medium of education as English. So, I am pretty much comfortable speaking English. I am required to write TOEFL (Test of English as Foreign Language) to study abroad. The exam requires me to speak about the given topic, answer few questions after listening to the lecture. While preparing for the exam, I felt the serious need of a voice dictionary which helps me find the pronunciation of fairly complex words and the meanings of the same. I have to search on internet for online dictionaries all the time for help. This added the pain of carrying my laptop everywhere I go. I thought of developing this vocal combo, an Android application which helps you access the functionalities of pronunciation and meaning check from your Android powered device.

The motivation behind choosing Android as the application development platform is its wide usage across the globe and it being an open source. I have hands-on-experience working with Java. So, I never felt pain in developing this application.

Chapter 2 - Background

Android operating system is structured as a stack of software components comprising applications, an operating system, run-time environment, middleware, services and libraries [1]. The architecture can be visualized as outlined in Figure 2.1.

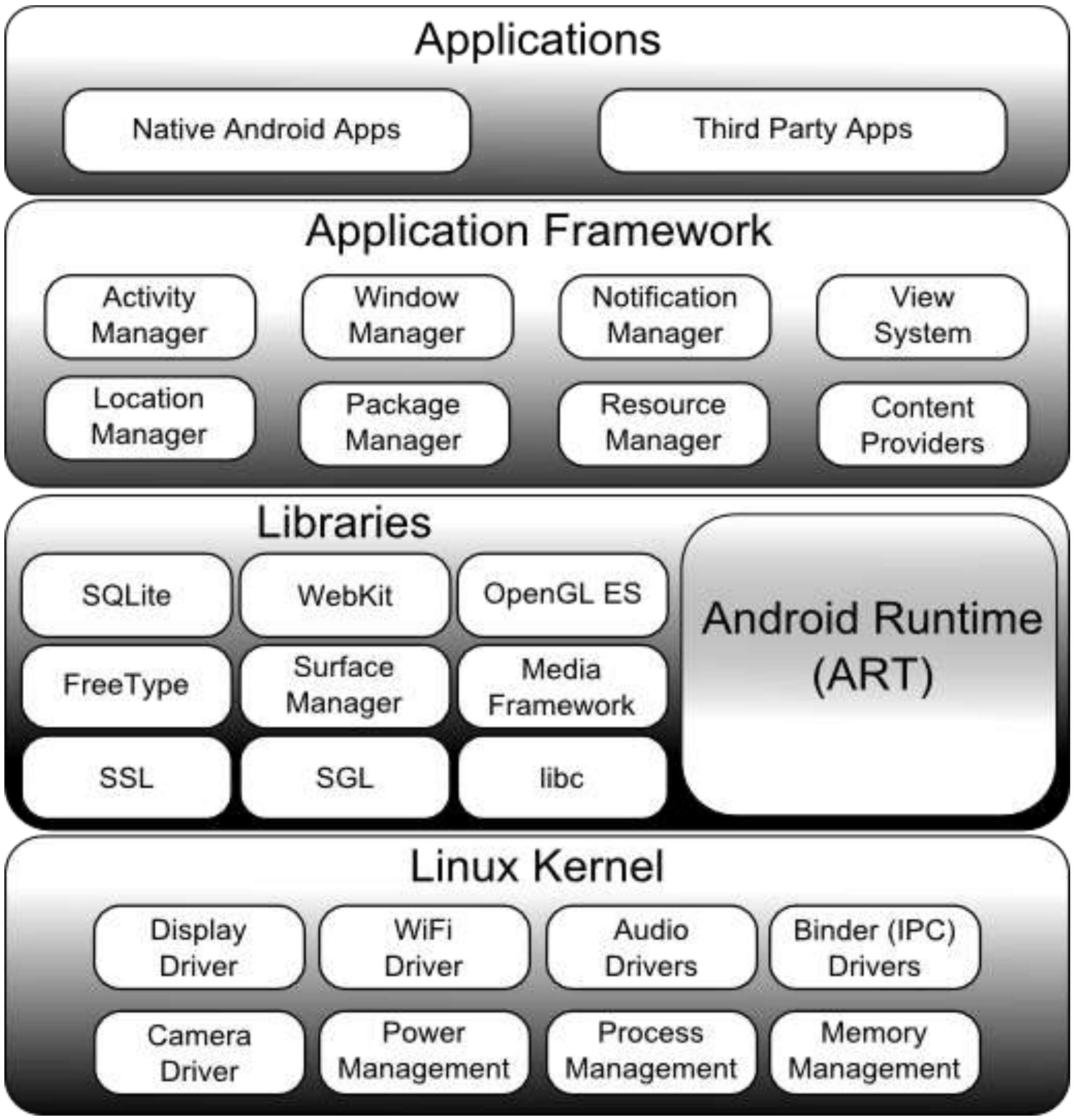


Figure 1 Android Architecture

http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture

To provide optimal application development and execution environment for mobile devices, each layer of the stack, and the corresponding elements within each layer, are tightly integrated and carefully tuned. In this chapter, I will discuss about each layer of the Android stack in detail.

The Linux Kernel

The Linux Kernel provides a level of abstraction between the device hardware and the upper layers of the Android software stack. The kernel handles all the things that Linux is really good at such as networking, a vast array of device drivers, which take the pain out of interfacing to peripheral hardware. The Kernel also provides preemptive multitasking, low-level core system services such as memory, process and power management in addition to providing a network stack and device drivers for hardware such as the device display, Wi-Fi and audio. Hence, Android takes the advantage of the efficiency and performance of the Linux Kernel.

Android Runtime-ART

The Android Runtime-ART comes up with a major component, Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android. When an Android app is built within Android Studio it is compiled into an intermediate bytecode format, referred to as DEX (Dalvik Executable Format). By default, Dalvik limits applications to a single classes.dex bytecode file per APK. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language.

When the application is subsequently loaded onto the device, the Android Runtime (ART) uses a process referred to as Ahead-of-Time (AOT) compilation to translate the bytecode down to the native instructions required by the device processor. This format is known as Executable and Linkable Format (ELF). Each time the application is subsequently launched, the ELF executable version is run, resulting in faster application performance and improved battery life. This contrasts with the Just-in-Time (JIT) compilation approach used in older Android implementations whereby the bytecode was translated within a virtual machine (VM) each time the application was launched.

Android Libraries

The Android development environment has a set of Java-based libraries which are specific to Android development facilitating the user interface design, graphics drawing and database access in addition to a set of standard Java development libraries that supports string handling, networking and file manipulation [2].

The following libraries are used in developing this application-graphics library which provides the drawing API, the util library provides the string conversions and XML manipulations, database library to provide access to database management classes, and webkit library to access the webpages from the application. In fact most of the above specified Java libraries are actually Java wrappers around a set of C/C++ based libraries. Secure Sockets Layer (SSL) communication, SQLite database management, audio and video playback, bitmap and vector font rendering, display subsystem and graphic layer management are implementations of the standard C system library (libc).

Application Framework

Application framework provides the environment to manage and run android applications. Few of the components in android framework are the activity provide which controls the activity stack, the content providers which allows the sharing of data with other applications, the location manager provides the location services to the application, the notifications manager which allows the application to display alerts and notifications.

Applications

Applications are located at the top of the Android software stack in the Android architecture. These comprise both the native applications provided with the particular Android implementation and the third party applications installed by the user after purchasing the device.

Chapter 3 - Related Work

3.1 Current System

The current system is a desktop application which requires the user to search for online dictionary in his laptop/desktop. Even there are very few websites that provide the combination of pronunciation and meaning check simultaneously.

3.1.1 Disadvantages

- Time Consuming
- No search history

3.2 Proposed System

Vocal Combo is an Android application which provides the functionalities of voice dictionary on portable devices like Android powered tablet/ mobile phone. This application can be easily extended in future to add any additional functionalities if required.

3.1.2 Advantages

- Portable
- Extendable

3.1.2 Purpose of the System

Vocal Combo is developed to provide the functionalities of a voice dictionary, pronunciation and meaning check on an Android powered device. This provides flexibility to the user. If the user wants to check the history of the words he/she has checked for, the history module provides this functionality.

3.1.3 Scope of the System

3.1.3.1 In Scope

The scope of the vocal combo is restricted to providing the meaning and pronunciation check functionalities. This application also provides the history check functionality for both pronunciation and meaning checks.

3.1.3.2 Out of Scope

Vocal Combo application requires the user to be connected to the internet all the time. The scope of the meaning check functionality is restricted to checking the meaning for words but not sentences or phrases.

3.1.4 Objective

The objective of the application is to provide flexibility of the user by providing the functionalities of both pronunciation and meaning check in a single application.

3.1.5 System Overview

After the user downloads the Vocal Combo application from the play store, he should install it on his Android powered device. The user can check the options available on the home page and proceed with the required functionality. To help him use the application effectively, he is provided with the help option which guides him through the application.

Chapter 4 - Requirement Analysis

Requirement Analysis is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement analysis phase arises. The software is initiated by the user needs. Requirement Analysis is the means of translating the ideas on the minds of the users (the input) into a formal document (the output of the requirement phase). After few weekly meeting with my Professor Dr. Mitchell L Neilsen, we came up with the following requirements for the application.

4.1 Functional Requirements

- **Pronunciation check**: The module speaks the text the user types in to facilitate the user with the correct pronunciation check.
- **Meaning check**: The module displays the meaning of the word the user types or speaks out to facilitate the user with the correct meaning check.
- **Pronunciation History check**: This module facilitates the user to check for the words for which he/she had checked the pronunciation for.
- **Meaning History check**: This module facilitates the user to check for the words for which he/she had checked the meaning for.
- **Help**: The help module facilitate the user to navigate through the application.

4.2 Non-Functional Requirements

Non-functional requirements describe aspects that are not directly related to the functional behavior of the system.

4.2.1 User interface and Human factors

- The user interface is designed such that the user requires very little, if not no knowledge of Android to access this application.
- The functionalities of each screen and navigation through the screens of the application is described in the help module.
- This application is user friendly and very interactive.

- This application prompts error messages if the user gives incorrect input or tries to access the functionalities of the application incorrectly.
- User is required to have basic knowledge of English to access the application.

4.3 Software Specifications

- Operating System: Windows 8.1
- Platform: Android SDK Framework 10 or higher
- Database: SQLite Database
- IDE: Android Studio IDE
- Technologies used: Java 1.6, Android, SQLite
- Emulator: SDK version 3.0 or higher

4.4 Hardware Specifications

- Processor: Pentium IV or higher
- Processor speed: 1.6GHz
- RAM: 512 MB
- Disk Space: 250 MB or higher
- Android Device: Any device with Android OS.

Chapter 5 - System Design

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements [3]. The system design depicts the overall product architecture, the subsystems that compose the product, and the manner in which subsystems are allocated to processors, the allocation of classes to subsystems, and the design of the user interface.

The system design is reviewed by examining the object-behavior model developed during object-oriented analysis and mapping required system behavior against the subsystems designed to accomplish this behavior.

UML is a standard visual modeling language widely used in system design for modeling business and similar processes, analysis, design, and implementation of software-based systems [4]. UML diagrams can be broadly classified into Structural and Behavioral diagrams. The static aspects of the system which forms the main structure are represented by classes, interfaces, objects, components, and nodes. The following are the structural diagrams-

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Behavioral diagrams capture the dynamic aspect like changing/ moving parts of the system. The following are the behavioral diagrams-

- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram

5.1 Use Case Diagram

A Use case diagram is a set of scenarios in which our system or application interacts with people, organizations, or external systems. The use case is an external view of the system that represents some action the user might perform in order to complete a task. The main components of a use case diagram are use cases, actors and their relationships. In its simplest form, a use case can be described as a specific way of using the system from a user's (actor's) perspective. It is used to describe a function provided by the system that yields a visible result for an actor. An actor describes any entity that interacts with the system.

Use Case Diagram for Vocal Combo

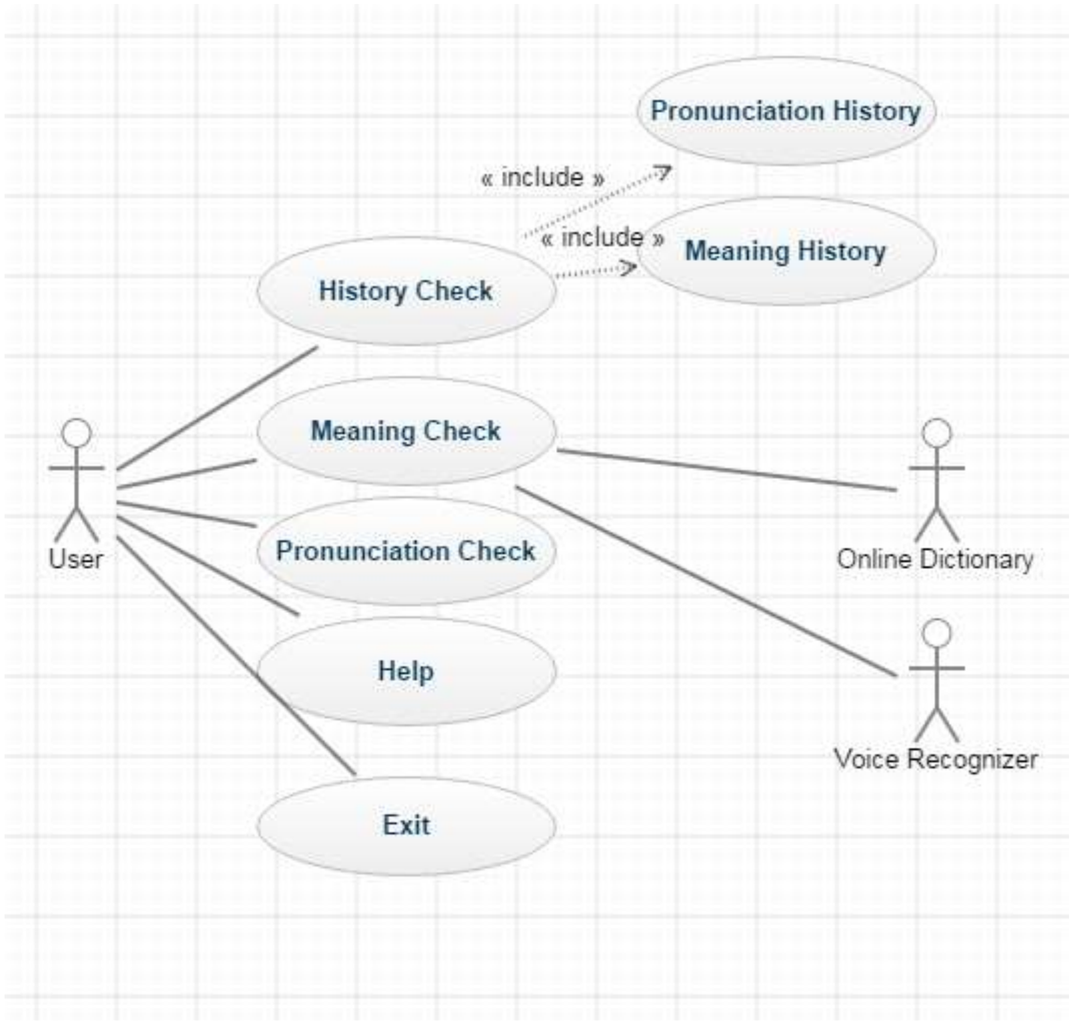


Figure 2 Use case Diagram

Actors:

In our model, the user, online dictionary, and voice recognizer are the actors who interacts with the system.

Use cases:

- **Meaning Check-** This use case denotes the sequence of actions required for the user to check for the meaning of the word either entered or spoken by him/her.
- **Pronunciation Check-** This use case denotes the sequence of actions required for the user to check the pronunciation of the word entered by him/her.
- **History Check-** This use case denotes the sequence of actions required by the user to check the pronunciation or meaning history. Hence the Pronunciation History and Meaning History are represented as sub use cases of the History check use case.
- **Help-** This use case represents the functionality of help provided by the application for user to navigate through the screens.
- **Exit-** This use case denotes the sequence of steps required by the user to exit the application.

Associations: Association exists whenever an actor is involved in an interaction described by the use case.

- **Associations of User:**
  The user is associated with History check, Pronunciation check, Meaning check, Help, and Exit use cases.
- **Associations of Online Dictionary:** The Online Dictionary use case is associated with the Meaning check use case as it displays the meaning of the words during the Meaning check functionality.
- **Associations of Voice Recognizer:** The Voice Recognizer use case is associated with the Meaning check use case as it captures the words spoken by the user during the Meaning check functionality**.**

5.2 Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram for Vocal Combo contains User, Meaning Check, Pronunciation Check, History, SQLiteDB, Pronunciation History, Meaning History classes.

- **User class-** The user class contains the userId, which is optional as attribute and speakWord, typeWord, clickButton as operations to represent the actions performed by the user while accessing the application.
- **Meaning Check class-** The Meaning Check class contains connectionId as attribute and openConnection to open connection with internet, getWord to get the input from the user, displayMeaning to display the meaning of the entered word, storeDB to store the words in Database, and displayErrorMessage to display error message in case of invalid input from user as operations.
- **Pronunciation Check class-** The Pronunciation Check class contains TTSObject as attribute and getText to get the input from the user, speakText to pronounce the entered word, storeDB to store the words in Database, and displayErrorMessage to display error message in case of invalid input from user as operations.
- **SQLiteDB class-** The SQLiteDB class contains DBConnectionObject as attribute and insertDB to insert the words in the Database, removed to remove words in case of clear history, retrieveDB in case of the history check, openConnection, closeConnection to open and close connections with the SQLite Database respectively are the operations.
- **History class-** The History class contains isMeaningHistory and isEmpty as attributes and clearHistory in case user chooses to clear search history, retrieveDB to establish Database connections and displayErrorMessage to display error message in case of empty history.
- **Pronunciation History-** The Pronunciation History contains isPronunciationHistory as attribute and displayPronunciationHistory to display the history of words the user have searched the pronunciation for. This class extends the History class.

- **Meaning History-** The Meaning History contains isMeaningHistory as attribute and displayMeaningHistory to display the history of words the user have searched the meaning for. This class extends the History class.

Class Diagram for Vocal Combo

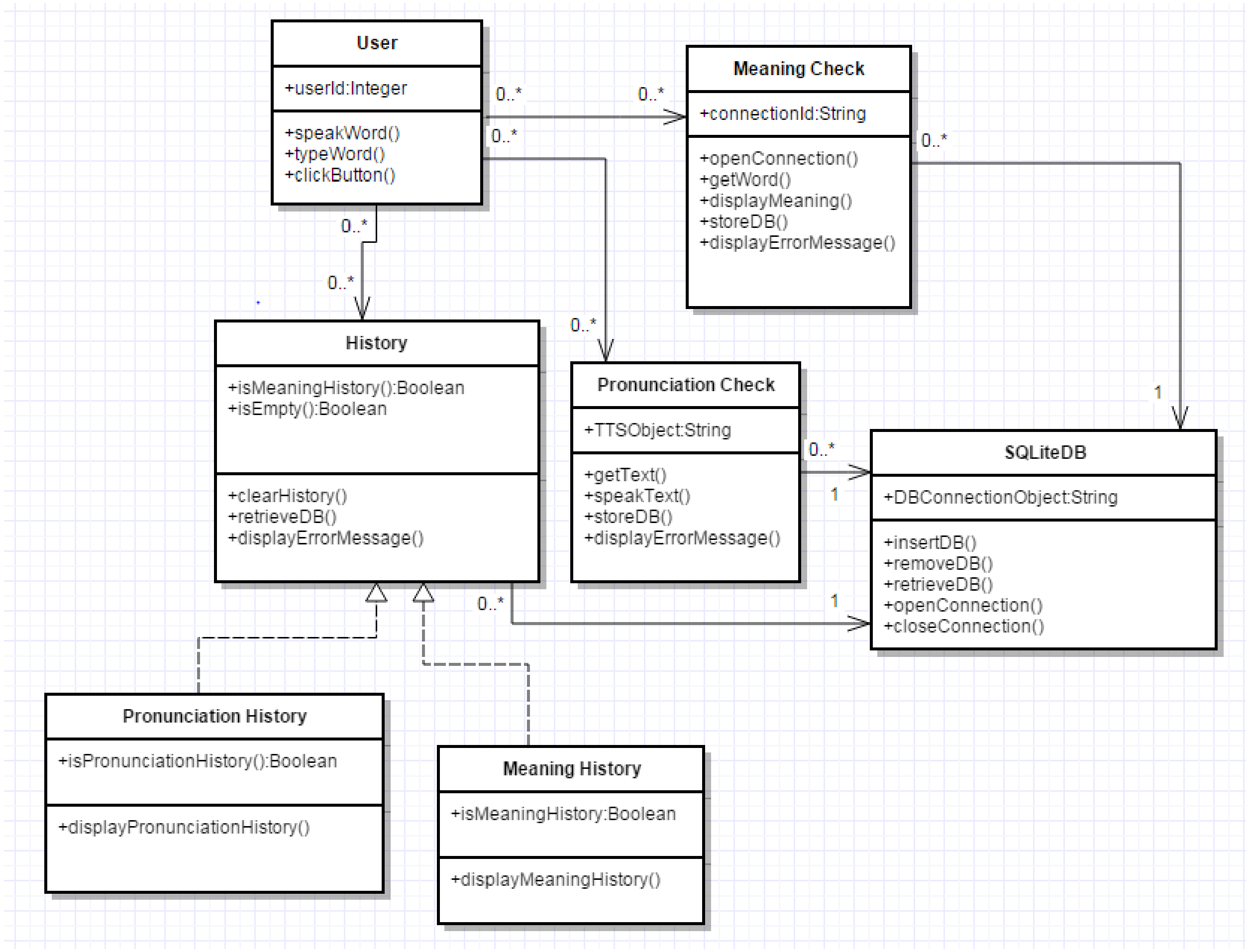


Figure 3 Class Diagram

5.3 Sequence Diagram

The sequence diagrams describe behavior as a sequence of messages exchanged among a set of objects. It is an interaction diagram that emphasizes the time ordering of messages. Graphically, a sequence diagram is a table that shows objects arranged along the X axis and messages, ordered in increasing time, along the Y axis. The main components of the sequence diagrams are-

Objects

Objects are anonymous instances of classes. They may also refer to instances of other things such as components, collaboration and nodes.

Links

A link is a semantic connection among objects.

Messages

A message is a specification of a communication between objects that conveys the information with the expectation that the activity will ensue.

The sequence diagram for the Vocal Combo contains the sequence of messages exchanged between the objects on the time axis.

Sequence Diagram for Vocal Combo

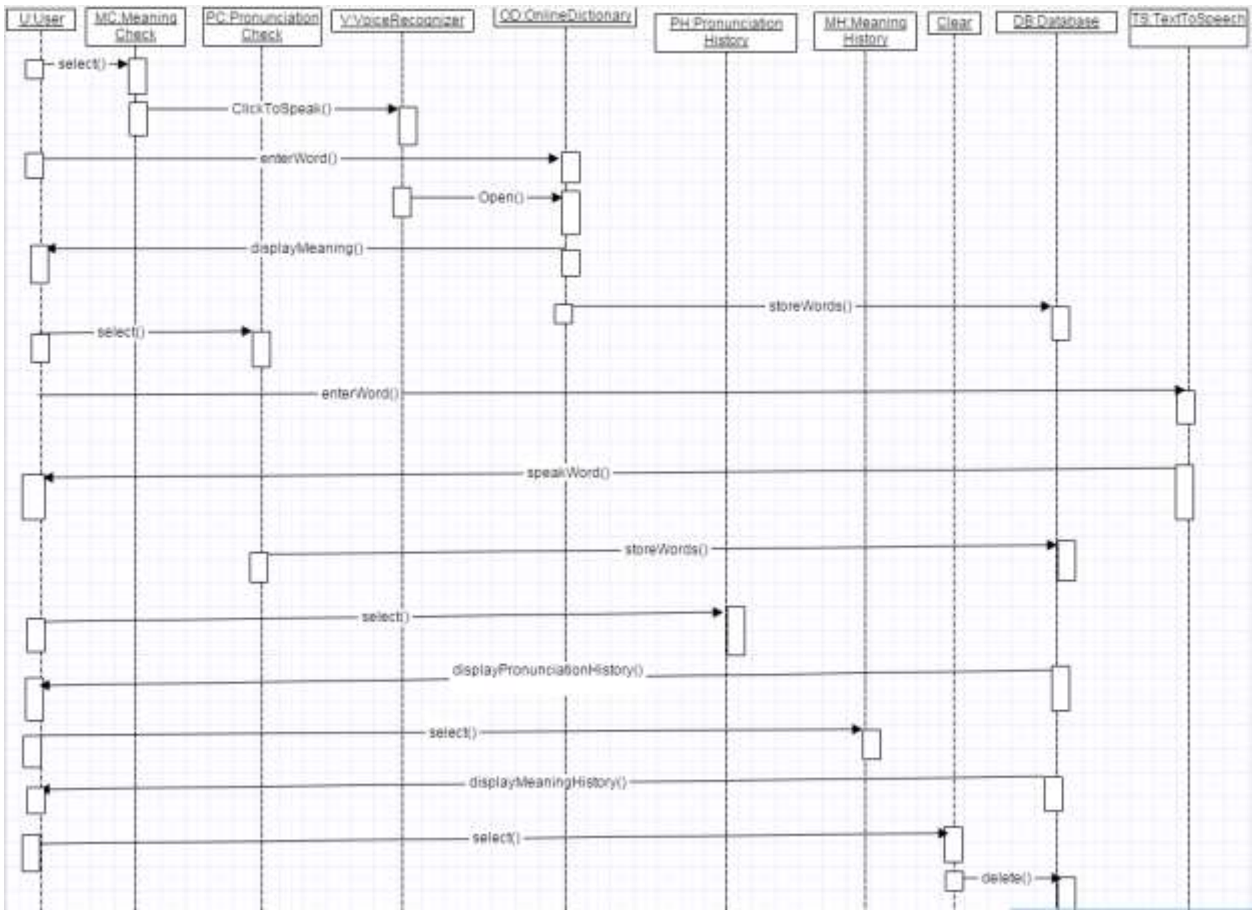


Figure 4 Sequence Diagram

5.4 State Chart Diagram

These diagrams describe the behavior of the non-trivial classes in our project. A state represents a collection of values for the object. A transition triggers a change in them. The following tools are used on the state chart diagram toolbox to model state chart diagrams:

State: A state represents a condition or situation during the life of an object during which it satisfies some conditions or waits for some events.

Transitions: A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied.

State Chart diagram for Vocal Combo

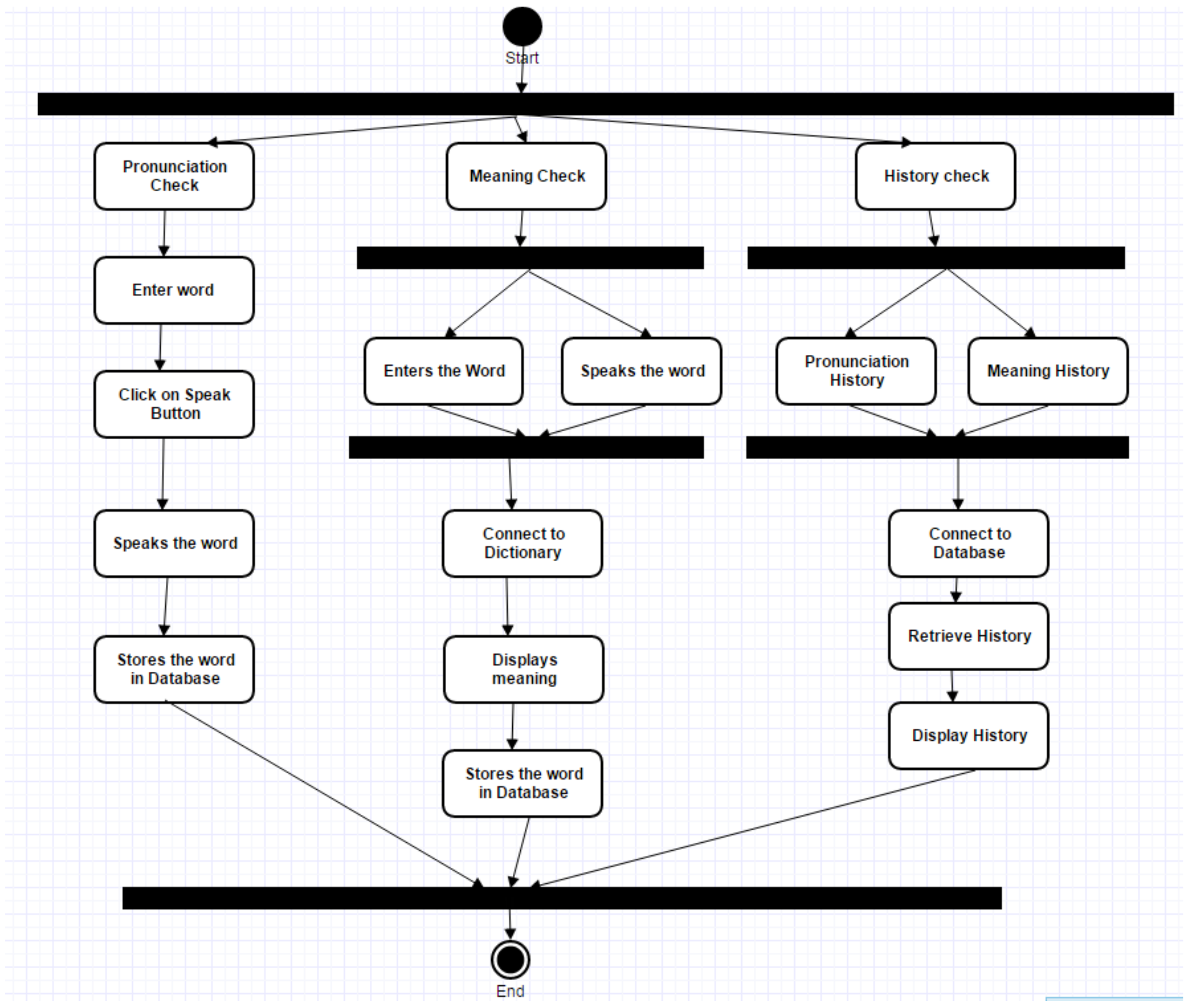


Figure 5 State Chart Diagram

5.5 Activity Diagram

Activity diagrams describe the workflow behavior of the system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by show activities that are conditional or parallel.

A fork is used when multiple activities are occurring at the same time. The branch describes what activities will take place based on a set of conditions. All branches at the same point are followed by a merge to indicate the end of the conditional behavior started by that branch.
The following tools are used on the activity diagram toolbox to model activity diagrams:

- Decisions: A decision represents a specific location in activity diagram where the work flow may branch based upon guard conditions.
- States: A state represents a condition or a situation during the life of an object during which it satisfies some condition or waits for some event.
- Transactions: A state transition indicates that an object in the source state will perform certain specified event occurs or when certain conditions are satisfied.
- Start states: A start state (also called initial state) explicitly shows the beginning of a work flow.
- End states: An end state represents a final state or terminal state.

Activity Diagram for Vocal Combo

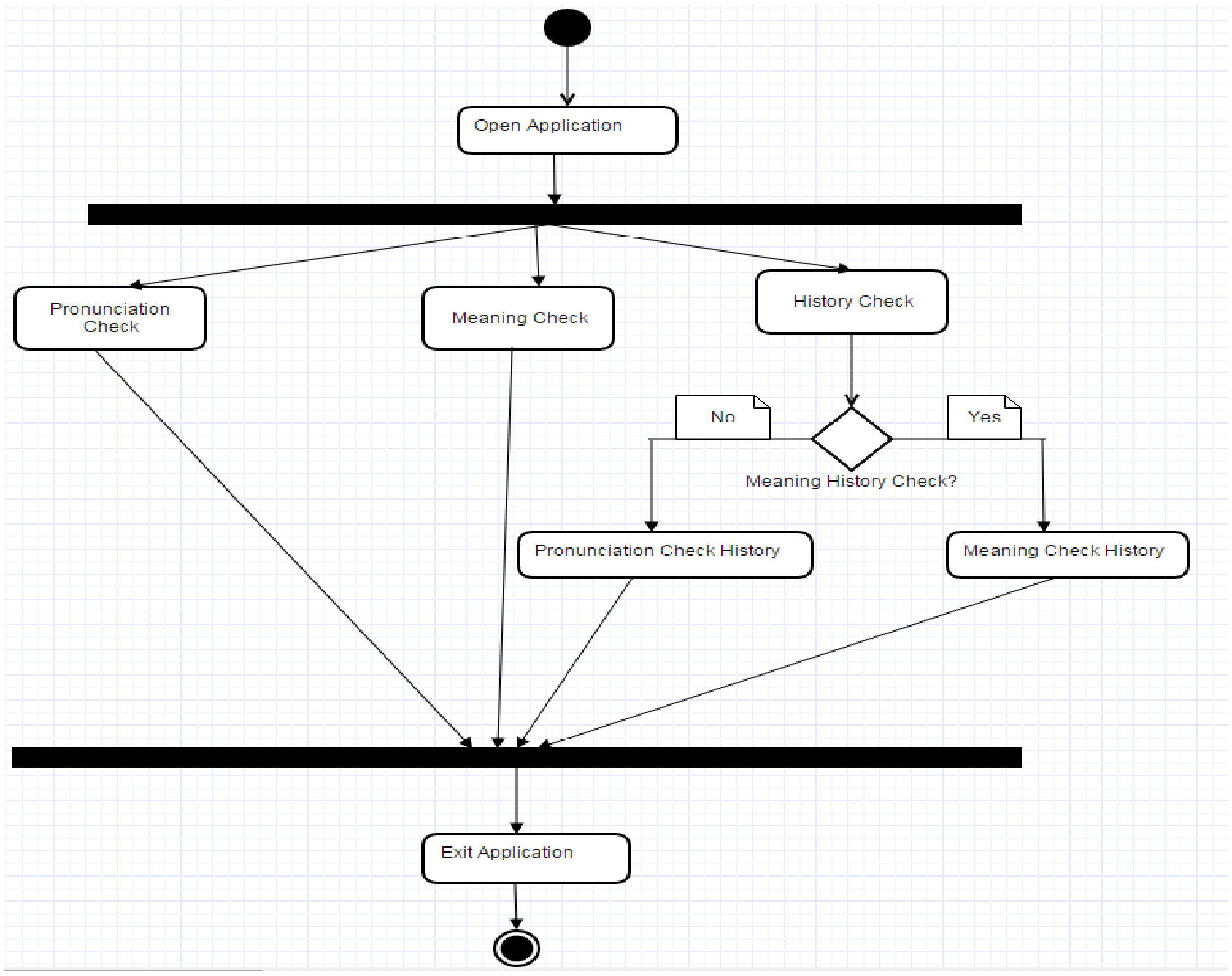


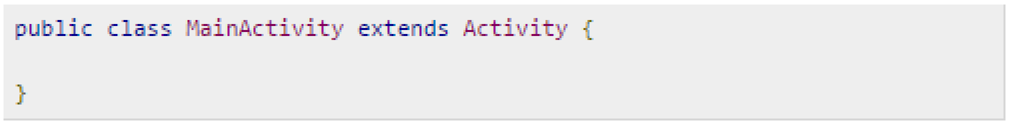
Figure 6 Activity Diagram

Chapter 6 - Android Application Components

Before actually proceeding with discussing the programming involved in developing this application, a brief introduction to the application components which plays a crucial role in developing any Android application is necessary. Application components are the building blocks of any Android application [5]. Major Android application components are as follows.

6.1 Activities

An activity in Android represents a single screen with user interface. An Android application typically consists of multiple activities that are loosely bound together. Among several activities for the application, only one activity is represented as Main Activity which is displayed while launching the application. Every time an activity starts, the previous activity, if any, is stopped but the data is saved. Every activity is implemented as a subclass of Activity class as follows-



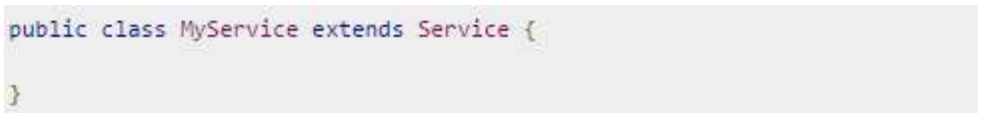
```
public class MainActivity extends Activity {

}
```

Vocal combo consists of 6 activity classes-

- MainActivity- This is the activity which is displayed upon starting the application. This has the buttons for each functionality of the system, pronunciation, meaning, history check, help and exit.
- SpeakingAndroid- This activity is responsible for the pronunciation check. When the user enters the text (word or sentence) for which he wants to check the pronunciation for and clicks the speak button, the application pronounces it for the user.
- Sec- This activity is responsible for the meaning check. When the user either types in the word or speaks out the word, this activity is responsible for providing him the meaning check for the word.
- Meaning- This activity is responsible for connecting to the online dictionary to provide the user the meaning for the word which he checks for.
- HistoryPage- This activity is responsible for providing the pronunciation check history and meaning check history to the user. The user can clear the history at any point of time.
- HelpScr- This activity displays the help screen which helps the user to navigate through the application.

6.2 Services

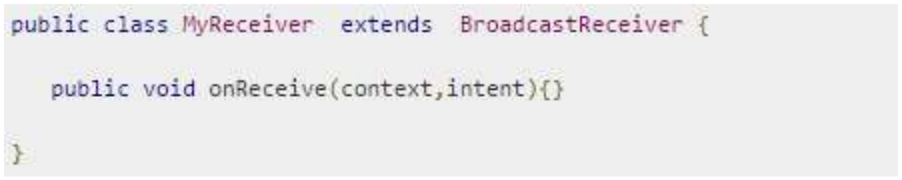
Service is a component that does not require user interface and are invoked by an activity with user invocation. Services performs operation without user interaction in the background. They do not have an independent thread but make use of the main thread from hosting process. Every service is implemented as a subclass of Service class as follows-



```
public class MyService extends Service {

}
```

6.3 Broadcast Receivers

Broadcast receiver responds to broadcast messages from various applications or the system. It is a component where the application user can register for system or application events like notification when the battery is low. Applications initiate the broadcasts to let other related applications or the system know that the data that is downloaded to the system and is available for use in the required applications. The broadcast receiver can statically registered via AndroidManifest.xml. The broadcast receiver extends BroadcastReceiver abstract class, requiring the developers to implement the onReceive() method of this base class as follows-

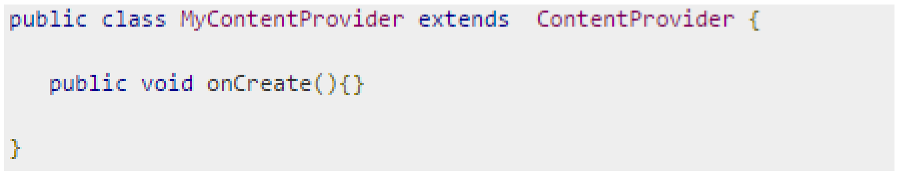


```
public class MyReceiver  extends  BroadcastReceiver {

   public void onReceive(context,intent){}

}
```

Content is used to start services or activities and intent is the object with the action used to register the receiver.

6.4 Content Providers

Generally, the data associated with any Android application is stored in SQLite database, which is embedded in Android. Content providers in Android facilitate the sharing of data among multiple applications which have the necessary permissions. The applications which requires to access or modify the data stored across different applications, needs to query the centralized repository through the Content Providers [6]. Every Content Provider is implemented as a subclass of the ContentProvider class. It must implement a standard set of APIs that enable other applications to perform transactions.



```
public class MyContentProvider extends  ContentProvider {

   public void onCreate(){}

}
```

6.5 Intents

Intents are used to provide the navigation between the activities. It provides the facility for performing late runtime binding between the codes in different applications. It is the abstract description of an operation to be performed. It can be used with-

- startActivity- to launch an Activity
- broadcastIntent- to send it to any interested BroadcastReceiver components
- startService(Intent) or bindService(Intent, ServiceConnection, int) to communicate with a background service.

Chapter 7 - Database Design

SQLite is an open source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation [7]. SQLite supports all the relational database features. To manage our own databases, we are provided with the classes from Android.database.sqlite package.

In this application development, we have used the SQLite database to store the words the user has checked the meaning/ pronunciation for. I have created two separate tables in SQLite database, Meanings, to store the words for which the user have searched the meanings for and Pronunciations, to store the words for which the user have searched the pronunciations for. When the user wants to check the Meaning/Pronunciation history, the application query the database to retrieve the data from the respective tables using cursors. The retrieved information is then displayed on the screens.

The user can choose to clear the history, in which case the pronunciation history and the meaning history are deleted from the database.

Chapter 8 - Implementation

Vocal Combo is designed as an Android application which helps the user as Voice dictionary by providing the functionalities of pronunciation and meaning check.
I have implemented the Vocal Combo in 4 modules-

- Pronunciation check
- Meaning check
- History check
- Help
- Exit

8.1 Pronunciation Check

This module manages to provide the pronunciation of the word/sentence when the user types in the text box and hits the speak button. I have used the TextToSpeech class of Android API to implement this functionality. TextToSpeech class synthesizes speech from text entered by the user for immediate playback. The text the user enters to check pronunciation is inserted into the database table, Pronunciation, for user reference if he/she wants to check the Pronunciation check history. I have used setLanguage method to set the language of TTS engine and speak method of TextToSpeech class to speak the text entered by the user.

myTTS.setLanguage(Locale.US)

myTTS.speak(speech, TextToSpeech.QUEUE_FLUSH, null)

8.2 Meaning Check

This module can further be divided into 2 sub modules depending on how the user give input to it. If the user types in the word for which he wants to check the meaning for, the word is stored in the array list before it is passed to the Meaning activity. If the user chooses to speak the text, on clicking the click to speak button, the voice recognition activity is enabled. To handle the results from the voice recognition activity, we use a class called RecognizerIntent [8] provided by the Android API. We make use of the following constants for supporting the speech recognition through starting an Intent.

- ACTION_RECOGNIZE_SPEECH- Starts an activity that will prompt the user for speech and send it through a speech recognizer.

- EXTRA_LANGUAGE_MODEL- Informs the recognizer which speech model to prefer when performing ACTION_RECOGNIZE_SPEECH.
- EXTRA_PROMPT- Optional text prompt to show to the user when asking them to speak.

When the application gets the user input, it stores it in the database table, Meaning, for user reference if he/she wants to check the Meaning check history. To get the meaning of the text, we use a class called WebView, which is a view that displays web pages. This class helps us to display the online content within the activity. In order for the Activity to access the Internet and load web pages in a WebView, we have added the INTERNET permissions to the Android Manifest file:

```
<uses-permission android:name="android.permission.INTERNET" />
```

I have used the http://www.thefreedictionary.com/dict.aspx?word= to provide the meaning for the text the user enters.

8.3 History Check

This module displays both the Pronunciation and Meaning history. Whenever the user types in a word to check the Pronunciation or Meaning, the words are stored in the Pronunciations or Meaning tables of SQLite database respectively. When the user hits the Pronunciation check button on the history page, the application retrieves the data from the Pronunciations table using cursors and displays them. When the user hits the Meaning check button on the history page, the application retrieves the data from the Meanings table using cursors and displays them. The user can choose to clear the history by clicking the clear button, all the data in the Pronunciations and Meaning table will be deleted and hence the query retrieves no data items to be displayed.

I have used following components to implement the history check-

- *ArrayAdapter*- Pulls data stored in array list after retrieving from the respective SQLite database tables.
- *ListView*- Displays data pulled by the ArrayAdapter in vertical scrollable list.

8.4 Help Module

After installing the application, if the user want the support regarding how to use the application or regarding navigating through the pages, the user can click the Help button on the home screen.

8.5 Exit Module

Once the user is done with using the application, he may choose to exit it. When he clicks the Exit button on the home page, all the data and operations will be saved and the application is exited.

Chapter 9 - Graphical User Interface

I made sure that the user interactive screens are easily understandable and the navigation through the application is very obvious. The screens of the application are discussed in detail in this chapter.

9.1 Splash Screen

The splash screen appears on the start of the application. This screen will display the different functionalities, Pronunciation check, Meaning check, History check, Help and Exit, available for the user. When the user clicks the button for required functionality, they will be navigated to the respective screens.

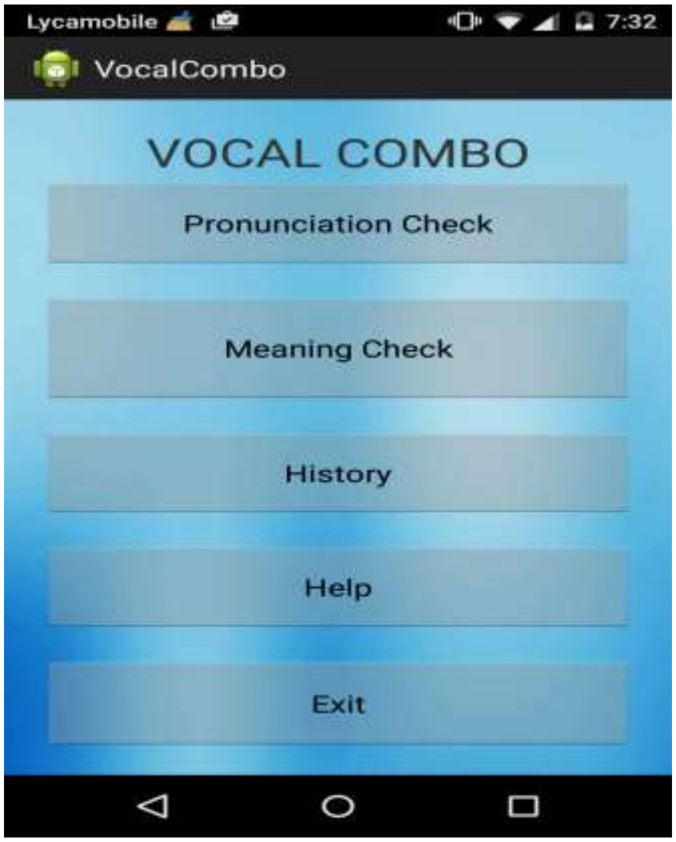


Figure 7 Splash screen

9.2 Pronunciation Check Screen

The pronunciation check screen provides the user text box, where the user types in the text for which he have to check the pronunciation for. When he clicks the speak button, the applications speaks the text for him.

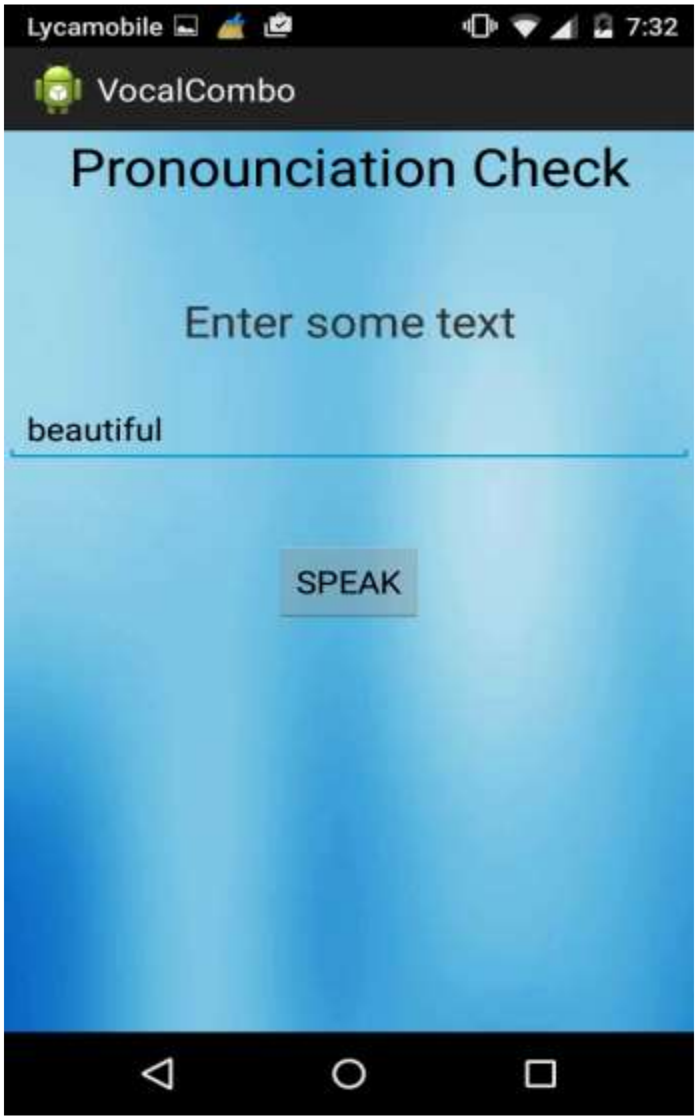


Figure 8 Pronunciation Check Screen

9.3 Meaning Check Screen

If the user want to check the meaning for a particular word, he can input the word to the application in one of the two ways-

- Type in the word in the text box and click on the Get Meaning button

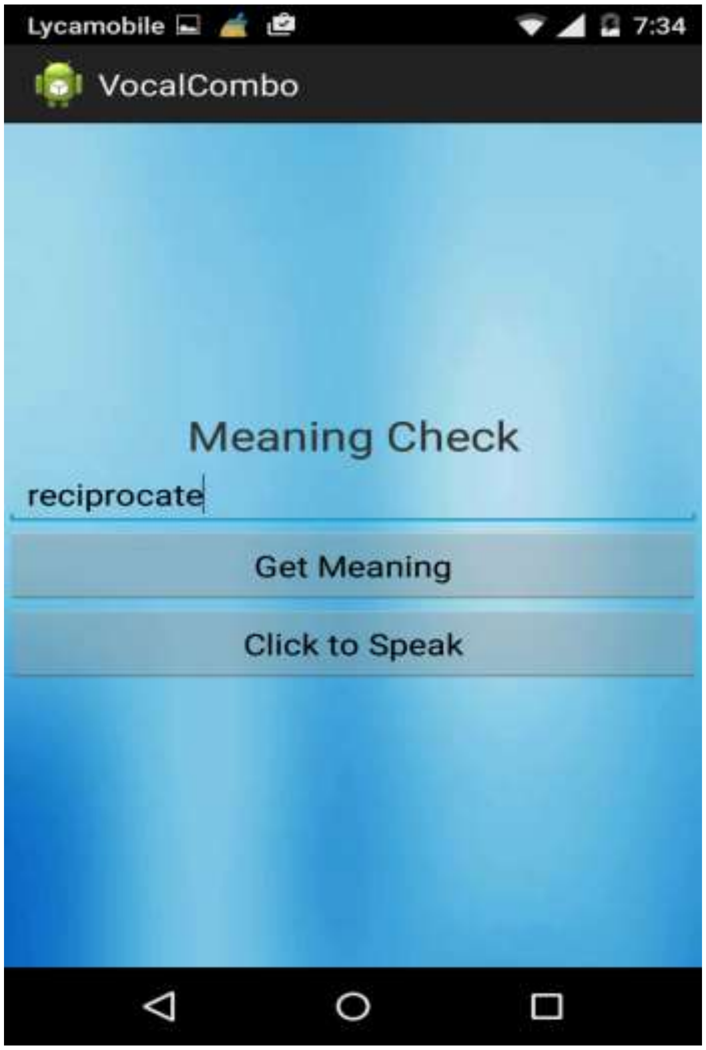


Figure 9 Meaning Check

- If the user wishes to speak out the word for which he wants to check the meaning for, he then clicks on Click to Speak button. The voice recognizer pops up to capture the word the user speaks out.

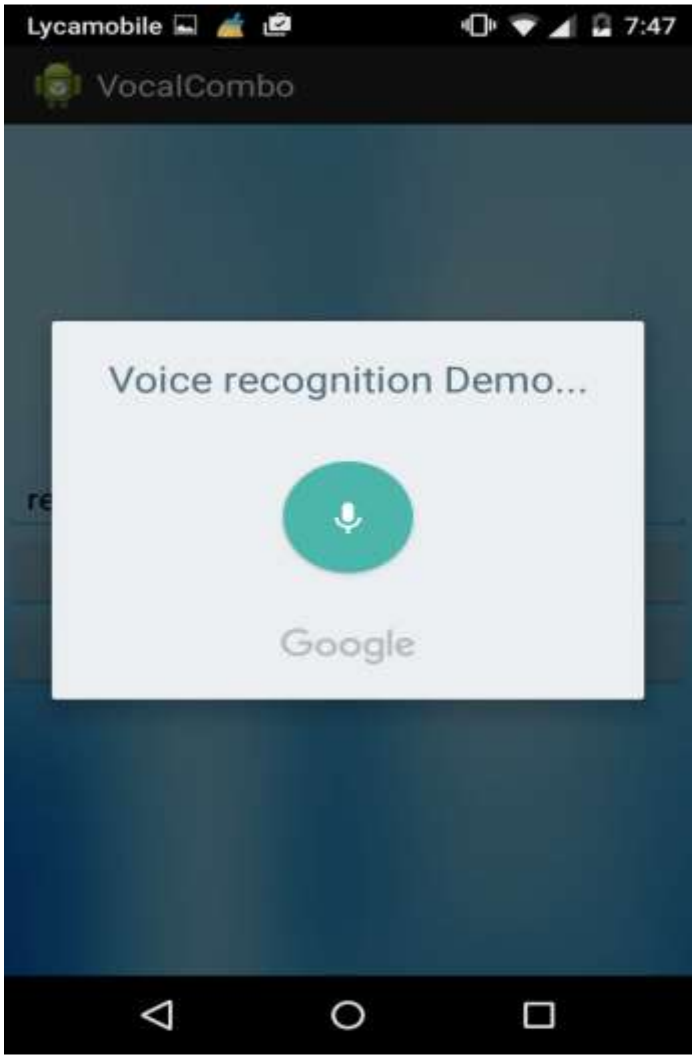
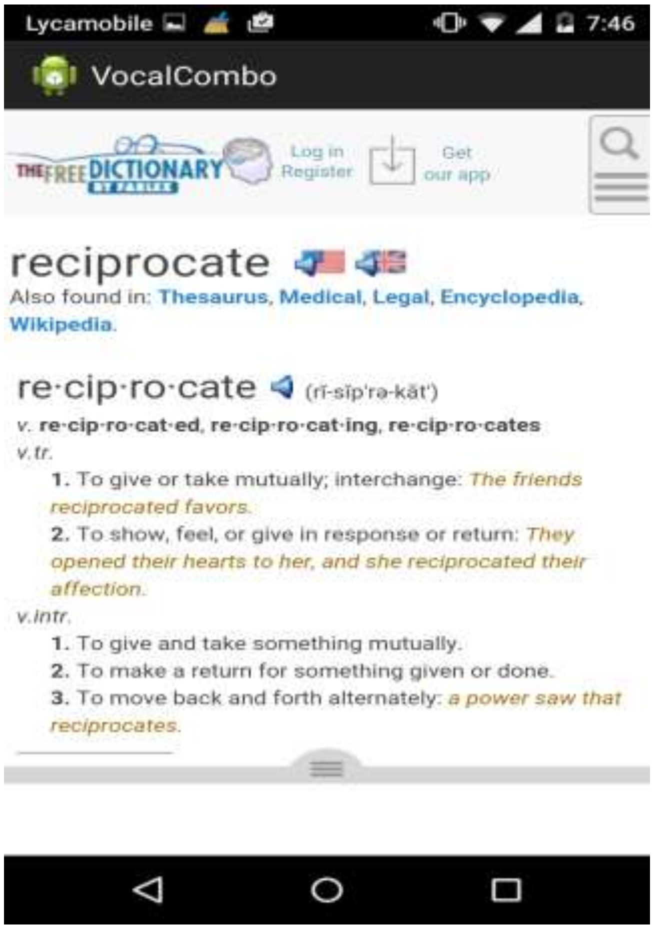


Figure 10 Voice Recognition Demo

- The application then opens the online dictionary to display the meaning of the word.



reciprocate

Also found in: Thesaurus, Medical, Legal, Encyclopedia, Wikipedia.

re·cip·ro·cate (rĭ-sĭp′rə-kāt′)

v. **re·cip·ro·cat·ed, re·cip·ro·cat·ing, re·cip·ro·cates**

v.tr.

1. To give or take mutually; interchange: *The friends reciprocated favors.*

2. To show, feel, or give in response or return: *They opened their hearts to her, and she reciprocated their affection.*

v.intr.

1. To give and take something mutually.

2. To make a return for something given or done.

3. To move back and forth alternately: *a power saw that reciprocates.*

Figure 11 Online Dictionary

9.4 History Screen

This history page displays the check history of the user.

- If the user wants to check the Pronunciation History, he clicks on the Pronunciation History button which redirects to the screen which displays the words the user have checked the pronunciation for.

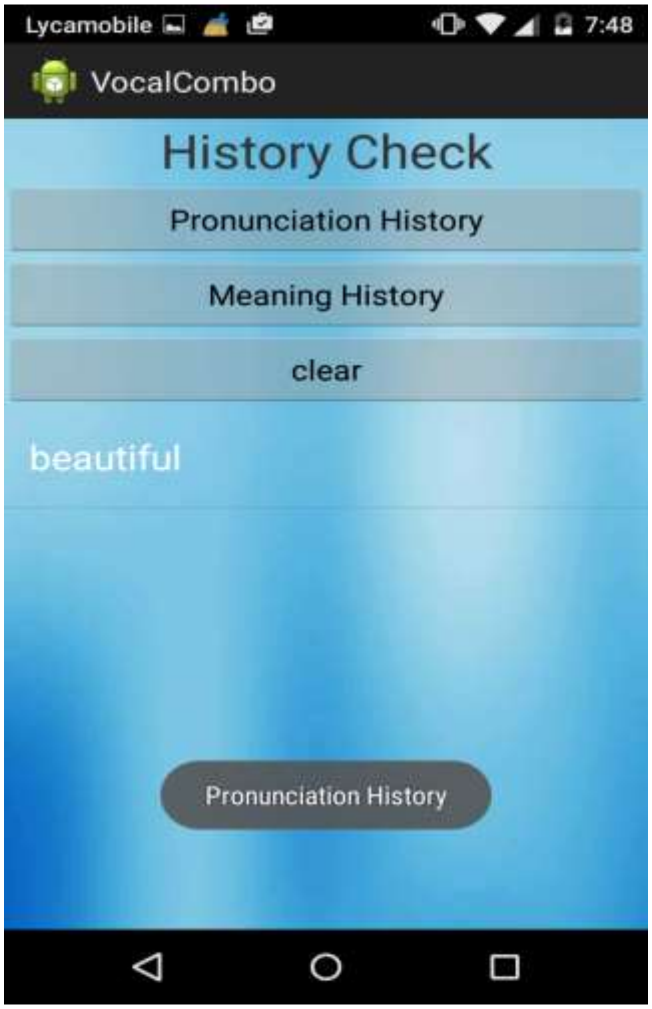


Figure 12 Pronunciation Check History

- If the user wants to check the Meaning History, he clicks on the Meaning History button which redirects to the screen which displays the words the user have checked the meaning for.

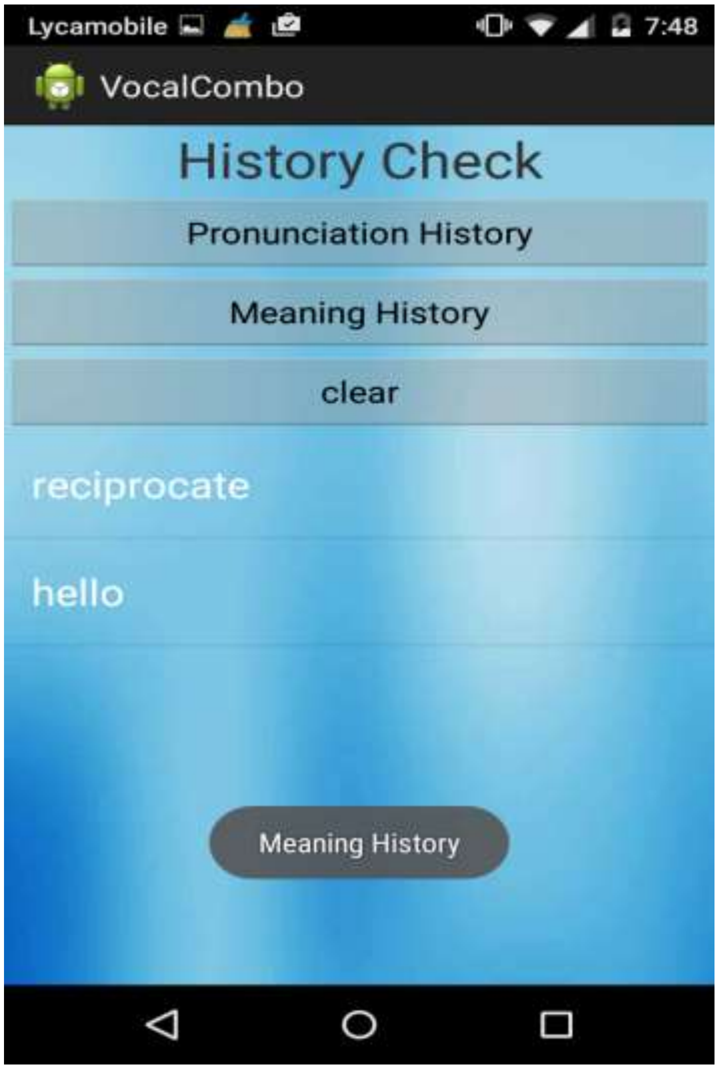
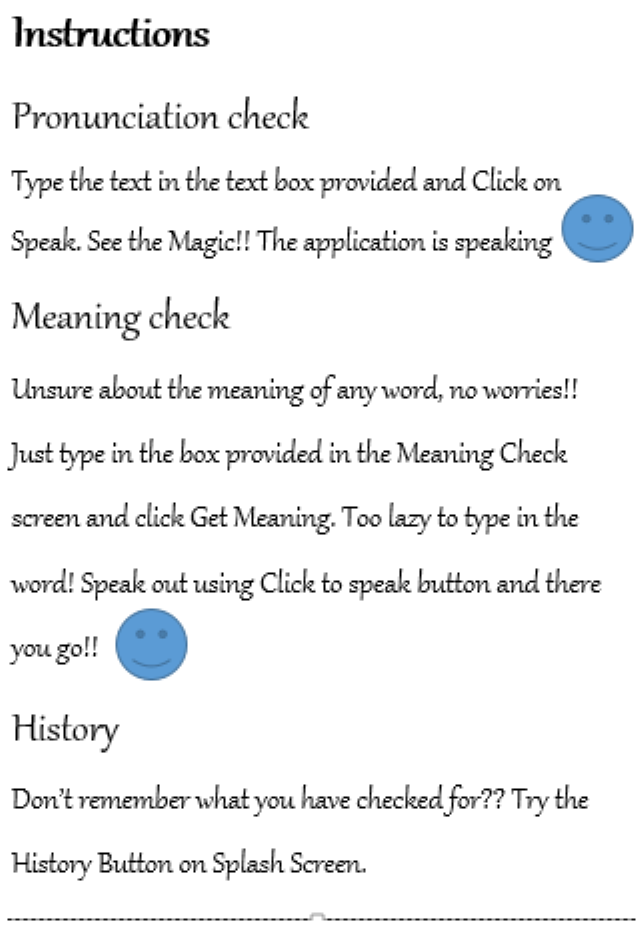


Figure 13 Meaning Check History

- In case the user wants to clear the check history, he clicks the clear button which deletes the check history and displays the message.

9.5 Help Screen

The help screen helps the user to navigate through the application.



Instructions

Pronunciation check

Type the text in the text box provided and Click on Speak. See the Magic!! The application is speaking

Meaning check

Unsure about the meaning of any word, no worries!! Just type in the box provided in the Meaning Check screen and click Get Meaning. Too lazy to type in the word! Speak out using Click to speak button and there you go!!

History

Don't remember what you have checked for?? Try the History Button on Splash Screen.

Figure 14 Help Screen

Chapter 10 - Testing

A primary purpose of testing is to detect software failures so that defects may be uncovered and corrected. This is a non-trivial pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of present software testing includes examination of code as well as execution of that code in various conditions as well as examining the quality aspects of code: does it do what it is supposed to do and do what it needs to do.

10.1 Testing Levels

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test.

- Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level.
- Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or altogether.
- System testing is a completely integrated system to verify that it meets its requirements.
- System integration testing verifies that a system is integrated to any external or third party systems defined in the system requirements.
- Regression testing tests new functionality in a program. It is done by running all of the previous unit tests written for a program, if they all pass, then the new functionality is added to the code base.
- Acceptance testing is conducted by a user to verify that the system meets the acceptance criteria.

10.2 Test Cases

In general, a test case is a set of test data and test programs and their expected results. A test case in software engineering normally consists of unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output and it validates one or more system requirements and generates pass or fail. I have tested the application on Samsung galaxy tab, whose results are summarized as below-

10.2.1 Test Case 1

Test Objectives: Navigation from Splash screen to Pronunciation check screen

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is currently on the Splash screen	User clicks the Pronunciation Check button	Directs to Pronunciation Check screen	PASS

Table 1-Test Case for navigation to pronunciation check screen

10.2.2 Test Case 2

Test Objectives: Navigation from Splash screen to Meaning check screen

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is currently on the Splash screen	User clicks the Meaning Check button	Directs to Meaning Check screen	PASS

Table 2-Test Case for navigation to meaning check screen

10.2.3 Test Case 3

Test Objectives: Navigation from Splash screen to History check screen

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is currently on the Splash screen	User clicks the History button	Directs to History Check screen	PASS

Table 3-Test Case for navigation to history check screen

10.2.4 Test Case 4

Test Objectives: User checks for Pronunciation

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is currently on the Pronunciation check page	User types in the word in the text box provided and clicks on Speak button	1. The application speaks the text for the user	PASS
	User does not type any text in the text box, but still clicks on the Speak button.	2. The application displays the error message, “Please Enter Word”.	PASS

Table 4-Test Case for pronunciation check functionality

10.2.5 Test Case 5

Test Objectives: User checks for Meaning by typing the input

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is currently on the Meaning check page	User types the word in the text box provided and clicks on Get Meaning button	1. The user is directed to a web page which displays the meaning of the word entered by the user.	PASS
	User does not type any word in the text box provided and clicks on Get Meaning button	2. The application displays the error message, “Please Enter Word”	PASS

Table 5-Test Case for meaning check functionality when user types input

10.2.6 Test Case 6

Test Objectives: User checks for Meaning by speaking the input

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is currently on the Meaning check page	User clicks on ‘Click to Speak’ button, speaks a valid word	1. A voice recognition box pops up and the user is directed to a web page which displays the meaning of the word entered by the user.	PASS
	User clicks on ‘Click to Speak’ button, speaks an invalid or does not speak any word	2. A voice recognition box pops up and the error message of “No Matches found, Speak Again” message is displayed.	PASS

Table 6-Test Case for** meaning **check functionality when user speaks the input

10.2.7 Test Case 7

Test Objectives: User checks for Pronunciation History

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is navigated from the Splash screen by clicking the	User clicks on ‘Pronunciation History’ button and the history is not cleared before-hand	1. The text words for which the user have checked the pronunciation for is displayed.	PASS

History button	User clicks on 'Pronunciation History' button and the history is cleared before-hand	2. An error message stating that the History is Empty is displayed.	PASS

Table 7-Test Case for pronunciation history check functionality

10.2.8 Test Case 8

Test Objectives: User checks for Meaning History

TEST CONDITION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/ FAIL
The user is navigated from the Splash screen by clicking the History button	User clicks on 'Meaning History' button and the history is not cleared before-hand	1. The text words for which the user have checked the meanings for is displayed.	PASS
	User clicks on 'Meaning History' button and the history is cleared before-hand	2. An error message stating that the History is Empty is displayed.	PASS

Table 8-Test Case for meaning history check functionality

10.2.9 Test Case 9

Test Objectives: Navigation from Splash screen to Help Screen

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is currently on the Splash screen	User clicks the Help button	Directs to Help screen	PASS

Table 9-Test Case for navigation to help screen

10.2.9 Test Case 9

Test Objectives: Exiting the application

TEST CONDITION	**INPUT SPECIFICATION**	**OUTPUT SPECIFICATION**	**PASS/ FAIL**
The user is currently on the Splash screen	User clicks the Exit button	Exits the application	PASS

Table 10-Test Case for exiting the application

Chapter 11 - Performance Profiling

Android Studio and the mobile device I have used provide profiling tools to record and visualize the rendering, compute memory and battery performance of the application [9].

11.1 Rendering Analysis Tools

11.1.1 Debug GPU Overdraw

The Debug GPU Overdraw shows how to visualize overdraw on the mobile device by color-coding interface elements based on how often they are drawn underneath. This helps in recognizing where the application might be doing more rendering work than necessary and hence can help in reducing rendering overhead. I did turn on the Debug GPU Overdraw option in developer options on the mobile device in which I have installed this application. The colors on the screen hints the amount of overdraw on the screen for each pixel.

Color	- **Overdraw**
True Color	- No overdraw
Blue	- Overdrawn once
Green	- Overdrawn twice
Pink	- Overdrawn thrice
Red	- Overdrawn four or more times

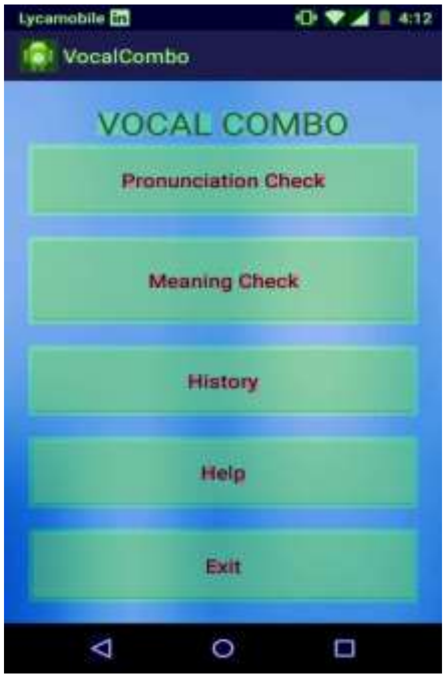


Figure 15 Debug GPU Overdraw for Splash Screen

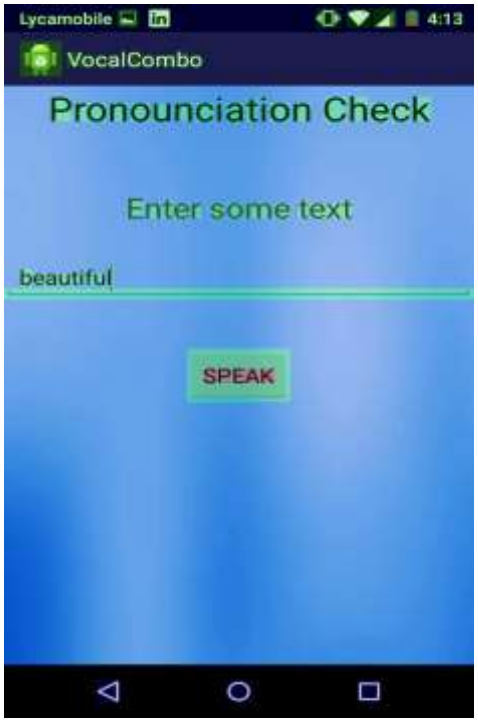


Figure 16 Debug GPU Overdraw for Pronunciation Check

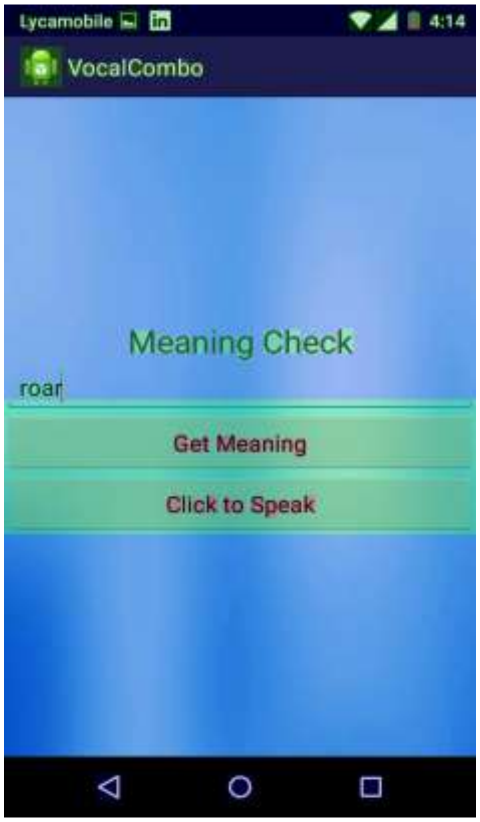


Figure 17 Debug GPU Overdraw for Meaning Check

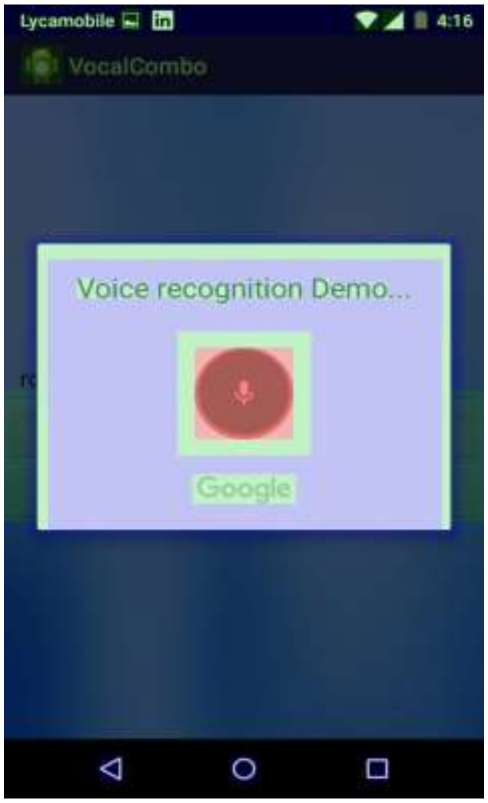


Figure 18 Debug GPU Overdraw for Voice Recognizer

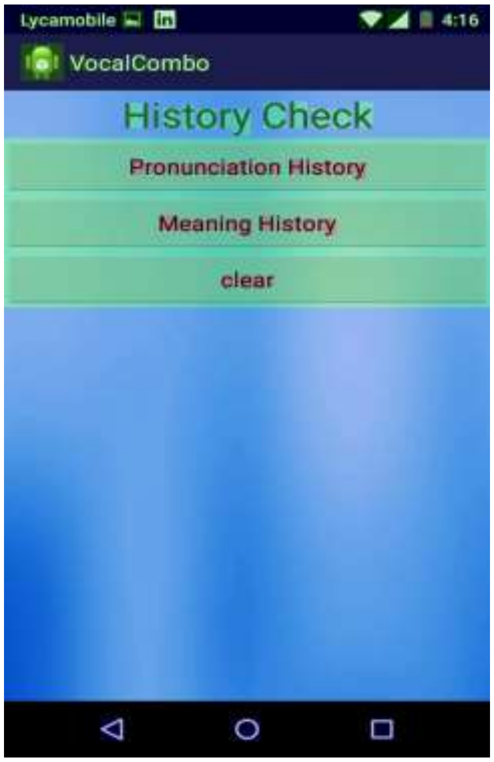


Figure 19 Debug GPU Overdraw for History Check

The Vocal Combo application has proved to have less GPU overdraw as there is no pixel with color red and a very few with color pink. There are few pixels with the green color implying the pixels are overdrawn twice which is common and cannot be avoided. The rest of the pixels on the screens are of the true color implying no overdraw.

11.1.2 Profiling GPU Rendering

Profile GPU Rendering gives a quick visual representation of how much time it takes to render the frames of UI window relative to the 16-ms-per-frame benchmark [10]. It helps us estimate the UI performance against the 16-ms-per-frame target and helps us finding the spikes in frame rendering time associated with user or program actions. To enable this tool, I did turn on the Profile GPU Rendering and I have chosen the On Screen as bars to overlay the graphs on the screen of the mobile device.

The tool displays a graph with the horizontal axis showing time elapsing, and the vertical axis showing time per frame in milliseconds. The vertical bars shown up on the screen, appearing from left to right, graphs frame performance over time. Each vertical bar represents one frame of rendering. The green line marks the 16 millisecond target. Every time a frame crosses the green line, the application is missing a frame, and there might be a pause in animations. The graph has colored sections representing the phase of the rendering pipeline.

- The **green** line represents 16 milliseconds. Any time a bar pushes above this line, there may be pauses in the animations.
- The **blue** section of the bar represents the time used to create and update the View's display lists.
- The **purple** section of the bar represents the time spent transferring resources to the render thread.
- The **red** section of the bar represents the time spent by Android's 2D renderer issuing commands to OpenGL to draw and redraw display lists.
- The **orange** section of the bar represents the time the CPU is waiting for the GPU to finish its work.

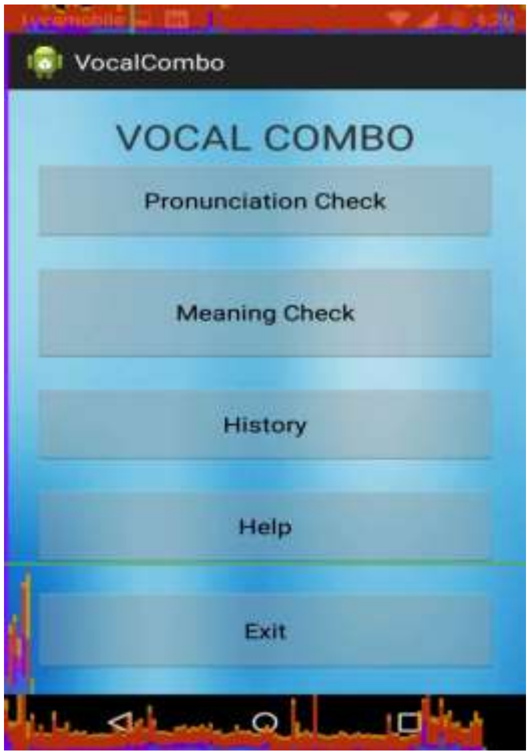


Figure 20 Profile GPU Rendering for Splash Screen

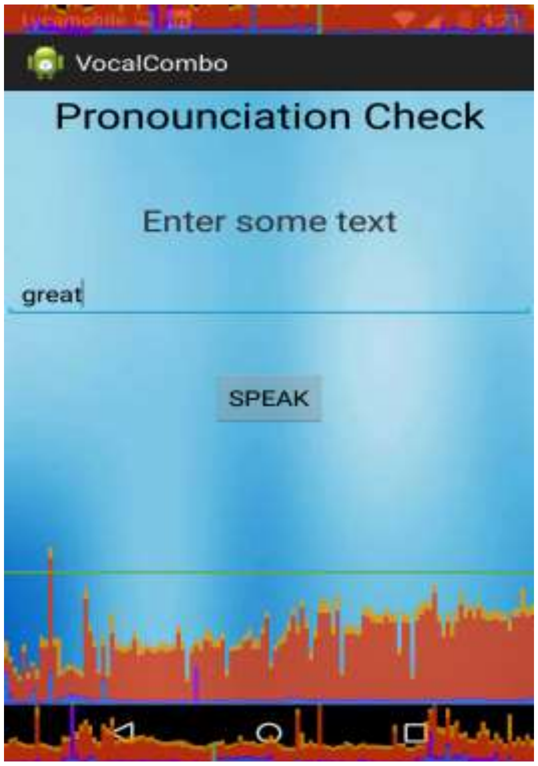


Figure 21 Profile GPU Rendering for Pronunciation Check

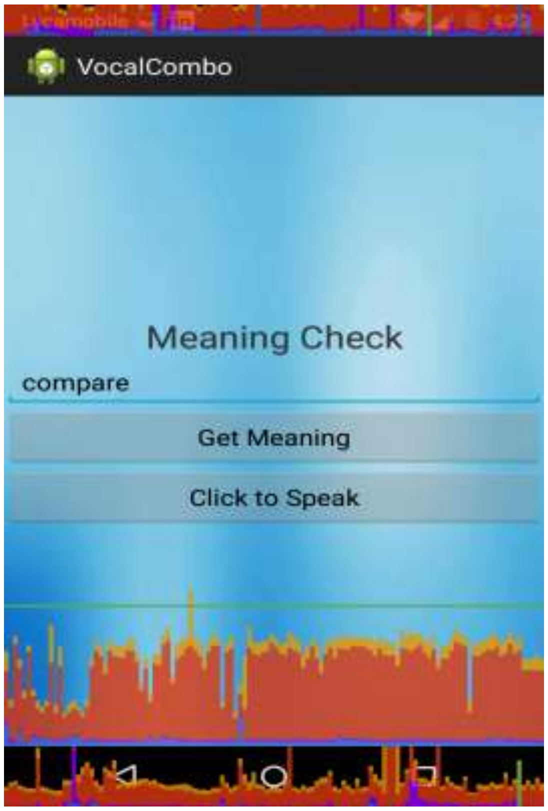
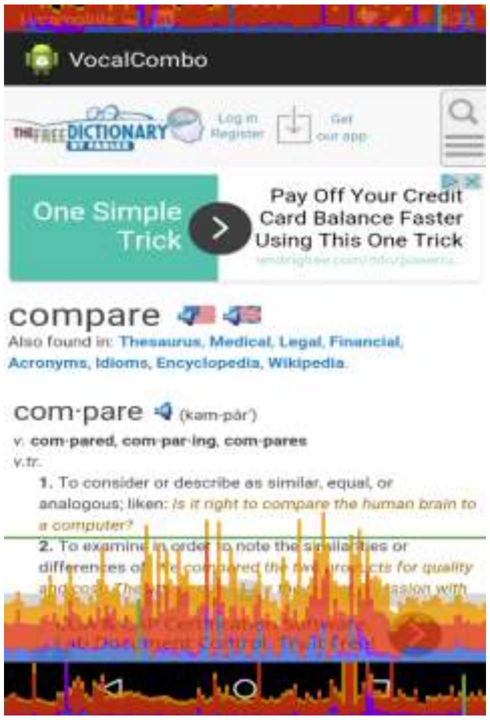


Figure 22 Profile GPU Rendering for Meaning Check



compare
Also found in: Thesaurus, Medical, Legal, Financial, Acronyms, Idioms, Encyclopedia, Wikipedia.

com·pare (kəm-pâr′)
v. com·pared, com·par·ing, com·pares
v.tr.
1. To consider or describe as similar, equal, or analogous; liken: *Is it right to compare the human brain to a computer?*
2. To examine in order to note the similarities or differences of: *We compared the two products for quality and cost.*

Figure 23 Profile GPU Rendering for Online Dictionary

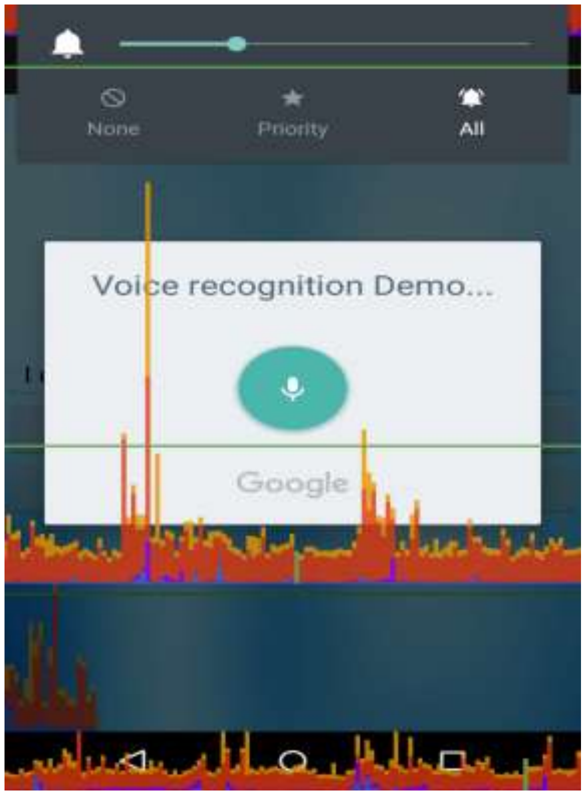


Figure 24 Profile GPU Rendering for Voice Recognizer

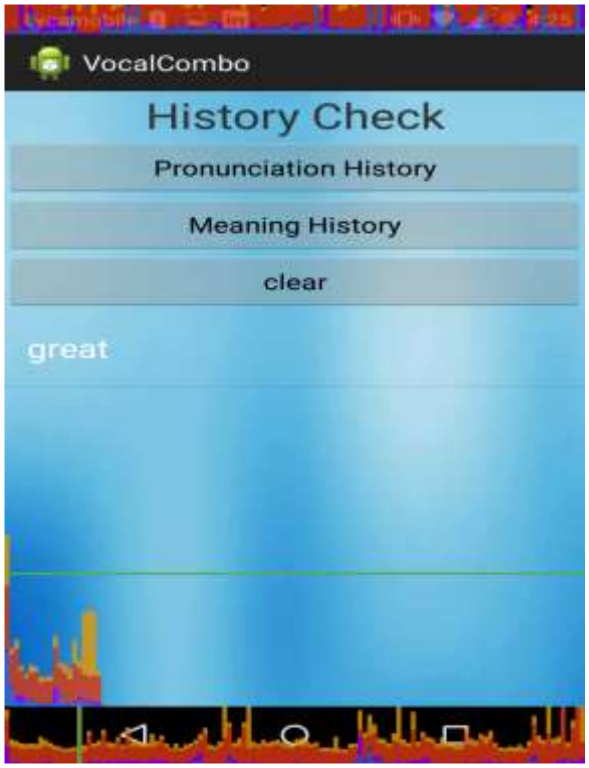


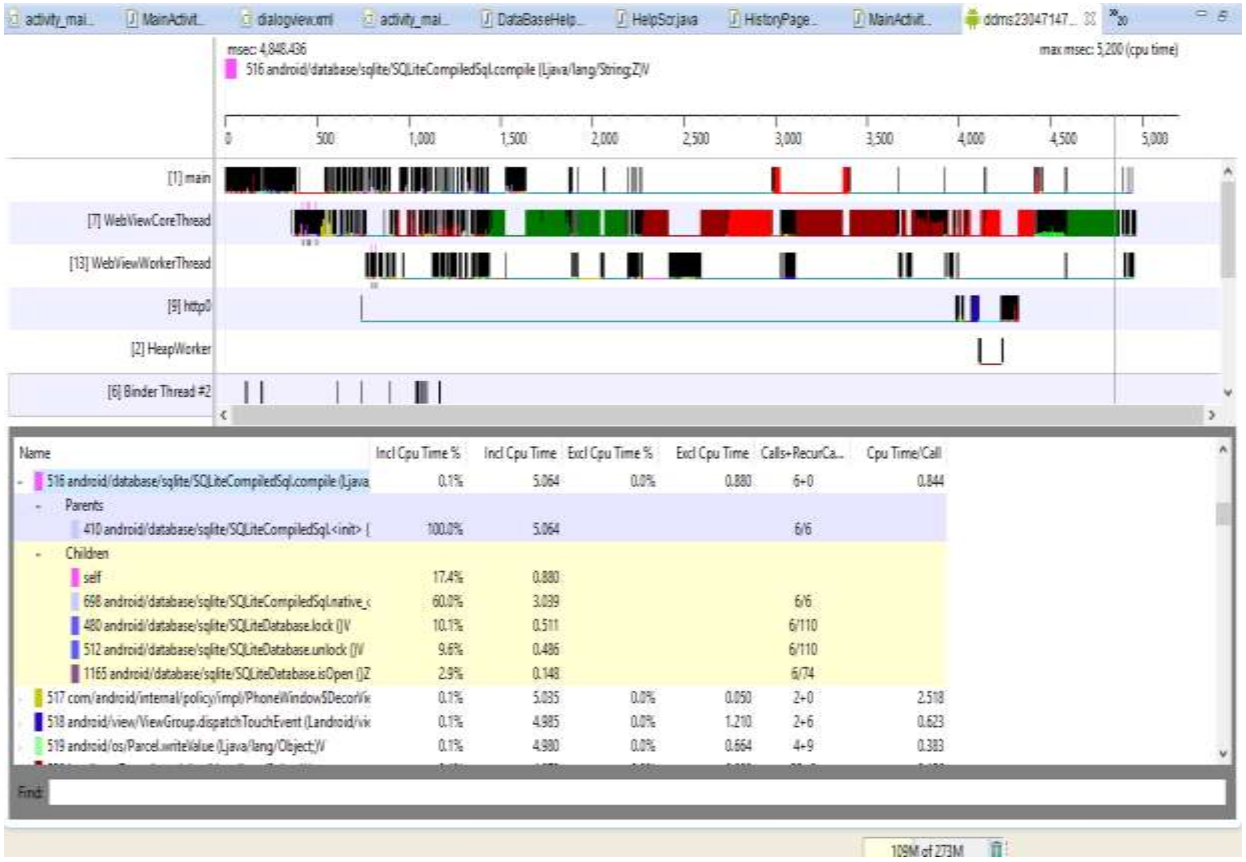
Figure 25 Profile GPU Rendering for History Check

The application shows good GPU Rendering. Except for the screens Online Dictionary and voice recognizer, almost all the vertical bars in all other screens are below the green line. The taller orange lines in these screens might be due to the fact that it takes CPU time to connect to internet.

11.2 Profiling with Traceview

Traceview is a graphical viewer for execution logs. It can help us debug the application and profile its performance [11]. I have created a trace log file using DDMS for the meaning check process. The Traceview generated b loading the log file displays the log data in two panels

- **A timeline panel** - describes when each thread and method started and stopped. Each thread's execution is shown in its own row, with time increasing to the right.
- **A profile panel** - provides a summary of all the time spent in a method. The inclusive time is the time spent in the method plus the time spent in any called functions. The exclusive time is the time spent in the method. The parent and children of the method is displayed in this panel.



Name	Incl Cpu Time %	Incl Cpu Time	Excl Cpu Time %	Excl Cpu Time	Calls+RecurCa...	Cpu Time/Call
- 516 android/database/sqlite/SQLiteCompiledSql.compile (Ljava	0.1%	5.064	0.0%	0.880	6+0	0.844
- Parents						
410 android/database/sqlite/SQLiteCompiledSql.<init> (	100.0%	5.064			6/6	
- Children						
self	17.4%	0.880				
698 android/database/sqlite/SQLiteCompiledSql.native_c	60.0%	3.039			6/6	
480 android/database/sqlite/SQLiteDatabase.lock ()V	10.1%	0.511			6/110	
512 android/database/sqlite/SQLiteDatabase.unlock ()V	9.6%	0.486			6/110	
1165 android/database/sqlite/SQLiteDatabase.isOpen ()Z	2.9%	0.148			6/74	
517 com/android/internal/policy/impl/PhoneWindow$DecorVie	0.1%	5.035	0.0%	0.050	2+0	2.518
518 android/view/ViewGroup.dispatchTouchEvent (Landroid/vie	0.1%	4.985	0.0%	1.210	2+6	0.623
519 android/os/Parcel.writeValue (Ljava/lang/Object;)V	0.1%	4.980	0.0%	0.664	4+9	0.383

Figure 26 Traceview for Meaning Check

I have created the traceview for the meaning check. If you hover the cursor over the processes in the timeline panel, you can actually check the time for which the process or the thread is running for. This helps us find the processes that are running for more than desired time and can be helpful in code optimization in such processes.

11.3 CPU Load Analysis

The below pie-chart shows the CPU load for the meaning check method.

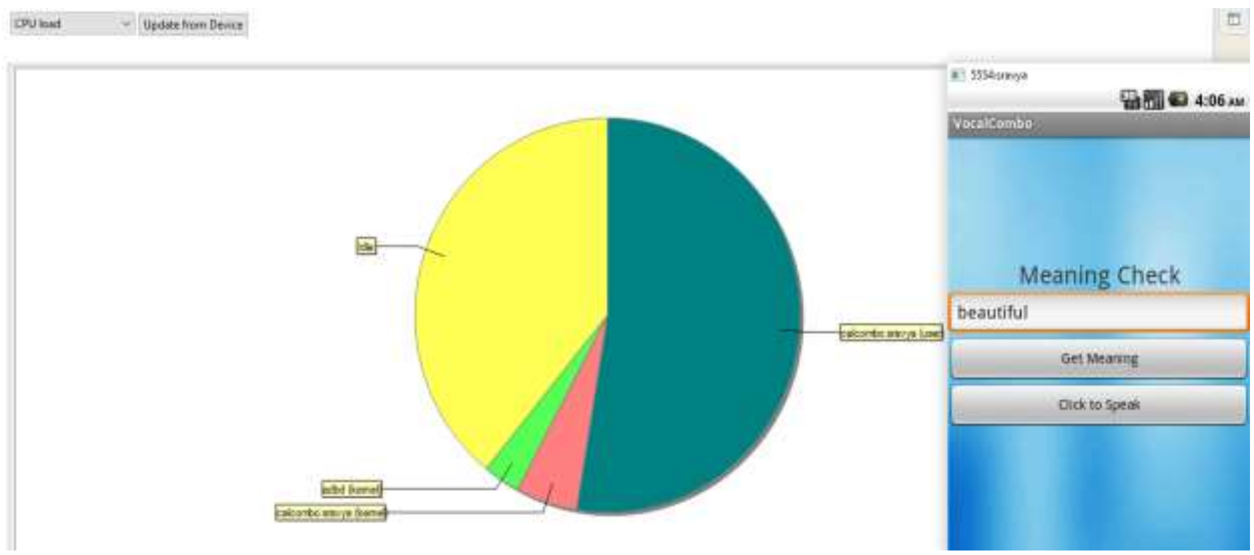


Figure 27 CPU Load Analysis for Meaning Check

Statistical data- com.vocalcombo.sravya(user): (51, 53%)

com.vocalcombo.sravya(kernel): (5, 5%)

From the above pie-graph we infer the peak CPU usage for the meaning check functionality of my application. The meaning check uses 53% of the CPU time as the meaning check functionality requires the application to get connected to the network, which cannot be avoided. This is the peak CPU load for the application and for all other functionalities the CPU load is below 20%.

11.4 Memory Usage Analysis

The below Memory Usage pie-chart shows the memory usage for the meaning check method in PPS in KB where PSS, Proportional set size, is a count of pages it has in memory, where each page is divided by the number of processes sharing it.

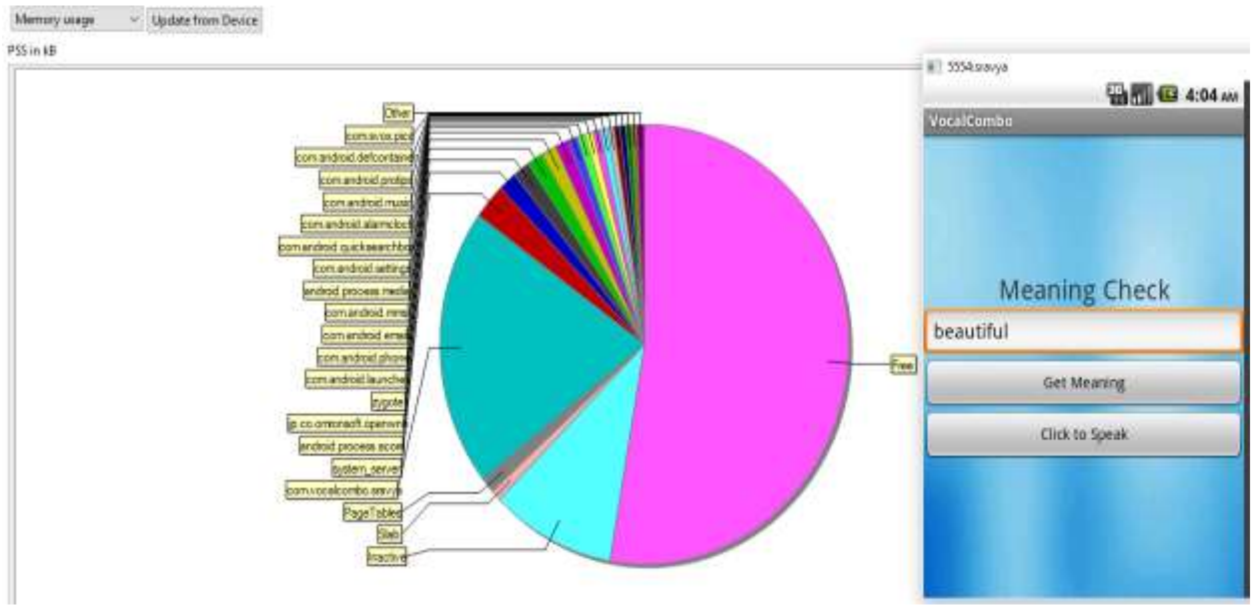


Figure 28 Memory Usage Analysis for Meaning Check

Statistical data- com.vocalcombo.sravya (108,470, 20%)

From the above pie-chart, we can infer that the meaning check functionality uses about 20% of the total memory available for the native and third part application development. The size of the .apk file of the application is about 800 KB and the application when installed occupies 1.3MB of the disk space, which is optimal among most of the available third party android applications that uses the dictionary functionalities.

Overall, the application screens have well designed user interface with minimum GPU overdraw and good GPU rendering. The traceview for the meaning check shows that except the process which uses the webView class that connects the application to the internet, all other process runs with in desired time frame which shows optimal performance of the application.

Chapter 12 - Conclusion

Though there are many websites which provides you the functionalities provided by this application, there are only a few which provides both the functionalities of Pronunciation check and Meaning check. Out of those, there are even less websites that displays the meaning of the words the user speaks out. This Android application provides the flexibility of installing this application on your Android powered smart phone or tablet. This can be really helpful for non-native English speakers and even for the native-English speakers who wants to check the meaning of complex words or play a pronunciation check game. The history check functionality helps the user to recollect the words for which he have checked the pronunciation and meaning for so that he can practice them. The Text to Speech and Speech to Text APIs provided by Android are very helpful in implementing this application. Developing this application boosted my confidence of developing mobile applications.

The TextToSpeech, used in pronunciation check, can be used in applications like reading out emails or texts while driving. The VoiceRecognizer, used in meaning check, can be used in applications to type any emails which user speaks out while driving. This application can be used to help disabled people with features like voice navigation, filling a form with voice input etc.

Chapter 13 - Future Work

The Vocal Combo application requires the user to be connected to the internet all the time he uses the application. One of the future works can be to extend the functionalities of the application so that it can work without internet. We can make use of some offline dictionaries or do the web crawling to integrate with the pronunciation and meaning check functionality. This application can be extended to use offline speech recognition using PocketSphinx. The application can be extended to provide the functionalities for different languages. It can also be developed as cross platform application using various mobile application development frameworks like Apache Cordova so that it can be compatible with any Operating System.

References

[1] "An overview of the Android Architecture"
http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture [Feb. 10, 2016]

[2] "Android-Architecture"
http://www.tutorialspoint.com/Android/Android_architecture.htm [Feb. 10, 2016]

[3] "Systems design" [Feb. 20, 2016]
https://en.wikipedia.org/wiki/Systems_design [Feb. 20, 2016]

[4] "The Unified Modeling Language"
http://www.uml-diagrams.org/ [Feb. 20, 2016]

[5] "Android-Application Components"
http://www.tutorialspoint.com/Android/Android_application_components.htm [Feb. 25, 2016]

[6] "Basic Components in Android Applications"
http://www.compiletimeerror.com/2013/01/basics-components-in.html#.VwQz1fn49pg [Feb. 25, 2016]

[7] "Android-SQLite Database Tutorial"
http://www.tutorialspoint.com/Android/Android_sqlite_database.htm [Mar. 02, 2016]

[8] "RecognizerIntent"
http://developer.Android.com/reference/Android/speech/RecognizerIntent.html [Mar. 06, 2016]

[9] "Performance Profiling Tools"
http://developer.android.com/tools/performance/index.html [Apr. 07, 2016]

[10] "Profiling GPU Rendering Walkthrough"
http://developer.android.com/tools/performance/profile-gpu-rendering/index.html#WhatYouNeed [Apr. 10, 2016]

[11] "Profiling with Traceview"
http://developer.android.com/tools/debugging/debugging-tracing.html [Apr. 07, 2016]