# IN721 2019 4.1 The "Welcome to Dunedin" App

### Introduction

Delivery of information for travellers is a good use case for mobile apps. The mobility is of value because people are often away from their desktop machines when travelling, and having an installable app (as opposed to a web site) means that the information content remains accessible when the internet is not available. In this practical, you are going to build a simplified travel information app for visitors to Dunedin. It will be simplified in the sense that it will have minimal content, allowing you to concentrate on the architecture and navigation.

### Information Architecture

As is typical for apps of this kind, you wish to present four categories of information to your users: activities, shopping, dining and services. You can represent your architecture schematically as:

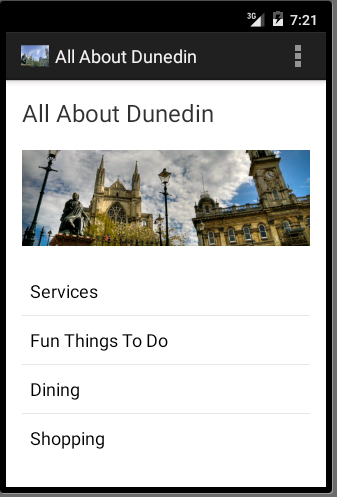
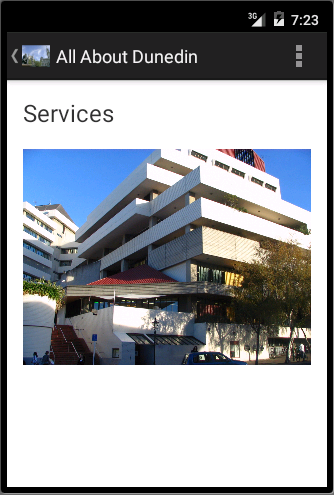
In a real app, the sub categories would provide lots of information and possibly additional functionality like connections to booking services. For our purposes, the sub categories need to present only identifying screen content (e.g. title text) and a single appropriate image. (See below for optional extensions.) Your primary focus for this task is on building clear and usable navigation which allows the user to move freely throughout the app and access all content.

### Application Architecture

The most straightforward way to build this application is with five activities: one for the main navigation screen and one for each sub category. Use this architecture, if you wish, or see below for some optional extensions and efficiencies.

### Task 1: Navigate with ListView

You may design your screens however you like (as long as they support the required behaviour). The following screenshots from my solution show one possible approach:

Main nav screen Services sub screen Activities sub screen

### Task 2: Navigate with a Navigation Drawer

Modify your opening Activity to show a full screen of content. Use a Navigation Drawer to allow the user to visit each of your subscreens.

### Optional Extensions

1. Add more content to your category pages. Include some descriptive text for each of multiple entries. Can you cope with the situation where you wish to display more entries than will fit on the screen? (**NB**: Making a ListView item hold complex data (e.g. an image and some text) is nontrivial. Feel free to explore this, but be aware that it is quite tricky. If you want to use ListView, it is fine to stick to single data elements (e.g. strings) so that you can use the existing ArrayAdapter classes. )
2. Simplify future extension and modification by using a single layout for all subcategory activities. Make sure that each screen still shows the correct title page and content.
3. Further reduce the amount of layout information you must maintain by using only two Activities - one for the main navigation and one for presenting content. Fill the content Activity dynamically from the code behind so that the user experience is the same as for the version with separate Activities for each sub category.
4. **Challenge Extension:** Eliminate hard-coding in your app by reading your content information in from an xml file. You will need to research this technique in the Android documentation, and to define your own XML data definition. If you would like to know how to do this, but don't feel like tackling it on your own, you can wait until we cover it in class later this term.