How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: Arta0613

Wani Reference

Description

Wani Reference is an app that allows you to browse and view the various Kanji available in WaniKani. The application will be written in the Java Programming language.

WaniKani is a website that utilizes a technique called Spaced Repetition System (SRS) to teach students Japanese Kanji and Vocab.

Wani Reference aims to bring a more native mobile feel to what is a web based application as a supplement to users of WaniKani. It also gives you the ability to set a widget that will update with a random “Kanji of the Day” so you can see something new every day on your homescreen.

Intended User

Users of the WaniKani website and anyone else interested in learning Japanese Kanji and Vocabulary.

Features

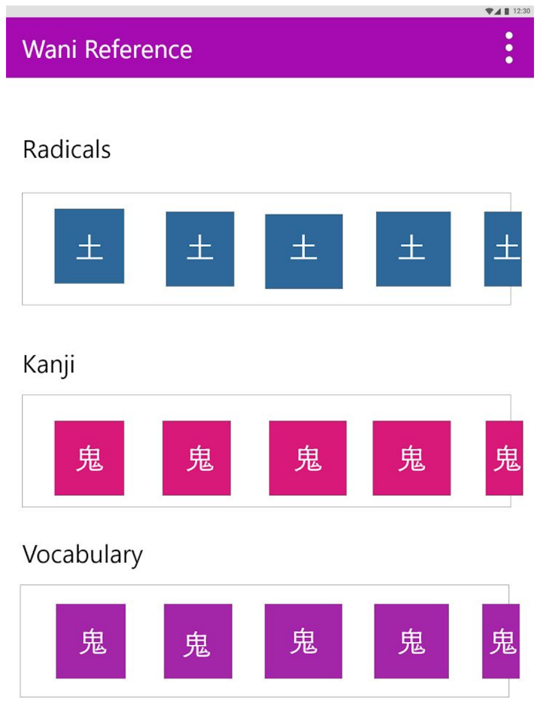
List the main features of your app. For example:

- Select from the first 3 levels of WaniKani (for free users)
- View Radicals, Kanji and Vocabulary from each level
- View Radicals Kanji and Vocabulary while offline
- Add widget on homescreen to see a random Kanji

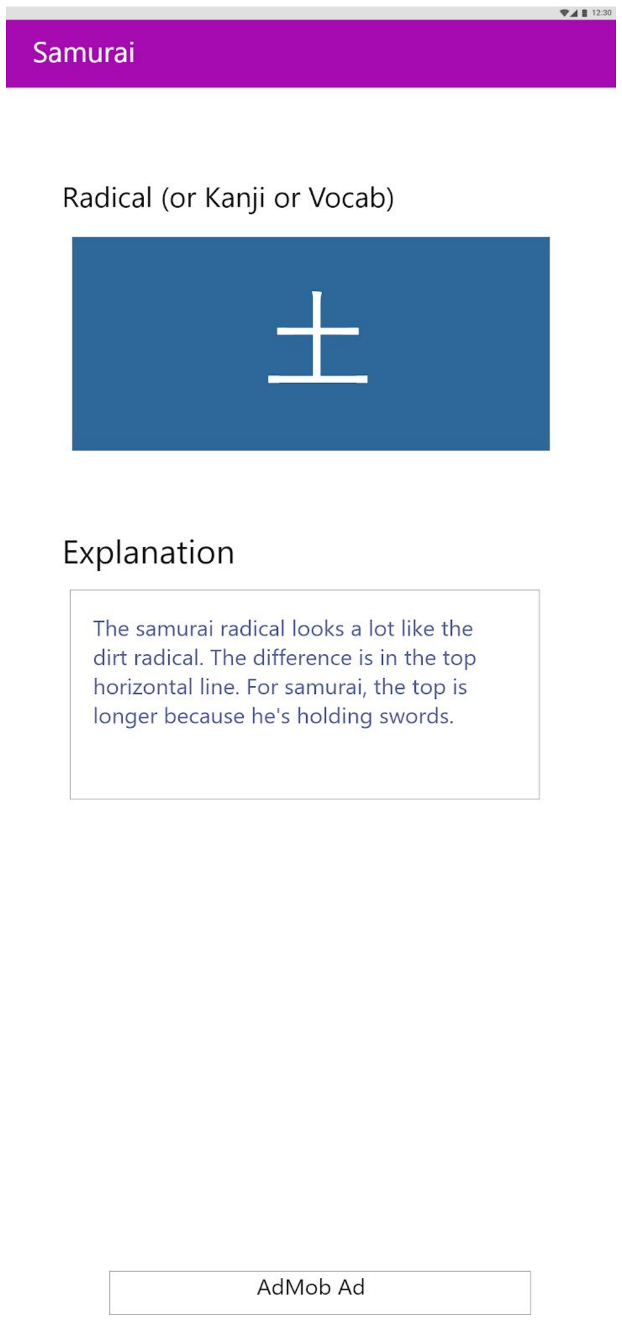
User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1:

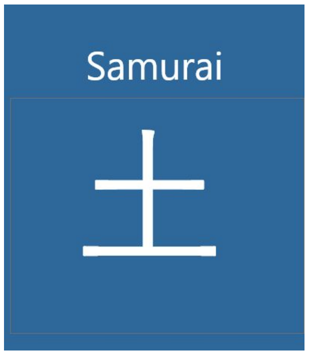

12:30
Wani Reference
Radicals
土
土
土
土
土
Kanji
鬼
鬼
鬼
鬼
鬼
Vocabulary
鬼
鬼
鬼
鬼
鬼

Screen 2



The samurai radical looks a lot like the dirt radical. The difference is in the top horizontal line. For samurai, the top is longer because he's holding swords.

Widget



Key Considerations

How will your app handle data persistence?

It will download subjects (data) on launch and store it offline using Room. It will be used in conjunction with ViewModels and LiveData

Describe any edge or corner cases in the UX.

A corner case could be that there is no initial data on launch because of no network connection. In this case some view could be shown to the user that they need to connect to the network once to download data.

Describe any libraries you'll be using and share your reasoning for including them.

Initial list of libraries determined to be used:

- Glide 4.11 -- For downloading image assets (specifically svg)
  https://github.com/bumptech/glide

- Android SVG 1.4 -- (As SVGs aren't directly supported, will utilize AndroidSVG as shown in glide sample app on github (Glide/Samples/Svg))
  https://bigbadaboom.github.io/androidsvg/

- Retrofit 2.8.1 -- Makes handling network requests much easier
- Retrofit converter-gson and rx adapter 2.6 -- tools that help with networking and converting callbacks into reactive streams
  https://square.github.io/retrofit/

- RxJava/Android 2.2.15 / 2.1.1-- tools to help reactive programming paradigm making async easier https://github.com/ReactiveX/RxAndroid

- Room runtime, compiler and rx java 2.2.5 -- data persistence with the help of Room and rx https://developer.android.com/topic/libraries/architecture/room

- Firebase analytics 17.4.2 -- analytics for what the users are are doing
  https://firebase.google.com/docs/analytics/get-started?platform=android

- Constraint layout 1.1.3 -- building powerful and performant layouts by keeping view hierarchies flat https://developer.android.com/training/constraint-layout

- Android Lifecycle viewmodel and extension 2.2.0 -- utilizing viewmodels and their extensions to keep the app running smoothly without worrying about lifecycles and memory leaks
  https://developer.android.com/topic/libraries/architecture/room

- Recyclerview 1.1.0 -- building out powerful and performant lists
  https://developer.android.com/jetpack/androidx/releases/recyclerview

- Play Services ads 19.1.0 -- to show ads on details screen
  https://developers.google.com/admob/android/quick-start

- Coordinator layout 1.1.0 -- powerful viewgroup to help coordinate the ui with animations and flow
  https://developer.android.com/jetpack/androidx/releases/coordinatorlayout

- CardView 1.0.0 -- quick tool for making easy cardview type layouts
  https://developer.android.com/jetpack/androidx/releases/cardview

- Material library 1.1.0 -- using components like the fab to create a responsive ui
  https://github.com/material-components/material-components-android/blob/master/docs/getting-started.md

These are most of the libraries that I can determine but versions might be changed, other libraries may be added, some may be removed based on the needs of the project. Project rubric required libraries however (AdMob and Analytics) will not be removed though their version may be upgraded or downgraded based on issues that may arise during implementation.

Describe how you will implement Google Play Services or other external services.

AdMob to show a small ad on the bottom of the details screen.
Firebase Analytics to log certain events like adding a widget or switching levels.

In terms of material designs, I will follow basic design guidelines as followed by the Android NanoDegree project "Make your app material" with elements such as transitions and layers, common theming etc.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.

You may want to list the subtasks. For example:
- Set up data classes (data coming from api)
- Set up Networking classes (retrofit)
- Set up gradle dependencies for above

Task 2: Implement Main UI

List the subtasks. For example:
- Build UI for MainActivity
- Download data and display first level

Task 3: Store offline

Describe the next task. List the subtasks. For example:

- Build database data class and store data offline

Task 4: Add multi level selection

Describe the next task. List the subtasks. For example:

- Identify pattern for multiple levels and build UI/set data

Task 5: Implement Detail UI

Describe the next task. List the subtasks. For example:

- Build Details UI and set data

Task 6: Add Widget feature

Describe the next task. List the subtasks. For example:

- Add widget feature

Task 7: Add Google Services

Describe the next task. List the subtasks. For example:

- Add google services integrations (analytics and admob)

Add as many tasks as you need to complete your app.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
  - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
- Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”