GitHub Username: jidekrm

Breeds

Description

Breeds is an educational quiz app for dog lovers of all ages. The quiz app presents you a picture of a dog and multichoice answer. Guess right, and you move up the ladder. Breeze will help improve your knowledge of dogs.

Intended User

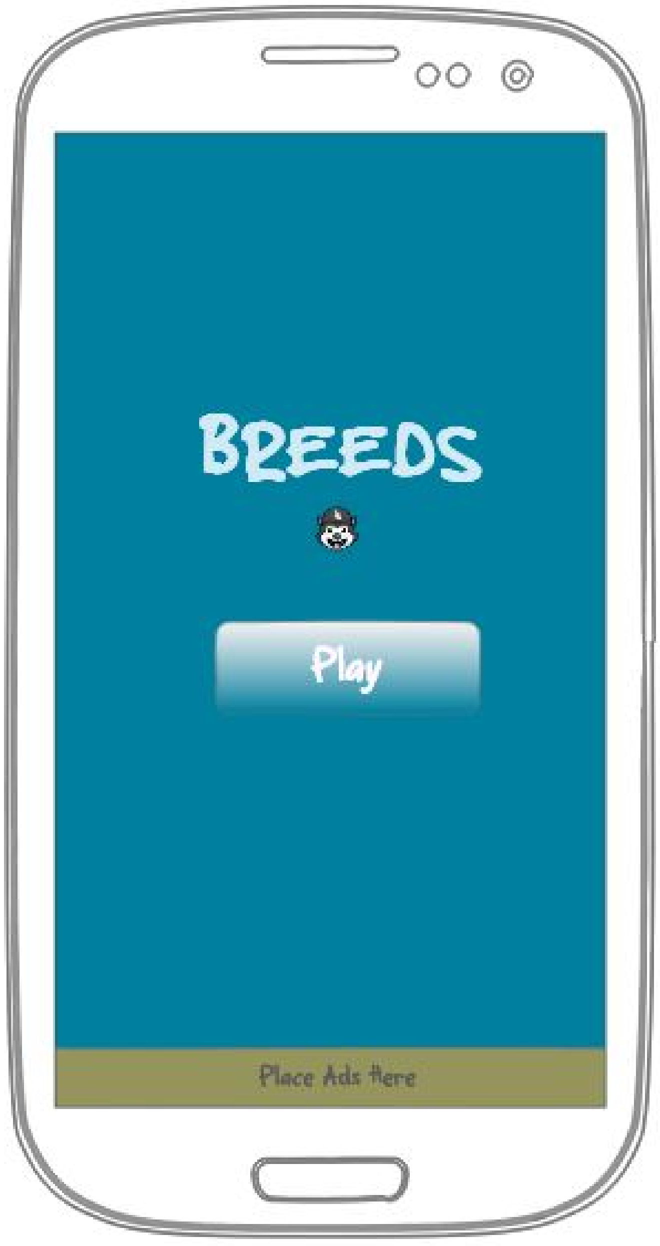
The primary target of the app are dog lovers of all age.

Features

- Breeds will be written solely in the Java programming language.
- Query https://dog.ceo/dog-api to obtain dog breed data.
- Store the queried data in a database.
- Present the queried data in a user friendly quiz format to the user.
- Inform the user the result of the quiz and provide the correct answer.
- Keep tab of the number of correct answers.

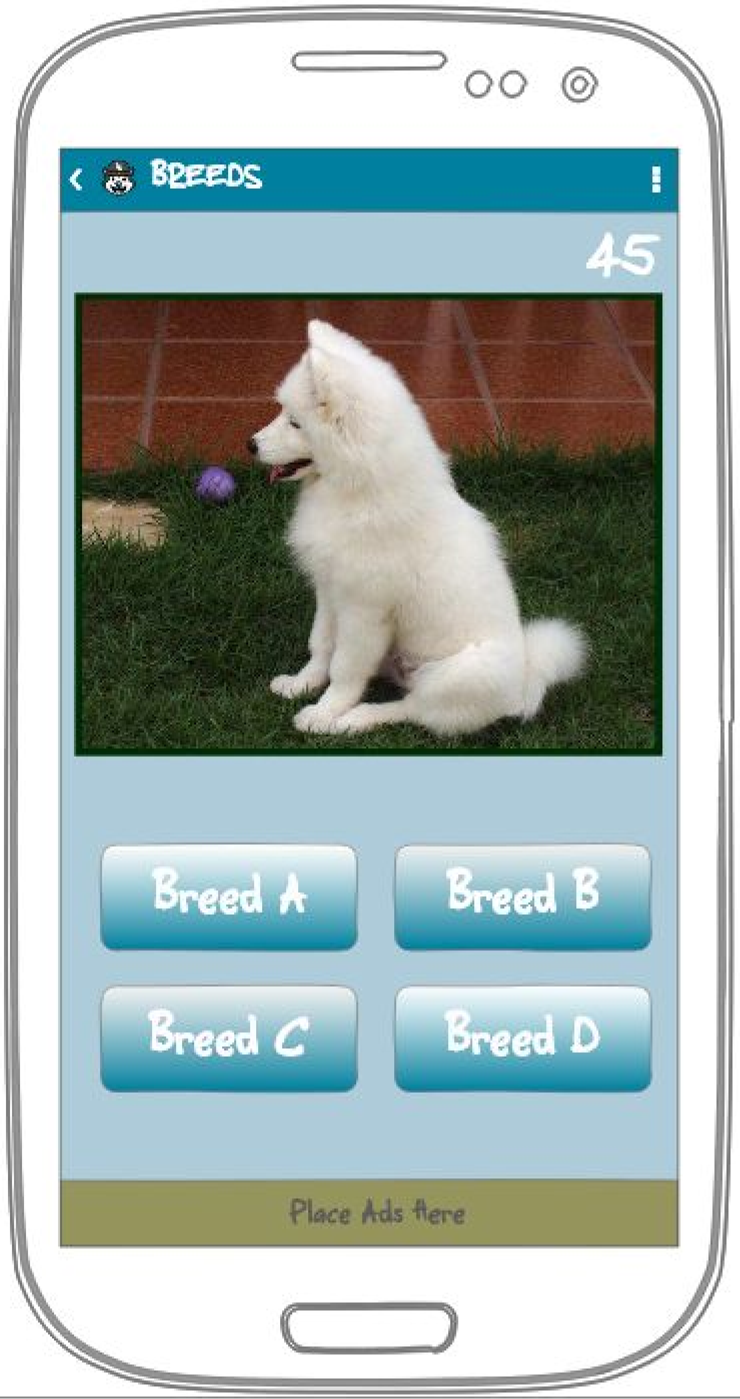
User Interface Mocks

Main Screen



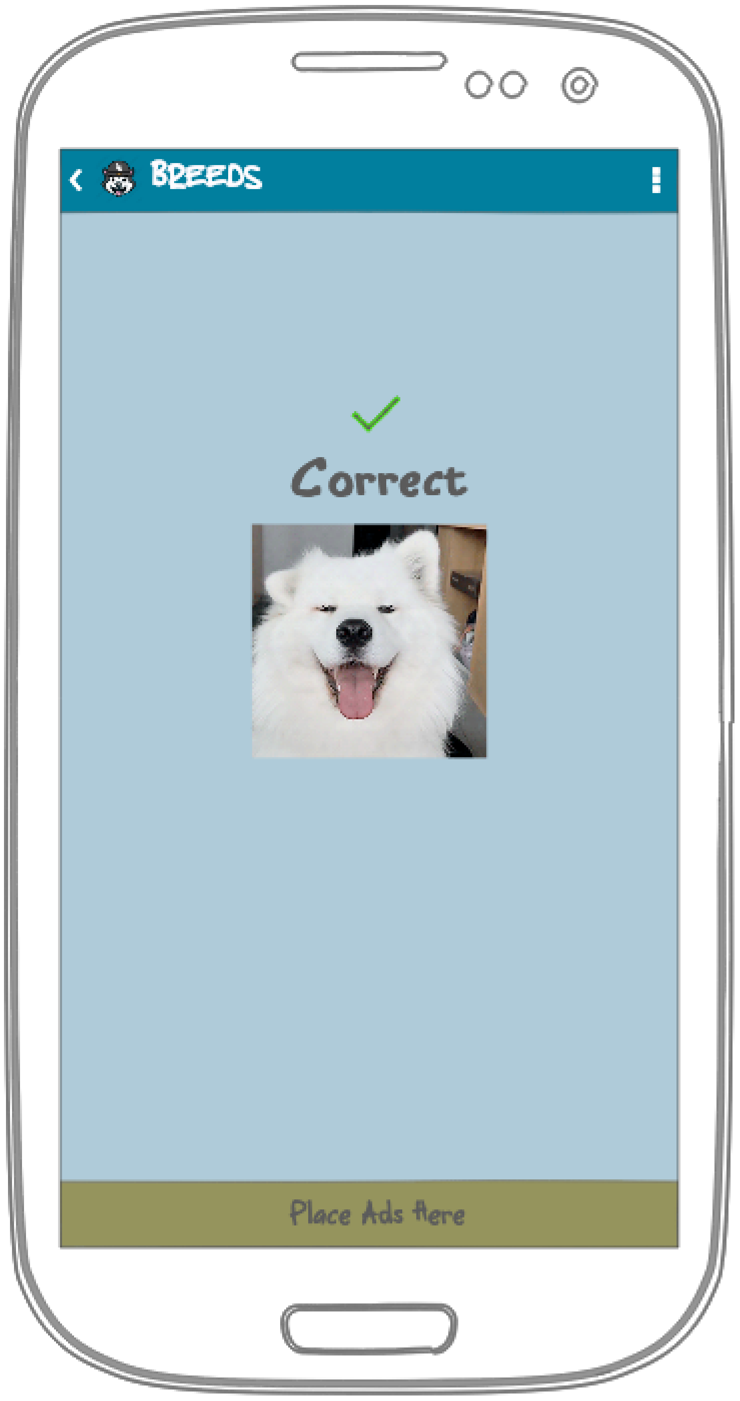
This will be the main page of the app. Here the user is directed to click on the Play button to launch a ‘Play’ screen.

Play Screen



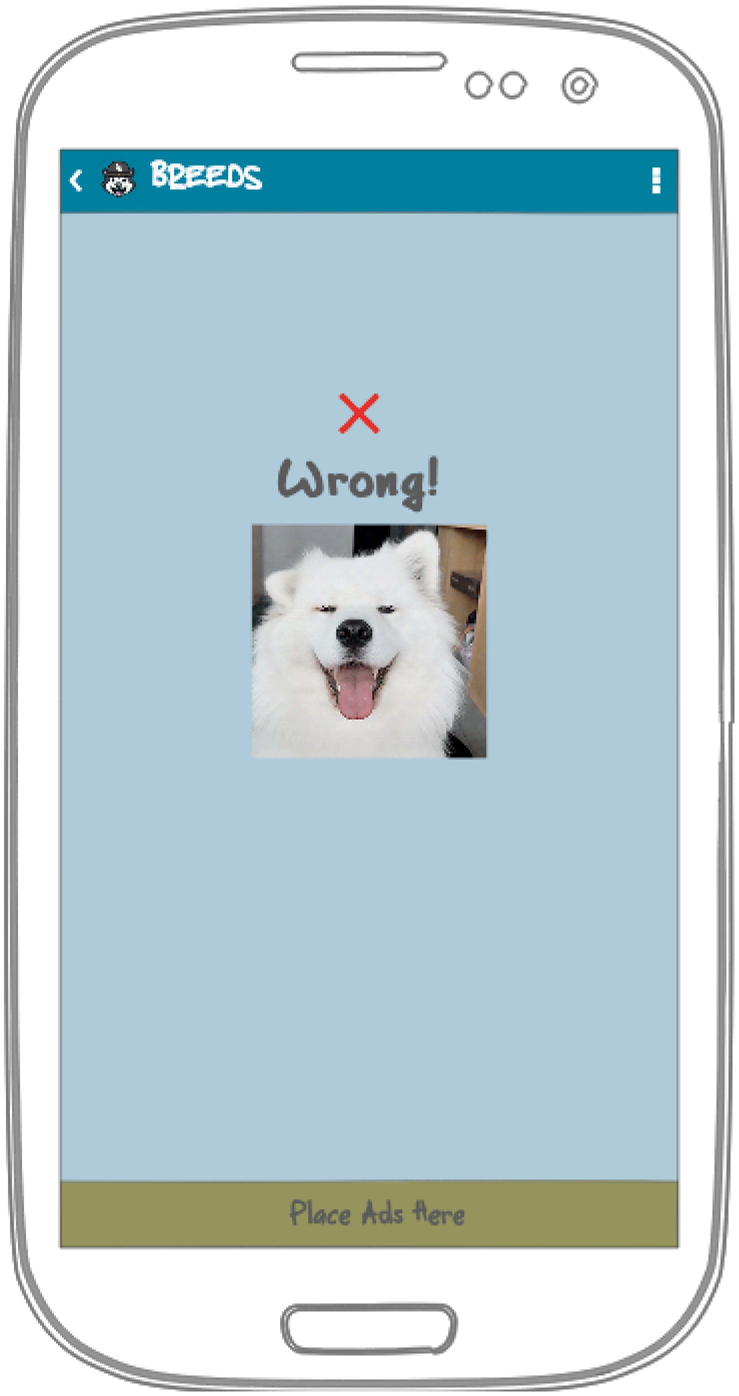
This is the 'Play' screen where the user is randomly shown the picture of a dog and asked to choose one of four options to accurately guess the breed of the dog.

Correct Screen



Once the User guesses right, the ‘Play’ Screen transitions into the ‘Correct’ screen.

Wrong Screen



If the User guesses wrong, the ‘Play’ Screen transitions into the Wrong screen.

Key Considerations

How will your app handle data persistence?

Data will be queried from https://dog.ceo/dog-api using an IntentService and stored in a database using Room.
The database is not expected to 'change' in real time but if for some reason the implementation goes is that direction, LiveData will be deployed to monitor the database change and present it to the UI. To avoid requering on screen rotation, ViewModel will be deployed.

Describe any edge or corner cases in the UX.

The secondary plan is to provide a capability for the 'Play' screen to be paused by the user. Once paused, the 'main' screen will be inflated with a 'Resume' button. If this is implemented, a pause button will be available on the 'Play' screen.

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

Picasso will be used to handle the loading and caching of images.

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

Google play services(Google Mobile Ads) will be used to for banner ads shown on each screen in the mockup.
A secondary implementation may include Google Analytics and Google Play Game services.

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

- Develop in Android studio 3.3.2 and name the app 'BREEDS'.
- Find a third-party library to help with the query and JSON. Add to gradle.
    - com.amitshekhar.android:jackson-android-networking:1.0.2
    - com.squareup.moshi:moshi:1.8.0
- Add other libraries to ease design. Must use libraries are identified below
    - Design library - com.android.support:design:28.0.0
    - Appcompat - com.android.support:appcompat-v7:28.0.0
    - Constraint library - com.android.support.constraint:constraint-layout:1.1.3
    - Picasso - com.squareup.picasso:picasso:2.5.2
    - Room - "android.arch.persistence.room:runtime:1.1.1
        - android.arch.persistence.room:compiler:1.1.1
    - Lifecycle - android.arch.lifecycle:extensions:1.1.1
        - Android.arch.lifecycle:compiler:1.1.1
        - android.arch.lifecycle:runtime:1.1.1

    - Espresso - com.android.support.test.espresso:espresso-core:3.0.2
- Enable DataBinding.
- Gradle- 3.3.2
- Minimum SDK should be set at 19.
- Target the latest SDK.
- buildToolsVersion - 28.0.3
- Define a style and theme to use for all UI.
- App theme extends AppCompat
- App uses an app bar and associated toolbars

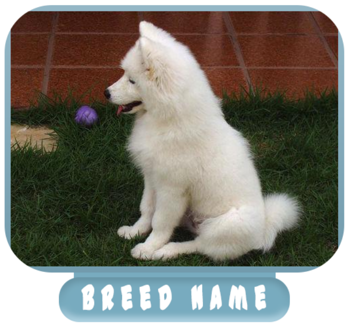
Task 2: Implement UI for Each Activity and Fragment

- Start with the 'Main' UI and add a layout to correspond to the mocks.
    - Add the 'Title' text- 'BREEDS'
    - Add a button and name it Play.
        - Create an intent onClickListener to launch the Play activity.
    - Apply the style.
- Build the Play screen and add a layout to correspond to the mocks.
    - Add and implement the Image view
        - Image should be loaded with Picasso.
        - The Image is randomly shown from database.
        - The breed name should be extracted from the url. For example, if the Url is https://images.dog.ceo/breeds/hound-afghan/n02088094_5150.jpg , then "hound-afghan" should be extracted and stored in a string variable "A".
    - Add the Buttons for the answer options.
        - One of the buttons name should be equal to the string variable "A".
            - An intent onClickListener here should launch the 'Correct' Activity.
        - There other three buttons should be randomly named from the API end https://dog.ceo/api/breeds/list/all . This should exclude the one stored in string variable "A".
            - An intent onClickListener on any of these buttons should launch the Wrong Activity.
    - Layout must include contentDescription and enables RTL layout switching.
- Implement the 'Correct' and 'Wrong' Screen and add a layout to correspond to the mocks.
    - A thumbnail of the correct answer should be shown on these screens.
    - These UI(s) will only be inflated for a transient period and returned to the Play UI.
    - Layout must include contentDescription and enables RTL layout switching.
- The Play screen image view and the 'Correct' and 'Wrong' screen should use a shared transition.

Task 3: Implement Widget

- Implement a widget to display random breed.

- On click, widget should launch the Play Activity.
- Widget should be implemented according to the mock below.



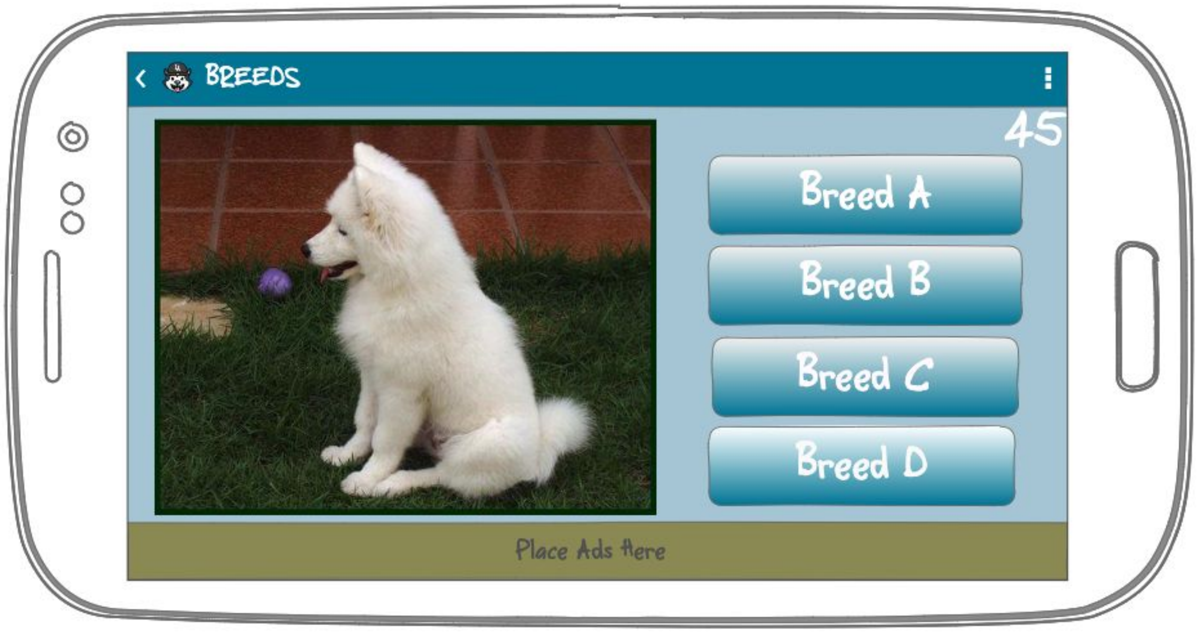
App widget

Task 4: Implement Google mobile ads

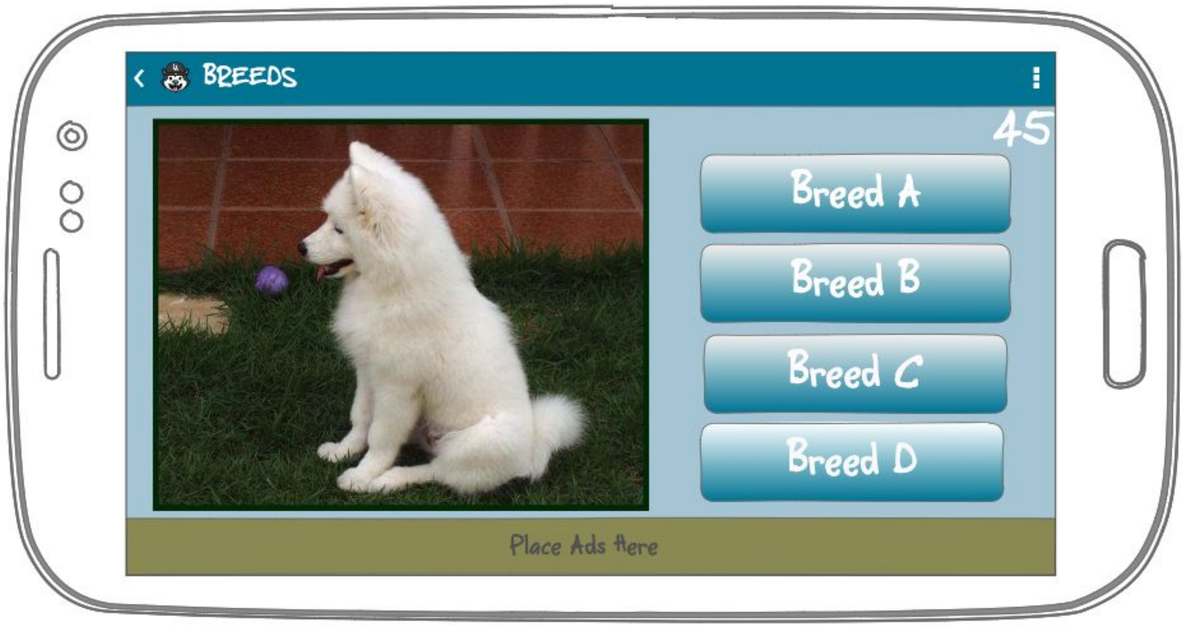
- Implement google mobile ads.
    - Add Banner Ads to every screen.
        - Banner ads in the 'Wrong' and 'Correct' screen can be excluded since they will only be displayed for brief period.
    - Interstitial Ads should be displayed after every 5 -10 play UI is inflated.

Task 5: LandScape and Tablet Implementation

- Landscape mode for the Play UI should be implemented with reference to the mock below while other UI(S) can remain same.



- The Play UI for Tablets should be implemented with reference to the mock below while others can remain same.



Task 6: Create Build Variants

- Consider creating a Paid variant.

Task 7: Handle Errors and Test

- Test each screen and button for response.
- Test to ascertain that the right layout is inflated.
- Test string variable “A” response.
- Log any error.
    - Any Play service error should be logged as advised by the Play services API doc.

Task 8: Polish

- Optional: Consider other styles using material design as guide.
- Take a second look at the layouts and ascertain that margins, padding and styles are consistent.
    - Final UI can slightly be different from the mock if it improves the UI look and feel.
- Extract all dimensions, values, colors and string to corresponding xml and res folders.
- Any drawable most be optimized for build variant and screen size.

Task 9: Google Play Games Services and Analytics

- Optional: Consider implementing the games services and analytics.
    - These are considered secondary in this app and should be put into consideration
    - Games services: A leaderboard should be considered.

Task 9: App Signing

Sign the app.