# Word Flashcards Project

User can practice memorizing meanings of words everyday using flashcards and add new flashcards to the list.

## Adding a New Flashcard:

User can add a new flashcard, which includes the word, its definition, and an optional example of the word usage.

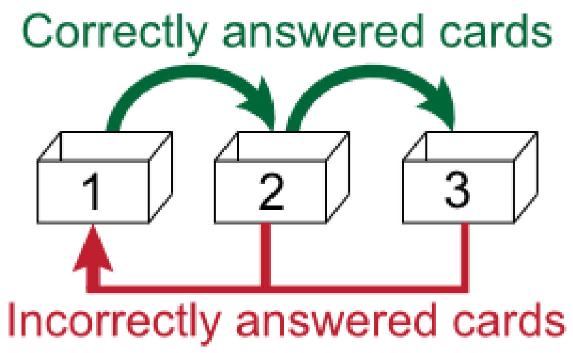
A new flashcard should first be saved to server (using */data/flashcards* route) and then to local database.

## Reviewing a List of Words:

User can start reviewing the list of words one by one. The list of words should be displayed based on a simplified version of the Leitner method, which will be explained below in more details.

On every displayed flashcard, the word should be shown, and user can choose to show the definition and example of the word. Also it should be possible to play the audio pronunciation of the word. When the meaning is shown, user should answer whether he/she knew the correct meaning of the word or not. After this answer, the next flashcard will be shown.

Any word can be in three different levels based on the memorization performance of the user, as shown in the figure below. At first, all the words are in level 1. If a user knows the meaning of a word, the level of the word will increase by one. On the other hand, if a user does not know the meaning, the word will move to level 1 again.



Every time a user starts reviewing the list of words, a subset of the words should be selected to be displayed based on the levels of words. The lower level words should be displayed earlier than the higher level words.

Level 1 words should always be in the displayed subset. A level 2 word should be displayed if the last time user reviewed it has been more than one day ago. And a level 3 word should be displayed if the last time user practiced it has been more than three days ago. In this way, the app makes sure that lower level words are practiced more frequently.

When a flashcard is reviewed by user, its level should be updated first in server and then in local database.

## Reminding the User to Practice Words Every Day:

Once every day, the app should remind the user to review flashcards by showing a notification in the notification bar.

## User Sign-Up:

A new user should be able to sign up with basic user info, at least including email, name and password. The sign-up should be done in communication with server (using */users/register* route).

## User Sign-In and Downloading Flashcards:

When user wants to sign in, if the user exists in local database, it is assumed that flashcards of the user are also available in the device database and user can enter the app without any extra server communication.

If user data is not found in local database, the app should perform an online sign-in for this user (using */users/login* route). If online sign-in is successful, it means this user has signed up before, so the app should then request the list of previous flashcards of the user from server. The user data and the downloaded flashcards should then be saved to local database.

**Other Requirements and Notes to Consider:**

1. For generating and playing the pronunciation of a word, you can use Android TextToSpeech API.

2. For setting a process to be executed sometime in future (e.g. reminding the user to review words), the AlarmManager service can be used in conjunction with BroadcastReceiver for your code to receive the alarm. For showing notifications, the Notification API can be used.

3. For list, database, and networking functionalities, you should use the same classes/libraries used for your assignments; that is RecyclerView for lists, Room for database, and Volley for networking.

4. Similar to the last assignment, the Backendless service should be used for the server-side REST API. Also all the server requests should include authorization credentials in the header, except for the sign-up and sign-in requests.

5. All processes in the code which require accessing some form of external resources should use a separate thread other than UI thread, which usually can be handled using the AsyncTask class. The main examples are accessing database, reading from or writing to files, and network operations.

6. While details of UI and view design of the app are not fully explained in the project description, the design and view of your app should be reasonable and suitable. It is recommended that you check a few examples of flashcard mobile apps to have a better design in mind for your app.

7. Some operations in your app may need special permissions. Make sure to include related permissions in the app manifest file, and implement runtime permissions for the cases needed like accessing device features and storage.

8. All possible errors, exceptions, and special cases in your app should be handled properly, including different errors caused by database, network and file operations, and other special functionalities in your app.

9. Your code should be neat and organized. Poor indentations, bad variable names, dirty and poorly-written code, unorganized code files, ignoring design patterns and principles you learned in the course, etc. can all reduce from your final score. Also in places that the code requires some clarification, you may need to add comments to your code.

10. You should not use any third-party library or package in your project, except for special cases which are allowed clearly in the project description or approved by the course team.

11. Any reasonable optional features implemented in your project beyond the main requirements are welcome and may add some extra points to your final project score. However, you should cover the main requirements before adding extra features.

12. While it is completely acceptable to search online or consult with your friends about some of your project's functionalities, your project code should be completely your own work, and you should completely understand every detail of the code.

13. Finally, you should respect all other guidelines and standards suggested by instructors throughout the course, either in the feedbacks to your assignments or in the classes and forum posts.

14. Similar to your assignments, your submitted project folder should also include the final APK file of your project.