Coursework Report - Andrew Walker

Mobile Applications Development

Mouse Times - California

1. Introduction

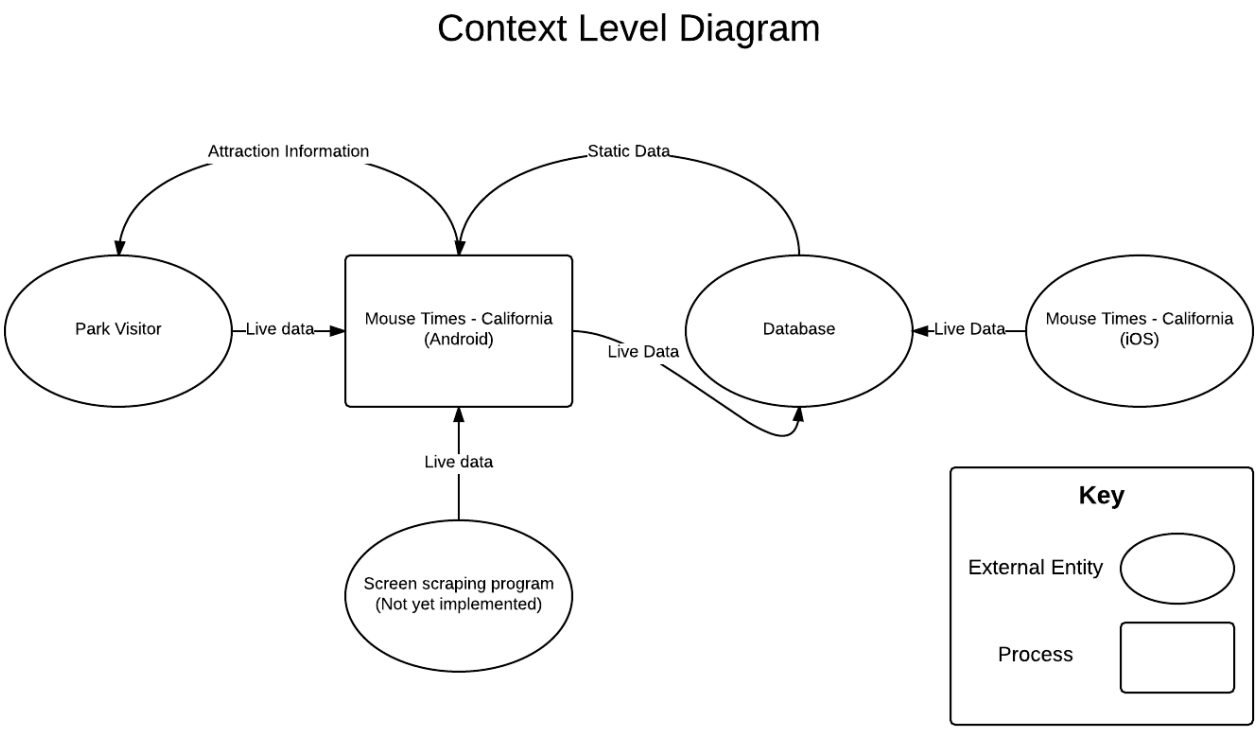
For my assignment, I decided to create a travel utility app for visitors to Disneyland Resort, California. The app would provide guests with all of the information they need to make the most of their visit, with features to help them get around the parks, avoid wait times and effectively plan their day.

My inspiration initially came from my most recent visit to Disneyland Resort in Anaheim, California. Disney tickets are expensive, and it was clear to me that visitors spend far too much time walking to closed attractions, and waiting in lines for the popular ones. Having gained valuable new skills the previous week at Apple's World Wide Developers Conference, I decided that I wanted to build an app that tracks wait times, and provides tools to help users effectively plan their day.

I first built an app for iOS which integrated both user input, and an API which returned FastPass information and live wait time data. The API was recently taken down after a complaint from Disney, so I needed a way to increase the volume of user uploaded data. This is what inspired me to create an Android version of the app. Having a database and server already set-up would mean that I could create a fairly large-scale app, while also improving the overall service, and allowing me to learn various aspects of Android development. Furthermore, this would help ensure that the user experience wasn't degraded by inaccurate information.

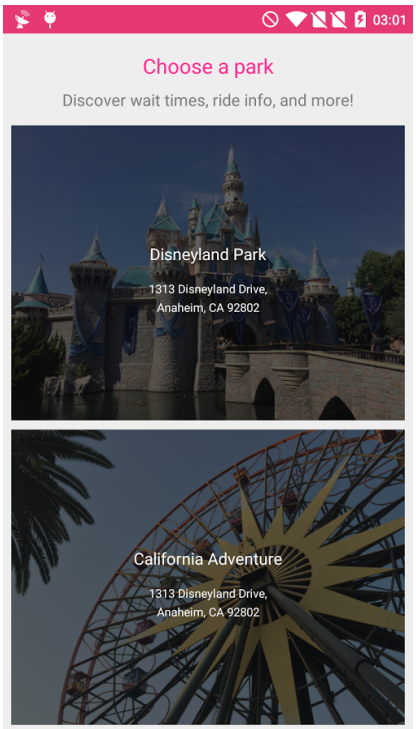
This report will cover the implementation of my idea, the software modelling I carried out, and also a final evaluation which will look at the success of the finished application. I will compare the final product against my original concept, similar applications, and competitors, and I will include feedback provided by users in my evaluation.

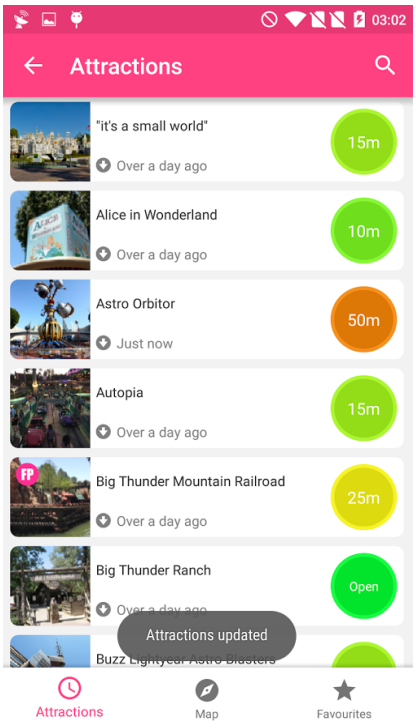
2. Software Modelling



3. App Implementation

Users are first presented with an activity providing the choice of either ‘Disneyland Park’ or ‘California Adventure’. Upon selecting their choice, they are then taken to the main section of the app. Initially I was using drawer navigation, until Google changed their guidelines to recommend tabbed navigation in certain circumstances. This was an obvious choice as all three tabs are of equal importance, and should be quickly accessible. I have provided an ‘up’ navigation item for quick switching between parks.

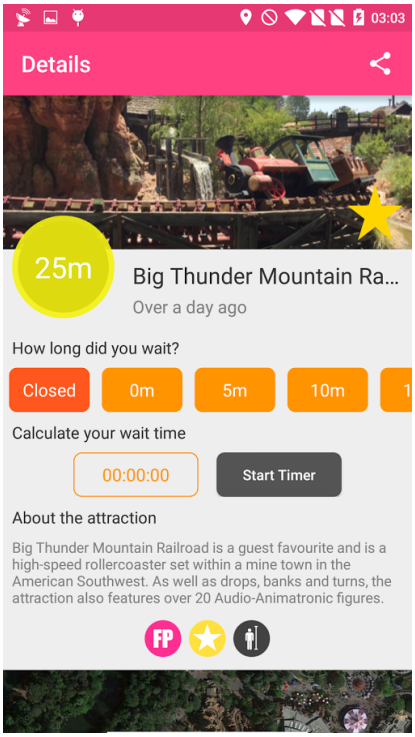


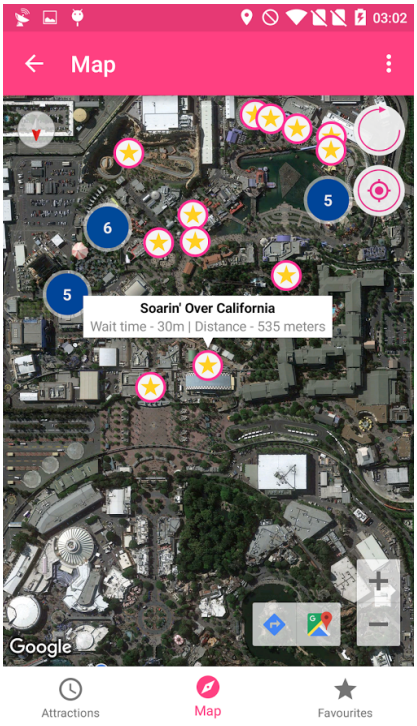


The first tab contains a full list of attractions for the selected park. Users can see the attraction name, a preview image, current wait time, whether the ride has FastPass availability, and the amount of time since the last update. The list can also be searched by attraction name which although useful, isn't vital as I have also sorted the list by alphabetical order anyway. Pull-to-reload is available, and I am also handling error responses with informative descriptions.

The 'Map' tab provides a full-screen map of the selected park. I am using custom rendered markers with Google's cluster utility to display all of the attractions without over-cluttering the map. Tapping a marker will present useful information such as the name, wait time, and distance from current location. The info box can be tapped to access the full attraction details. The top-right side features 'return to default', and 'centre on current location' buttons. I also display the default zoom buttons for users with older, less-responsive screens. Users can switch between map types with the menu item in the action bar.

The 'Favourites' tab is much the same as the 'Attractions' tab, other than displaying only the user's favourite attractions. For this reason, I decided to eliminate the ability to search. Pull-to-refresh is still present.





The final section of the application is the attraction detail activity. Users can access this view from any of the tabs. In terms of information, this activity includes the attraction name, a large header image, the latest update time, a short description, attraction requirements/features (height restrictions, single rider availability etc.), and also a map displaying their location and distance to the attraction. As the app relies heavily on user input, I have made it easy for park visitors to input data. They can either use a handy timer to calculate how long they have waited, or if they already know, they can select a time in increments of 5. The timer will resume after the app has been reopened. Finally, users can favourite attractions, and share them with via an action bar menu item. It's also worth mentioning that attractions without wait times (such as shows) have a simple 'Open' or 'Closed' selection choice, and no timer.

The app uses material design throughout, including on each of the interactive elements, and all icons. I tried to follow Google design guidelines as closely as possible, without losing consistency between this and it's iOS counterpart.

4. Evaluation

My target was to develop an Android version of my iOS application that would retain the same (and in some cases improved) functionality, while adapting the design to fit the operating system. This gave me a concept to work towards that I feel I have achieved.

Design

I wanted to create an application that would not only improve upon the functionality of similar apps in this category, but also one that would look and feel better to use. At the time of starting this project, I wasn't able to find a single application that fulfilled my design criteria, or usability requirements. The app 'Wait Times for Disneyland' has a very utilitarian design, which doesn't fit with the general Disney experience. It is however easy to use, although I would put this down to being feature-light, rather than well-designed. Similarly, 'Disney World Lines', an app designed by reputable company Touring Plans, has a very uninspired design with most of the UI elements taken directly from an old version of iOS.

I feel that I have achieved a strong balance of usability and a modern, fun design. I have used bright colours throughout, and have put a lot of effort into ensuring font weights/sizes are legible in direct sunlight on smaller screens. To make the app easy to use and quick, I had to implement navigation that would allow the user to quickly switch between sections. Initially I wanted to use a fixed top tab bar, however this would make it impossible to pan on the map fragment. I therefore settled on using drawer navigation, however as I mentioned, was able to later remove this for a bottom tab bar.

Functionality

When conducting research of competitors, I saw a lot of potential for improvement in the information-side of the app. While most apps present wait times in a logical way, none of them provide any attraction information. Having found myself having to Google attraction features in the past, it was clear that information such as FastPass availability, and height restriction details would be extremely useful. I also recognised that many users would prefer to view wait times as a map, rather than a list. That way they are able to see attractions nearby with reasonable queues.

I intended to bring all of the functionality of my iOS app over to Android, and I have been successful in doing this, with only a few small exceptions.

Features eliminated due to time constraints -

- Android Wear app (iOS version features an Apple Watch app)
- More complex animations such as parallax effects
- Custom map InfoBox/annotation on markers
- Park tips section

Features improved upon -

- Error response handling now provides more descriptive messages
- Small bugs present in the iOS version have been avoided
- Interaction animations are improved in some cases

Adherence to guidelines

An as iOS user and developer, I spent a substantial amount of team researching Google's design guidelines to ensure I understood the purpose and use cases of particular UI elements. I feel that I have successfully developed an application that maintains brand identity between the operating systems, but one that also works the way users have come to expect. The 'Disney World Lines' app is an example that has completely ignored the guidelines, and is something that I aimed to avoid.

An example of this the use of the 'Up' navigation button. I recognised that this should be used in situations where there is a static, predetermined location, so I used it to provide a quick way to return to the 'Park Select' activity. Within the detail activity would have not been suitable as there are multiple entry points.

Fonts and icons are also consistent with the rest of the system, and as previously mentioned, all of the interactive elements feature material design animations.

Improvements

Given more time, I would have liked to take material animations further, particularly upon activity start. The app does feature some fade in/out animations which I would like to replace with the splash-like animations present throughout Android.

Despite a strong combined user base, many of the attraction wait times can go long periods without being updated. This of course would not be a problem had I still been able to utilise the live API. As a replacement for this, I would like to build a screen-scraping program. I was also able to previously display live FastPass return time information, which would again be made possible.

Performance could also potentially be improved. The attractions list rows are comprised of a rather excessive number of layers, which I suspect could be reduced, improving scrolling frame rate. Similarly, the detail view has a small delay before presenting, which I would like to reduce.

5. Sources

Code

ImageHelper/getRoundCornerBitmap - **http://stackoverflow.com/questions/12550311/rounded-corners-for-an-imageview-in-android** (Modified)

ImageHelper/scaleCenterCrop - **https://github.com/lingochamp/ShareLoginLib/blob/master/share/src/main/java/com/liulishuo/share/util/BitmapUtil.java** (Modified)

Icons

Any icons I have not created myself were sourced from - **https://design.google.com/icons/**

Images

Most attraction images were taken myself. The remainder are copyright-free images from Wikipedia.