Summary

Structure and Architecture

Project has been separated into packages, mainly based on their parts in the project. This allows the parts that work together to be in the same package, which their visibility would be minimized and also makes it easier to change code as these parts would usually need to be altered at the same time. Anyone looking at these packages would at least be able to identify the working parts of the application, without the need to fully dive into each class.
Architecture used here is a MVP, with mainly passive views. The architecture is simple enough to suit a project of this size very well and does not overcomplicate the flows. MVP helps modularize the code that is needed for a feature, which allows several team members to work on different modules separately and then wire them together to complete the feature. It also makes the code very clean, which helps with later changes or code reviews and allows for easy swapping of any libraries, when needed.

Libraries used

- Dagger 2: Dependency injection library
- Retrofit 2: REST library used for communicating with the backend
- Logging Interceptor: To create logs for retrofit requests
- Picasso: Image loading library
- Gson: Json parsing library
- Omise: Used to provide view components for the payment form, validation utility classes and payment token generation client
- Espresso: For UI testing
- Espresso-Idling-Concurrent: For creating idling resource registry to help with UI testing
- Okhttp3-Idling-Resource: Idling resource for Okhttp client to help with UI testing of views with network interaction
- Logging-Interceptor: To add logging capabilities to retrofit

Assumptions

- For payment section, no address information is required.
- Validation is done only for credit card number and not any other fields, only null checks are done for those.

- ListView is used as per requirements, instead of the newer and preferred component RecyclerView
- There is no caching needed, so data on the list is fetched fresh every time the screen re-opens.
- Minimum sdk level is 17 and target is 26

UI/UX and Security principles

UI/UX

- All sizes are multiples of 8dp, as per material design guidelines
- All layouts have a 16dp margin and all inner elements have an 8dp margin
- Any element that can be tapped has a selected state
- The main flow has a right to left slide transition to show the user they are making progress
- Animation on the success screen to show the user their transaction has been successful and their flow is done

Security

- A dummy proguard signing config has been set up, which would secure the contents of the APK to an extent when it is replaced with real values
- Omise sdk public key has been stored in a file that has not been committed to source control and is then used in the build file. This would ensure that the keys are not exposed to others
- The security code on the payment section is input into a password type EditText, which hides the code

Improvements

There are certain things that can be done better, such as

- Error handling can be improved by creating a parent custom error and different children of that error could be thrown from the data layer and then easily handled by the presentation layer. For now, a simple string is used as proof of concept
- Presenters may not need to be hidden behind an interface as they are coupled tightly with views anyway
- List items can be displayed in a card, to follow better UI principles