

# PROJECT REPORT

## ABSTRACT

---

Languee, a language learning tool on android platform using Java programming language is designed for users to improve their learning efficiency. Languee includes 4 major features - writing, flashcard, AI revision, and quiz. It supports both Traditional Chinese and English locale setting. Daily reminder can be set to remind users continue their learning progress.

## HOW TO USE

---

For installing Languee application, an APK file can be found in the “build” folder. Move “Languee.apk” file to the android phone. Open and install it using the default file manager application on the android smartphone.

For testing, the source code is provided in the “source” folder. Select the “source/App” folder in Android Studio to test.

Tested Android Studio Version	Hedgehog   2023.1.1 Patch 2 (Windows 11 Enterprise 23H2) Jellyfish   2023.3.1 (macOS Sonoma 14.4)
Special Dependencies	com.google.ai.client.generativeai:generativeai:0.3.0 com.google.guava:guava:31.0.1-android org.reactivestreams:reactive-streams:1.0.4 commonmark:commonmark:0.21.0

For using Languee, a complete user manual is provided and not covered here. Please read “README.md” for details instead.

## BASIC STRUCTURE

---

### LAYOUT

In the project, there is only one activity, “MainActivity”, used throughout the application life cycle. For viewing different pages, fragments are used. “NavigationView” by google android material is used for navigating to different pages. When a navigation item is selected, it will create a new Fragment based on which item is clicked. Navigation outside the navigation panel can also be done using the public method in “MainActivity”

```
public void NavigateToFragmentByFragment(Fragment fragment)
```

The layout design in most of the fragments are “RelativeLayout”. For some pages, such as the flashcard main pages, dynamic loading of the flashcard-sets is done by creating a layout resources (e.g. flashcard\_set) and adding it into a linear layout view inside the main fragment.

### LOCALIZATION

Localization is mainly done by resources switching where most of the displayable texts are stored in a value resources file under “resources/values” folder. An additional value folder, “values-zh” is created for storing the Chinese version of the texts. The “UpdateLanguage()” public method in “MainActivity.java” and “AppLanguage.java” shows the implementation details.

## NOTIFICATION

Daily reminder is created for reminding users to continue their learning progress. A notification will be pushed every day at specific time configured by the users in setting page. This scheduled notification is implemented using the idea of alarm manager from the system service and broadcast receiver. A notification channel is created when the main activity is created. When the application is about to be terminated, a specific time is set to the alarm manager for triggering a pending intent. When the pending intent is triggered, the intent itself will be passed to “MyBroadodcastReceiver” and the notification push is handled inside the receiver class using “NotificationCompat.Builder”. It is noted that the custom receiver has to be declared in the manifests file first.

IMPORTANT: Due to the implementation of repeating notification using “setRepeating()” method by AlarmManager, the time of the notification pushed is not exactly match with the user setting, especially when using an emulator in Android Studio [1].

## DATA STORAGE

All the data in the application, including quiz, flashcards, writing and settings are stored locally in the shared preferences. No special database is implemented in this project. For storing an image of a flashcard, the image data is encoded to a string type using Base64 encoding scheme. Hence, even if the original image is deleted by the users, it has no effect on the image in the flashcard feature. For quiz feature, a copy of the flashcards is also saved to another shared preference. These implementation details avoid referential integrity problems.

<table><tr><th>NotePref</th></tr><tr><td>NoteCount : int note_title_i : string note_content_i : string note_datetime_i : long</td></tr></table>	NotePref	NoteCount : int note_title_i : string note_content_i : string note_datetime_i : long	<table><tr><th>settingsPref</th></tr><tr><td>language : string geminiAPIKey : string enableNotification : bool notificationHour : int notificationMinute : int</td></tr></table>	settingsPref	language : string geminiAPIKey : string enableNotification : bool notificationHour : int notificationMinute : int	<table><tr><th>QuizSetPrefs</th></tr><tr><td>QuizSetCount : int quiz_set_title_i : string quiz_set_question_count_i : int quiz_set_correct_count_i : int quiz_set_source_count_i : int quiz_set_datetime_i : long quiz_set_question_i_j : string quiz_set_correct_answer_i_j : string quiz_set_user_answer_i_j : string quiz_set_source_i_j : string</td></tr></table>	QuizSetPrefs	QuizSetCount : int quiz_set_title_i : string quiz_set_question_count_i : int quiz_set_correct_count_i : int quiz_set_source_count_i : int quiz_set_datetime_i : long quiz_set_question_i_j : string quiz_set_correct_answer_i_j : string quiz_set_user_answer_i_j : string quiz_set_source_i_j : string	<table><tr><th>FlashCardPrefsi</th></tr><tr><td>CardCount : int set_i_card_j_setID : int set_i_card_j_front : string set_i_card_j_back : int set_i_card_j_image : string set_i_card_j_datetime : long</td></tr></table>	FlashCardPrefsi	CardCount : int set_i_card_j_setID : int set_i_card_j_front : string set_i_card_j_back : int set_i_card_j_image : string set_i_card_j_datetime : long
NotePref											
NoteCount : int note_title_i : string note_content_i : string note_datetime_i : long											
settingsPref											
language : string geminiAPIKey : string enableNotification : bool notificationHour : int notificationMinute : int											
QuizSetPrefs											
QuizSetCount : int quiz_set_title_i : string quiz_set_question_count_i : int quiz_set_correct_count_i : int quiz_set_source_count_i : int quiz_set_datetime_i : long quiz_set_question_i_j : string quiz_set_correct_answer_i_j : string quiz_set_user_answer_i_j : string quiz_set_source_i_j : string											
FlashCardPrefsi											
CardCount : int set_i_card_j_setID : int set_i_card_j_front : string set_i_card_j_back : int set_i_card_j_image : string set_i_card_j_datetime : long											
<table><tr><th>MessagePrefs</th></tr><tr><td>MessageCount : int message_sender_i : string message_text_i : string message_datetime_i : long</td></tr></table>	MessagePrefs	MessageCount : int message_sender_i : string message_text_i : string message_datetime_i : long	<table><tr><th>SetPrefs</th></tr><tr><td>SetCount : int set_count_i : int set_title_i : string set_datetime_i : long</td></tr></table>	SetPrefs	SetCount : int set_count_i : int set_title_i : string set_datetime_i : long						
MessagePrefs											
MessageCount : int message_sender_i : string message_text_i : string message_datetime_i : long											
SetPrefs											
SetCount : int set_count_i : int set_title_i : string set_datetime_i : long											

## FILTERING AND SORTING

In all the features except Ai revision, users can create and save multiple instances of the item. Filtering by title of the item using keywords with case-insensitive exact match is provided for users. Flashcard and quiz can further filter by the contents. Sorting is also available which users can sort the list by date (from latest to oldest) and title (alphabetic order).

## HOME PAGE

When first enter the application, users are always directed to the home page, where they can view their learning progress and shortcuts to perform actions. The progress show the number of articles, quiz attempted and flashcards information. For action part, as looking for the meaning of a word is one of the most frequently action performed by language learner, a shortcut for asking AI to explain a word is created. Four predefined shortcuts is created in the bottom as well.

## MAIN FEATURES

---

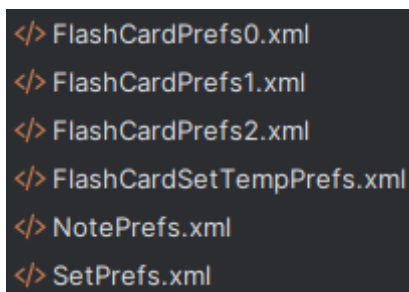
### WRITING

The writing feature is basically a simple note taking function. Users can write article and each article wrote by users are consider as an object of a java class “Note”, which contains title, content, and datetime. “WriteMainFragement” is used for showing the main page where users can view and select their articles. To delete a set, press and hold the item for a short time and a dialog will pop up to ask for the confirmation to delete the article. “WriteCreateFragement” is used for editing and creating a new article. Besides, a check button is provided for quickly navigating to the ai revision page for proofreading. The content of the article is passed to ai fragment using Bundle.

### FLASHCARD

The flashcard function consists of 3 fragments, main, create and view, as well as two java helper classes “Flashcard” and “FlashcardSet”. Main fragment is similar to the one in writing feature. In create fragment, users can add a card and fill in the front and back text. Additionally, users can add an image by selecting photos from the android gallery using “PickVisualMediaRequest”. In view fragment, users can click edit button to edit the flashcards, tap the flashcards to flip it and view the back text, and swipe to left or right to view the next or previous card. A simple animation is done here using animate() function of a view object. Swiping is implemented using a class “OnSwipeTouchListener” inherited from “OnTouchListener”.

In data management, the image shows an example of 3 flashcard-sets. All the sets basic information are stored in “SetPrefs”. It stores the information of the number of sets created and the titles, number of cards and datetime created of each set. Shared preferences with prefix, “FlashCardPrefs” are used to store the information of the cards of a set, including front text, back text and image data. For passing the selected set information to view fragment, an additional “FlashCardSetTempPrefs” is created so that data consistency can be assured.



```
<> FlashCardPrefs0.xml
<> FlashCardPrefs1.xml
<> FlashCardPrefs2.xml
<> FlashCardSetTempPrefs.xml
<> NotePrefs.xml
<> SetPrefs.xml
```

NOTE: Bundle is not used here because our group has not clearly clarified how data are passed to different fragments in early stage. As it can function normally, we decided not to modify it and to leave it as an optional improvement in the future when we have time.

### QUIZ

Quiz feature can be considered as an extension of the flashcard feature, where users can make use of the premade flashcard-sets to generate questions to answer. There are 3 fragments considering quiz features as well. Main fragment has no significant difference to the two above. Create fragment fetches the existing flashcard-sets and allows users to select which sets to be adopted. Besides, users can set a specific number of questions. If it is smaller than the sum of all adopting flashcard, it will randomly pick cards from all the sets. In attempt fragment, unlike the flashcard features, users cannot edit the quiz, meaning that no quizzes can be modified after creation. In a quiz, users are required to answer questions by entering the back text of each question as each question is the front text of a flashcard.

In considering viewing and attempting to a quiz, quizzes are divided into two states, completed and not completed. Hence, when user finishes a quiz, it can be reviewed and correct answers will be shown, as well as the mark.

To store the quiz data, each quiz has title, datetime, questions, correct answers, user answers and other relevant data.

## **AI REVISION**

Google Gemini Ai is used in the application. There is only one fragment for the feature. Users can send requests to the ai and get responses. To facilitate some common action, users can choose 3 modes when using the feature – free, proofread, and summarise. Free mode is asking ai without any predefined prompt. Proofread is asking ai to proofread an article where users do not need to specify the prompt. “Please proofread the following article for any grammatical errors, typos, or awkward phrasing.” is used to get the best results. Summarise mode uses the prompt, “Your task is to summarize the following article into three points. Avoid technical jargon and explain it in the simplest of words.” The prompts are set up referencing Prompt Engineering Guide [2].

The chat history is stored and provided to Gemini Ai for referencing the context of the chatting. It stores in a simple shared preferences and when users decide to clear chat history, all the data will be deleted so that a new chat thread can be created.

## **UNRESOLVED ISSUES**

---

### **TECHNICAL**

Repeated notification is not precisely pushed based on the time set by users. By reading Android documentation, one-time exact alarms have to be used which increase the complexity of the implementation and waste more system resources.

### **FUNCTIONAL**

In proposal, weekly tests are planned and labelled as “Hard” based on its implementation difficulty. During the development stage, several attempts of using Gemini API to generate questions based on user articles and flashcards are made but failed to format the Ai responses by prompt engineering. More advanced Ai model (e.g. Gemini Pro 1.5) may solve the issue but it requires subscription payment. Using other free Ai model may solve this issue. Besides, due to the inexact time of notification, we have decided to separate the scheduling and quiz function. However, it is still possible to be integrated together in the future.

## **REFLECTION**

---

In android development, one of the challenges is adapting to the new API. Some methods that are widely used in old version could be deprecated or even not applicable when using the latest API. One of the examples is the “PendingIntent.getActivity()” where the documentation does not mention that the flag, FLAG\_IMMUTABLE or FLAG\_MUTABLE has to be used. The Android Studio Editor does not provide enough hints and most of the example usages from the Internet are a few years ago, especially those using Java, which slow down the development progress a lot.

## REFERENCE

---

[1]“AlarmManager,” Android Developers.

[https://developer.android.com/reference/android/app/AlarmManager#setRepeating\(int,%20long,%20long,%20android.app.PendingIntent\)](https://developer.android.com/reference/android/app/AlarmManager#setRepeating(int,%20long,%20long,%20android.app.PendingIntent))

(accessed May 13, 2024).

[2]“Getting Started with Gemini – Nextra,” [www.promptingguide.ai](http://www.promptingguide.ai).

<https://www.promptingguide.ai/models/gemini>

(accessed May 13, 2024).