CMU Sphinx Development

Assignment 1

Class

Special Topic

Tutor

Paul Morris

Date Last Revised

Friday, 16 June 2023

Table of Contents

Stage 1 – Research Technologies

In stage one, the research team examined PocketSphinx and Sphinx4 technologies in order to determine the most suitable technology stack to proceed with throughout the study.

In this phase, researchers downloaded such integrated development platforms like Android Studio, IntelliJ IDEA, and NetBeans, alongside the PocketSphinx and Sphinx4 speech recognition libraries in order to determine the most appropriate technology stack to proceed with going forward into the future of the study.

The research team determined that the most suitable technology stack to proceed forward with would be Android Studio, alongside PocketSphinx. This is the result of PocketSphinx consisting of two important factors related to the use of speech recognition in this study, these are speed and portability (CMUSphinx, n.d.).

Stage 2 – Learn New IDE and Programming Languages

The research team utilized stage two of the study to gain knowledge within Android Studio which they would proceed to use in later stages of the study. The team acquired this knowledge by developing a simple text to speech application using Java and XML.

Phase two of this study was carried out throughout the duration of ten weeks. During this period, the development team focused on learning the ropes of Android Studio, alongside the Java and XML programming languages.

The team performed this by, first and foremost obtaining knowledge in the layout of folder structures within the Integrated Development Platform (as seen in Figure 1). In addition to this, learning the fundamentals of Java and XML (as seen in Figure 2 and 3). This included a portion of advanced programming techniques such as, acquiring the knowledge to save and load user settings (as seen in Figure 4), alongside applying XML styles, and application themes and colours. Succeeding this, the development process was relatively streamlined with continuous research for basic information and weekly team meetings performed throughout.

Figure 1

Folder Structure

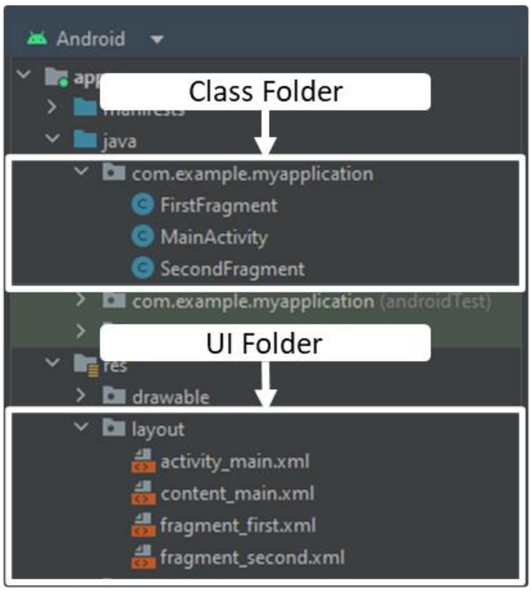
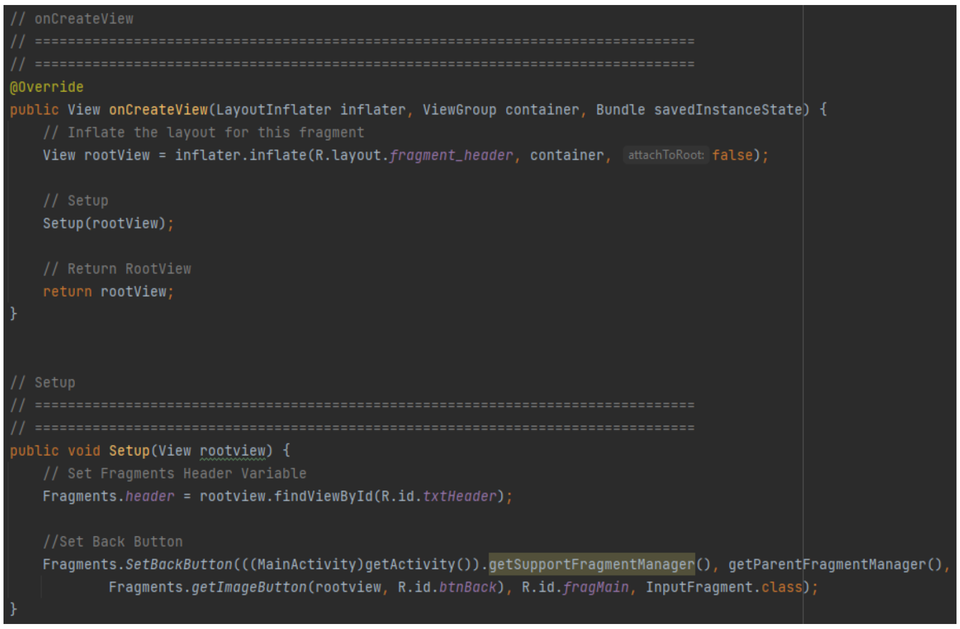


Figure 2

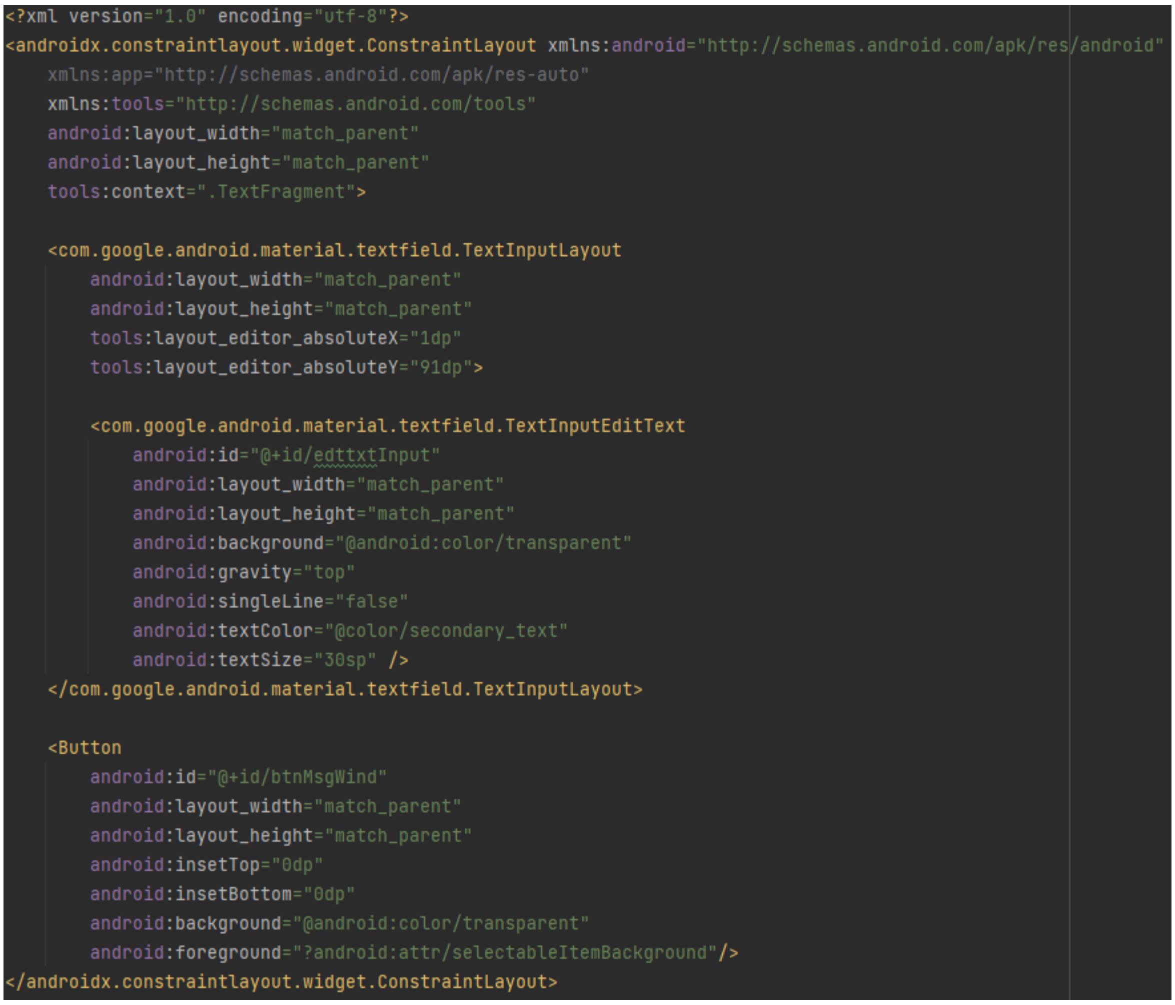
Basic Java Example



```java
// onCreateView
// ================================================================================
// ================================================================================
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View rootView = inflater.inflate(R.layout.fragment_header, container,  attachToRoot: false);

    // Setup
    Setup(rootView);

    // Return RootView
    return rootView;
}


// Setup
// ================================================================================
// ================================================================================
public void Setup(View rootview) {
    // Set Fragments Header Variable
    Fragments.header = rootview.findViewById(R.id.txtHeader);

    //Set Back Button
    Fragments.SetBackButton(((MainActivity)getActivity()).getSupportFragmentManager(), getParentFragmentManager(),
            Fragments.getImageButton(rootview, R.id.btnBack), R.id.fragMain, InputFragment.class);
}
```

Figure 3

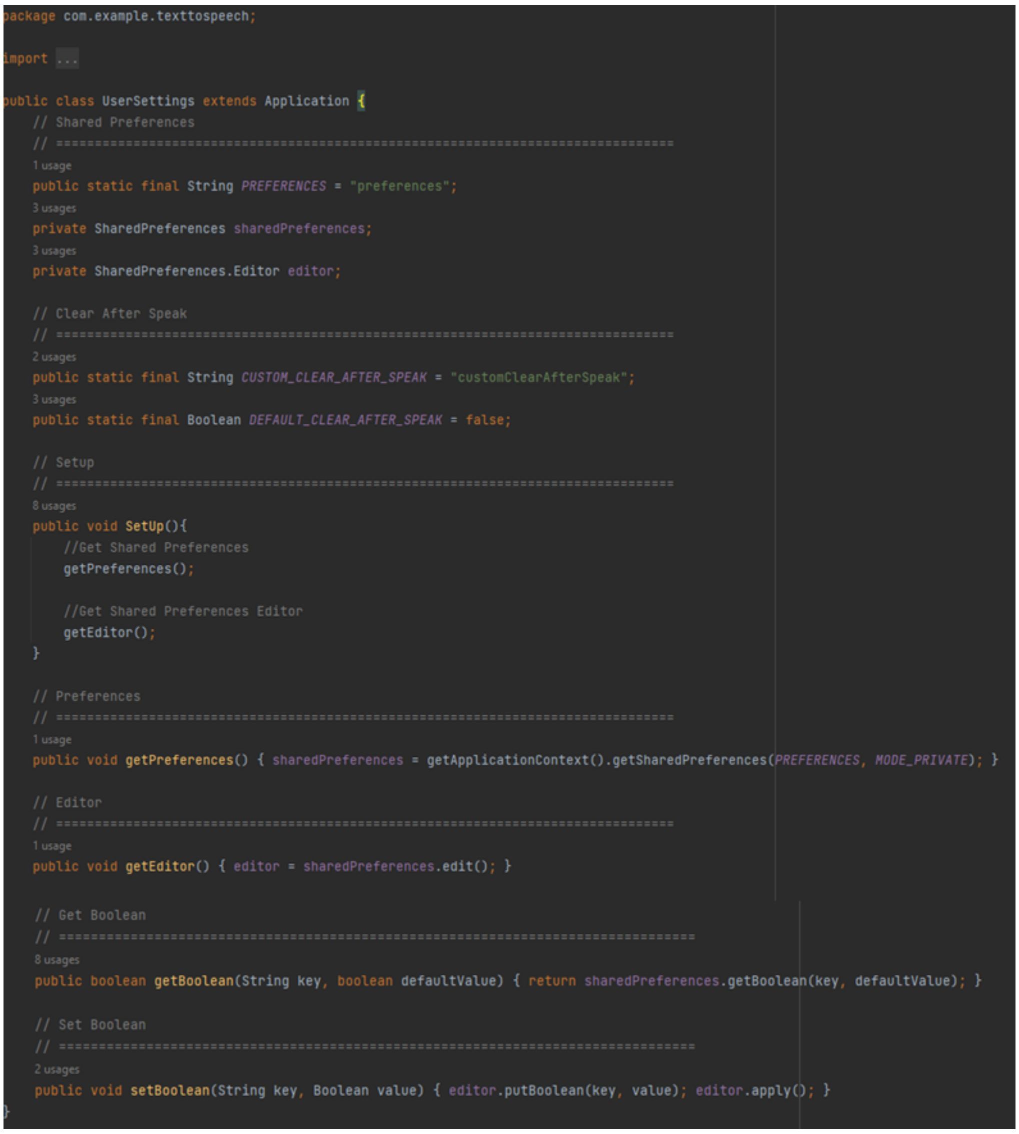
Basic XML Example



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TextFragment">

    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:layout_editor_absoluteX="1dp"
        tools:layout_editor_absoluteY="91dp">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/edttxtInput"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@android:color/transparent"
            android:gravity="top"
            android:singleLine="false"
            android:textColor="@color/secondary_text"
            android:textSize="30sp" />
    </com.google.android.material.textfield.TextInputLayout>

    <Button
        android:id="@+id/btnMsgWind"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:insetTop="0dp"
        android:insetBottom="0dp"
        android:background="@android:color/transparent"
        android:foreground="?android:attr/selectableItemBackground"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 4

Save and Load User Settings



```
package com.example.texttospeech;

import ...

public class UserSettings extends Application {
    // Shared Preferences
    // ================================================================================
    1 usage
    public static final String PREFERENCES = "preferences";
    3 usages
    private SharedPreferences sharedPreferences;
    3 usages
    private SharedPreferences.Editor editor;

    // Clear After Speak
    // ================================================================================
    2 usages
    public static final String CUSTOM_CLEAR_AFTER_SPEAK = "customClearAfterSpeak";
    3 usages
    public static final Boolean DEFAULT_CLEAR_AFTER_SPEAK = false;

    // Setup
    // ================================================================================
    8 usages
    public void SetUp(){
        //Get Shared Preferences
        getPreferences();

        //Get Shared Preferences Editor
        getEditor();
    }

    // Preferences
    // ================================================================================
    1 usage
    public void getPreferences() { sharedPreferences = getApplicationContext().getSharedPreferences(PREFERENCES, MODE_PRIVATE); }

    // Editor
    // ================================================================================
    1 usage
    public void getEditor() { editor = sharedPreferences.edit(); }

    // Get Boolean
    // ================================================================================
    8 usages
    public boolean getBoolean(String key, boolean defaultValue) { return sharedPreferences.getBoolean(key, defaultValue); }

    // Set Boolean
    // ================================================================================
    2 usages
    public void setBoolean(String key, Boolean value) { editor.putBoolean(key, value); editor.apply(); }
}
```

Stage 3 – Create Basic Speech to Text Application

In stage three of the study, the research team utilized the skills acquired in phase two of the study to develop a small speech to text application. The team implemented this, utilizing the PocketSphinx speech recognition library.

Stage two of this study was carried out throughout the duration of a week. Over this period, the development team dedicated their time to obtaining an operational version of the PocketSphinx speech recognition library within Android Studio.

Unfortunately, the research team found this phase tedious to a certain degree. However, the development team's dismay was not out of spite. PocketSphinx was last released in two thousand and sixteen. This, as you may have conceptualized, places the PocketSphinx speech recognition library in an odd situation for modern development. The library should not be considered outdated, however, with the mobile development community ever changing, features for Android Studio which once existed or did not exist in two thousand and sixteen cause a problematic development environment.

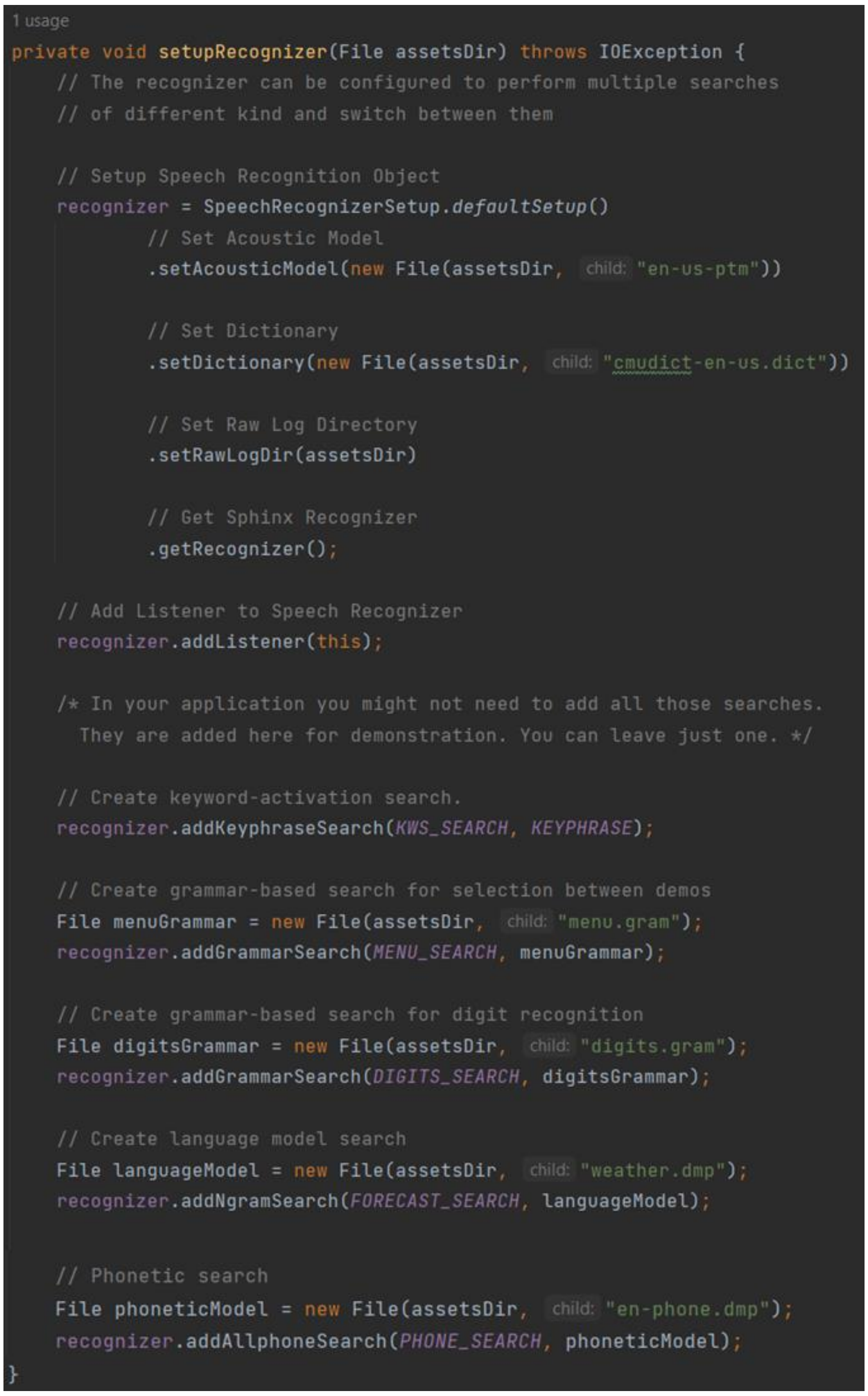
Unfortunately, the majority of tutorials and documentation located online to create a speech recognition application utilizing PocketSphinx were vastly outdated, with some going as far back as two thousand and eleven. The primary concern with utilizing these resources were they utilized matured editions of the Android Studio IDE which consisted of exceedingly dissimilar build features from the modern iteration of the integrated development platform.

An example of a problematic feature which did not exist in two thousand sixteen is background microphone access succeeding Android API v9.0. Following the release of Android API v9.0 in two thousand and seventeen, only applications operating in the foreground (or a foreground service) could capture audio input. If an application without a foreground service began to capture, the application would continue running, however, it would receive silence, even if it was the only application capturing audio at the time (Android, 2023). The development team circumvented this issue by downgrading the Android Studio development API version to eight-point-zero.

Succeeding the development team's success with overcoming the various obstacles, the process was relatively simple to get a functional speech to text application running. Unfortunately, the speech recognition was exceptionally inaccurate. A code example of the speech to text application may be found in Figure 5.

Figure 5

Speech to Text Code Example



```
1 usage
private void setupRecognizer(File assetsDir) throws IOException {
    // The recognizer can be configured to perform multiple searches
    // of different kind and switch between them

    // Setup Speech Recognition Object
    recognizer = SpeechRecognizerSetup.defaultSetup()
            // Set Acoustic Model
            .setAcousticModel(new File(assetsDir,  child: "en-us-ptm"))

            // Set Dictionary
            .setDictionary(new File(assetsDir,  child: "cmudict-en-us.dict"))

            // Set Raw Log Directory
            .setRawLogDir(assetsDir)

            // Get Sphinx Recognizer
            .getRecognizer();

    // Add Listener to Speech Recognizer
    recognizer.addListener(this);

    /* In your application you might not need to add all those searches.
      They are added here for demonstration. You can leave just one. */

    // Create keyword-activation search.
    recognizer.addKeyphraseSearch(KWS_SEARCH, KEYPHRASE);

    // Create grammar-based search for selection between demos
    File menuGrammar = new File(assetsDir,  child: "menu.gram");
    recognizer.addGrammarSearch(MENU_SEARCH, menuGrammar);

    // Create grammar-based search for digit recognition
    File digitsGrammar = new File(assetsDir,  child: "digits.gram");
    recognizer.addGrammarSearch(DIGITS_SEARCH, digitsGrammar);

    // Create language model search
    File languageModel = new File(assetsDir,  child: "weather.dmp");
    recognizer.addNgramSearch(FORECAST_SEARCH, languageModel);


    // Phonetic search
    File phoneticModel = new File(assetsDir,  child: "en-phone.dmp");
    recognizer.addAllphoneSearch(PHONE_SEARCH, phoneticModel);
}
```

Stage 4 – Create and Implement Adapted Acoustic Model

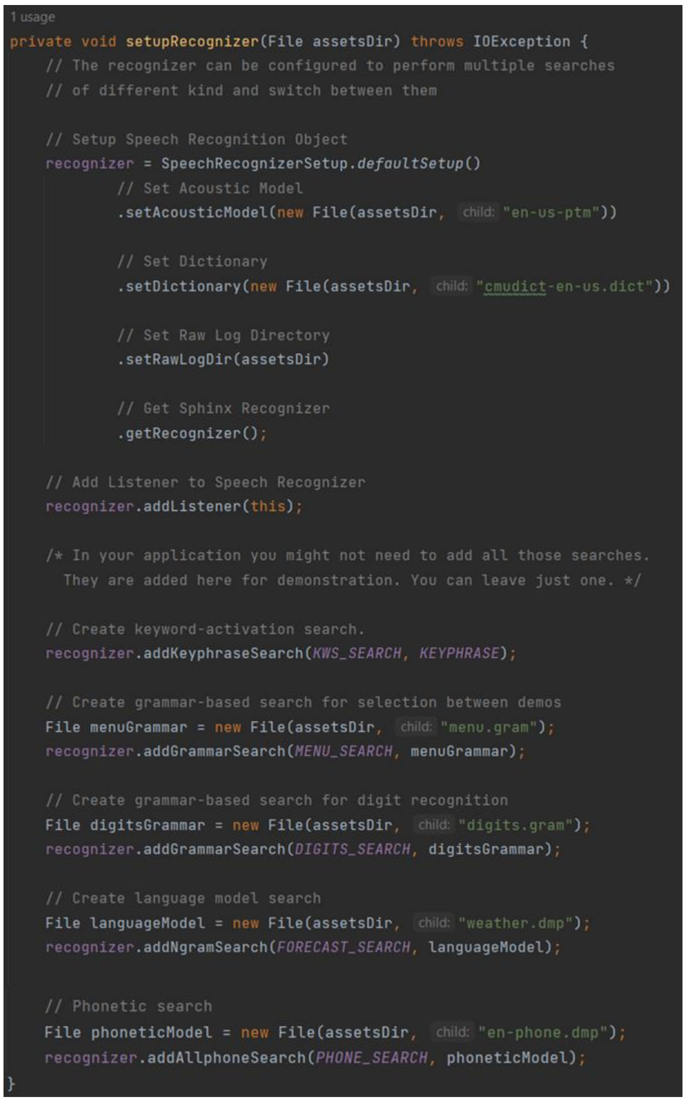
The research team used stage four of the study to train an acoustic model on Google Translate's female English voice utilizing SphinxTrain. In addition to this, the team implemented the trained acoustic model in the small speech to text application developed during phase 3.

Phase four of this study was performed over the course of three days. During this period, the development team focused on recording audio, creating an adapted acoustic model, and implementing the adapted acoustic model within the created speech to text application.

The team performed this by, first and foremost utilizing a microphone to record audio snippets of Google Translate's female English voice. Succeeding this, the development team followed the Training CMU Sphinx Speech Recognition tutorial to adapt the acoustic model inside of a Linux virtual machine. Following this, the research team transferred the acoustic model on to a Window device before proceeding to implement the model within the small speech to text application developed throughout phase 3. The aforementioned development process was relatively streamlined with little to no research performed. An example of the code modification performed for phase four may be shown in Figure 6 and 7.

Figure 6

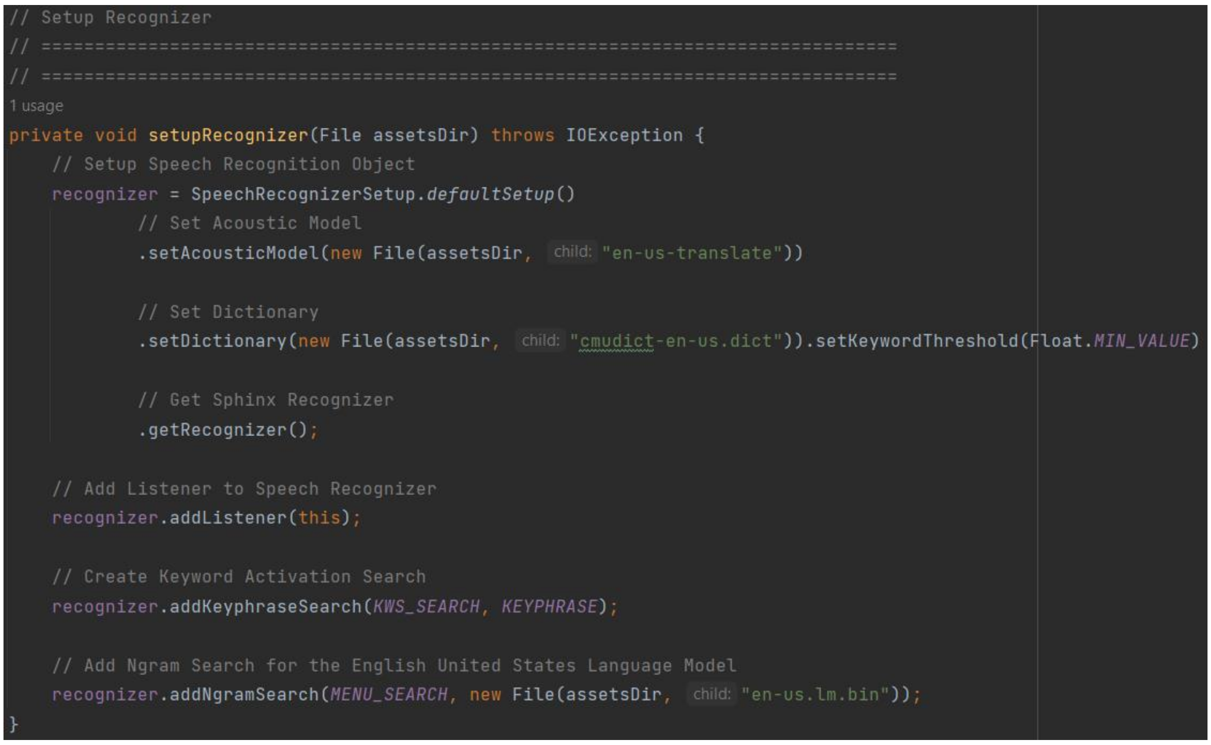
Speech Recognition Setup Before Adapted Acoustic Model Implementation



```
1 usage
private void setupRecognizer(File assetsDir) throws IOException {
    // The recognizer can be configured to perform multiple searches
    // of different kind and switch between them

    // Setup Speech Recognition Object
    recognizer = SpeechRecognizerSetup.defaultSetup()
            // Set Acoustic Model
            .setAcousticModel(new File(assetsDir,  child: "en-us-ptm"))

            // Set Dictionary
            .setDictionary(new File(assetsDir,  child: "cmudict-en-us.dict"))

            // Set Raw Log Directory
            .setRawLogDir(assetsDir)

            // Get Sphinx Recognizer
            .getRecognizer();

    // Add Listener to Speech Recognizer
    recognizer.addListener(this);

    /* In your application you might not need to add all those searches.
      They are added here for demonstration. You can leave just one. */

    // Create keyword-activation search.
    recognizer.addKeyphraseSearch(KWS_SEARCH, KEYPHRASE);

    // Create grammar-based search for selection between demos
    File menuGrammar = new File(assetsDir,  child: "menu.gram");
    recognizer.addGrammarSearch(MENU_SEARCH, menuGrammar);

    // Create grammar-based search for digit recognition
    File digitsGrammar = new File(assetsDir,  child: "digits.gram");
    recognizer.addGrammarSearch(DIGITS_SEARCH, digitsGrammar);

    // Create language model search
    File languageModel = new File(assetsDir,  child: "weather.dmp");
    recognizer.addNgramSearch(FORECAST_SEARCH, languageModel);


    // Phonetic search
    File phoneticModel = new File(assetsDir,  child: "en-phone.dmp");
    recognizer.addAllphoneSearch(PHONE_SEARCH, phoneticModel);
}
```

Figure 7

Speech Recognition Setup After Ada[ted Acoustic Model Implementation



```
// Setup Recognizer
// ================================================================================
// ================================================================================
1 usage
private void setupRecognizer(File assetsDir) throws IOException {
    // Setup Speech Recognition Object
    recognizer = SpeechRecognizerSetup.defaultSetup()
            // Set Acoustic Model
            .setAcousticModel(new File(assetsDir,  child: "en-us-translate"))

            // Set Dictionary
            .setDictionary(new File(assetsDir,  child: "cmudict-en-us.dict")).setKeywordThreshold(Float.MIN_VALUE)

            // Get Sphinx Recognizer
            .getRecognizer();

    // Add Listener to Speech Recognizer
    recognizer.addListener(this);

    // Create Keyword Activation Search
    recognizer.addKeyphraseSearch(KWS_SEARCH, KEYPHRASE);

    // Add Ngram Search for the English United States Language Model
    recognizer.addNgramSearch(MENU_SEARCH, new File(assetsDir,  child: "en-us.lm.bin"));
}
```

References

Android. (2023, May 22). *Sharing Audio Input.*

https://developer.android.com/guide/topics/media/sharing-audio-input#pre-behavior

CMUSphinx. n.d. *Before You Start.*

https://cmusphinx.github.io/wiki/tutorialbeforestart/#technologies