POLITECNICO
MILANO 1863

Smart Tourist

Design and Implementation of Mobile Applications
Design Document

Fabio Codiglioni, Alessandro Nichelini

A.Y. 2019/2020

Contents

1 Introduction

This is the *design document* (DD) of the ***Smart Tourist*** iOS application developed by *Fabio Codiglioni* and *Alessandro Nichelini* in the context of the *Design and Implementation of Mobile Application* course at Politecnico di Milano. The authors have been tutored by Bending Spoons for the usage of some technologies that are described later. The document explains the most important design choices we made and the motivations behind them, with specific focus on the Redux-like architecture adopted.

2 General Overview

Smart Tourist is a multi-device application for iOS and iPadOS.

2.1 Assignment

For the reader convenience, the assignment is reported below:

Assignement: Smart Tourist

Help your users when they are exploring new cities!

By using the localisation systems on the smartphones and the google places API, notify the user when something interesting is nearby.

Connect to wikipedia, to retrieve interesting informations on the point of interest selected.

Search and filters among several interesting places of the city you are visiting, save them as favourites so you can check on them later.

Attach some pictures to those pins and eventually share them with your friends!

2.2 Features description

2.2.1 Exploring

The user is given with a navigation map filled with the city's point of interest. She has the chance to control which attraction to see on a map. She can choose between *Nearest places*, *Popular places* and *Favourites*. Each attraction is shown on the map and in a table view. They are clickable from both position to open the corresponding detail view.

In the map, two circles are displayed to help the user better understand distances. These circles correspond to the distance she can cover in 5 and 15 minutes. Circles' radius is automatically updated with information from user profile and thus they fit always with user's pace.

2.2.2 Dynamical exploring

SmartTourist works well in both crowded cities and small towns. The user will be always provided with a right amount of attractions. Where not so many attractions are available, the app will automatically show her less known attraction and particular point of interest.

2.2.3 Learning

The user can open the detail view of both the city and attractions. She will be provided with useful information about the point of interest such as pictures, Wikipedia description, useful links and a shortcut to open turn-by-turn navigation.

2.2.4 Notifications

The user is notified when she is nearby a "top location". The notification comes with a picture of the attraction and lets her either to open the the detail view of the attraction or to open turn-by-turn navigation.

2.2.5 Keep track of favourite attractions

The user can view the attraction of her current city or she can have a preview of other cities attractions. In both case she can keep track of them by adding them in the list of favourite attractions. The user is also provided the ability to open a worldwide map with all her favourite places.

2.2.6 Contribute (proof of concept)

SmartTourist is manly based on free data. When a detailed description is missing, the user is given the chance to contribute.

3 Architectural design

4 Data design

5 User interface

5.1 Screenshot

Here follow some screenshot of the application in both Light and Dark mode.

5.1.1 Mapview

Mapview - nearest

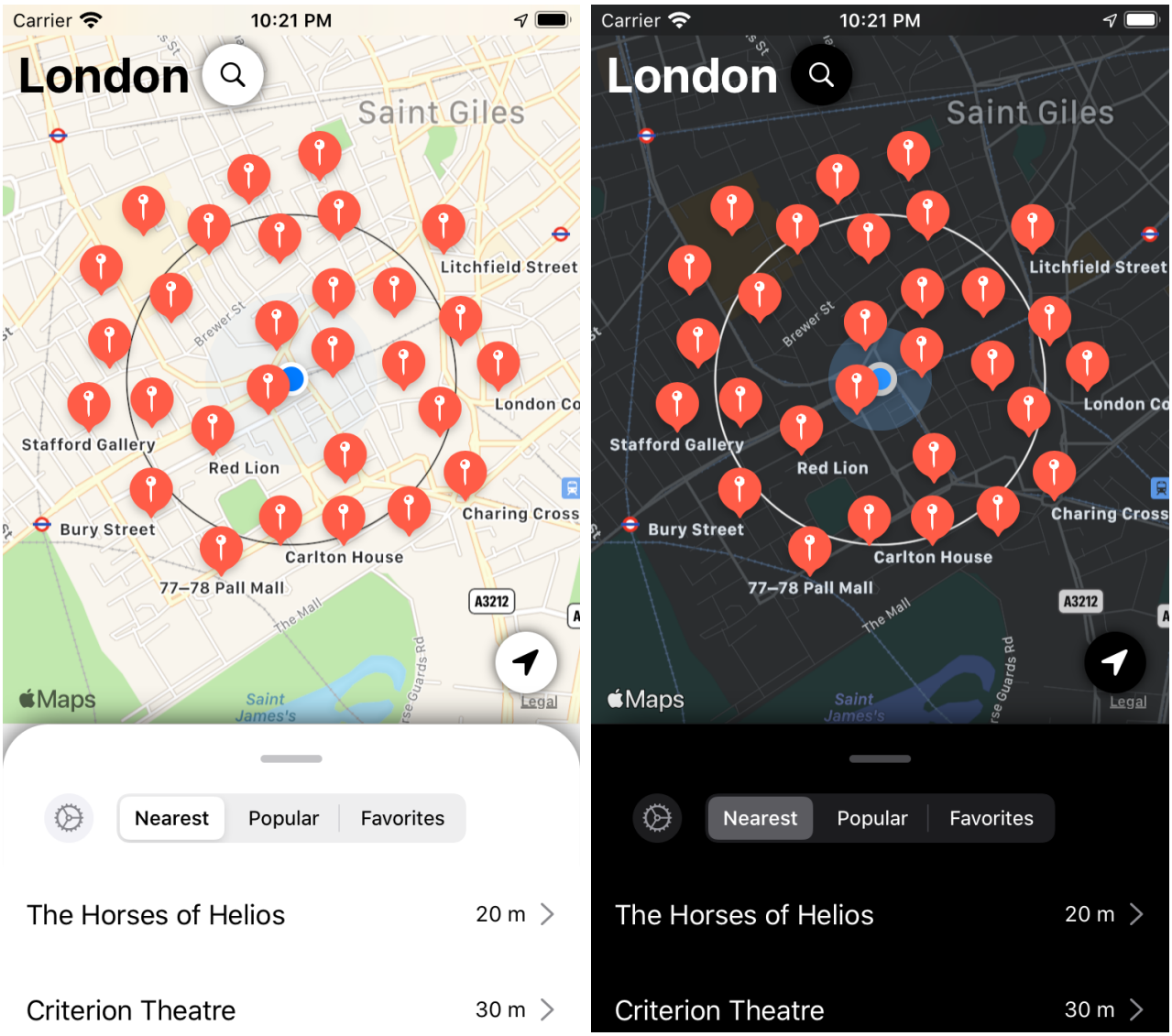

Figure 5.1: Map view with nearest tab selected in both light and dark mode

5.1.2 Mapview - popular

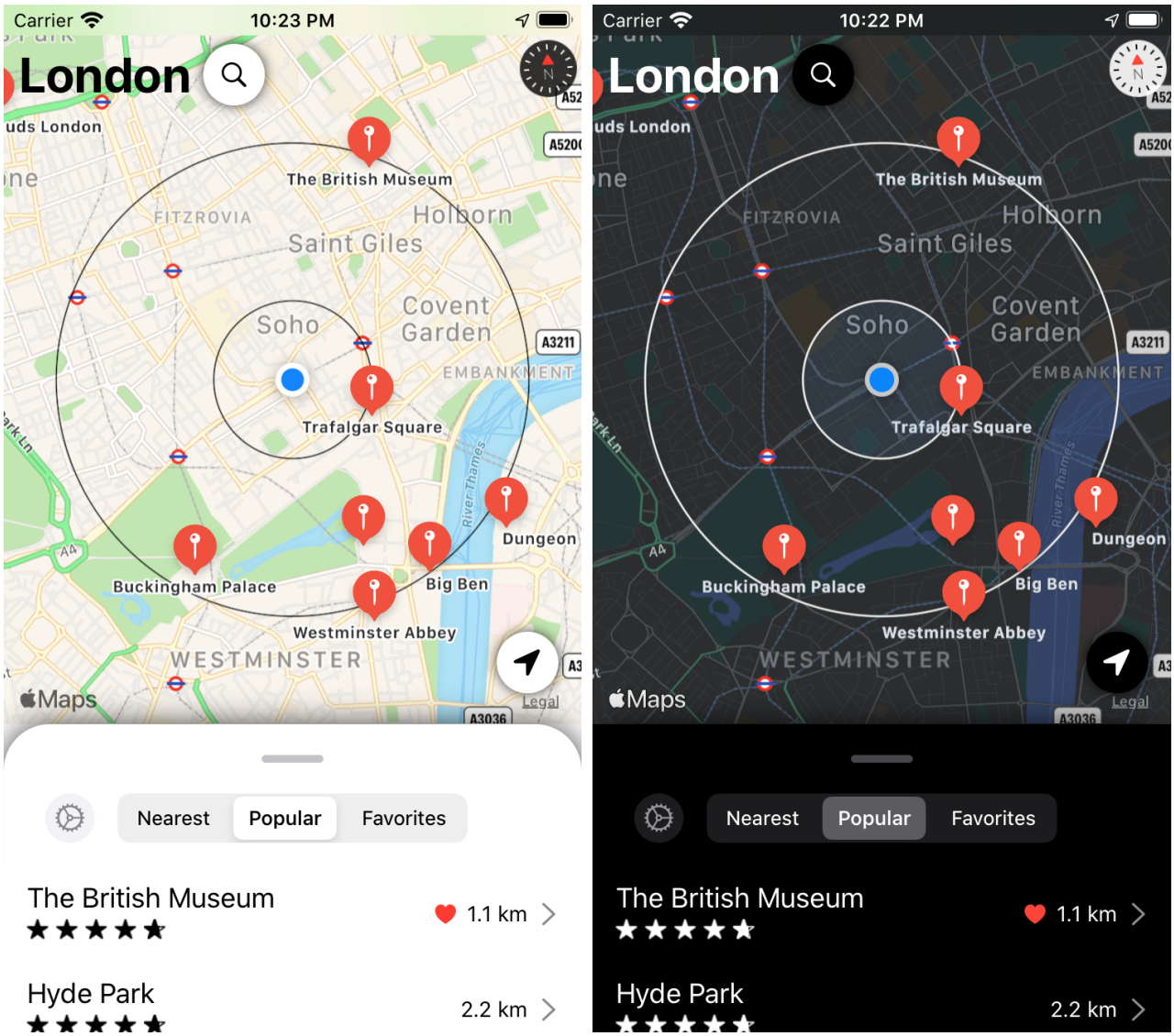

Figure 5.2: Map view with popular tab selected in both light and dark mode

5.1.3 Mapview - favourite

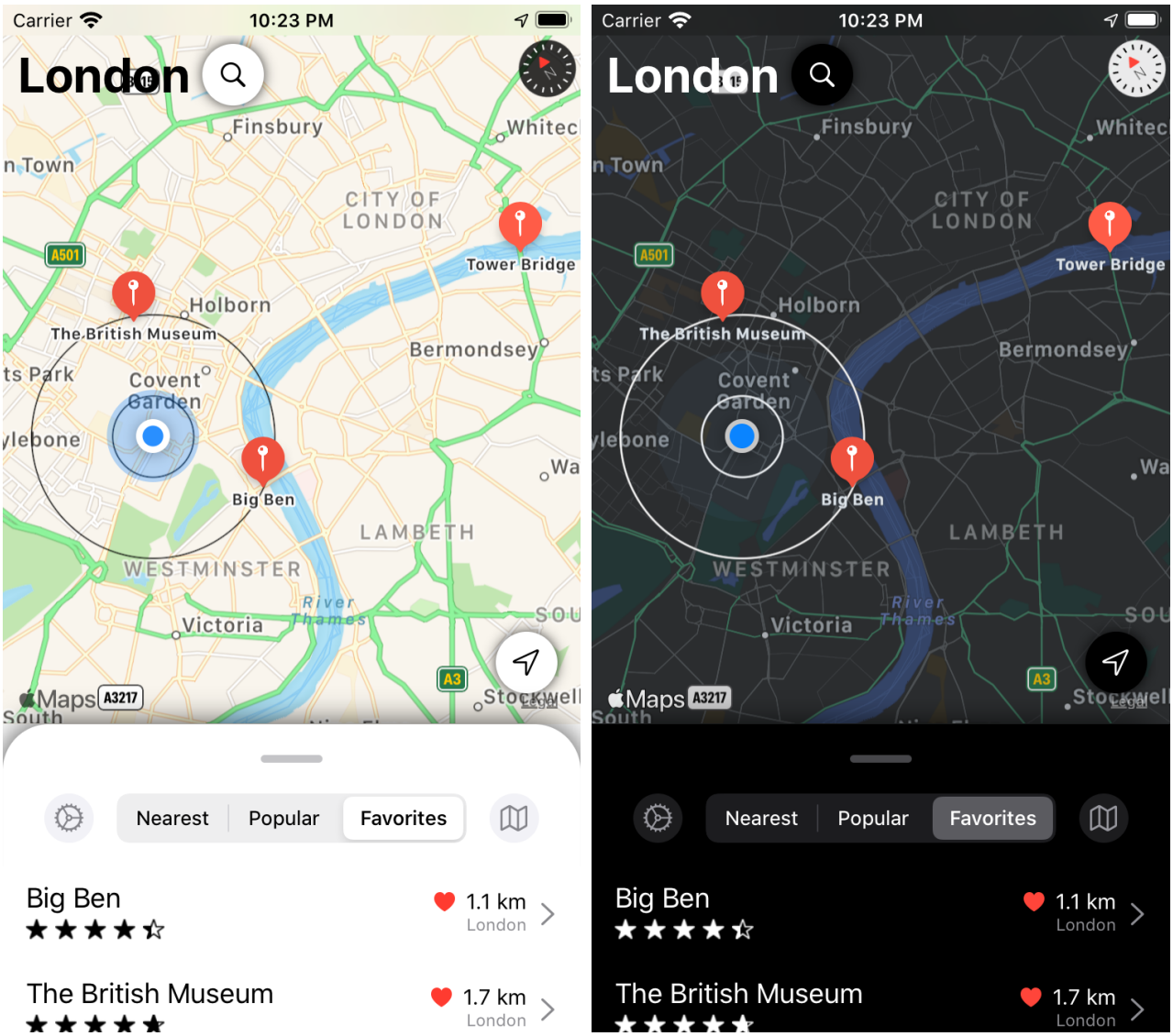

Figure 5.3: Map view with favourite tab selected in both light and dark mode

5.1.4 Global favourite view

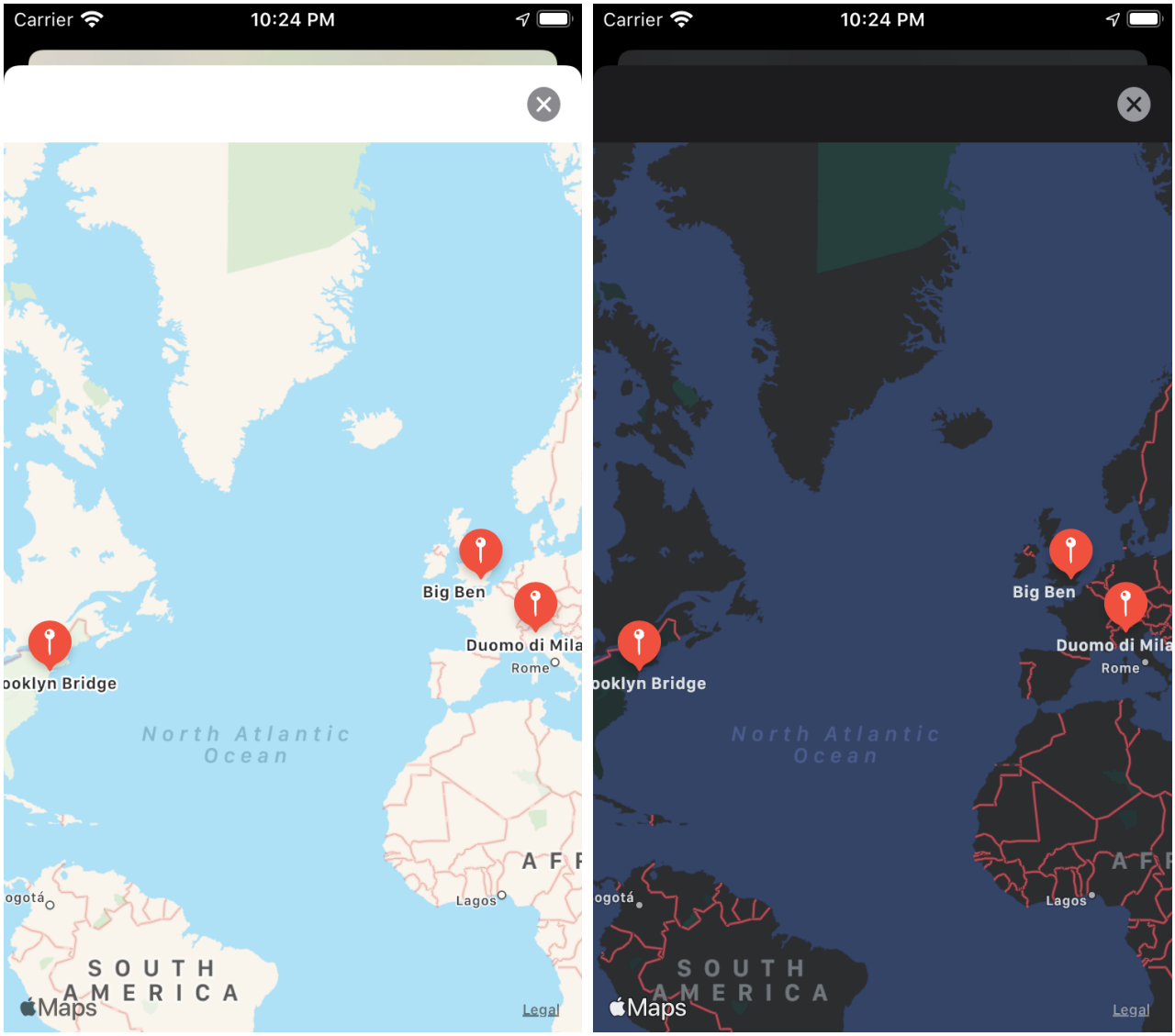

Figure 5.4: Favourite worldwide view in both light and dark mode

5.1.5 Attraction detail view

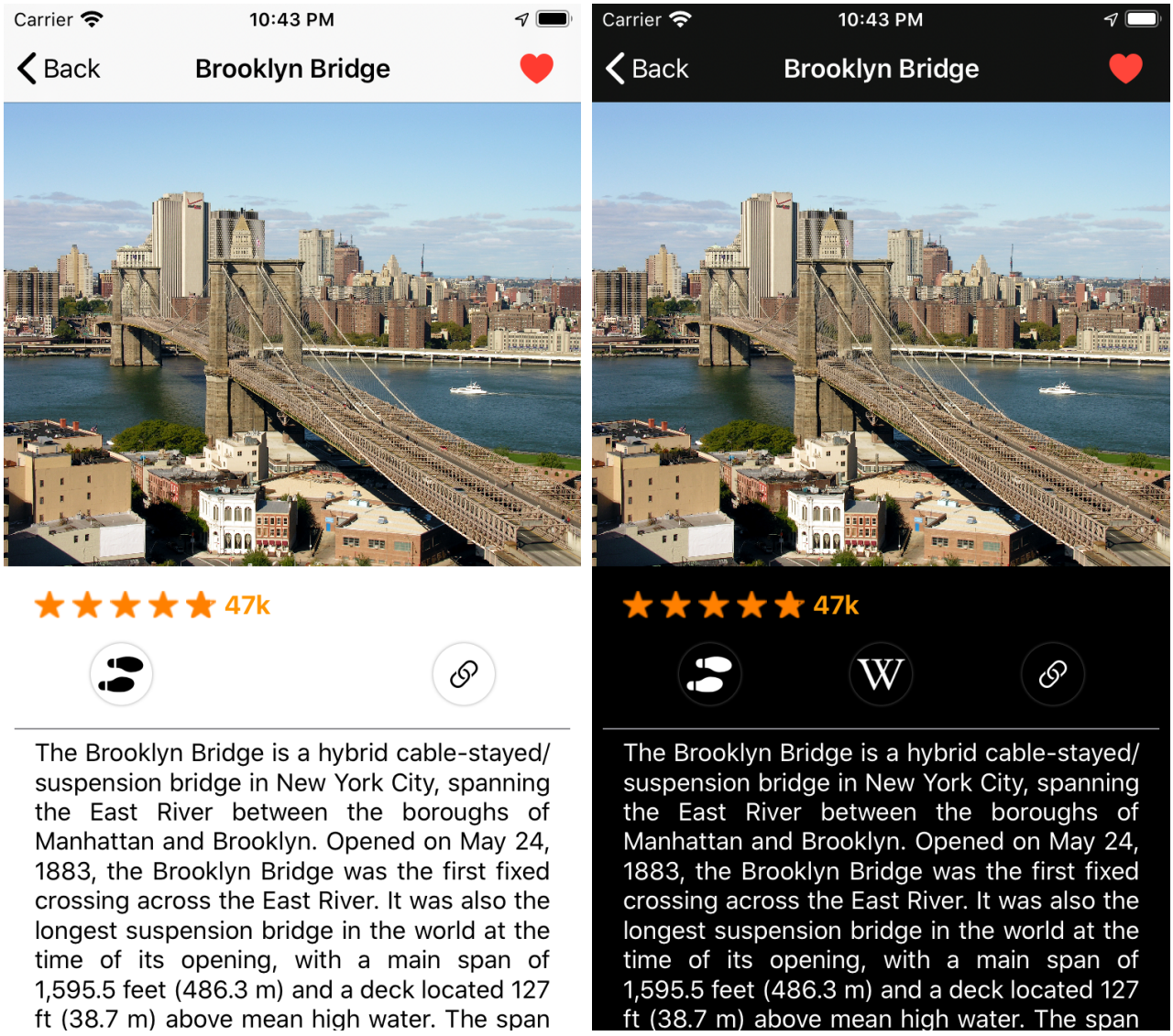

Figure 5.5: Attraction detail view in both light and dark mode

5.1.6 City detail view

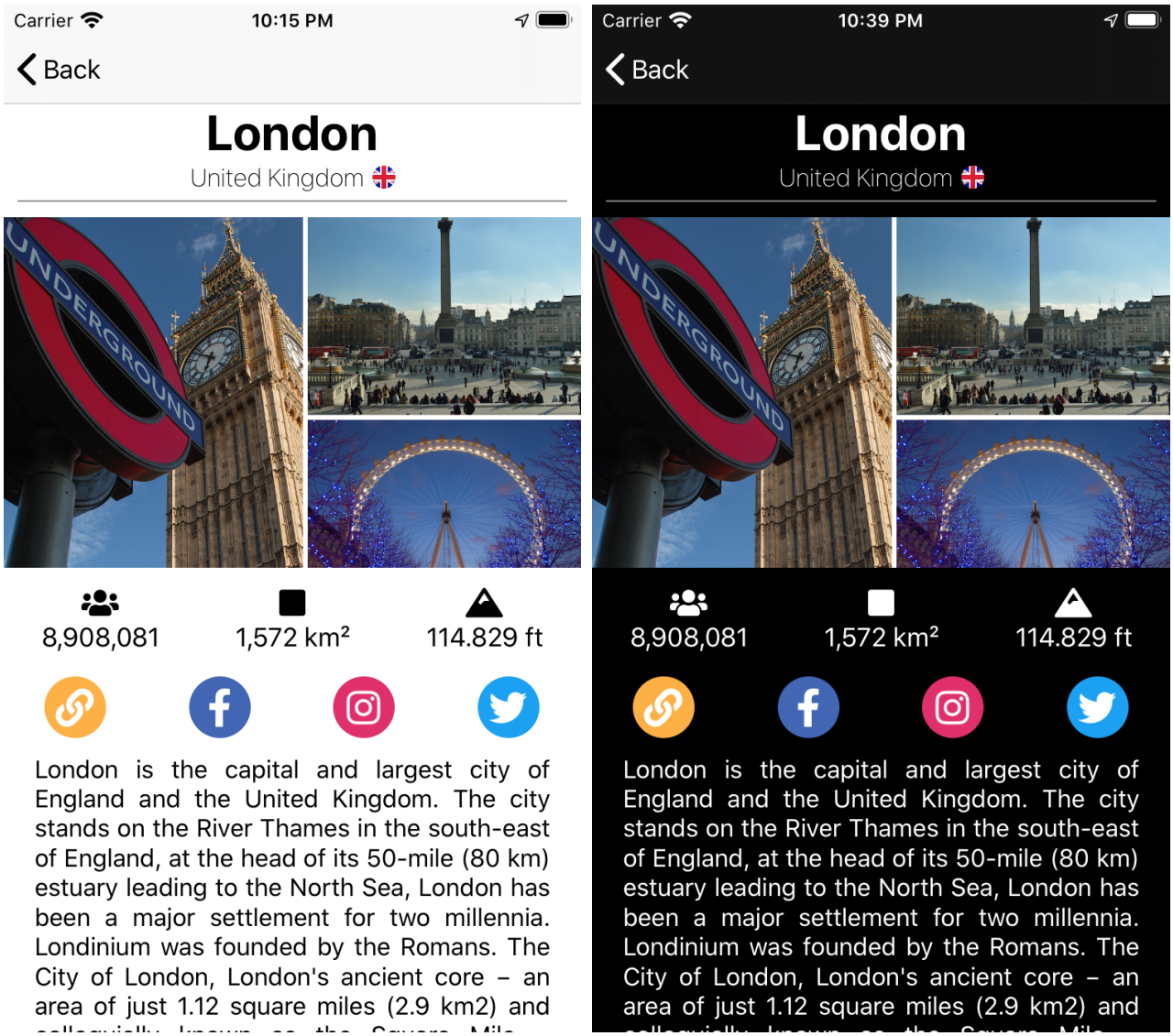

Figure 5.6: City detail view in both light and dark mode

5.1.7 Settings view

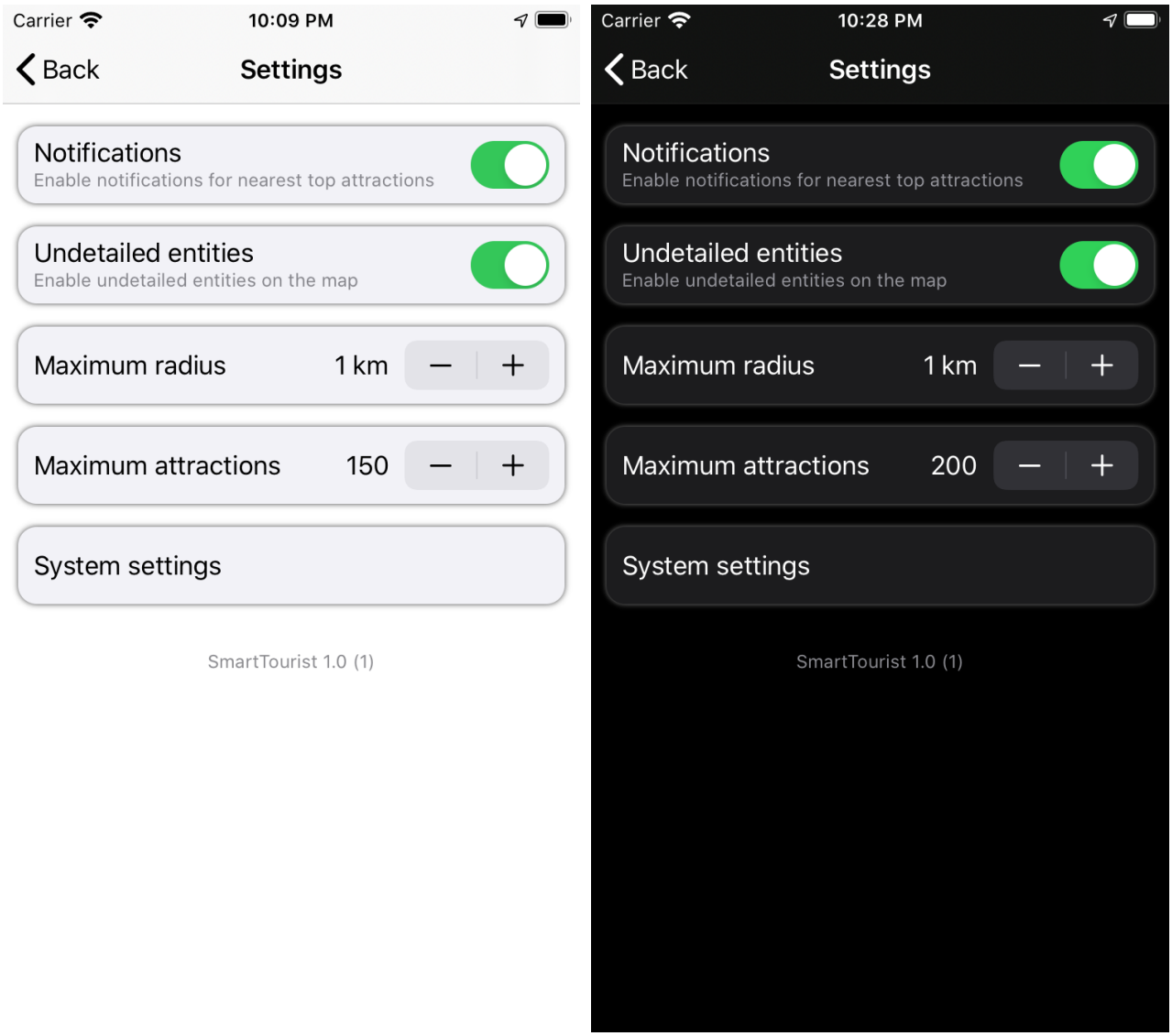

Figure 5.7: Settings view in both light and dark mode

6 Notifications

7 Services and libraries

7.1 Internal libraries

SmartTourist uses consistently internal libraries and services:

- ***MapKit***: for location gathering and maps.
- ***Core Motion***: for accessing accelerometer, gyroscope, pedometer, and environment-related events.

7.2 External services

SmartTourist relies almost only on *free data and services*. Here, it follows the description of the external services used.

- ***Wikipedia***: all displayed attractions are taken from the Wikidata knowledge base. The given API is quite basic: attraction entries are retrieved with a SPARQL query embedded in a http API call. Details and pictures are retrieved with standard HTTP API calls from different endpoints.

```
1
2       SELECT DISTINCT ?place ?placeLabel ?location ?image ?instance ?
            phoneNumber ?website ?wikipediaLink
3       WHERE {
4           SERVICE wikibase:label { bd:serviceParam wikibase:language "en,
                it" }
5           SERVICE wikibase:around {
6               ?place wdt:P625 ?location .
7               bd:serviceParam wikibase:center "Point(\(location.longitude)
                    \(location.latitude))"^^geo:wktLiteral .
8               bd:serviceParam wikibase:radius "\(radius)" .
9           }
10          ?place wdt:P31 ?instance  .
11      ?place wdt:P18 ?image .
12      OPTIONAL {?place wdt:P1329 ?phoneNumber}.
13      OPTIONAL {?place wdt:P856 ?website} .
14      OPTIONAL {?wikipediaLink schema:about ?place;
15          schema:inLanguage "en";
16          schema:isPartOf [ wikibase:wikiGroup "wikipedia" ]} .
17      }
```

Listing 7.1: SPARQL query for attraction retrieving

```
1           SELECT DISTINCT ?city ?cityLabel ?country ?countryLabel ?
                population ?area ?elevation ?link ?facebookPageId ?
                facebookPlacesId ?instagramUsername ?twitterUsername ?image ?
                coatOfArmsImage ?cityFlagImage ?countryCode ?wikipediaLink
                WHERE {
2           BIND( <http://www.wikidata.org/entity/\(cityId)> as ?city ).
3           OPTIONAL {?city wdt:P17 ?country}.
4           OPTIONAL {?city wdt:P1082 ?population}.
5           OPTIONAL {?city wdt:P2046 ?area}.
6           OPTIONAL {?city wdt:P2044 ?elevation}.
7           OPTIONAL {?city wdt:P856 ?link}.
8           OPTIONAL {?city wdt:P2013 ?facebookPageId}.
9           OPTIONAL {?city wdt:P1997 ?facebookPlacesId}.
10          OPTIONAL {?city wdt:P2003 ?instagramUsername}.
11          OPTIONAL {?city wdt:P2002 ?twitterUsername}.
12          OPTIONAL {?city wdt:P18 ?image}.
13          OPTIONAL {?city wdt:P94  ?coatOfArmsImage}.
14          OPTIONAL {?city wdt:P41 ?cityFlagImage}.
15          OPTIONAL {?country wdt:P297 ?countryCode}.
16          OPTIONAL {?wikipediaLink schema:about ?city;
17            schema:inLanguage "en";
18            schema:isPartOf [ wikibase:wikiGroup "wikipedia" ]}.
19          SERVICE wikibase:label { bd:serviceParam wikibase:language "en".
                }
20      }
```

Listing 7.2: SPARQL query for city detail retrieving

- ***Google***: Places ratings are the only data retrieved from Google services. This was due to the lack of a reliable free alternative. For the simplicity of the task, we decided not to rely on the Google SDK for iOS and to manually make request to the Google API.

7.3 External libraries

The app uses lots of third parties libraries the can be roughly divided into three kind: architectural libraries, back-end libraries and front-end libraries.

Kind	Library	Description
Architectural	Katana	Katana is a modern Swift framework for writing iOS applications' business logic that are testable and easy to reason about. Katana is inspired by Redux.
	Tempura	Tempura is a holistic approach to iOS development, it borrows concepts from Redux (through Katana) and MVVM.
Back-end	DeepDiff	DeepDiff tells the difference between 2 collections and the changes as edit steps.
	Fuse	Fuse is a super lightweight library which provides a simple way to do fuzzy searching.
	Alamofire	Alamofire is an HTTP networking library written in Swift.
	SigmaSwiftStatistics	It is a collection of functions that perform statistical calculations in Swift. It can be used in Swift apps for Apple devices and in open source Swift programs on other platforms.
	SwiftyXMLParser	Simple XML Parser implemented in Swift.
Front-end	PinLayout	Extremely fast views layouting without auto layout. No magic, pure code, full control and blazing fast. Concise syntax, intuitive, readable and chainable. PinLayout can layouts UIView, NSView and CALayer.
	FlexLayout	Angular Flex Layout provides a sophisticated layout API using Flexbox CSS + mediaQuery.
	Cosmos	This is a UI control for iOS and tvOS written in Swift.
	ImageSlideshow	Customizable Swift image slideshow with circular scrolling, timer and full screen viewer.
	MarqueeLabel	MarqueeLabel is a UILabel subclass adds a scrolling marquee effect when the text of the label outgrows the available width.
	FontAwesome	Use Font Awesome in your Swift projects.
	FlagKit	Beautiful flag icons for usage in apps and on the web.

8 Testing

9 Effort spent