

OPENWORDS APP

DESIGN DOCUMENT

Marc Bogonovich, Ph.D.

布馬克

Contributors

Diyue Bu, Jerry Shao, Jonathan North Washington, Shashank Mahendranath, Sandesh Katkamwar, Zi Wang, Nitin Ainani, Ma Lei, Qindan Nie, Fang Li, Jianqing Guan, Mayukh Das

Authors of the Openwords file format

Jerry Shao, Jonathan North Washington, Serhiy Vernei, Marc Bogonovich, Daniel Whyatt, Alejandro Andrade

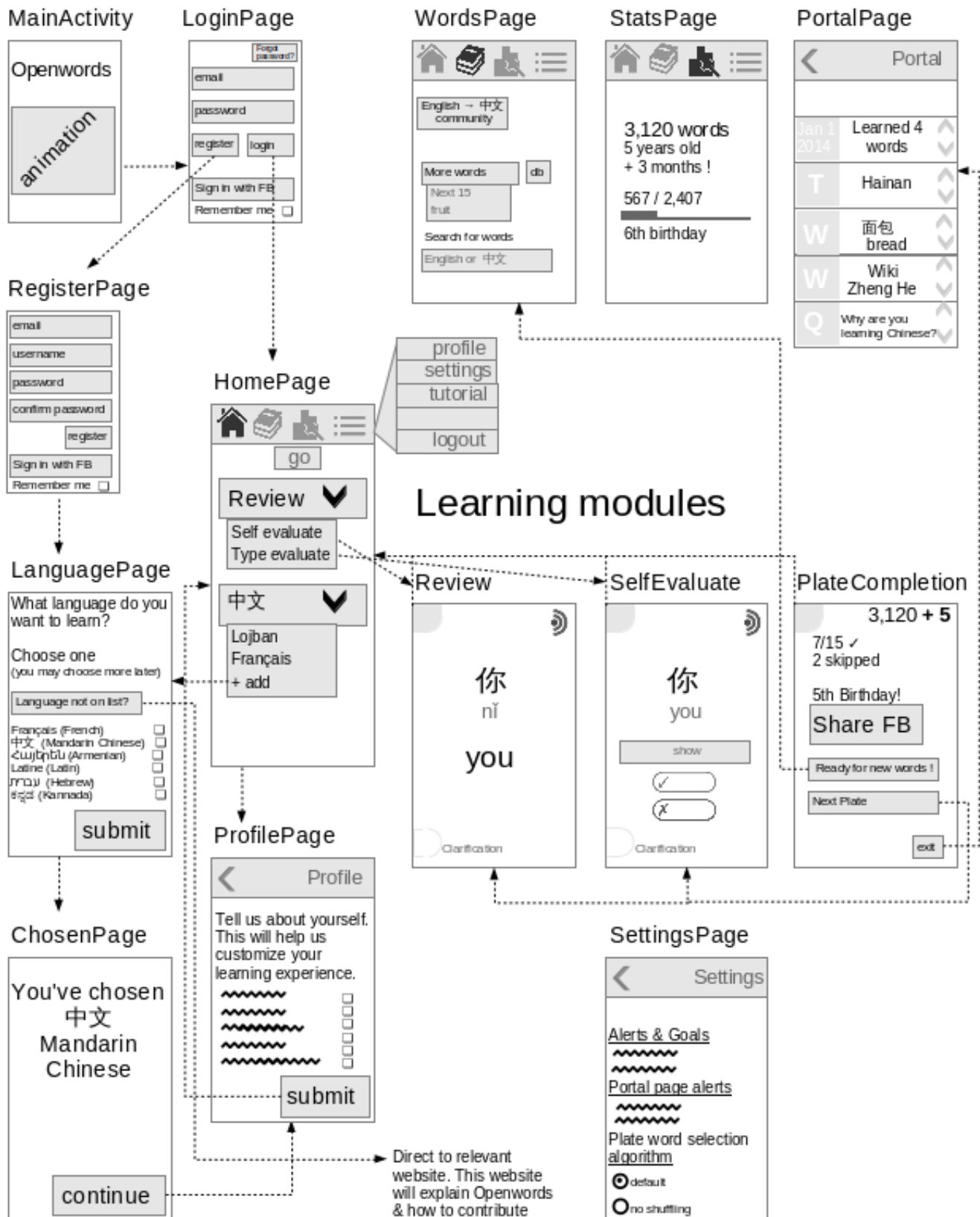
A complete overview of the Openwords 1.0 app, including:

- 1) Page network diagram
- 2) Naming conventions
- 3) Screenshots
- 4) Widgets
- 5) A complete account of widget behavior
- 6) Local Databases
- 7) Server Databases
- 8) Inter-database communication

The app will be introduced with a Page network diagram showing the connections between pages. A list of terms will follow for the pages. Next, we will illustrate individual pages, name their widgets and describe the behavior of the widgets. The Openwords tutorial is not listed in the page network diagram (PND), but is described at the end of this document.

PAGE NETWORK DIAGRAM

Upon tapping the Openwords icon within their Android system, users will come to the home screen. This diagram shows the screenshot topology from that initial screen.



NAMING CONVENTIONS

The following section details the naming conventions that will be used within the Openwords program.

We need to name xml pages & widgets (Layout).

- The page id activity_nameofpage (activity_login).
- For buttons, activity_name_widgetTypeName_functionDescription
- Three elements [activity where the widget exists], [widget]
 - Examples:
 - loginPage_TextView_answer
 - homePage_Button_testPageGo
 - homePage_item_words
 - if multiple examples of the same kind of widget exist use the above naming convention and add _00, _01, _02, etc.

We need to name objects in Java.

- Pages will be named in the following way: LoginActivity, SelfEvaluateActivity, MainActivity etcetera.
- For objects that match xml widgets.
 - They can receive simpler names.. So for example, if the corresponding widget is named:
 - start_activity_button_testActivityGo
 - then the Java button would be named:
 - testActivityGo
- For objects that **do not** match xml widgets

Intents.

- Intents will be named in the following way: PlaceIntent
- Where Place is the page that is opened by the intent

Variables.

- We need a convention for naming variables. We currently lack a convention.

Functions.

- Functions will be named with a description of that functions action
- *sometimes* we will add the return type (what if anything the function returns)

We need to name strings.

- Strings need to be clearly named, indicating in what page they are located, and with what widget/s they are associated.
- If Strings are associated with multiple pages, or multiple widgets, then these rules can be relaxed.
- ActivityName_associatedWidget_purpose
- ActivityName_associatedWidget_purpose_00
- ActivityName_associatedWidget_purpose_01 etc.

Class names.

- Classes must start with a capital, but beyond that built in convention we do not have our own convention yet.

ON STARTUP

Check for the storage of user login and password in [Storage_file_01](#). (See the widget, [introPage_CheckBox_rememberMe](#))

Next, from [Storage_file_01](#) we should know the last language that the user had been using. ***That should tell us how to populate various fields within the program.*** Additionally, what specific user-language specific databases should be loaded needs to also come from [Storage_file_01](#) ([Personal_db](#), [User_Performance_db](#), [User_Words_db](#))

MAINACTIVITY

The MainActivity is very simple. The page will display the Openwords logo and name. An animation will follow while the program is loading. The animation will be The Openwords logo at the top, with the word for “word” in several languages rising from the bottom of the screen, until coming slowly to a rest. The words will be different sizes and colors.

We'll need to use the Android animation class in order to produce this animation.



MainActivity continued...

Author: Jianqing

The MainActivity is the first page user will see after opening the app. The page has two elements: the first one is the Openwords logo and below that is the animation part.

1. Openwords logo:

Source: http://www.openwords.com/Openwords09_files/openwords_logo_c_oranged_h_97.png

Format: PNG

Size: 14,906 bytes

Height * width: 417 * 97 (pixels)

Position and other information: android:layout_marginTop="39dp"

2. Animation part:

This part contains six string elements and one function for animation. The string elements are aligned at the bottom of this page. When this page is loaded, string elements will raise from the bottom of page.

I. String:

Each string contains the word "word" in other languages. For example, "מילה" in Hebrew and "शब्द" in Hindi.

Names of these widgets:

In /Openwords/res/values/styles.xml, named string elements following this convention:

flashword_string_homePageLift01

flashword_string_homePageLift02..., etc.

II. Animation function:

Function name: void lift_word_from_bottom(int stringName, int distance)

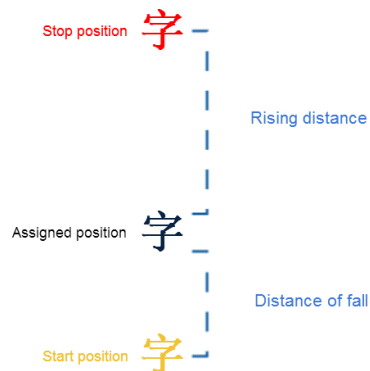
Function purpose: lift string from one position to another

Require package: android.view.animation.*

Description:

First, we get the string by input stringName in hex. Then we set an animation translator. The four parameters of class TranslateAnimation are (xFrom, xTo, yFrom, yTo). Since we just need to word lifting vertically, the xFrom and xTo are 0. The yFrom sets the start position and the yTo is the stop position.

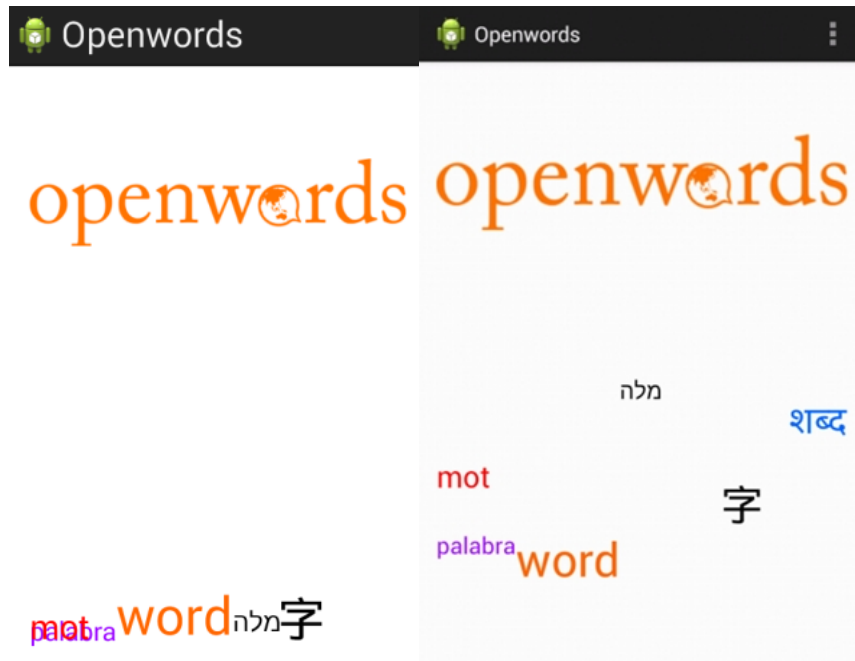
Illustration:



When we insert textView for each string element, each one has a location. Those position determine the strings' location horizontally but during the animation the strings start at (y + distance_of_fall) and stop at (y - rising_distance) vertically. (The y is the current coordinate) The duration is set to be a fixed value which means all string elements start and end at the same time.

3. Connection:

After this page is loaded, user can touch screen to skip the animation and go to LoginPage

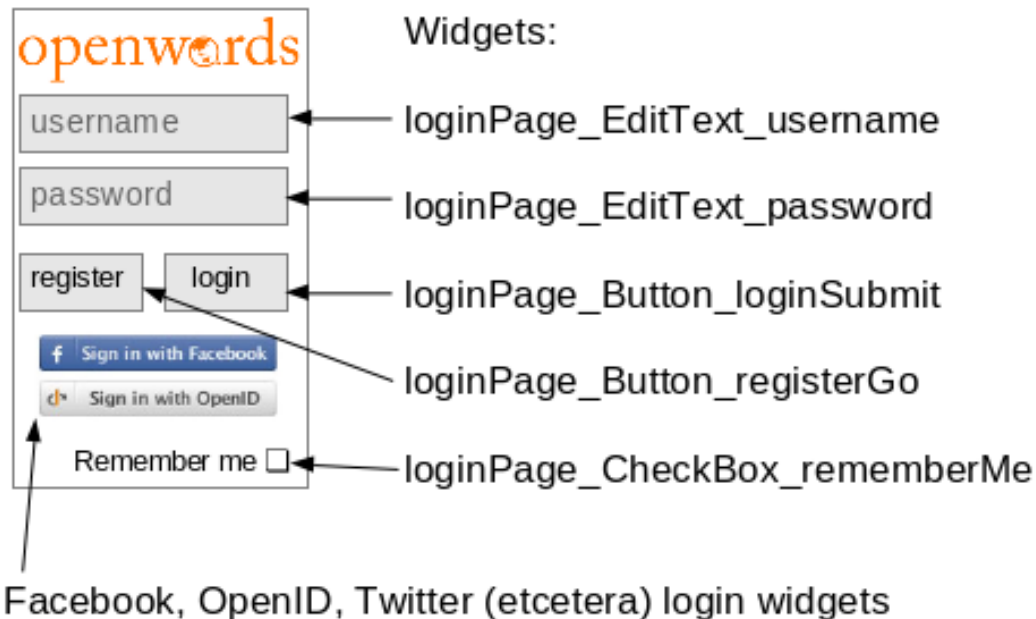


Varun Kothari improved the MainActivity in the following ways

- 1) Developed code to remove the notification bar
- 2) Detected if the animation was in landscape or portrait mode
 - Using the class WindowManager & the method "getWindowManager().getDefaultDisplay()"
- 3) Scaled the height of the the height of the rising word animations (and position of the logo image) by the height of the display (in landscape or portrait)

LOGINPAGE

The LoginPage will allow users to login or register. They'll be able to register via our own system or via Facebook, OpenID, or Twitter (etcetera). This page will be skipped if the user is already logged in. The precise widget names are provided for all widgets except the Facebook/OpenID login widgets and the logo image widget.



This page will have regular login page behavior. The important thing to specify here is that when a successful login occurs, the user is connected to their personal databases. When a successful registration occurs, a users' personal databases must be created. If a user provides incorrect or unallowed entries in login or registration, an error notification needs to be provided. This can be in the form of an Android "Toast."

1. *introPage_EditText_username* – standard registration login behavior

2. *introPage_EditText_password* – standard registration login behavior

3. *introPage_Button_loginSubmit*

Input: User clicks on the button.

Logic:

Check if email matches db entry (via query *Users_db*)

Check if password matches db entry for email (via query *Users_db*)

Output:

If email does not match db entry:

Toast "username or password error"

If email matches db entry but password does not match password entry:

Toast "password incorrect"

If email matches db entry and password matches password entry:

Connect to server databases. *User_Performance_db*, *Personal_db*, *Portal_Page_db*

Change the activity to the **HomePage**. (also destroy **MainActivity**)

4. *introPage_Button_registerSubmit*

Output: Personal User databases must be created, on the server and on the local device.

5. *introPage_CheckBox_rememberMe*

Input: User checks on the button

Logic: If off then condition A, if on then condition B

Output: If A erase contents of a **Storage_file_01**. If B store username and password in a **Storage_file_01**. This will then be used to automatically login in future clicks on.

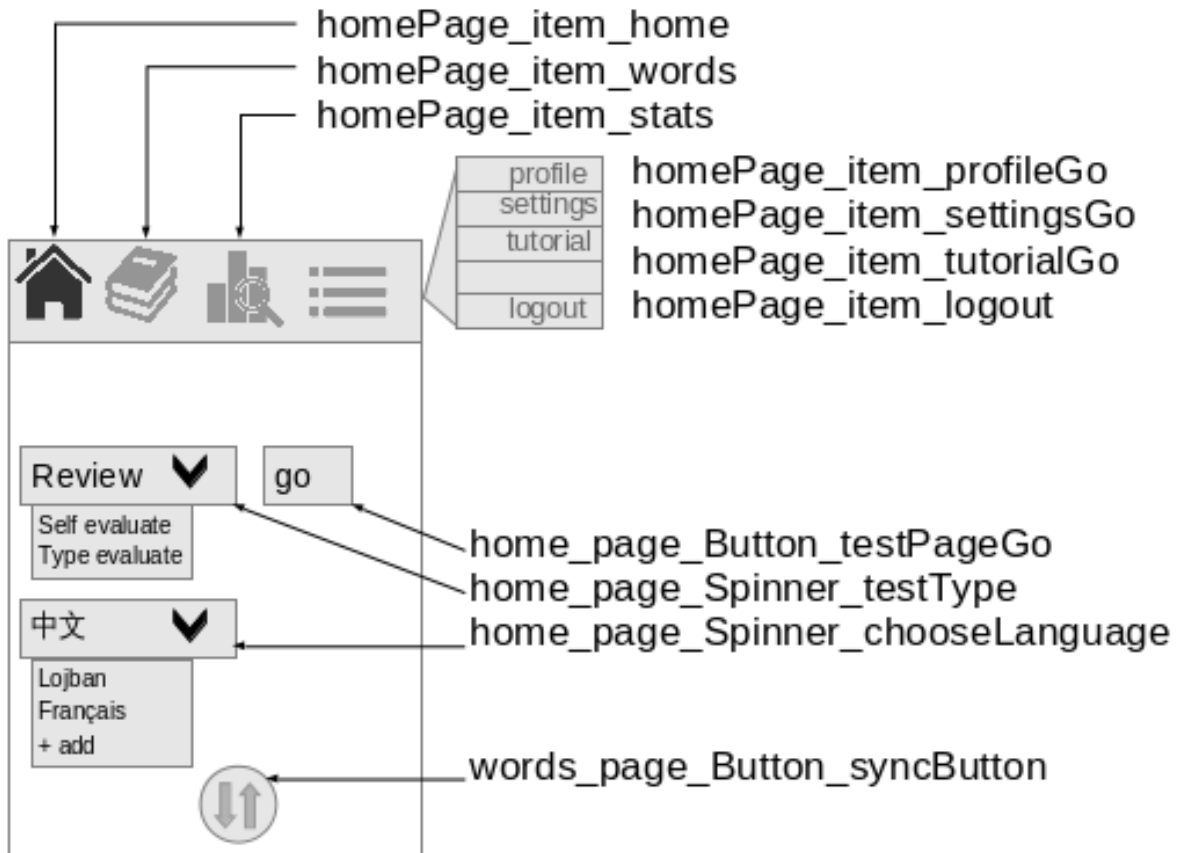
Notes:

1.0 Should also contain a Facebook login button. This has already been POCed by Sandesh Katkamwar.

We should allow Openwords to be started started *offline*.

HOME PAGE

The HomePage will be the central directory page. From here, the user/learner will be able to access three areas. 1) The learning pages, 2) The auxiliary pages (WordsPage/StatsPage/PortalPage), and the ProfilePage & LanguagePage.



1. *homePage_item_profileGo* - Change the activity to the **ProfilePage**.

2. *homePage_Spinner_ChooseLanguage*

Input: User clicks on the *Spinner*

Logic: The Dropdown is populated by the *Personal_db* (which holds the languages the user has chosen).

Output: The languages that will be displayed will be the following. The original language the user decided to learn. And any other language the user has begun. And finally, “+ add” will be the final option – to add a language to learn.

Case A

Input: User clicks on one of the languages.

Logic: *OnClickListener?*

Output: *Personal_db*, and *UserPerformance_db*, and any other user/language specific database will need to be switched (when the user sets up a new language including for the first time, these databases are created as user-language combinations). Then repopulate the appropriate text fields within pages.

An entry in *Storage_file_01* should be altered so that the current language is highlighted. The latter will indicate what language to start on in future initiations of the program.

Case B

Input: User clicks on “New Language”

Logic: **OnClickListener?**

Output: Change the activity to **LanguagePage**.

3. *homePage_Spinner_testType*

Input: User clicks on the button

Logic: The dropdown is populated by the available test settings (right now just Review and SelfEvaluate). In the future, the available tests should be in a language settings file “**LANGUAGENAME_settings**,” peculiar to the L2 (and potentially the L1->L2 combination).

Output: The “**Test_page test types**” are loaded in the dropdown.

Input: User clicks on a test type.

Logic: OnClickListener

Output: The current Test_page test type is stored in **Storage_file_01** (This will be the type of test that the user will go to if the user clicks on *homePage_Button_testPageGo*).

Notes: *There is a problem here that needs to be resolved. If the user is currently in a plate (within a different test type). I think the answer is that we ignore this problem. Wipe the old plate.*

4. *homePage_Button_TestPageGo*

Input: User clicks on the button

Logic: Using an OnClickListener - **Storage_file_01** should indicate several things on the start-up of the **Test_page**.

- 1) Check if a plate is currently active. If it is, go to the current location on that plate (position stored in **Storage_file_01**, plate stored in **Plate_db**).
- 2) If a plate is not active, organize a plate using the **Plate_algorithm**. This means the **User_Performance_db**, and **User_Words_db** and must be opened and read.
- 3) **Test_page** Widget values should be populated (Changed) by instructions from above numbers 1) or 2). In other words the values of the widget values should be changed in response to the 1st elements of the **Plate_db**. Essentially, this should be accomplished automatically, if the **Test_page** is properly designed to respond to the **Plate_db** and **Storage_file_01**.

Output: Change the activity to the **Test_page**. Either a new plate will be opened with X words. Or the user will go to the next problem in the existing plate.

5. *homePage_item_words* - Change the activity to the **WordsPage**.

6. *homePage_item_stats* - Change the activity to the **StatsPage**.

7. *homePage_item_portal* - Change the activity to the **PortalPage**.

8. *homePage_item_profile* - Change the activity to the **ProfilePage**.

9. *homePage_item_settings* - Change the activity to the **SettingsPage**.

10. *words_page_Button_SyncButton*

Input: User clicks on the button.

Logic: Many things will occur.

Output: The following databases will be synced on the server. **Personal_db**, **User_Words_db**, **User_Performance_db**.

Note: Note that the name of widget 8 begins with “words_page” - this is because it appears on the **Words_page**. The features in widget 8 may be 1.0. Syncing best practices and protocols need to be explored. A user may need get a message pop-up indicating that some records will be overwritten (If they have been edited). Perhaps they should be shown which records will be downloaded, and which records will be uploaded.

LANGUAGEPAGE

The LanguagePage will allow the user to select the languages they wish to study. Users will be directed through this page the initial run through the app. They will be connected to the default L1->L2 database.

We will essentially have to keep track of every L2 language that is available for each L1. This will probably be hard-coded at first for each L1. Thus the languages that need to be displayed in this page will be part of the strings of the program.

The user will be able to select one language at a time (This interface on the right should show radio buttons rather than check boxes). The user selects a language. If the user has selected a language, when the user clicks the submit button

widget name: *languagePage_Button_submitLanguage*
that language is placed into the Personal_db (not to be confused with the Users_db), and that language is highlighted as the current language of the user.

What language do you want to learn?

Choose one
(you may choose more later)

Language not on list?

Français (French) ☐
中文 (Mandarin Chinese) ☐
Հայերեն (Armenian) ☐
Latine (Latin) ☐
עברית (Hebrew) ☐
ಕನ್ನಡ (Kannada) ☐

submit

CHOSENPAGE

The ChosenPage is very simple. The page simply confirms for the user, the language they have chosen to study. The user should see this page only once, and it should direct the user to the ProfilePage, through the only widget on the page:

widget name: *chosenPage_Button_continue*

The illustration on page 2 sufficiently represents the ChosenPage.

PROFILEPAGE

The ProfilePage will start very simple. This page will possess a set of questions for the user. Eventually, questions to learn more about the user will be directed to the Portal_page as well. In the 1.0 we will simply need a few simple questions.

The data from the ProfilePage will be stored in the Personal_db.

We have designed several questions. The developer on this page has the right and responsibility to implement these questions with the widgets they would like to use.

SETTINGS PAGE

Android has a built in SettingsPage. However, we've decided to design/develop our own for the following reason. We will be implementing Openwords on other platforms as soon as we can.

Several of the settings listed here do not need to be functional in the 1.0.

There are currently three sections.

- 1) Alerts and Goals.

If we implement a goals/scheduling and notification system, this is where that system will be modulated from. Thus this does not need to be developed immediately. Think notifications in the Android system.

- 2) Portal Page

Setting related to what users wish to receive in the Portal page. This data can go to the Personal_db.

- 3) Plate word selection algorithm

Several options will be available for how the user cycles through word problems. Our 1.0 will have a default Plate word selection algorithm. Additional options can be controlled here.

WORDSPAGE

The WordsPage is the location where users will acquire new words to study. This page will connect to different L1 → L2 databases (but primarily the default word db), allow the presentation and selection of different selections of particular L1 → L2 databases - and provide users the autonomy they need to search for and add new words. For example, users will need to A) Be able to (build and) connect to their alternative L1 → L2 database B) view unique selections of database entries, such as a class list of words C) and of course be able to just identify and download the default sequence of lists of words.

Note that the home page widget does not specify what page it belongs to, the reason for this is that this widget is located in many pages. Other widgets are located in more than one page.

The words page is fairly complex and not all elements will be added immediately. Sets will need to be identified with a specific kind of word identifying file on the server (a set file simply needs to identify the database and the words in the set). Additionally, the sets download interface will need to allow users to view the words from the sets, and highlight those they want and don't want – download every word in the set, or only those that are selected. All words in a set are selected by default.

Whenever words are added to the User_Words_db, the system should clear or refresh the relevant bin and provide the user with a toast, indicating that the words have been added.

1. *wordsPage_Button_homeGo* - Change the activity to the **WordsPage**.

2. *wordsPage_Button_communityGo* - Go to the L1 → L2 community webpage.

3. *wordsPage_Button_syncButton* - This same widget appears in the home page. This button syncs several items in the server and the local device. The following databases will be synced on the server. *Personal_db*, *User_Words_db*, *User_Performance_db*. For the most part this will mean that the server database will need to be overwritten by the data on the local device. The exact logic still requires some more thought. We'll need to work out when to overwrite and when not to overwrite when the user moves from one device to another.

4. *wordsPage_Button_myWordsGo* will currently go nowhere. It will go to a page which will allow a user to browse and search their corpus.

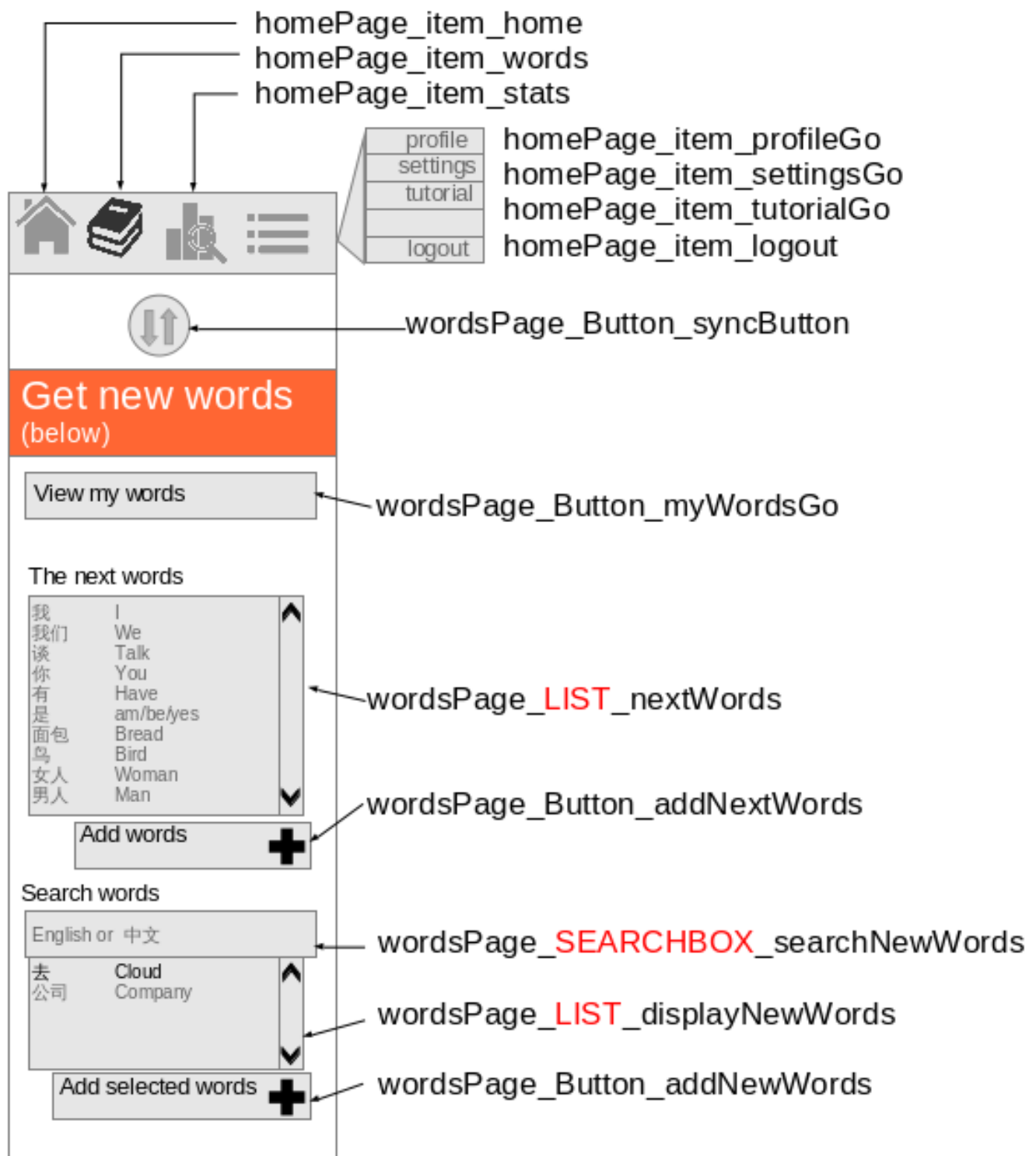
5. *wordsPage_LIST_nextWords* - This list will contain the ten next words that the user will download to their personal words (*User_Words_db*). This list will need to be filled from a query to the *Global_Words_db*. There will be a default word sequence (the order that the words are presented to the user) recorded in the *Global_Words_db*. A query of the *User_Words_db* will show the ten lowest unfilled words in that list. So for example if the user has words 1-21,23,26. Then Their next ten would be 22,24,25,27,28,29,30,31,32,33. A query of the *Global_Words_db* will select these words. To populate this widget, *Storage_file_01* can be filled with the words' rank, L1, and L2 fields.

6. *wordsPage_Button_addNextWords* - When the user clicks on this button, the same query process as described in the previous widget must take place, **but additionally**, the appropriate records from the *Global_Words_db* must be downloaded into the *User_Words_db* (server and local device).

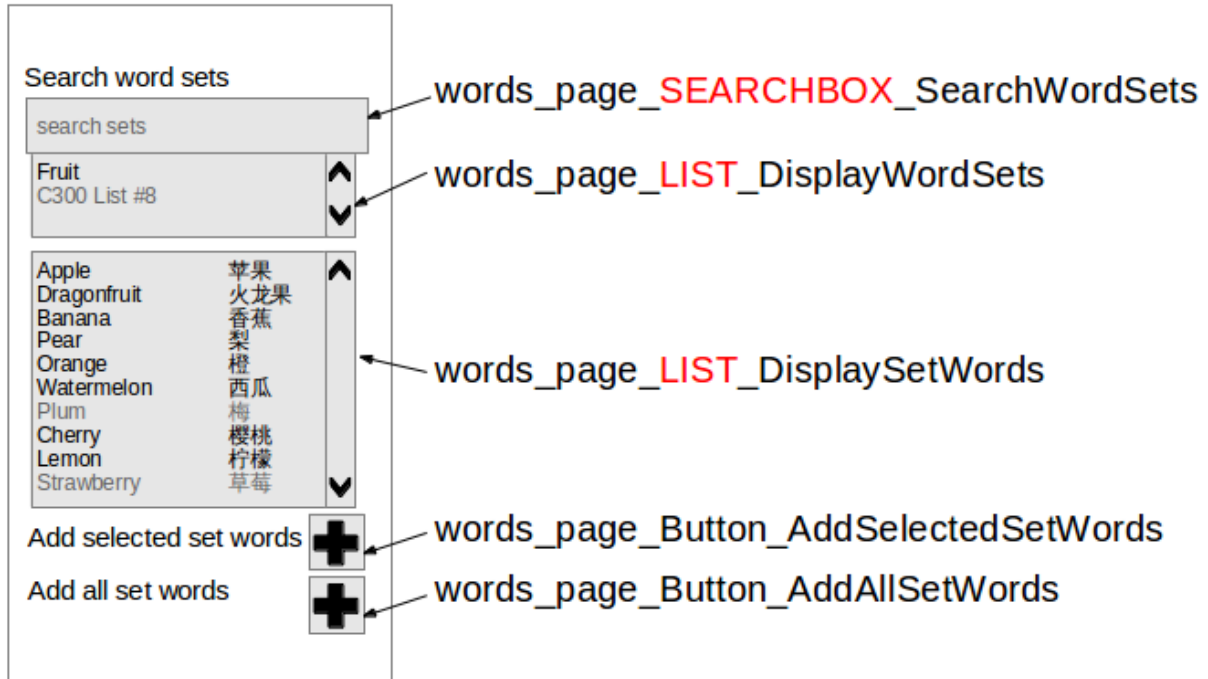
7. *wordsPage_SEARCHBOX_searchNewWords* - Here the user will enter text and when they press enter, a query based on the users' entry will be performed on the *Global_Words_db* (on both L1 and L2 fields) and the L1 and L2 fields of each record will be displayed below in the widget named *wordsPage_LIST_displayNewWords*.

8. *wordsPage_LIST_displayNewWords* - This list will receive input from queries performed in *wordsPage_SEARCHBOX_searchNewWords*. *Storage_file_01* can hold the information that needs to be displayed. The user should be able to highlight words that they wish to add to their list, again *Storage_file_01* can represent which words (identified by their id or rank) of those highlighted have been chosen for download.

9. *wordsPage_Button_AddNewWords* - reading from *Storage_file_01*, when the user clicks on this button the words that have been identified will be queried from the *Global_Words_db* and downloaded into the *User_Words_db* (server and local device).



The following illustration (below) displays additional functionality that could be added to the Words_page, namely a word set search tool. This functionality is not needed in the 1.0, but it will be very valuable eventually. Sets will need to be identified via some mechanism. Possibly, sets could be referenced by a Set_file, which identifies the words in the set (and database from which they derive).



STATSPAGE

The StatsPage will start out very simple. The purpose of this page is to display various kinds of progress. Now, it could simply display the number of words the user knows, and the number of words they will need to learn in order to reach the next Birthday.

A progress bar could display the progress to the next Birthday.

statsPage_PROGRESSBAR_birthdayProgress – Displays progress on the way to the next birthday.

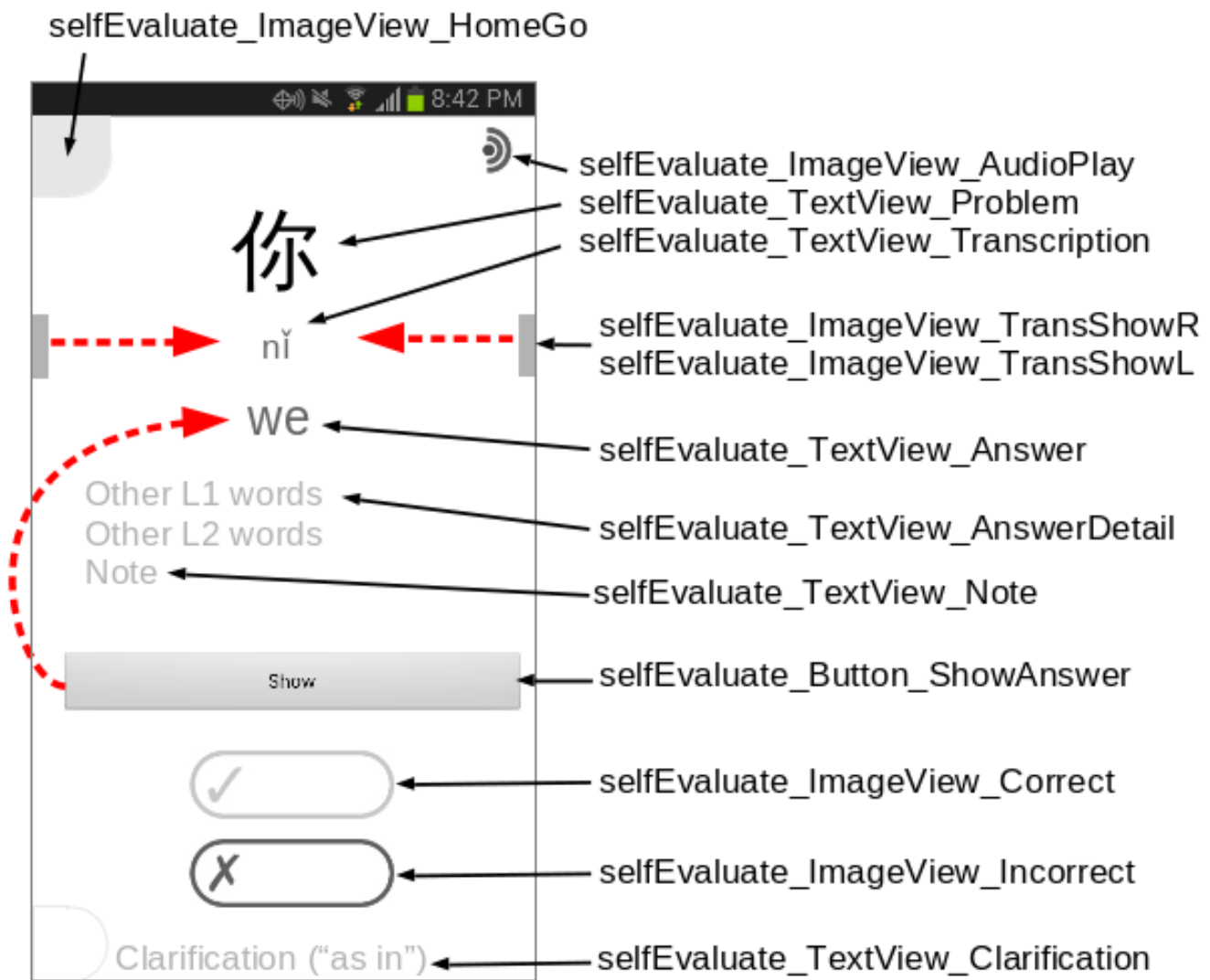
The percentage of the bar filled = Number of words beyond the current VocabularyAge divided by the number of words needed to achieve the next age.

statsPage_TextView_birthdayProgress – Displays the number of words needed to achieve the next Birthday, and the number of words so far learned to get there.

SELF-EVALUATE PAGE (NAME IS JUST SELF-EVALUATE)

The SelfEvaluate page is the main testing page in the 1.0 app. We expect the user to spend the most time here. The entire page is set up on an Android “ViewPager.” Gesture listeners are set up to respond to a swipe to the right or left – the screen moves forward in the stack of cards, or backwards depending on the direction of the swipe.

Like all test pages, the SelfEvaluate page is organized around the concept of “the Plate” and a corresponding Plate_db. A plate is 15 problems long, 15 words. The user goes through these 15 words and depending on the test (or Review) performs the required actions and advances through the problems. As the user proceeds, a variable in Android’s “SharedPreferences” keeps track of the progress through the plate (1-15). If the user is on the 4th problem/card of a plate the value in SharedPreferences is used to read from the appropriate line in the Plate_db to populate the widgets that need to be displayed in the SelfEvaluate page.



On Gesture Listeners:
Screen respond to swipes to the
right and left.

The behavior of the SelfEvaluate page is reasonably complex. We have worked out a procedure for its operation, and this will be easier to explain verbally. However, some notes should be recorded here.

The display of some widgets needs to depend on the language that is being learned. For example for people learning Chinese, the following widgets should be displayed.

```
se_page_ImageView_TransShowR  
se_page_ImageView_TransShowL  
se_page_TextView_Transcription
```

For other languages where the transcription is not relevant, this should not be displayed.

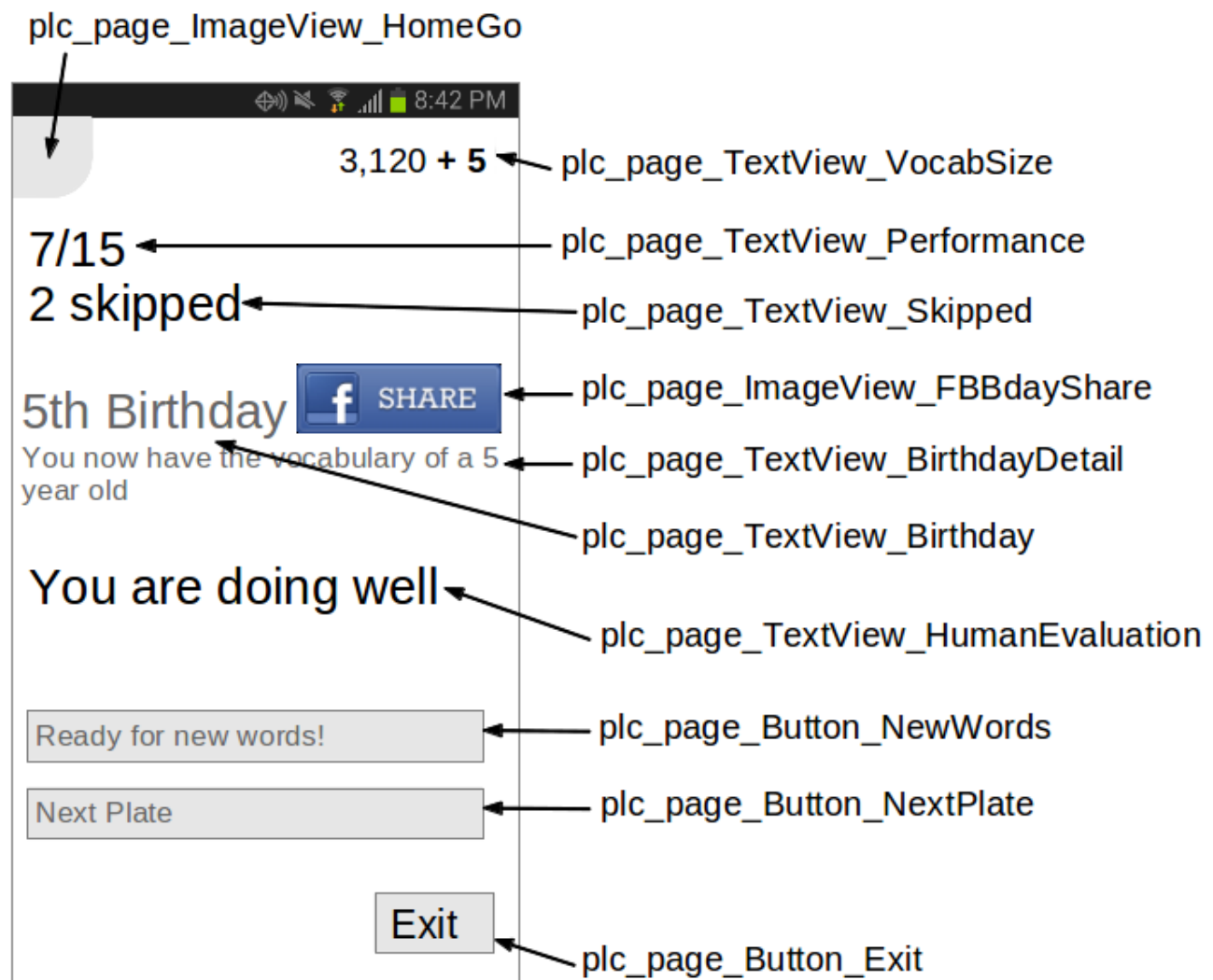
REVIEW_PAGE

The Review_page is very similar to the SelfEvaluate page except that it does not require a show button to display an answer, or correct/incorrect buttons (image views) to allow the user to self evaluate. The Review_page, simply need to have several of the relevant fields from the User_Words_db displayed in TextView widgets to the user. If nothing else this could be the SelfEvaluate page without the show button, correct or incorrect buttons, or the buttons which display the transcription.

PLATE_COMPLETION_PAGE

The Plate_completion_page will need to display differently depending on the test type that has just been completed.

Additionally, The Plate_completion page will display certain widgets about progress depending on whether the user passes certain performance thresholds. For example, every time a user passes a birthday, the Facebook share Birthday button needs to appear along with the year. This should stay until they've shared the birthday, or the next Birthday is achieved.



On Gesture Listeners:
Only allow movement to the left.

Every time a user finishes a “plate” (15 problems) the user should see the `Plate_completion_page`. This page will be populated by information from the `Plate_db`, just like the previous 15 slides. Some widgets will be populated from the `User_performance_db`

plc_page_TextView_VocabSize – The current vocabulary size should be calculated from the `User_performance_db`, but should be stored somewhere else as well. For example, we could place this number in “`SharedPreferences`.” When a word is added (or removed) from “being known” by the user, then this variable should be updated. Calculations of `VocabSize` should occur on entering the `Plate_completion_page`, including either additions or deletions depending on performance on the 15 words of the Plate.

plc_page_TextView_Performance – Calculated from the `Plate_db performance` field, this will display the number of words the user got correct out of the total plate size (15).

plc_page_TextView_Skipped – Calculated from the `Plate_db performance` field, this will display the number of words the user skipped.

plc_page_ImageView_FBBdayShare – This button will allow a user to share their progress, once they achieve a new Birthday. The Birthday will be calculated by the Vocabulary size of the user in comparison to an array of Vocabulary age values. The array will need be a variable in our program. Sherif (1950) gives us values for the first 4 years (we'll get better values in the future). Age 1 = 3 words, Age 2 = 272 words, Age 3 = 896 words, Age ~4 = 1222 words. The variation between individuals is incredible, but we can pick a reasonable median number. A second reference put the vocabulary of average American English speakers at 21,416 by age 16, ascending roughly linearly from Age 4. Thus we get some rough estimates for the remaining intervening ages.

Age 5 = 2905 words

Age 6 = 4588 words

Age 7 = 6271 words

Age 8 = 7953 words

Age 9 = 9,636 words

Age 10 = 11,319 words

Age 11 = 13,002 words

Age 12 = 14,685 words

Age 13 = 16,368 words

Age 14 = 18,050 words

Age 15 = 19,733 words

Age 16 = 21,416 words

Once a users `VocabSize` increases above one of these values, this `ImageView` should be displayed. When clicked the `ImageView` should share a message on Facebook. A Twitter equivalent could be included.

plc_page_TextView_Birthday – Will be populated from a string and a calculation from the `User performance db`.

plc_page_TextView_BirthdayDetail – same as above.

plc_page_TextView_HumanEvaluation – Doesn't need to be on 1.0. A string, a human readable statement about progress. Use your creativity here on what can be said and under what conditions.

plc_page_Button_NewWords – Take the user to the `Words_page`.

plc_page_Button_NextPlate – Take the user to the next plate. All the necessary calculations must take place for the construction of a plate. The `User_performance_db` along with the `Word_selection_algorithm` will be used to select words for the next plate.

plc_page_Button_Exit – Take the user to the `homePage`.

OPENWORDS_DATABASES

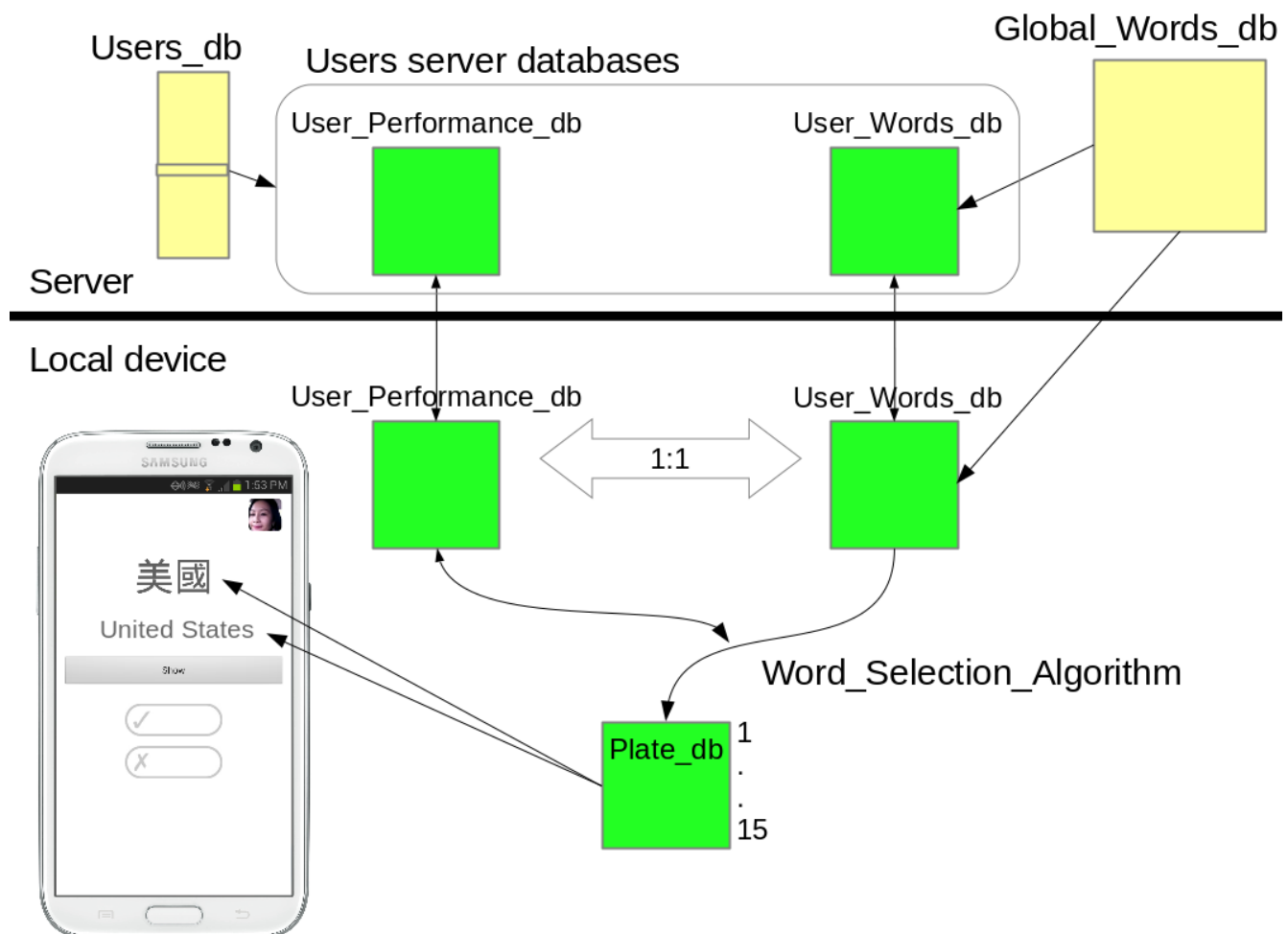
The following is a graph showing a global overview of the Openwords databases, on the server and the local machine. This graph does not contain databases related to the Portal_page. The fields contained within these dbs will be displayed in the following pages.

The Global_Words_db is a superset of the User_Words_db. The User_Words_db is mirrored on the server and the local device (alternatively the local User_Words_db is a subset of the server side User_Words_db). The Global_Words_db has the same structure (or is just the same) as the Openwords file format. Thus, it is a complex relational db.

The User_Performance_db is much simpler than the User_Words_db, and contains only one table, holding several fields of user/learner performance data. The User_Performance_db has a 1:1 relationship with the **connections** table of the User_Words_db (connected by the id field of each table).

The Plate_db is simply a holder database. This db holds the information needed to construct the 15 problems in the “plate.” The plate is the 15 vocabulary problems the user cycles through.

Finally, the Users_db holds fields related to the user. The purpose is to allow the user/learner to connect with their dbs (Their dbs will need to be identified by entries in the Users_db).



GLOBAL_WORDS_DB (OPENWORDS FILE FORMAT)

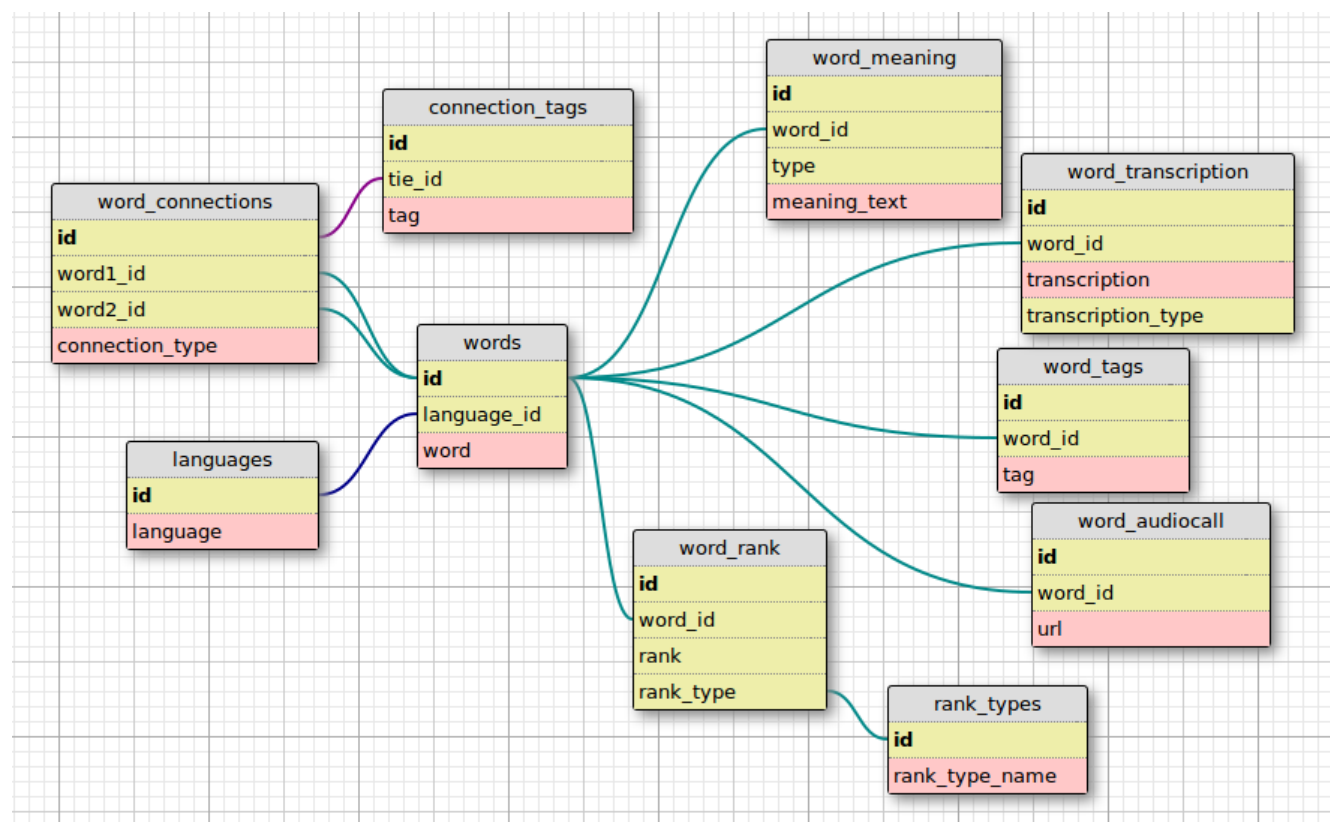
The Global_Words_db is a database on the server, it has the same structure as the Openwords file format. There are two main fields of the Global_Words_db, playfully known as the “Tower of Babel” and the “God table.” The Tower of Babel is the 'words' table. This table contains all the words of all the L1 → L2 combinations (or the default versions of them). There is a 'languages' table that identifies the language of a word. The God table is the 'connections' table. This table is the table most comparable to a “flash card” - matching one word in an L1 with one word (or words) in an L2. The ids of the records of this table match in a 1:1 relationship with the ids of the User_Performance_db.

The documentation for the Openwords_file_format (Global_Words_db) is located here.

https://github.com/Openwords/Openwords_file_format/wiki

The 'word_meaning' table has L1 word *definitions* and L1 word *clarifications*. The table can also hold L2 word clarifications and meanings (expressed in either the L1 or L2). Clarifications on the connection (between the L1 and L2 words) need to be added. Currently we only have a connections tag.

We have recently added one more table to complete this database, which makes the Openwords file format work fully with the Openwords app. A table containing commonality rank (of the connection or word) or a ranking of words in order of their sequence to be learned. These fields are needed when the user selects new words (and queries the Global_Words_db) within the Words_page. Now, the current version of the Global_Words_db will support most activity within the Openwords program (once downloaded to the device).



The 'word_transcription' table contains transcriptions of the words. These include the IPA

transcriptions, relevant for all words in all languages. This table will also include transcriptions relevant for only particular languages, such as the Pinyin Romanization system for Mandarin Chinese.

The 'word_audiocall' predictably contains the url or path to the audio file to be played for a word.

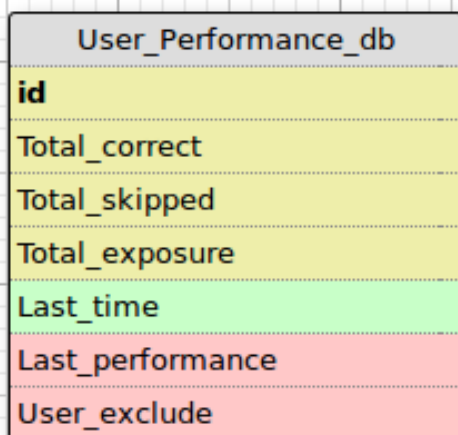
The 'word_rank' table contains the rank of the word in the language. This could be the commonality of the word in the language, or represent the sequence of words learned by a child in development. Ultimately it is essentially the default decision of the order in which words are learned – this rank determines the order in which learners go through words in the language (learners start with lower ranks and move up).

Finally, there is a 'word_tags' table to place miscellaneous tags onto words.

The User_Words_db is a subset of the Global_Words_db, including only some of the records, and only words that are within the L1 and L2 languages.

USER_PERFORMANCE_DB

The User_Performance_db (below) has several fields. The id field matched 1:1 with the 'connections_table' of the Global_Words_db. The fields are as follows. Total_correct holds the number of times the user has gotten this word correct. Total_skipped is the number of times the user has skipped the word. Total_exposure is the total number of times the user has been exposed to the word (the total number of wrong answers = Total_exposure - Total_skipped - Total_correct). Last_time contains the time_stamp of the the last time the user saw the word. Last_performance is how the user performed in their last time exposed to the word - correct, incorrect, or NULL (skipped). User_exclude is a field to record a users preference that they no longer wish to see the word in tests.

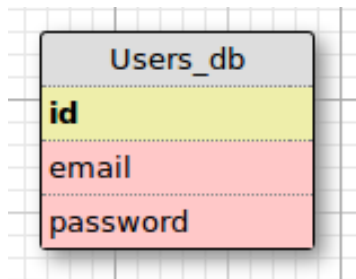


User_Performance_db
id
Total_correct
Total_skipped
Total_exposure
Last_time
Last_performance
User_exclude

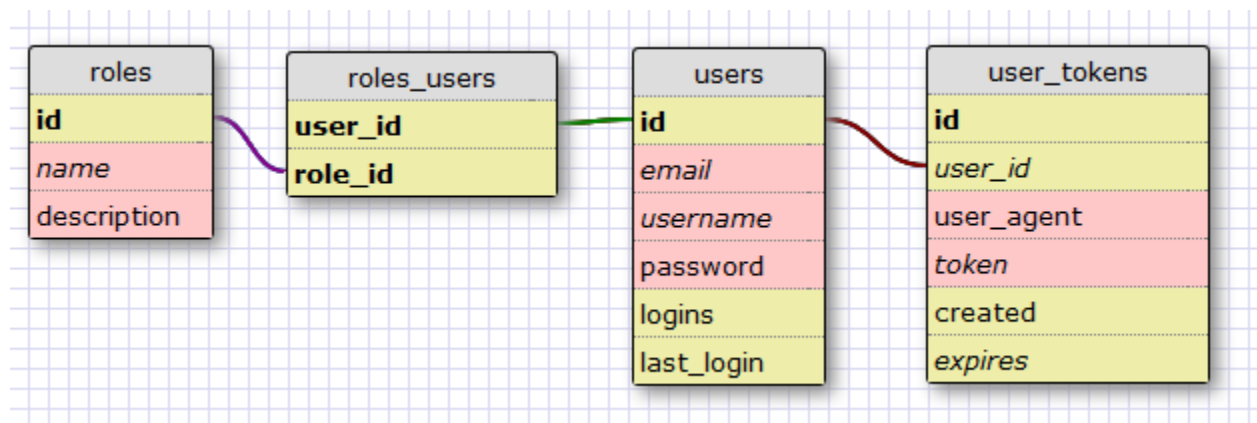
USERS_DB

The Users_db (below) has increased in complexity slightly. Originally this db had very few fields, simply allowing a user to associate with their databases on the server. There is an 'id' field (identifying the user), an 'email' field, and a 'password' field. The original Users_db and modified Users_db are listed below. The modified db allows for role specification of the user with the 'roles' table (e.g. Teacher, Student, Individual), and 'user_tokens' table allows for automatic login to proceed in a secure manner.

Original version



Modified version



PLATE_DB

The Plate_db (Right) has fields to hold data to populate 15 individual problems or cards. The user/learner runs through 15 individual problems, reviewing, self evaluating, or being automatically evaluated and the Plate_db supplies the raw data for the problems. And when the user performs (e.g. gets a problem correct or incorrect) the Plate_db collects that data.

The Plate_db is an intermediary to the Front-end. At the end of 15 problems or cards, the user reaches a 'plate completion page' and the data that were recorded to the Plate_db are sent to the User_Performance_db. And the Plate_db is again filled (using the Word_Selection_Algorithm) from the User_Words_db.

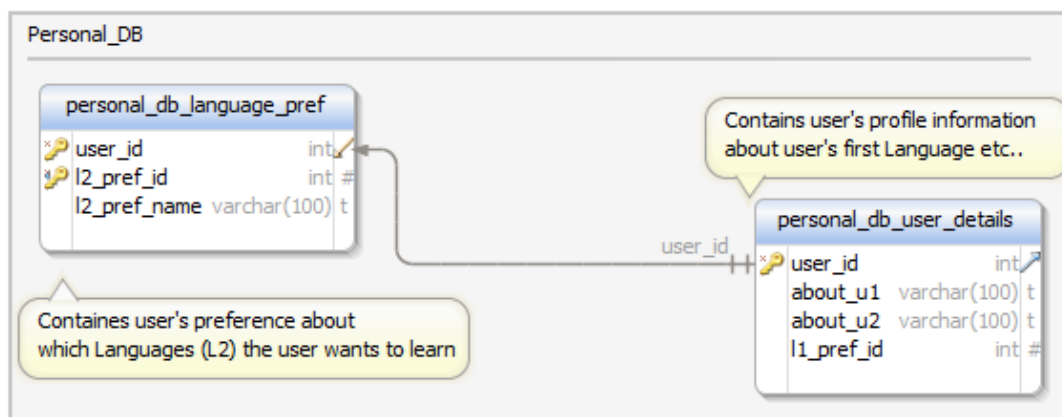
The 'connection_id' holds the id of the record in the 'connections' table (of the Users_Words_db) that corresponds to a problem in the user's current plate. If necessary the 'connection_id' could be used to pull more data from the Users_Words_db. 'Performance' holds the performance of the user for the word TRUE (correct), FALSE (incorrect), NULL (as yet unseen/skipped). 'Test_type' contains the type of test for which the Plate_db has been filled (for the 1.0 this could be Review, SelfEvaluate, in later versions we can include Hearing, Minimal_pairs, L2_recognition_type etc.). 'Audio_call' contains the url or path to the audio file. 'Clarification' contains any clarification that might be necessary for a particular word (or connection). Three generic fields allow a test to hold several separate bits of information for display in review or problems (Problem_01, Problem_02, Problem_03). Two generic answer fields exist to allow the same flexibility for answers (Answer_01, Answer_02).

Plate_db	
id	
Connection_id	
Performance	
Test_type	
Audio_call	
Clarification	
Problem_01	
Problem_02	
Problem_03	
Answer_01	
Answer_02	

PERSONAL_DB

Personal_DB is a collection of tables aimed at storing each user's profile/personal information (including the words db they are working from, e.g. Default Deutsch -> English). The tables that constitute this collection are:

- *personal_db_language_pref* (*user_id*, *l2_pref_id*, *l2_pref_name*) :- Stores all the chosen languages that a user chooses to learn . There can exist multiple choices against one "user_id". "l2_pref_id" and "l2_pref_name" store the language id and and the name of the language respectively. As is clear from the above description, the table necessarily has a composite primary key <user_id,l2_pref_id>.
- *personal_db_user_details*(*user_id* , *about_u1* , *about_u2* , *l1_pref_id*) :- Stores the profile details of the user. Primary Key is the "user_id" attribute. Currently what information we wish to keep has not been completely determined. However, "l1_pref_id" attribute contains the language ID of the user's 1st language (mother tongue/L1). The value in this attribute will facilitate in generating the options for the user when the user wishes to choose language to learn on the 'LanguagePage'



Generated using DbSchema

TUTORIAL_PAGE

The Tutorial_page will provide very simple instructions on how to use the Openwords app. Our 1.0 version of the the Tutorial page will simply be a series of *flat* images showing:

- 1) Use of the testing modules.
- 2) The Words_page.
- 3) The StatsPage.
- 4) The Portal_page.

PORTALPAGE DATABASES

The Openwords app will make revenue through targeted marketing, with information gathered from the Portal_page and ProfilePage informing what will be shown on the Portal_page. The following databases will hold and record the necessary information.

WORD_SELECTION_ALGORITHM

We will implement a very simple algorithm for the 0.9. We have decided that this will be an object of scrutiny, user testing, and research.