README - Write up

Here's a brief overview giving some insight into my approach and my next steps.

Note: Developed against Android Studio Ladybug | 2024.2.1 and Gradle 8.9,
I have also included an APK of the project in the root directory if required as well as demo showing the loading and error states which can be hard to recreate.

- **Priorities**: My main focus was to implement the core functionality as specified in the ACs, specifically the networking and UI and navigation for both pages.
    - I felt like these were the most critical tasks as there's not much of app without them.
- **With more time:** assuming there were no more features required to develop, I would:
    - Add UI tests
    - Improve the UI on the breeds photos screen/grid
    - Perform some accessibility testing
    - Profile the app (especially for recomposition issues)
    - Improve the UI polish as detailed below
    - Add an app icon
    - Prep the release build of the app (minification, obfuscation etc.)
- **Time spent:** Approximately 4-5 hours on the project, plus the time taken to write this document
    - I wanted to constrain the amount I could spend so that it was clear what I had to prioritise and
      decisions I had to make as well as exemplifying what I could achieve in a reasonable amount of time.

Approach

- Read through the requirements and tips emails
- Setup a starter code base from the Android Studio template
- After that I put together a plan of the list of tasks I wanted to tackle

Plan

I put together a prioritised list of the tasks to tackle:

- ~~**Add navigation to a second screen**~~
- ~~**Add dog api**~~
- ~~**Add dog repo and mapping**~~
    - ~~Unit test mapping logic~~
- ~~**implement network call on first screen and display content**~~
    - ~~Call repo from ViewModel~~
    - ~~Pipe results to state~~

    - ~~Display content on screen~~
- ~~**implement network call on dog photos and display content**~~
    - ~~Call repo from ViewModel~~
    - ~~Pipe results to state~~
    - ~~Display content on in grid~~
- ~~**Add error handling and loading states**~~
- ~~Unit test for any business logic~~
- ~~**Tidy up UI on Breed list screen**~~
- **Add unit tests for viewModel state logic**
- **Add UI tests**
- **Tidy up UI on Breed photos screen**
- *Everything else - I didn't think I could get anymore than the above done in time I allotted*

Visual Design Improvements

If I had more time, Although visual design isn't my forte, here's a few improvements I would consider:

- Improving the Photo grid on the breeds photos screen so they sit a bit more cohesively together and don't look as jarring when there are portrait photos.
- Maybe a bit more distinction between the different items on the DogBreed list
- Adding a some color (e.g. adding color to the TopBar)
- Refining some of the padding/margins on the dogs homepage
- Add some simple animations to make the app more fluid and transitions less jarring
- Adding an app icon and potentially a splash screen
- Whilst not strictly Visual Design but UI, ideally I would add content descriptions for the dog photos coming back from the API (however I don't think would be possible as this data isn't return with the image)
- Whilst not essential, supporting a dark theme

Technical discussion

I thought I might explain a bit about my details of technical decisions:

- I kept the number of packages light to make it easier to navigate for yourselves, however would change this as the amount of code increased
- I frequently left small classes like Models where they were used rather than extract them out due to the small size of the codebase
- I deputised writing UI tests due to the fact it was quicker for me to manually test however I would be keen to implement them if this were continued

- It seems the breed data from the API is a bit inconsistent as some breeds have the words in their name reversed and some don't, this make it harder to show this to a user in human readable way

If I had more time, I would look to implement the following changes:

- The current dependency injection solution isn't ideal and is a bit clunky, however is fine for an app of this scale
    - As the app scales, using something like Koin or Hilt would be ideal, being more robust and maintainable
- I would look to improve the colors, spacing etc. to be a singular theme object
- Add caching for the backend requests for better performance as well as image caching
- Add more `@Preview` functions for some of the UI
- Add static analysis tool like detekt and clean up any code smells

---