Android Proficiency Exercise

Overview

The purpose of this exercise is to assess candidate developer's Android technical proficiency, coding knowledge and style. The exercise involves build a "proof of concept" app which consumes a REST service and displays photos with headings and descriptions. The exercise will be evaluated on coding style, understanding of programming concepts, choice of techniques, quality of the final product and also by the developer's process, as indicated by the trail of git commits.

Specification

Create an Android app which:

1. Ingests a json feed from https://dl.dropboxusercontent.com/s/2iodh4vg0eortkl/facts.json
    - The feed contains a title and a list of rows.
    - You can use a third party json parser to parse this if desired.
2. Displays the content (including image, title and description) in a ListView
    - The title in the ActionBar should be updated from the json data.
    - Each row should be dynamically sized to the right height to display its content - no clipping, no extraneous white-space etc. This means some rows will be larger than others.
3. Loads the images lazily (don't download them all at once, but only as needed).
4. Implement a refresh function allowing the data & view to be updated
    - Use either a refresh button or pull down to refresh.
5. Should not block UI when loading the data from the json feed.
6. Each row of the table should look roughly like the following image:

Additional Guidelines

1. Use a Github repository to manage the source code. A clear Git history showing your process is required. Commit your changes to git in small chunks with meaningful comments.
2. The app should target Android version 7.1 (v25). Don't worry about backwards compatibility for this task.
3. The list should scroll smoothly. As much data as possible should be cached.
4. Comment your code where necessary.
5. Polish your code as much as possible.
6. Feel free to use any best-practice open-source libraries/examples you need, just be sure to give credit. Do not use beta versions of any libraries.
7. Include at least two UI unit tests; one that asserts the state of the screen when set up with all data present, and one that asserts the state of the screen when in an error state.
8. Handle screen rotation efficiently.