```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1" android:versionName="1.0" package="com.paad.GREVocab"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <application android:label="@string/app_name" android:icon="@drawable/icon"
    android:debuggable="true">
        <activity android:label="@string/app_name" android:name=".GREVocab">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:label="About Us" android:name=".mydev" class=".mydev" />
        <activity android:label="Wordlist A" android:name=".a" class=".a" />
        <activity android:label="Wordlist B" android:name=".b" class=".b" />
        <activity android:label="Wordlist C" android:name=".c" class=".c" />
        <activity android:label="Wordlist DE" android:name=".de" class=".de" />
        <activity android:label="Wordlist F" android:name=".f" class=".f" />
    </application>
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```

```
package com.paad.GREVocab;

import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.ListActivity;
import android.content.Intent;
import android.content.res.Resources;
import android.net.Uri;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import java.io.InputStream;
import java.util.ArrayList;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

public class GREVocab extends ListActivity
{
  ArrayList<String> items;
  String[] mean;
  TextView selection;
  public GREVocab()
  {
    ArrayList localArrayList = new ArrayList();
    this.items = localArrayList;
    String[] arrayOfString = DictionaryDatabase;
    this.mean = arrayOfString;
  }

  public void onCreate(Bundle icicle)
  {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    TextView localTextView = (TextView)findViewById(R.layout.main);
    this.selection = localTextView;
    try
    {
      InputStream localInputStream = getResources().openRawResource(R.layout.main);
      NodeList localNodeList = DocumentBuilderFactory.newInstance().newDocumentBuilder().
      parse(localInputStream, null).getElementsByTagName("word");
      int i = 0;
      while (true)
      {
        int j = localNodeList.getLength();
        if (i >= j)
```

```
      {
        localInputStream.close();
        ArrayList localArrayList1 = this.items;
        ArrayAdapter localArrayAdapter = new ArrayAdapter(this, r.layout.main,
        localArrayList1);
        setListAdapter(localArrayAdapter);
        return;
      }
      ArrayList localArrayList2 = this.items;
      String str1 = ((Element)localNodeList.item(i)).getAttribute("value");
      boolean bool = localArrayList2.add(str1);
      i += 1;
    }
  }
  catch (Throwable localThrowable)
  {
    while (true)
    {
      StringBuilder localStringBuilder = new StringBuilder("Exception: ");
      String str2 = localThrowable.toString();
      String str3 = str2;
      Toast.makeText(this, str3, 2000).show();
    }
  }
}

public boolean onCreateOptionsMenu(Menu paramMenu)
{
  getMenuInflater().inflate(R.menu.menu, paramMenu);
  return true;
}

public boolean onKeyDown(int paramInt, KeyEvent paramKeyEvent)
{
  boolean bool = super.onKeyDown(paramInt, paramKeyEvent);
  int i;
  switch (paramInt)
  {
  case KeyEvent.BACK:
    Intent localIntent = new Intent("Complete");
    setResult(-1, localIntent);
    finish();
    break;
  case KeyEvent.CAMERA:
    return i;
    Toast.makeText(this, "Pressed Camera Button", 0).show();
    i = 1;
    break;
  case KeyEvent.ENDCALL:
     finish();
     break;
  default:
     break;
  }
  return false;
}
```

```
public void onListItemClick(ListView paramListView, View paramView, int paramInt, long
paramLong)
{
  TextView localTextView = this.selection;
  String str = this.mean[paramInt];
  localTextView.setText(str);
  Animation localAnimation = AnimationUtils.loadAnimation(this, R.anim.animate);
  localAnimation.reset();
  this.selection.startAnimation(localAnimation);
}

public boolean onOptionsItemSelected(MenuItem paramMenuItem)
{
  switch (paramMenuItem.getItemId())
  {
  case R.id.develop:
    Intent localIntent1 = new Intent(this, mydev.class);
    startActivity(localIntent1);
    break;
  case R.id.setting:
    Toast.makeText(this, "Enjoy The Colors!!!", 1).show();
  case R.id.submenu1:
    getListView().setBackgroundColor(0xffffff);
    break;
  case R.id.submenu2:
    getListView().setBackgroundColor(0xFFFFAF);
    break;
  case R.id.submenu3:
    getListView().setBackgroundColor(0x778899);
    break;
  case R.id.submenu4:
    getListView().setBackgroundColor(0x0000CD);
    break;
  case R.id.submenu5:
    getListView().setBackgroundColor(0x228B22);
    break;
  case R.id.submenu6:
    getListView().setBackgroundColor(0x2FOE6C);
    break;
  case R.id.submenu7:
    getListView().setBackgroundColor(0x8B4513);
    break;
  case R.id.submenu8:
    getListView().setBackgroundColor(0x8A2BE2);
    break;
  case R.id.submenu9:
    getListView().setBackgroundColor(0x3CB371);
    break;
  case R.id.submenu10:
    getListView().setBackgroundColor(0xFF6347);
    break;
  case R.id.submenu11:
    getListView().setBackgroundColor(0x9370DB);
    break;
  case R.id.submenu12:
    getListView().setBackgroundColor(0x838B83);
    break;
```

```
    case R.id.submenu13:
      getListView().setBackgroundColor(0xFFFFFF);
      break;
    case R.id.submenu14:
      getListView().setBackgroundColor(FF0000);
      break;
    case R.id.fact:
      AlertDialog.Builder localBuilder = new AlertDialog.Builder(this).setTitle("FAQs").
      setMessage
      ("It's a Basic GRE Vocabulary software...\nBrowse Through The Words And Click To See
      its Meaning...");
      GREVocab.1 local1 = new GREVocab.1(this);
      AlertDialog localAlertDialog = localBuilder.setNeutralButton("Close", local1).show();
      break;
    case R.id.quit:
      finish();
      break;
    case R.id.inf:
      Toast.makeText(this, "Program Size: 1024Kb \nDistribution Restricted", 1).show();
      break;
    case R.id.nxt_lesson:
      Intent localIntent3 = new Intent(this, b.class);
      startActivity(localIntent3);
      break;
    case R.id.feedback:
      Uri localUri = Uri.fromParts("sms", "09646864149", null);
      Intent localIntent4 = new Intent("android.intent.action.VIEW", localUri);
      startActivity(localIntent4);
      break;
    }
    return false;
  }
}
```

DATABASE ACCESSING

```
package com.example.android.searchabledict;

import android.app.SearchManager;
import android.content.ContentValues;
import android.content.Context;
import android.content.res.Resources;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.provider.BaseColumns;
import android.text.TextUtils;
import android.util.Log;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.HashMap;

public class DictionaryDatabase
{
    private static final String TAG = "DictionaryDatabase";
    public static final String KEY_WORD =
SearchManager.SUGGEST_COLUMN_TEXT_1;
    public static final String KEY_DEFINITION =
SearchManager.SUGGEST_COLUMN_TEXT_2;

    private static final String DATABASE_NAME = "dictionary.XLS";
    private static final String FTS_VIRTUAL_TABLE = "FTSdictionary";
    private static final int DATABASE_VERSION = 2;

    private final DictionaryOpenHelper mDatabaseOpenHelper;
    private static final HashMap<String,String> mColumnMap =
buildColumnMap();
    public DictionaryDatabase(Context context)
      {
        mDatabaseOpenHelper = new DictionaryOpenHelper(context);
    }
    private static HashMap<String,String> buildColumnMap() {
        HashMap<String,String> map = new HashMap<String,String>();
        map.put(KEY_WORD, KEY_WORD);
        map.put(KEY_DEFINITION, KEY_DEFINITION);
```

```
        map.put(BaseColumns._ID, "rowid AS " +
                BaseColumns._ID);
        map.put(SearchManager.SUGGEST_COLUMN_INTENT_DATA_ID, "rowid AS
" +
                SearchManager.SUGGEST_COLUMN_INTENT_DATA_ID);
        map.put(SearchManager.SUGGEST_COLUMN_SHORTCUT_ID, "rowid AS "
+
                SearchManager.SUGGEST_COLUMN_SHORTCUT_ID);
        return map;
    }

    public Cursor getWord(String rowId, String[] columns) {
        String selection = "rowid = ?";
        String[] selectionArgs = new String[] {rowId};

        return query(selection, selectionArgs, columns);

    }

    public Cursor getWordMatches(String query, String[] columns) {
        String selection = KEY_WORD + " MATCH ?";
        String[] selectionArgs = new String[] {query+"*"};

        return query(selection, selectionArgs, columns);
}

 private Cursor query(String selection, String[] selectionArgs,
String[] columns) {
        /* The SQLiteBuilder provides a map for all possible columns
requested to
         * actual columns in the database, creating a simple column
alias mechanism
         * by which the ContentProvider does not need to know the real
column names
         */
        SQLiteQueryBuilder builder = new SQLiteQueryBuilder();
        builder.setTables(FTS_VIRTUAL_TABLE);
        builder.setProjectionMap(mColumnMap);

        Cursor cursor =
builder.query(mDatabaseOpenHelper.getReadableDatabase(),
                columns, selection, selectionArgs, null, null, null);

        if (cursor == null) {
            return null;
```

```
        } else if (!cursor.moveToFirst()) {
            cursor.close();
            return null;
        }
        return cursor;
    }

    private static class DictionaryOpenHelper extends SQLiteOpenHelper
{

        private final Context mHelperContext;
        private SQLiteDatabase mDatabase;

        private static final String FTS_TABLE_CREATE =
                    "CREATE VIRTUAL TABLE " + FTS_VIRTUAL_TABLE +
                    " USING fts3 (" +
                    KEY_WORD + ", " +
                    KEY_DEFINITION + ");";

        DictionaryOpenHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
            mHelperContext = context;
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
            mDatabase = db;
            mDatabase.execSQL(FTS_TABLE_CREATE);
            loadDictionary();
        }

        private void loadDictionary() {
            new Thread(new Runnable() {
                public void run() {
                    try {
                        loadWords();
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }
            }).start();
        }

        private void loadWords() throws IOException {
            Log.d(TAG, "Loading words...");
```

```
            final Resources resources = mHelperContext.getResources();
            InputStream inputStream =
resources.openRawResource(R.raw.definitions);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));

            try {
                String line;
                while ((line = reader.readLine()) != null) {
                    String[] strings = TextUtils.split(line, "-");
                    if (strings.length < 2) continue;
                    long id = addWord(strings[0].trim(),
strings[1].trim());
                    if (id < 0) {
                        Log.e(TAG, "unable to add word: " +
strings[0].trim());
                    }
                }
            } finally {
                reader.close();
            }
            Log.d(TAG, "DONE loading words.");
        }

        public long addWord(String word, String definition) {
            ContentValues initialValues = new ContentValues();
            initialValues.put(KEY_WORD, word);
            initialValues.put(KEY_DEFINITION, definition);

            return mDatabase.insert(FTS_VIRTUAL_TABLE, null,
initialValues);
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
            Log.w(TAG, "Upgrading database from version " + oldVersion
+ " to "
                    + newVersion + ", which will destroy all old
data");
            db.execSQL("DROP TABLE IF EXISTS " + FTS_VIRTUAL_TABLE);
            onCreate(db);
        }
}
```

INTENT OF CREDIT AND DEVELOPERS

```
package com.paad.GREVocab;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;

public class mydev extends Activity
{
  public void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(R.layout.screen2);
    Button localButton = (Button)findViewById(btnClick2);
    mydev.1 local1 = new mydev.1(this);
    localButton.setOnClickListener(local1);
  }
}
```