# Granular Mobile Interview

**Design and Implementation Decisions**

**App Design**

I am using an MVP inspired design for the app. Even though the exercise is rather straightforward, I still wanted to use some kind of acceptable design pattern.

**Database**

There is no need to have a Database since there are no queries that we need to run for this assignment. I could have just stored the JSON file as retrieved, but I understand the requirements, so I decided to use ObjectBox database because I think it is really cool and makes things very simple as compared to Room or SQLite, or maybe even Realm. Assignment description mentions that a 3rd party database may be used for this exercise.

**Data Visualization**

I am using ListView (with ViewHolder pattern) to show the data. I would have used RecyclerView as it is better optimized, but I recall the assignment description mentioned the words "ListView", so I thought maybe the reviewers want to see how I will deal with a long list without using a RecyclerView. ListView can be used without a ViewHolder, but RecyclerView requires a ViewHolder. Adding ViewHolder to the ListView brings some level of optimization.

**Image Loading**

I know the assignment description discourages use of 3rd party libraries, except for Database. I could have used Volley for image loading, but that would mean use of NetworkImageView and implementing an ImageLoader. I did not want to restrict my code to use NetworkImageView.

In the interest of time (I wanted to finish this in under 3 hours) and to avoid boilerplate code, I am using Picasso for image loading as it provides caching and lazy loading. I know how Picasso works (basically built on top of LRUCache.

**Networking**

I am using the Networking class provided in the skelton app, which uses Volley. Volley request is made on a background thread and the response is provided on the main thread - so as such I did not need to implement an AsyncTask or an IntentService. One design decision I made in terms of data fetching is that when data is fetched, database is populated with the fetched data. When app needs to show data, the model first checks if the database has data and if it doesn't only then it requests data from network. Also, keeping in mind the Activity lifecycle, I am not updating the UI directly from the model, and even the callback for the data fetched is invoked after safety check. Presenter (which also implements the data receiver) informs the Model that it should not invoke the callback in case of orientation change, but the data is still saved in the database, and that prevents future network calls.

**JSON Parsing**

Generally, I use Gson to parse JSON response, but in this case the JSON was quite simple and also, I did not want to use a 3rd party library unnecessarily.

**Support for various Android sizes**

I did not implement Layout for landscape view, even though it is quite straighforward. Since the view was relatively simple, I couldn't find any justification of creating another Layout for landscape view. For the different sizes - again, the View is quite simple, so as such there is no need to customize for different sizes - I went ahead and created different dimensions (dimens.xml) for different screen sizes to showcase one way this can be done without creating different layouts. The principle is the same though if you want to create different layouts.