# JapaneseReps Post-mortem

Finn Bright

Repo name: seng440_assignment1_fjb56

“JapaneseReps” is an app where you can browse a selection of word lists and use flashcards to memorize Japanese words and phrases. New words can be learnt from an API that collects 5 random Japanese words at a time and settings can be used to change ordering of the cards.

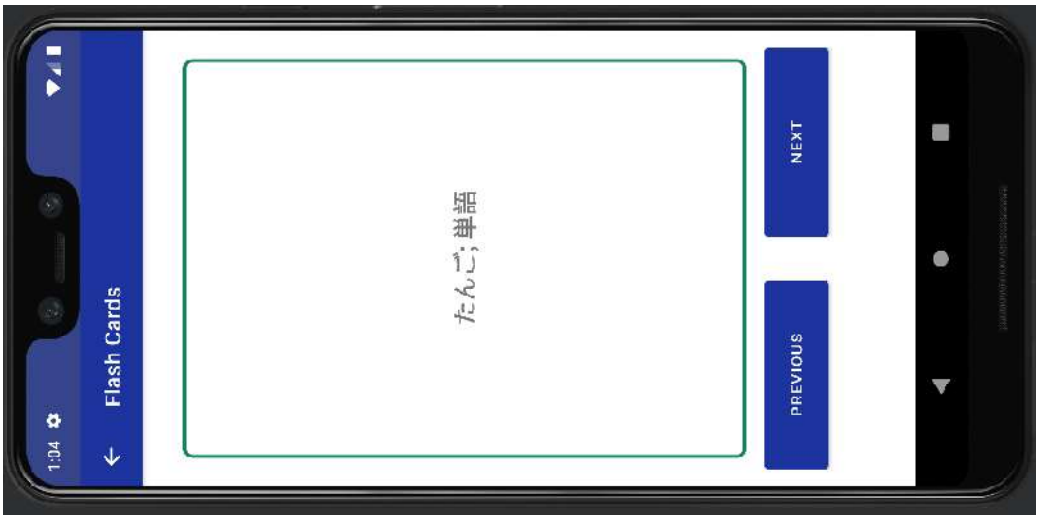

*Figure 1 Flashcards.*

## Development Process

I used plain Kotlin for this project. Found some good tutorials for setting up most of the app, though accessing specific recycler view lists within a recycler view list was a pain until I realised it was really simple to do by passing in the right arguments to the navigation method. A lot of the problems were overcome by using the navigation variables e.g., the flashcard fragments (Figure 1) just loop to the same fragment view but with the next pair of words in a list. I had initially planned to have a text input to translate words into Japanese with the assistance of an API but for some reason the POST request worked everywhere except for in the app.

## Checklist

### 1. Screens

This app is made up of a home, lists, and settings page alongside some children within those fragments. The overall structure of the app was done with a single-activity multiple fragment architecture. I ignored the full-screen fragment.

### 2. Intents

The intents are accessed from the Flash Cards overview page. They take the Japanese word from the RecyclerView item and search for it on one of three websites.

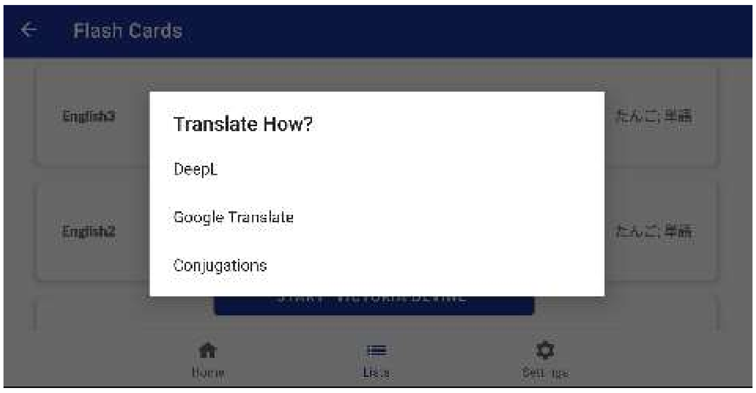

*Figure 2 Intents.*

## 3. RecyclerView

Three RecyclerViews with adapters are used. The first is to view the flashcard lists available, and the second to view the contents of each respective list and the last to view 5 random words given by an API. All use CardView widgets within a linear layout to populate the RecyclerView.

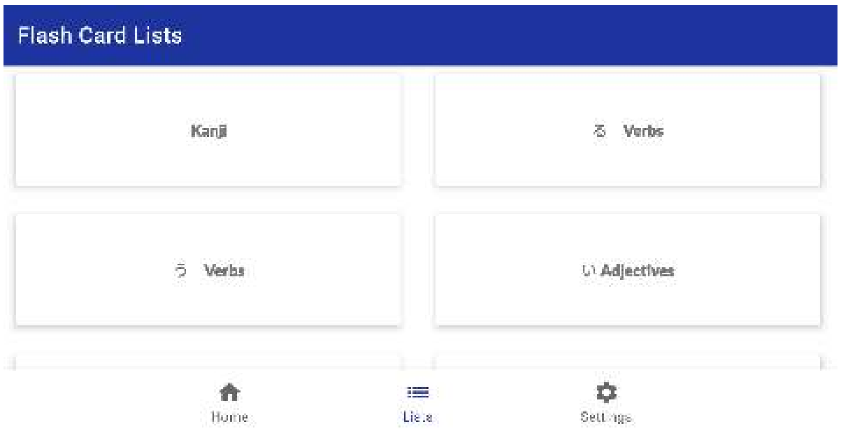

*Figure 3 Flashcard lists.*

## 4. Widgets

The widgets with listeners used in this app include the Buttons, Switches, CardViews, a TextView, and RadioButtons. The flashcard TextView has a listener so that on selection it triggers the animation to flip the card.

## 5. Layouts

ConstraintLayouts are used for most of the fragments within the app. LinearLayouts and RelativeLayouts have been used to make the CardView tiles (e.g., Figure 3).

## 6. Landscape/Horizontal View

Most of the recycler views naturally scroll to accommodate the horizontal view. The other fragments without them have used a ScrollView to hold content. The FrameLayout combined with ConstraintLayouts and margins has also been useful for ensuring content fits into the available space.

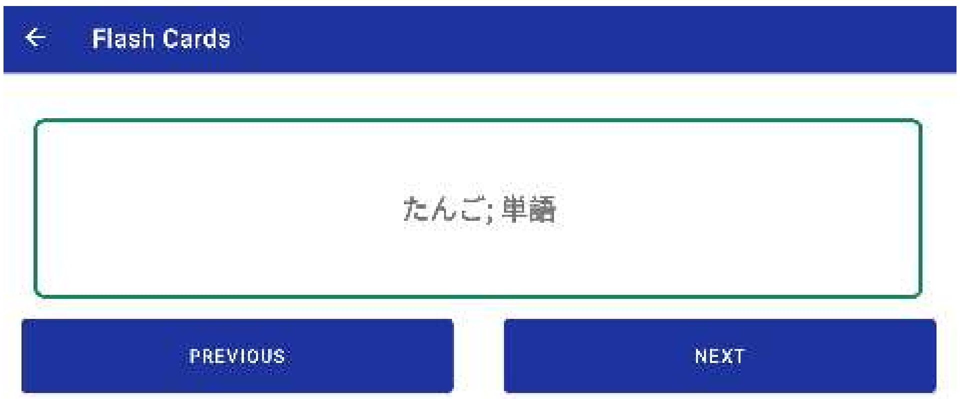

*Figure 4 Horizontal view of the flashcards, no scrolling necessary.*

## 7. String Resources

Everything except for the content adapted into the RecyclerViews has static string resources.

## 8. Default Definitions

Japanese has been provided as an alternative language for the application.

## 9. Toast

Any time a setting is changed a toast appears.

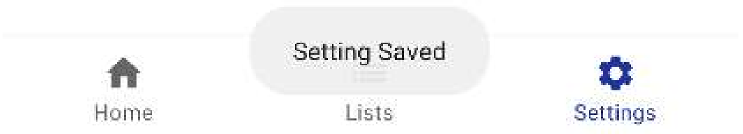

*Figure 5 Settings modified.*

## 10. Coroutines

A timer has been implemented to give the user a reminder via Toast message. A coroutine has also been used for receiving data from the network call to an API that returns 5 random Japanese words.

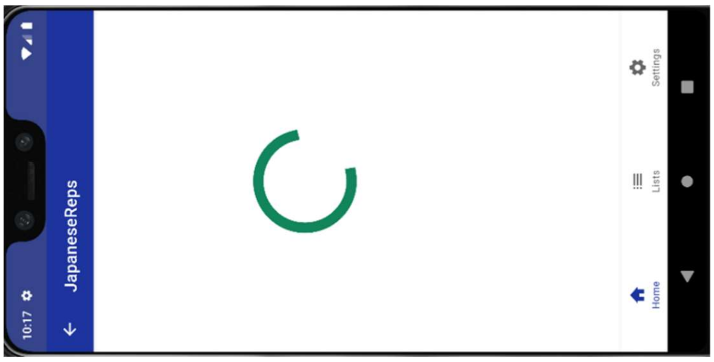

*Figure 6 Loading screen for 5 random words screen.*

## 11. Animation

Animations have been used for the transition of most fragments (not including the BottomNavigationView). The flashcard is also animated such that it flips 90 degrees around the x-axis and is replaced by another CardView widget for the other 90 degrees.

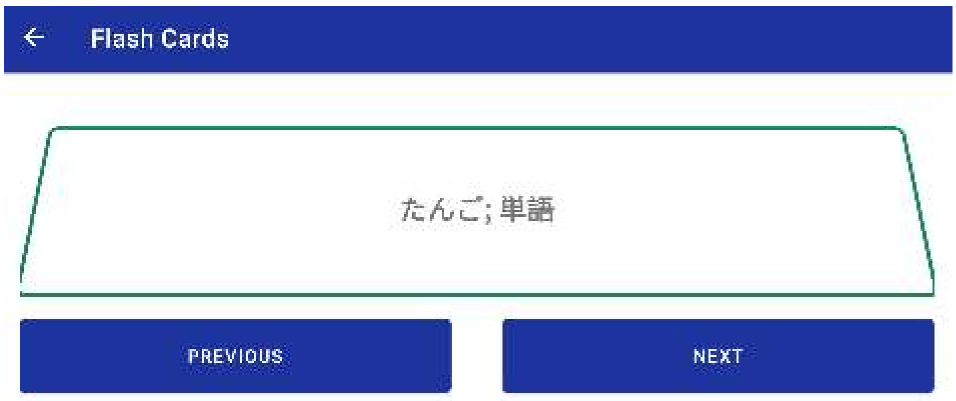

*Figure 7 Flashcard mid-flip.*

## 12. Two other Android API features

- The Gson library is used to retrieve flashcard data from a Json file.
- SharedPreferences is used to save the settings.

## Extra

- Navigation from a RecyclerView to a RecyclerView to every flashcard. The individual flashcards loop to itself with two different animations for the next or previous buttons. This transition is not saved to the back stack, so you don't have to spam the back button to get to the previous RecyclerView.
- The dark theme also works for every widget by using the correct style for widgets such as CardView.
- A BottomNavigationView has been implemented so that the user can toggle between these three fragments at any time.

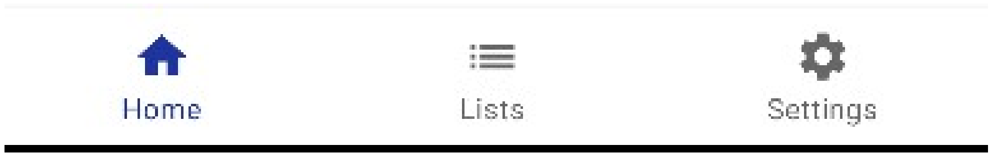

*Figure 8 Bottom navbar.*

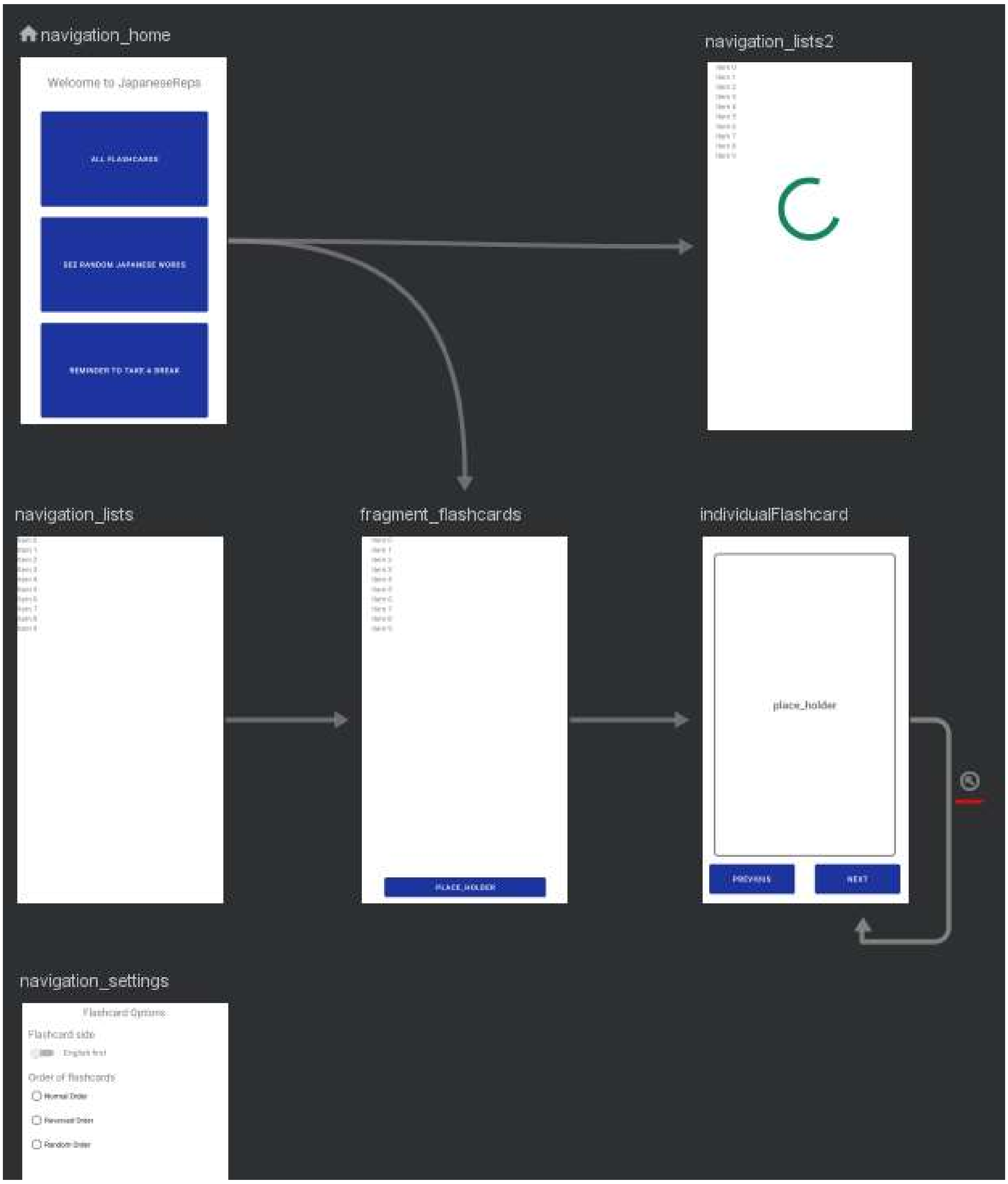

*Figure 9 Fragment navigation.*