WPI

FreedomTour:

A Freedom Trail Itinerary Planning App

Project Team:

Samuel Mailand sfmailand@wpi.edu

Project Advisors:

Professor Wilson Wong

Professor Lane Harrison

Abstract

FreedomTour is an itinerary-based iPhone application aimed not only at providing the tourists of Boston a helpful way to plan their day on The Freedom Trail, but also creating a template that can be applied to any major city. While there are applications currently on the market that either provide information about the Trail or offer planning capabilities, none exist that allow for the user to fully plan their day in Boston, both on and off The Freedom Trail. The current version of this application allows users to add restaurants, museums, public parks, and Trail locations to any number of different itineraries, but the concepts developed within this application could be expanded to include any walking tour, or even be used by tourists to plan an entire visit to a city.

Table of Contents

Table of Figures

Table of Tables

Chapter 1 Introduction

Planning a day trip in an unfamiliar location can often be a daunting task. There are thousands of different possible stops, the majority of which are likely not of interest to the planner. In the past, travel agencies have been heavily used to ensure that a trip goes well, but this is becoming less true as technology evolves and becomes a more common aspect in everyday life. Since the Internet has become more prevalent, anybody with a computer, tablet, or smartphone can do the necessary planning online without ever talking directly to a person, and it can often be done for free. As the number of adults who own a smartphone grows, approximately sixty-four percent of Americans at the beginning of 2015, the average person has an increasingly convenient and centralized way to plan and modify their trip themselves instead of through somebody else, such as a travel agent.

There are, however, few applications that allow for the sort of planning that a tourist would do for a trip to a city. In order to work towards addressing this lack of products, an application was developed that focused on planning a day on the Freedom Trail. With 4 million tourists visiting The Freedom Trail annually[1], and at least five different iOS applications dedicated to the Trail, there is a clear market for technology to be available to the public for touring the historic location. The majority of applications currently in the

[1] (The Freedom Trail Foundation, 2015)

marketplace, however, do not take complete advantage of what the new technology has to offer. For example, if users want to go somewhere not on the Freedom Trail, they currently have to use other applications such as Yelp to find that information. If an application only allows for users to enter in information that they've already found instead of providing recommendations, then the app has no more utility than a notebook and a map.

To develop an application that would allow for outings in a city to be planned, we researched the applications available on the iOS App Store, which were divided into two main types, apps that dealt specifically with The Freedom Trail, and those that were designed for general planning purposes, of which both types had their own benefits and drawbacks.

Many different frameworks, libraries, and architectures are used within this application, all of which are discussed in further detail in Chapter 3. Requirements and user stories provided the path for the development of this application, which broke down into eight different iterations, each of which is linked to a specific feature of the application. After we developed requirements and user stories, we moved on to determining major classes for the application. Chapter 6 discusses the graphical design of the application and Chapter 7 provides an evaluation of the Freedom Trail App. The paper closes with future work and conclusions.

Chapter 2 Background

Currently Available Applications

Before we began development of the new application, we reviewed what was currently available. This was not only the best way to ensure that the application did not already exist, but was an excellent way to learn from the mistakes of others. While ten applications were reviewed in total, four are described below. These four applications were selected because they best represented the four different categories of vacation planning mobile apps found in the marketplace. *WorldMate* represents the application specifically targeted at recording plans that one has already made. *FreedomWalk* shows applications which focus only on conveying information about The Freedom Trail, and *Freedom Trail* is the best example of poor user interface design. The final application, *Discover Boston*, comes the closest to the type of application that this project aims to create, but lacks in a number of important areas, including the inability to save locations and a large number of its features being blocked behind seemingly random pay walls.

WorldMate

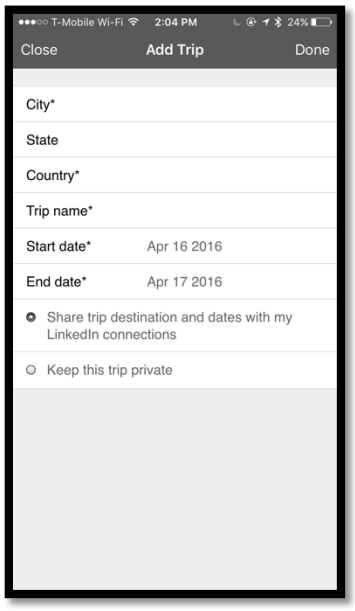


Figure 1 - Creating a New Itinerary

When first creating an itinerary with *WorldMate*, the user is immediately prompted to enter in the specific trip details, including the City, State, Country, and trip dates. This step cannot be skipped, and all the information on the screen must be entered, meaning that there is no way to avoid filling out every piece of information.

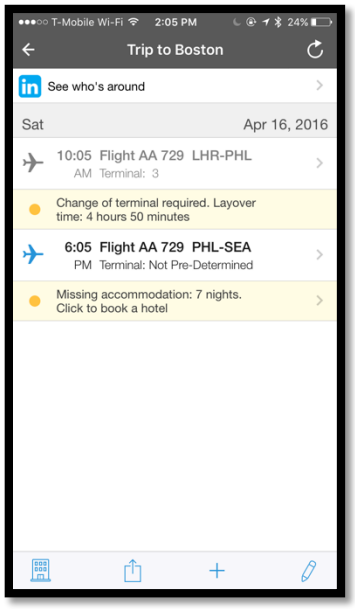


Figure 2 - Adding Flight Information

Once the main details have been entered, the user can add any type of stop or accommodation that he/she wishes to. The example above shows what would happen if a flight were to be added. In the case of flights, information is auto populated based on the airline and flight number, as it appears to be pulled directly from the airline's information pages dynamically. Hotels, car rentals, other ground transportation, a general meeting, and other information can also be added. The buttons at the bottom of the page allow the user to find a hotel, add locations, share the trip, or edit

the name of the trip. The refresh button at the top of the screen refreshes any dynamic info, such as flight data.

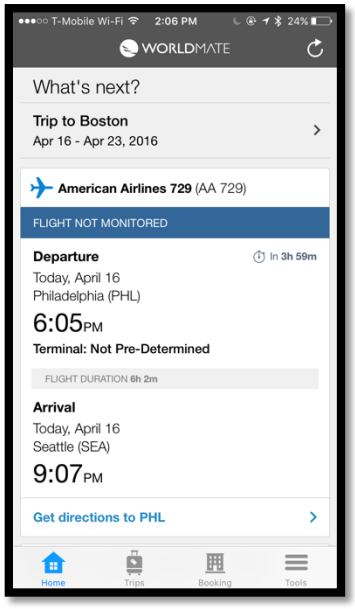


Figure 3 - Viewing Important Information

On the main screen, where all itineraries are shown, there is a section of the application that is dedicated to showing the user the important information that relates to upcoming events. For example, in Figure 3 there is flight information being shown that relates to the "Trip to Boston" itinerary. If there were multiple travel notices coming up, the screen would show all relevant information so that the user knew everything that would be happening within the foreseeable future.

Overall, the itinerary specific applications, demonstrated above by WorldMate, are simply not designed in a way that would work for tourists visiting the Trail. Tour guide applications need to provide information to the user, whereas these applications all assume that the user already knows his/her plans.

FreedomWalk

Figure 4 - Trail Overview

This application opens on an overview of the tour. This application was last updated in 2010. Buttons are shown that likely once directed the user to some relevant websites and videos but none of them currently work, likely due to it being outdated. Such issues, probably caused by the lack of updates for this app, are present throughout the entire application.

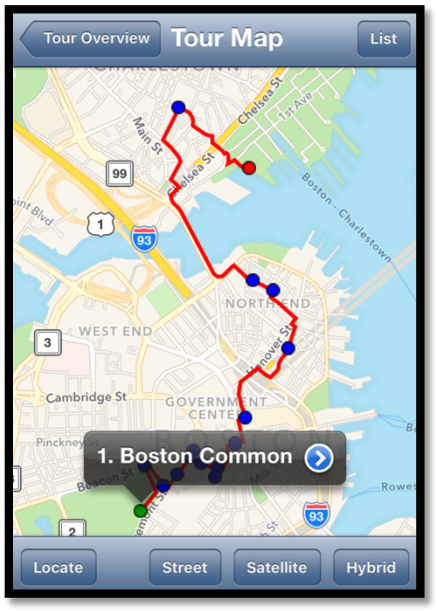
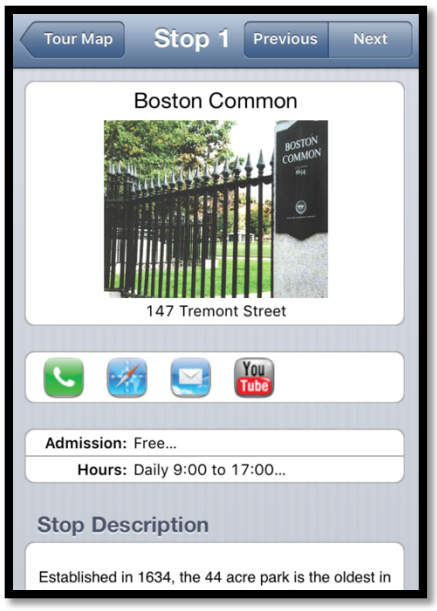


Figure 5 - Tour Map

Tapping on the tour overview text brings the user to a map of Boston, with the Trail shown and each of the locations displayed as a dot. Selecting a dot brings up the name of that location in a bubble, which can then be selected again to get more information about the historic landmark.



The next page shows the details about a specific Trail location, including a paragraph or two about the location, the admission fee, and the daily hours. Similarly to the first screen, it has icons that likely used to do relevant actions, but no longer work due to the lack of updates.

Figure 6 - Trail Location Details

The largest problems with this application are the outdatedness and the heavy reliance on the map for navigation. It being outdated means that it often crashes and its UI is not consistent with present day iOS, making it confusing to the user. The heavy use of maps makes the navigation feel awkward and adds levels to the menu hierarchy that make finding the important information more difficult.

Freedom Trail

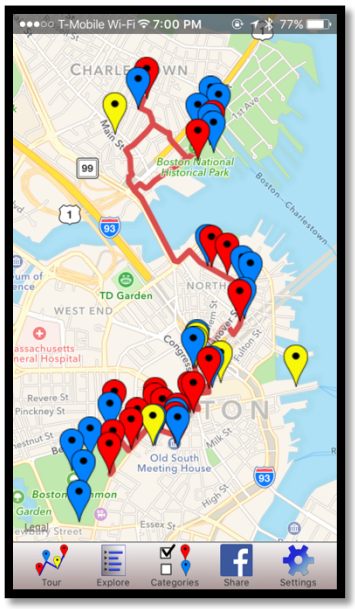


Figure 7 - Main Screen

The opening screen for this application shows a map of Boston with various pins placed on it, representing points of interest in the city that relate to the Trail. There is no key to what each of the colors mean, and not all of them are stops on the Trail.

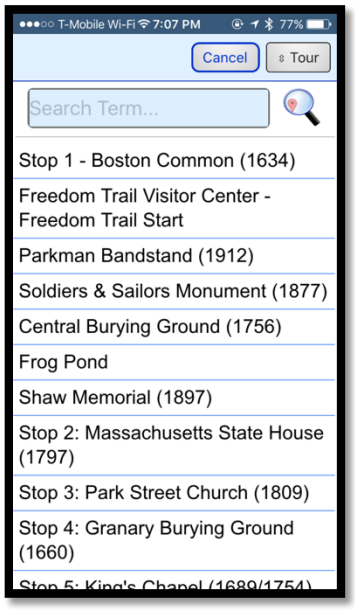
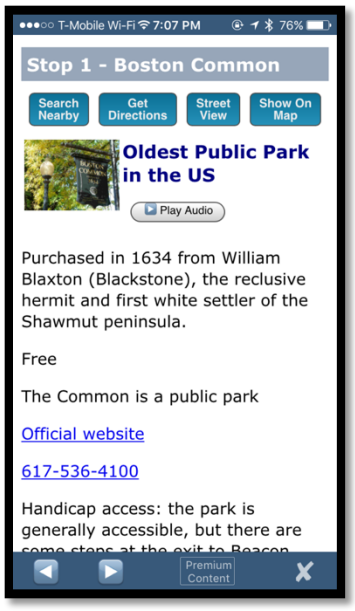


Figure 8 - Points of Interest

Tapping on the “Tour” tab button brings up this page, which shows the names for all of the pins on the map in Figure 8.



Each location on the map has a corresponding details page, which appears to be a page in a web browser, instead of an actual page in the application.

Figure 9 - Trail Site Detail Page

This application also focuses only on locations on The Freedom Trail, but it falls into a separate category due to its confusing user interface. It uses what appears to be a tab view, a standard design within iOS, but the behavior of that navigation system is entirely different from the standard UI practices of the operating system (OS). Instead of each tab representing a single view, where all the other tabs are still visible, these tabs bring up different pages that overlay themselves onto the tab bar. This overlay makes it difficult for the user to create a mental model of what is occurring within the UI, meaning that navigation becomes confusing.

Discover Boston

Figure 10 - Discover Boston Main View

The *Discover Boston* app for iPhone is similar to Yelp in many ways, as it stores various points of interest that the user can sort through, with the primary difference being that it only has locations in the Boston area. It also provides other information relating to Boston, such as weather, transit directions, and even a section about musical artists who were originally from the city.

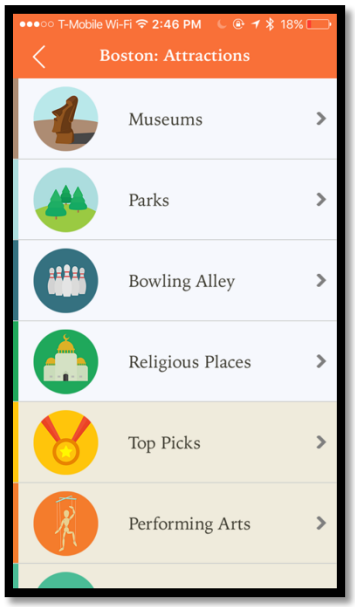


Figure 11 - Attraction Types

After tapping a location type, in this case "Attractions", a more detailed list is shown. The darker options in the user interface (UI) are attraction types that are only available for paid "VIP" members.

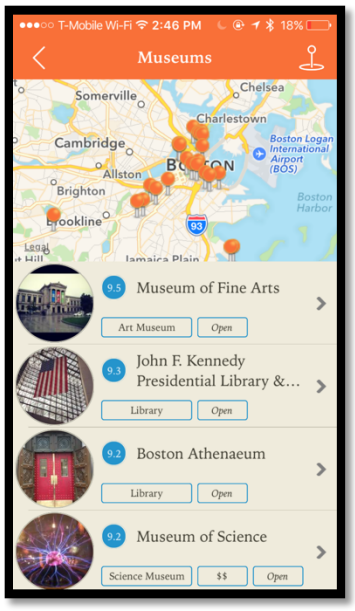


Figure 12 – Museums

The next screen, reached by clicking "Museums", shows specific locations that the user would be able to visit. The top half of this screen shows a map, with each of the locations being represented by a pin. Zooming in and out is possible, and the pins can be tapped to provide more information. This bottom half of the shows whether the location is open, the type of museum, a picture of the location, and a rating that has been pulled from FourSquare. The placement of the two halves of the screen are completely static, meaning that both are always shown, not just one or the other.

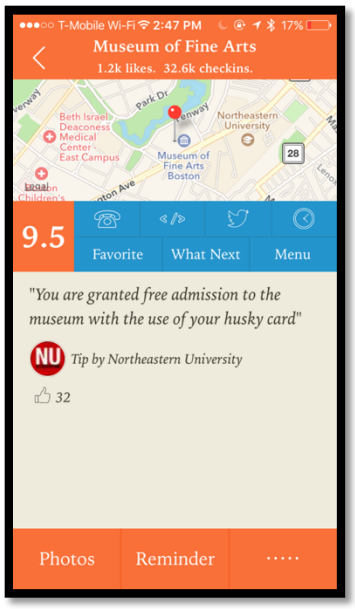


Figure 13 - Location Details

This information page allows the user to view specific details about a single location. It contains information such as the location's phone number, hours, and a menu if it is a restaurant. The menu button is always shown, even if the location is not a restaurant. A "What Next" button is also shown, which provides the user with a list of locations that he/she might want to visit next.

This application provides a wide variety of information to the user about the different attractions across the city of Boston. It does an excellent job of providing reviews, general information, and directions to these locations, but falls short when it comes to planning and general use. While it does have a "What's Next" button, which provides recommendations on where to go next, it does not allow the user to plan a day of stops. It also struggles to find the balance between making the application free to use while still making money. The application is free to download, and offers a couple free features, but a large portion of it is behind various pay walls, which could cause the user to pay a fairly large sum of money if they want everything unlocked. This is not ideal for a tourist that may only use the application once before deleting it.

Limitations

Outdated

The most significant and obvious issue of these apps is how often they are updated. As of December 2015, six out of the eight apps mentioned above had not been updated in the app store within the last year; five of the apps had not been updated in nearly a year and a half, and two seemed to be abandoned, having not being updated for five years. The *FreedomWalk* application (Figures 4, 5, and 6) is the oldest Freedom Trail application on the Apple App Store, and has not been updated since February 18th, 2010. The app still uses iOS's old layout style and is built to fit on the smaller displays of iPhone models older than

the iPhone 5, which was released nearly 6 years ago, in September 2012. On any larger phone, the app no longer fills the entire screen.

User Interface

Since the *FreedomWalk* application was updated, iOS 7 was also released, which completely changed the look, feel, and navigation of the entire user interface for iOS devices. iOS moved from a skeuomorphism[2] model to the more modern flat design in this update. The standards of user interfaces within the smartphone world are always rapidly changing, and making sure to stay current can make app navigation less intuitive for users.

[2] A design concept of making items resemble their real-world counterparts.

Chapter 3 Methodology

Iterative Development

The development of this application was completed using the iterative software development life cycle[3]. This process fit the project the best, as it was important to get the minimal viable application developed as quickly as possible to begin testing, and then iterate on that initial progress. Each feature of the application constituted an itineration.

First Iteration: Minimal Application

The initial application involved incorporating the most minimal information into the application. We first added the location names to the table view framework that Apple provides, followed by navigating to a page of information (dummy info initially) when the location is clicked. This information page was the first custom UI built for this app.

Second Iteration: Improved Navigation

This iteration focused on improving navigation within the application, and navigation to The Freedom Trail destinations. It enabled the "Get Directions" button that was already on the information page for each location. This linked to Apple's built in Maps for directions.

Another piece of navigation that was improved in this iteration was the navigation between the information pages of each location. Instead of having to go back to the main

[3] The software development lifecycle consists of six stages: requirement gathering, analysis, design, implementation, testing, deployment, and maintenance.

list of locations, this allowed the user to be able to tap on arrow icons to immediately go to the next Trail stop. The main table is separated out into two tabs, one for Trail locations and one for the planned itinerary. While at this stage, the itineraries page is not used; enabling this navigation is important to the continued development of the app.

Third Iteration: Itinerary Building

This third iteration involved the actual planning of an itinerary. From the Trail locations table, a user was able to slide the location over and select "Add" to add it to the current itinerary. This location was then disabled (grayed out) so it cannot be added again, but will appear on the itinerary tab.

Fourth Iteration: Saving Itineraries

The next step was to enable itineraries to be saved, and then displayed in the itinerary tab. Saving an itinerary involved saving its name. These itineraries then showed up in the itineraries tab. This change also meant that whenever a new itinerary was being created, the locations being added showed up in an itinerary named "New Itinerary" at the top of the list of saved itineraries to keep it separated from the others.

Fifth Iteration: User Interface

Prior to this iteration, the interface that the application had was not exactly what the final design showed, so the pieces that had been completed needed to be moved around

slightly. The first change was to move the two tabbed main screen of the app to include “My Itineraries” and “Popular Itineraries” instead of the list of Trail locations and “My Itineraries”. The second change was to add a “Create Itinerary” button on the “My Itineraries” page that directed the user to a new page that had two tabs on the bottom of the page, one for the list of locations, which was the same table-view used in previous iterations, and one for the locations added to the itinerary already.

With the completion of this fifth iteration, the minimum application that would be able to be published to the app store was completed. While this application could have been published and used by tourists, this iteration did not mark the end of development. In order to ensure that the user had the best experience possible, we wanted to ensure that dynamic data was used for places such as restaurants and museums, as this would mean that that information would be as up to date as possible. At this point, the application also did not allow users to edit it once it was created, which is not ideal, since it caused mistakes to be permanent.

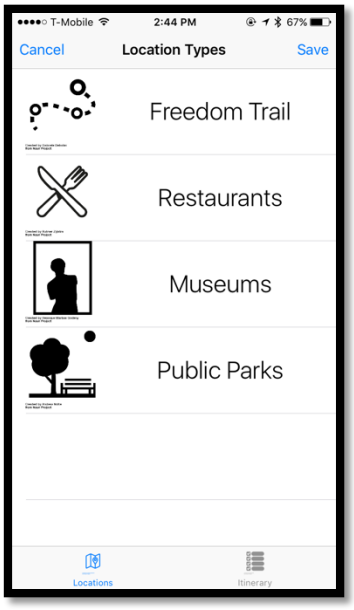


Figure 14 - UI of the Selection Menu for Location Types

Sixth Iteration: Yelp Integration

The previous iterations had allowed only Trail locations to be added to the itinerary; however, functionality to add other points of interest, such as restaurants, to the itinerary was also needed. To do this, we used the Yelp API. The navigation table was altered slightly to include different categories (Trail Locations, Restaurants, etc.), so that the added locations were not in one list that would have been difficult to sort through.

Listing all restaurants in the area would be difficult to accomplish and keep up to date, so instead of names, the restaurant section contained types of restaurants (Italian, American, etc.). These types could be added to the itinerary just like any other location, but were able to be clicked on to bring up Yelp locations from within the app. Clicking on these locations caused the Yelp app to be brought up and perform a search for that type of restaurant and

the results were displayed like a traditional Yelp search. After viewing the restaurant information in Yelp, the user then was able to direct himself back to the *FreedomTour* application to add the restaurant.

Seventh Iteration: Editing An Itinerary

The next step after enabling all of the different points of interest to be added to the user's itinerary was to allow the user to edit the itinerary beyond simply deleting an item. This iteration focused on editing the order of an itinerary that the user is working on, as well as itineraries that have already been created.

Eighth and Final Iteration: Time Estimation

This iteration focused on providing the user with a time estimate for different stops on their itinerary. A start time for the itinerary that can be edited by the user was added at the top of the list and it also used a combination of average time at a location and the travel time found through the Google maps API to estimate how long an itinerary will take, and when the user will arrive at each location.

iOS Architecture

The development of this application was done in Swift[4], as it is Apple's new standard for development on iOS. On top of being easier to code and read, Swift also manages memory

[4] Swift is Apple's latest programming language to be used for both iOS and Mac development. It replaces Objective-C, which had been used for development since the original iPhone was released.

automatically, meaning mistakes are less likely. This section covers the various frameworks, patterns, and libraries used within this application.

Model-View-View-Model

The model-view-viewmodel (MVVM) design pattern is heavily based on the model-view-controller (MVC) pattern that many software designers are already familiar with. The only difference between MVC and MVVM is that there's a "view model" class, and that the view controller no longer has access directly to the model. The primary piece of this pattern is the view-model, as it contains all of the information and states that the view uses. Unlike MVC, there is no circularity, as the view-model is referenced by the View, but is not aware of the view itself. This is also true of the model, which is referenced in the view-model, but is unaware of the view-model existing. Figure 15 illustrates MVVM[5].

[5] http://artsy.github.io/images/2015-09-24-mvvm-in-swift/mvvm.png

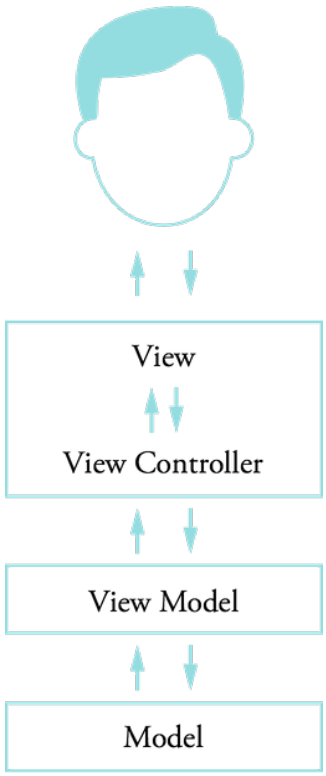


Figure 15 - MVVM Diagram

Observer Pattern

The observer pattern is one where a specific object/class is able to post or subscribe to specific actions that are taken from anywhere else within the application. This is a useful tool because the classes involved do not need to know anything about each other. One class can simply post a notification, and any number of other classes will see that notification and perform the desired action. Within The Freedom Trail application, this is used whenever an itinerary is updated. Whenever any change is made to an itinerary, it posts a notification that is used to make sure that other classes have the updated itinerary information. For example, if a location is added to an itinerary, then the view that shows

all of the itineraries locations will need to be refreshed; otherwise, it will not contain this new location. Similarly, if an itinerary name is updated, the view that displays every itinerary that has been saved will need to make sure that its list is refreshed. Both of these classes would "subscribe" to the itinerary updating notification, so when the itinerary is modified, they are immediately alerted and can run their respective *refresh* functions.

OAuthSwift

OAuth is protocol used to connect the resources of various applications and website. It is used by sites such as Twitter, GitHub, and Dropbox. In order to use it on iOS, there is a library that has been created called *OAuthSwift*, which is hosted on GitHub[6]. Within this application, this library will use OAuth1.0 to connect to the Yelp API and retrieve restaurant information.

iOS Databases

Similarly to Android, iOS uses SQLite for storing and persisting data. The method for storing and retrieving data is, however, abstracted into the NSCoding protocol. Any NSObject can be stored using this method, which covers any data types or custom objects. Figure 16 below shows the encode function, which must be implemented in any class that stores data using the NSCoding protocol. The *encodeObject* function requires only the object and the key with which that object will be associated.

[6] https://github.com/OAuthSwift/OAuthSwift

```
func encodeWithCoder(aCoder: NSCoder) {
        aCoder.encodeObject(name, forKey: nameKey)
        aCoder.encodeObject(photo, forKey: photoKey)
        aCoder.encodeInteger(rating, forKey: ratingKey)
}
```

Figure 16 - NSCoding Example

Data Sources

Yelp

Using Yelp to find local restaurants is a key part of this app, as the user will likely want to add meals to their trips. The initial plan with this integration was to incorporate deep linking[7] to allow the Trail app and Yelp to transfer information between each other. For example, the Trail app would direct to Yelp with a specific search, and then the user would select a restaurant, and that restaurant's information would be transferred back to Trail app.

Unfortunately, in order for the Yelp information to transferred back to The Freedom Trail app, Yelp would need to implement deep linking into their app to recognize when it is being called from the Trail app. Without working with Yelp and having their software engineers program this functionality into their application, it is not possible.

[7] Deep linking is a methodology for launching a native mobile application via a link. Deep linking connects a unique URL to a defined action in a mobile app, seamlessly linking users to relevant content.

Since the original plan was not an option, a secondary plan was developed. This plan involves calling the Yelp API directly and pulling and displaying restaurant information directly within The Freedom Trail app. If the user would like more information about the specific restaurant, they will be able to click on the restaurant within the Trail app and it will open the Yelp app directly. They can then view the information they want, navigate back, and add the restaurant if they desire.

Figure 17 shows a query URL provided by Yelp to access results from its API.

```
https://api.yelp.com/v2/search/?location=Worcester, MA&limit=1&radius_filter=1000&category_filter=thai
```

Figure 17 - Yelp API Call

The query allows for the restaurant type and location to be specified, as well as the distance from the location (in meters), and the total number of results that are returned. The returned data from Yelp provides a large amount of information on a location, including name, address, phone number, an image URL that can be displayed, as well as the Yelp link, which can be used by the user to navigate to the Yelp app or website for more information. Figure 18 shows an edited response, which includes only the information used in the final application, the full results can be seen in Appendix A.

```
"businesses": [
      {
         "rating": 3.0,
         "mobile_url": "http://m.yelp.com/biz/thai-time-
worcester?utm_campaign=yelp_api&utm_medium=api_v2_search&utm_source=-KjeymOM8cXvmxhHxr2iJQ",
         "review_count": 66,
         "name": "Thai Time",
         "snippet_text": "I really like the fried rice with chicken or steak. The tea is decent too. Food isn't
expensive either.",
         "image_url": "https://s3-media3.fl.yelpcdn.com/bphoto/trK6MGOCmvvOJWzEbmtZYg/ms.jpg",
         "id": "thai-time-worcester",
          "coordinate": {
                "latitude": 42.2709097,
                "longitude": -71.8068917
            },
         }
      }]
```


Figure 18 - Modified Yelp Response

The pieces of the Yelp response that this application used are shown in Figure 18. These include the GPS coordinates, the business rating, name, and Yelp URL, as well as a picture to show next to the name of the restaurant.

Apple Maps

Apple has provided a URL scheme for developers to use within their apps. Figure 19 shows the scheme that allows for the developer's app to open directions in the built-in *Apple Maps* app.

```
http://maps.apple.com/?saddr=San+Jose&daddr=San+Francisco&dirflg=w
```

Figure 19 - Apple Maps Directions URL

The URL shown above is broken into three sections: the source address, the destination address, and the method of travel. The method of travel for the purpose of this app will be the 'w' flag, for walking, as the user will be walking the Trail.

Twitter

The group that manages The Freedom Trail has a Twitter account that they use to post relevant information. Integrating the feed from this account would be incredibly helpful for allowing users to quickly see any important information. This information could be closures, changes in the hours of operation of an attraction, or special events that are happening. For example, in October, 2015 the Freedom Trail offered their first ever lantern guided tour and on February 12, 2016 they announced that they were not doing any tours due to the cold weather. This type of announcement integration is ideal for those users that don't have a Twitter account, or don't know how to navigate well online, as they will not have to worry about finding the information, since it will be in the app.

TestFlight

Testing of this application was done through TestFlight, a system provided by Apple specifically for having people test applications on their phones. The system essentially works as a sort of specialized App Store, where the developer of an application can invite others to download the application through the *TestFlight* app. The application is then updated automatically when a new beta version is pushed. This integration makes it significantly easier for applications to be tested by non-developers. Before TestFlight was

released, the only way for anybody to test iPhone applications was to have access to the source code and then compile and load it onto their own phones, rather than anybody being able to be added to the testing network and being able to download the application whenever updates were pushed. TestFlight also provides a feedback button, which allows the testers to directly send feedback to the developer.

Chapter 4 Requirements

User Types

There are three types of users that this application originally planned to focus on learners, planners, and those who are already on the Trail.

Planners

The planner is somebody using the application with the goal of creating an itinerary to follow for when he begins walking the Trail in Boston.

In the case that the planner has never used the app before, he will want to easily create his first itinerary and begin adding locations to it. Unlike the learner, the planner does not need to have detailed information on the Trail's landmarks when creating his itinerary, as he plans on doing the majority of his learning once he arrives in Boston. This means that there should only be a brief summary for each of the Trail's location; just enough to give him an idea of what he might expect once he begins his walk. In the event that he is curious about other historical landmarks in the area that are not part of The Freedom Trail, he should be able to just as easily add those stops to his planned itinerary.

At some point in the day it is likely that the planner will need to take a break from walking the Trail to find a place to eat, drink, shop, or even just use the restroom. While these types of stops should be able to be planned, they must also easily be able to be added in during the walk, as the user might decide that he wants something to eat after he has already

started walking the Trail. The suggestions provided by the app must also be location sensitive, as walking halfway across the city is likely not ideal.

In the event that the planner cannot complete his itinerary in one sitting, he should be able to save it with a custom name and come back to it at a later date. This saving feature also allows him to be able to save multiple itineraries for the event when he wants to spend multiple days on the Trail focusing on something different each day.

Before he starts his walk, it is also possible that the planner does not know a great deal about what there is around The Freedom Trail and might need some help. To help with this, the app will always provide a list of the most popular routes taken by other users. These pre-defined itineraries can be reviewed by the planner and added to his personal itinerary list if desired.

Once the walk has begun, the planner will be given directions to the next stop on his itinerary in the case that he is not sure where to go. Once at the location, a page of historical and other relevant information will be displayed to the user. There will also be a "nearby" section to show what other stops are in the area. These stops can be added to the itinerary during the middle of the user's walk without any trouble, as the itinerary is editable at any time.

Users on The Freedom Trail

These users are the ones using the application while in Boston, whether they are downloading the app in the city for the first time, or are planners who are now walking

their itinerary. For this user, convenience and speed will be key. This user likely has never been to the city before and needs the important pieces of information available to him without having to enter in any information, as planning applications such as *WorldMate* require.

Learners

The learner is somebody who is only using the application to learn more about The Freedom Trail, but has no desire to plan a trip to the Boston to walk the Trail. Through the initial design planning, it became clear that putting more features into this use would compromise the functionality and usability of the primary focus of the project: planning an itinerary.

User Stories

Itinerary Creation and Modification

- As a tourist, I want to be able to browse Freedom Trail Locations and add them to an itinerary so that I can easily keep track of where I will be traveling.
- As a tourist, I want to save multiple itineraries with custom names, which I can view/edit later so that I can have a variety of options to view in the future.
- As a tourist, I want to be able to add stops, such as restaurants, to my itinerary so that I can take a break from my day to grab something to eat.

- As a tourist, I want the restaurants that are recommended to me to be nearby the other stops on my itinerary, that way I don't have to walk halfway across Boston just to have lunch.
- As a tourist, I want to be able to see when I will arrive at specific locations based on a set start time, so I can have a better feel for my day.
- As a tourist, I would like to see what the most popular non-Trail points of interest are on The Freedom Trail so that I don't need to wonder whether it will actually be important and interesting.

Starting and Completing a Walk

- As a tourist, I want to be given directions to each location, but not lose my place in my itinerary so that I don't have to keep scrolling through the locations that I've already visited.
- As a tourist, I want to be able to modify my itinerary at any time without having to start the entire walk over, because nothing ever goes as planned and the plan needs to be flexible.
- As a tourist, I want to have a brief summary, as well as a more detailed description of each Freedom Trail location I visit so that I can learn something on my walk.
- As a tourist, in the case that the location is not on The Freedom Trail, I want to be able to easily see all of the restaurant's information, including its reviews, overall

rating, menu, and some pictures so that I can get more information before I decide if that's where I want to go.

- As a tourist, I want to be able to find locations that are nearby so that I can add them to the itinerary so that I add spots in the moment that aren't on the Trail.

Look up Information About Any Freedom Trail Landmark at any Time

It is important that the application has this feature for two main reasons. The first is so that users can see information about the landmark while planning their visit. This will allow them to see whether the location is interesting to them and see admission cost, which might impact whether they visit that spot. The second reason is to allow the app to be used by anybody what is looking for information about the Trail.

Build an Itinerary for his/her Trip

The user needs to have the ability to select the specific points of interest that he/she would like to visit, as walking the entire Trail might not be an option. This itinerary should also be editable at any time so that the user can add, remove, or change the order of stops. The app will also recognize the physical order of the landmarks on the Trail and ask the user if they would like to use that order in the case that it is different from what is being entered. This would just help to save travel time, as the user would then not have to go backwards on the Trail to a spot they might already have passed.

View and Add Restaurants to the Itinerary

Either before or while the user is walking the Trail, they should be able to easily look up and add restaurants to the itinerary. Restaurants should be pulled up based either on current location or nearby based on the locations already in the itinerary (user can decide when finding one). The Yelp API will provide the restaurants and reviews for the user so that they can have as much information as possible when deciding where to go.

Directions

Once the user has started his walk, the application will go to the information page of the first stop. On this page there will be a button for getting directions to this location. This will redirect to an outside GPS navigation service (Google or Apple Maps), which they can follow to the desired location. The Freedom Trail app will continue to run in the background and provide a location aware notification once the user has reached the destination.

Using another navigation application such as Apple or Google maps would leverage the features and benefits of those applications, while still allowing the user to jump seamlessly back in to the tour guide app. It also gives them the option to not click the navigation button and either follow the brinks on the Trail to the next location, or take their own path if they see something else along the way. Once the user is done at that location, he can click to move on to the next landmark.

Chapter 5 Analysis

This section contains is a detailed list of each of the components of the application, and how they fall into the model, view, view-model architecture.

Models

Location

The *Location* model represents any location that could possibly be added to an itinerary. There is also a FreedomTrail and Yelp location, which both extend the location class.

Itinerary

The *Itinerary* model represents a single itinerary that a user creates.

View Models

ItitinerariesViewModel

The itineraries view model is used by the view controller to work with itineraries. It maintains an array of all of the itineraries that the user has created, as well as all processing done to those itineraries.

FreedomTrailLocationModel

Since the locations on The Freedom Trail are never changing, a model was created to store all of the Trail's information, which can be accessed by the proper view controllers.

Controllers

LocationTypesTableViewController

This table view controller shows the four different types of locations that the user can add to the itineraries: Trail, restaurant, museum, and public park.

LocationDetailViewController

The details for each location of the itinerary is shown here, including the picture, summary, description, and the button to get walking directions to the location.

ItineraryLocationsTableViewController

The various locations that an itinerary holds are shown in this, which include the name of the location, and its picture.

ItineraryTabTableViewController

This controller was the tab controller that switched between the locations that could be added to the itinerary, and the locations that had already been added to the itinerary.

LocationTableViewController

This controller provides the information about specific Trail locations, such as the Old North Church.

SegmentedViewController

This controller allows for the first page of the application to switch between saved itineraries, and the stored popular itineraries.

ItineraryListTableViewController

This controller is used for showing the different saved itineraries that the user has created.

ParkTypeTableViewController

This controller shows the different public parks within Boston, it pulls the parks from Yelp directly.

MuseumTypeTableViewController

This controller shows the different museums within Boston, it pulls the museums from Yelp directly.

RestaurantTypeTableViewController

This controller shows the different types of restaurants that Yelp allows users to search by.

YelpLocationsTableViewController

This controller shows Yelp restaurant results that it pulls from Yelp's servers. It provides the information for displaying the star ratings, names, and number of reviews that the start rating is based on.

Chapter 6 Design and Development

Interface Design

Already Existing Designs

While there was a variety of applications already on the market for building an itinerary, or walking The Freedom Trail, none of the applications that currently exist find a balance between the two that puts the focus on a tourist building an itinerary to visit a city, or in this case, The Freedom Trail.

Creating an Itinerary

Itinerary applications appeared to be more popular than those for The Freedom Trail, but all of them focused on building a complete schedule for more formal trips, such as a vacation or a business trip. All of them required a large amount of information to be entered in first before anything else could be done. When a tourist downloads a guide application so that finding landmarks to visit is easy, having to enter in large amounts of data is likely going to be a deterrent. In this regard, keeping the required information for creating an itinerary to a minimum is incredibly important.

Use of Maps

A common design decision that all of The Freedom Trail apps and *Discover Boston* shared was the over use of a map element as the main mode of navigation, with *FreedomWalk*

relying most heavily on the use of a map. While maps have a place in applications, using them as the primary mode of navigation can often be crippling to the user experience.

The second screen of the *FreedomWalk* app only showed a map with the line for The Freedom Trail and dots that represented each of the Trail's stops (Figure 5). The major problem with relying so heavily on maps is that information is often not obvious to the user. It is unclear exactly what each dot on the map represents, as only once the user taps on the dot does it display the name of the location. The small dots also pose a problem due to the fact that they are fairly small. Apple's guidelines recommend that every icon is at least 44 pixels by 44 pixels, which is not the case for the dots on the map. While the map provides a good representation of the Trail stop's physical location in the city, the need for the user to zoom to the dot, carefully tap on such a small icon, and then tap on the name of that location just to get information is much more complicated than it should be.

Discover Boston attempts to find a balance between relying only on maps, and showing a list of each location in its application (Figure 12). This balance is, however, problematic due to the inability to make either half of the screen be minimized if so desired. This means that the user is left with two halves of a small screen to view two UI's that each require a full screen to fully function.

Initial Design Sketches

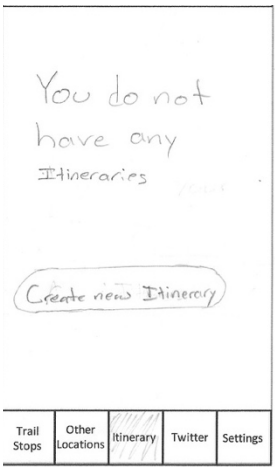


Figure 20 - Initial Design Part 1

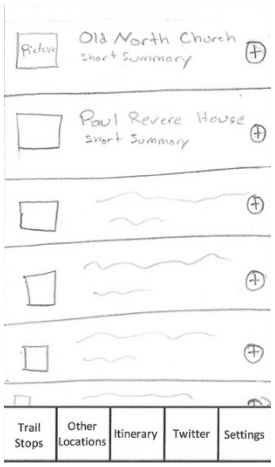


Figure 21- Initial Design Part 2

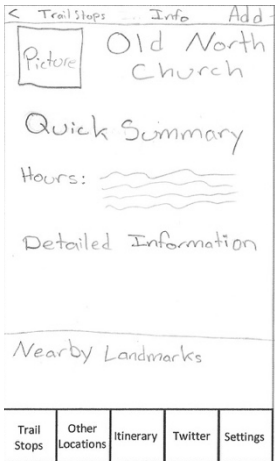


Figure 22- Initial Design Part 3

Figure 23- Initial Design Part 4

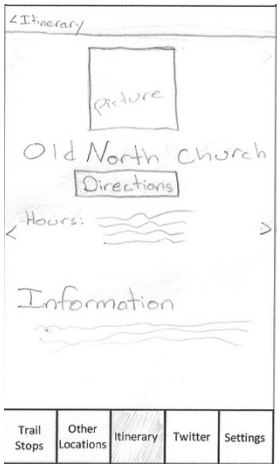


Figure 24- Initial Design Part 5

Figure 25- Initial Design Part 6

When we began designing the application, we first turned to the basic tabbed view provided by Apple. The thought with this layout was that everything that the app offered would be immediately clear to the user, as of the different pieces of the information would be broken into the separate tabs, such as the Trail locations and their information would

only be accessible under the “Trail stops” tab, and the itinerary information would only be accessible under the “itinerary” tab.

It, however, quickly became clear that an interface that was as clearly defined as that was not possible for this type of application, as it was difficult to confine each possible workflow of the application to a single tabbed view. For example, a user could get to Trail information from both the “Trail Stops” tab (Figure 21), and from the Itinerary Tab (Figure 24), which would lead to the same exact page being shown under two different tabs.

The plan was to have the application open on the Itinerary tab, since the focus of the application would be building itineraries. To make sure that the user knew what to do next, there was a message to them with a button to start creating a new itinerary. This is where the first problem arose. There were two options for where in the interface this button could lead, the first being to just pop up a new window that had all of the different locations to add, and the second being to go to the “Trail Stops” tab. Both of these were, however, problematic from a user interface point of view.

With the first option, the button moved the focus away from the tabs and into another window, which can be seen if the tabs are cut off the bottom of Figure 20. The problem with this is that the new page that pops up is showing information that is already accessible from the tabs that were already there. Another problem with this was that this page would only be accessible if the user had never saved an itinerary before. Once an itinerary was created in this other window, the user would have to create the second one

using the tab controllers. Having two very different methods of doing the same thing in an interface is confusing to a user, which is why this UI was never pursued.

The problem with the second option is slightly more straightforward. If the “Create New Itinerary” button simply leads to another tab in the application, it would be unclear to the user that the switch had happened. Even in the case that the user realized what had happened, it still is not good design practice to have two buttons perform the exact same action.

The initial design was also focused on the idea that there would be users whose primary use of the app would not be to walk the Trail, but rather to simply learn about it. This meant that they had to be able to see information about locations without building an itinerary or start a walk. This extra information screen (Figure 21) was removed when the focus of the application shifted away from this type of user.

Design Concept

When designing an interface that a large number of people will use, there is inevitably going to be a large number of skill levels that a user has. First time users of any application need to be able to navigate the app without trouble, but expert users will likely want shortcuts for the features that they use often. One popular example of this technique is copy and pasting on any modern computer.

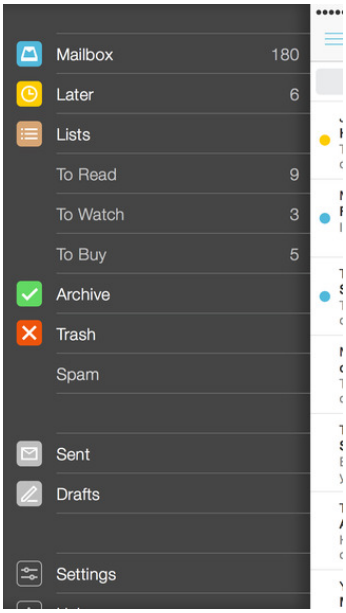


Figure 26 - Mailbox Side Menu

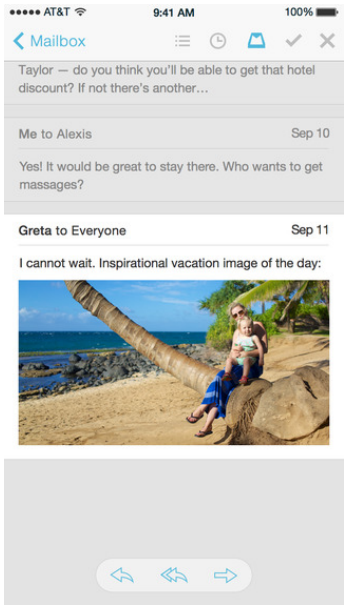


Figure 27 - Mailbox Message View

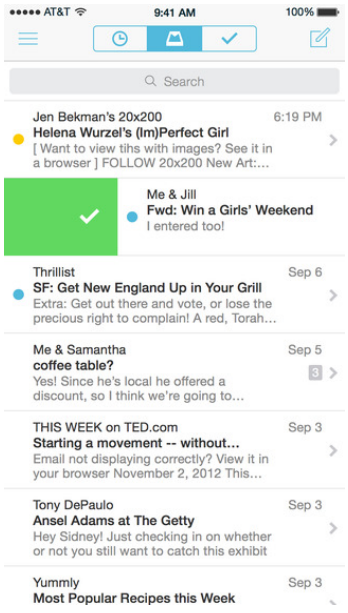


Figure 28 - Mailbox Slide to Archive

Another good example of an application that has multiple ways to access the same information or perform the same action is the email app, Mailbox. For example, both Figure 27 and 28 show a way to archive a message using two separate methods. Figure 27 shows the screen the user sees when he is viewing an email, which has a check button in the top right corner to *archive* the message. Figure 28 shows the table-view that the user sees displaying all of his emails, where he can slide the cell to the right to perform the same archiving action. This not only ensures that the action can be taken from either screen, but also that basic users have a more straightforward (but slower) method of archiving mail, while advanced users have a faster way to quickly process their emails.

Figure 26 and Figure 27 also show the various methods of navigation within the app. Figure 26 shows the hidden hamburger menu that the user can use to move between folders, while Figure 28 shows the top tab bar, which provides the same functionality, just without showing the specific folder name.

For the initial release of this application, we made the decision that there would be no map component besides the directions between locations. This decision is due to the confusion and difficulty that can be caused when the primary mode of navigation in an application is a map. While navigating with a map is not ideal, it is important to take into account that people like to visualize their day on a map as a way to see where they're going, or where they've been. It is for this reason that it is possible that a "View Map" button could be added in future releases. This addition is, however, not a vital part of itinerary planning, so it was left out of the first version.

Besides the removal of a map component, it was also important to view what other features do people look for in an itinerary, and what features are not needed. The largest complaint about the current Trail applications are that they are entirely static, and offer no benefit over a printed brochure or map. Pulling nearby locations, such as restaurants or museums, was also another common request in both the Trail specific and itinerary applications. This makes sense, as somebody visiting a city for the first time will likely not

know exactly what the “hot spots” are. *Discover Boston* is the closest example to dynamically pulled data available for download, and was used as a reference point.

Final Application

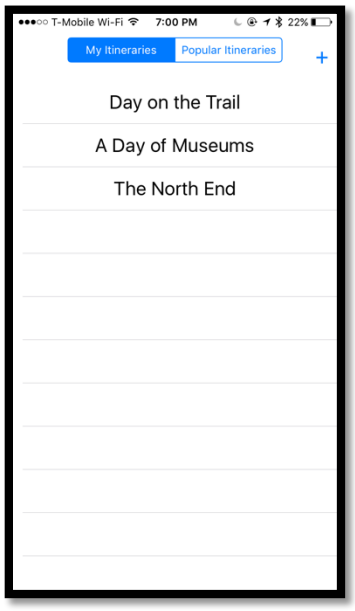


Figure 29 - Final App, Main Screen

The application opens to show the user all of the various itineraries that they've created.

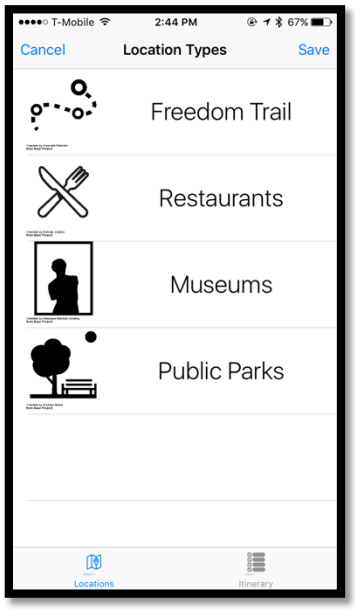


Figure 30 - Final App, Location Types

When the user taps on the "+" button on the initial screen, it brings him to this page. In order to start creating an itinerary, absolutely no information is needed, meaning the user can immediately access the locations that can be added. We made this decision after seeing how itinerary specific applications required a large amount of information to begin creating a new itinerary. Since the user is using this application to find locations to travel to, it made more sense to make the location details be quickly navigated to.

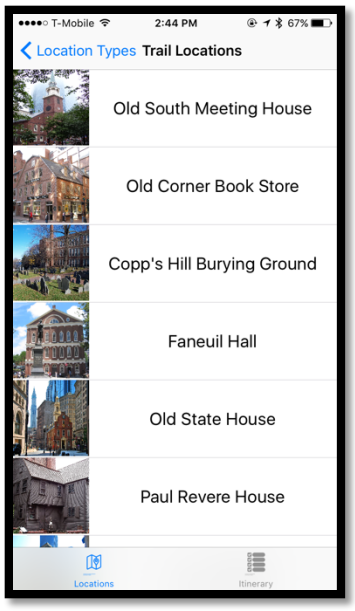


Figure 31 - Final App, Trail Locations

Instead of showing the various locations of The Freedom Trail on a map, which can be difficult to navigate, we decided to have it in a clear list. Each location's name and picture is clearly shown to make it easy to view when walking in Boston.

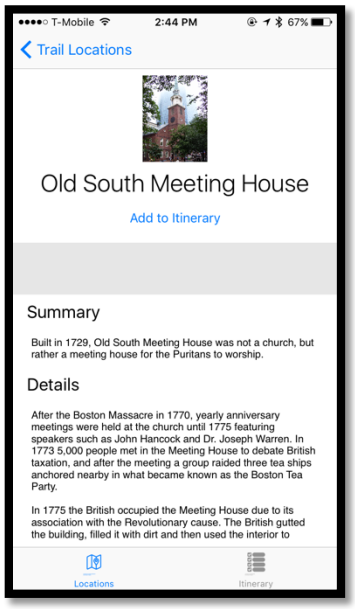


Figure 32 - Final App, Location Details

The majority of users for this app will not know what each of these stops on the Trail are, so it was important to make sure that they could view the location's information before adding it to the itinerary. Allowing them to view it meant that they would have a general idea of what to expect when walking the Trail, before they even got to the location. An "Add to Itinerary" button is included on this page.

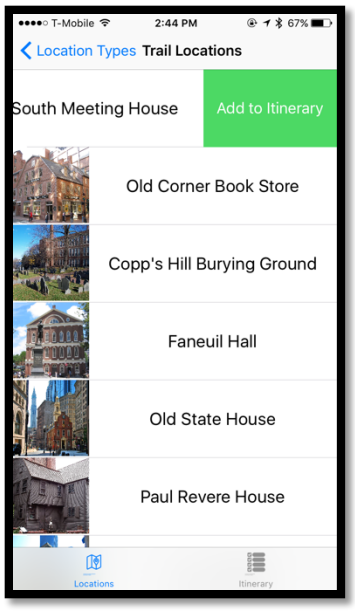


Figure 33 - Final App, Swipe to Add

Using the idea of allowing the user to have two different ways of completing a task, swiping to add an itinerary was added. This additional way to add a location to the user's itinerary removes a screen, which then allows it to be done more quickly if the user wants to.

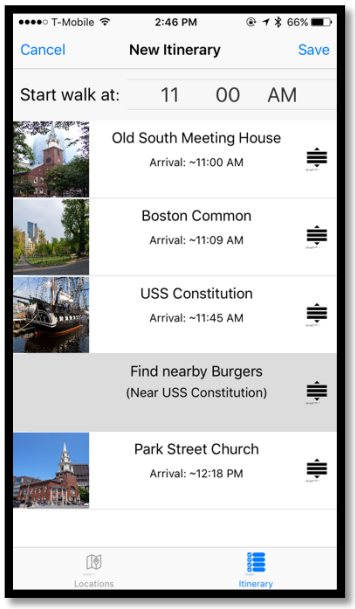


Figure 34 - Final App, Restaurant Placeholder

After many locations have been added, the itinerary page will look like this. It includes a start time, as well as every location that the user has added, with an arrival time that is calculated using the walking directions from Google. A gray location in the itinerary represents a restaurant type, and the user must tap it to select the specific restaurant.

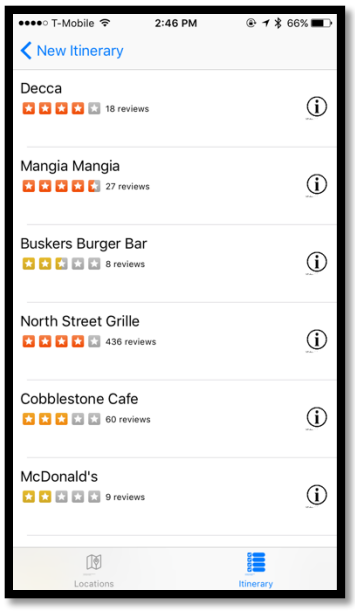


Figure 35 - Final App, Yelp Restaurants

Selecting a restaurant is done on this page, which pulls results dynamically from Yelp, using the GPS coordinates of the nearby itinerary stops on the previous page (Figure 35). This dynamic information is one of the concepts that was drawn from the *Discover Boston* application, with the difference being that these locations can still be added to the itinerary.

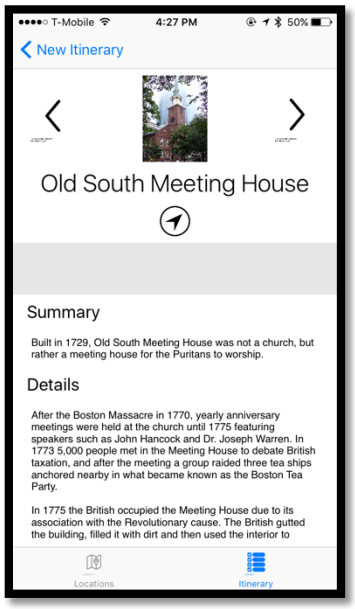


Figure 36 - Final App, Location Directions

Once a location is selected in an itinerary, a page that is similar to the info page shown before is displayed. The difference between this and the page shown before is that it now provides the user the option to get walking directions to the location, and the user can go to backwards and forwards in the application without going back to the main itinerary schedule page.

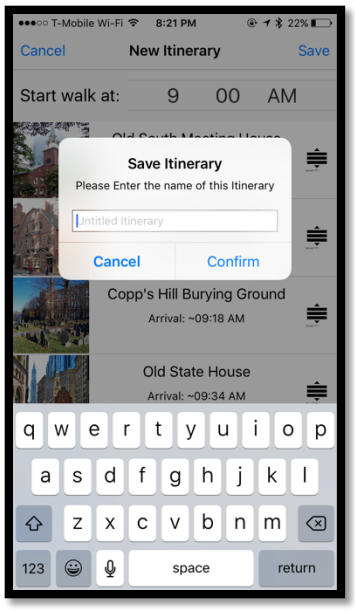


Figure 37 - Final App, Saving an Itinerary

Once a user is satisfied with their itinerary, they can save it under a custom name. Unlike the other itinerary applications, which ask for this type of information first, this was done last because saving is not the most important feature of the application. Although it is important to planners who save their plans, the name is not required to modify and create an itinerary, so it should not be intrusive for those users who might not need to save.

Architectural Design

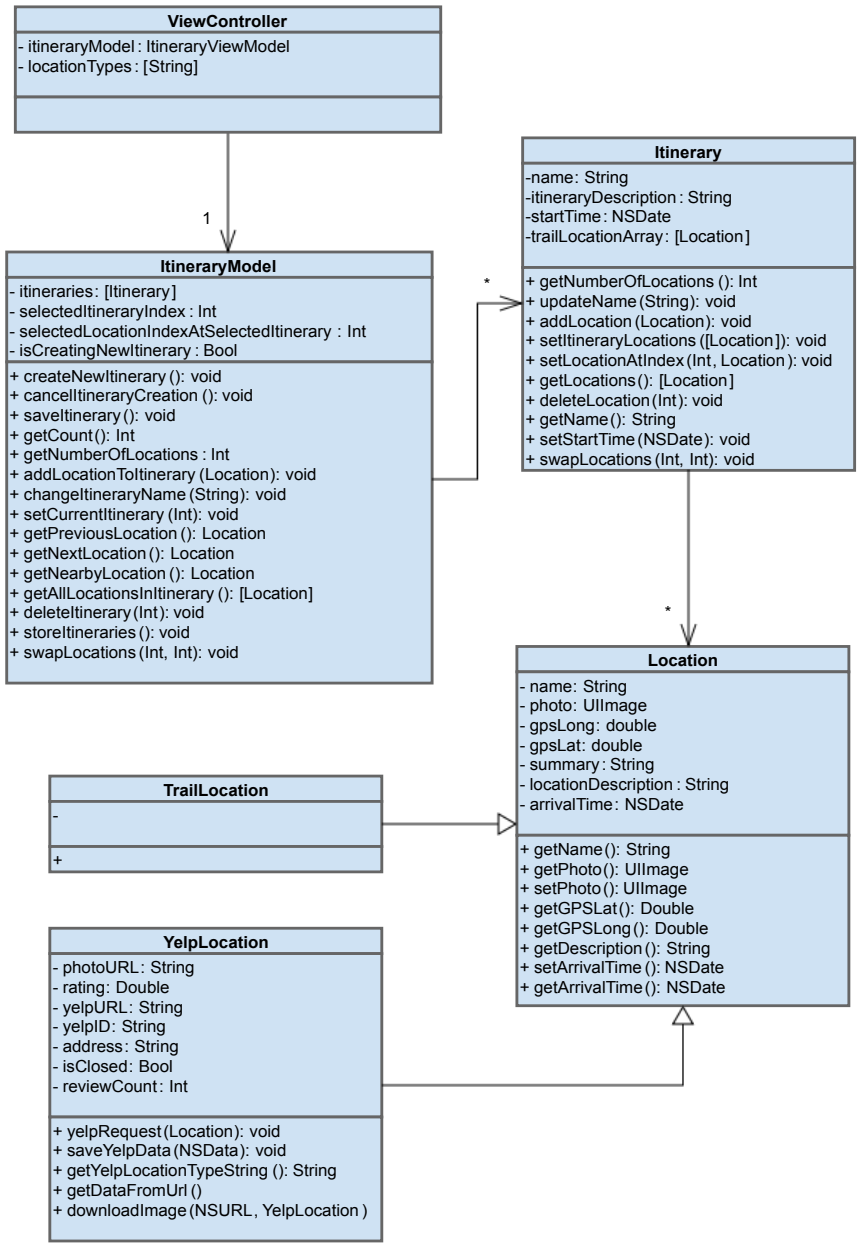


Figure 38 - Itinerary Class Diagram: View controllers is generalized to save space. Trail Location was added to allow for flexibility in future development.

Observer Pattern

There was initially some difficulty keeping each of the screens of the application up to date with the proper information. There were often times when locations were added to

the itinerary, but did not show up, which would be confusing to the user. The observer pattern was used as a way to make sure that this did not happen. Every time the itinerary was updated, it caused an event to be fired, which every single screen (controller) was watching for. When the controller saw that the itinerary update event has been fired, it ran a function which updated the information that it had to display to the view, which therefore meant that the user always saw updated and accurate information.

Yelp

Yelp results were used to populate the application with restaurant, museum, and public park information. The main reason for using Yelp was to make sure that the application did not become outdated as quickly as the other apps on the iOS store. Pulling Yelp data dynamically, instead of storing it all in the application itself, means that the information will stay current.

Twitter

Although the original design for the application included a Twitter component, after looking into the option, a decision was made to not implement it. While the closure information would be incredibly helpful for those looking to walk on the Trail, Twitter has specific guidelines for how Tweets are displayed and used, and the desired use in this app would not be allowed. An alternative that is far more flexible, is having the information pulled from a server that can be updated. This is described in the future work section.

Chapter 7 Evaluation

Usability tests

To test the usability of this application among users that had never used it before, a total of ten students were asked to create an itinerary that included at least 5 Freedom Trail locations, one restaurant, one public park, and one museum. This testing was performed on the campus of WPI (no testing was done directly on the Trail). The most significant areas that users noted were the lack of feedback when adding locations, and it being unclear that how restaurants were added and selected. The way the application is currently designed, when a location is added, it does not give the user any indication that the addition was successful. Users recommended either having a small pop up that says "Added", or even changing to the itinerary screen to show the location in the context of the rest of the itinerary.

Having the restaurant type be added, and then selected the specific location from within the itinerary was also slightly confusing to users, even with the grayed out box. One suggestion was to more explicitly say that the box needed to be tapped again to select a specific restaurant, while another recommended that all restaurants just be added by search, so the user could always add their own restaurant preference, not have it recommended to them. The final recommendation that people wanted was to be able to have a visual of the itinerary that they were going to walk. They didn't necessarily need

the map for navigation, but said that just having it be a static map would have been nice to have.

Table 1 - Summarization of Problems and Suggestions from User Testing

Problem or Suggestion	**Number of Users**
"Location Types" tab should make it more clear that the locations can be tapped to reveal more specifics	6 out of 10
Adding a location to the itinerary should show that it has been added	10 out of 10
Wanted to be able to add specific restaurants	8 out of 10
Wanted to be able to change a restaurant if they moved it between a different location (to keep it nearby)	4 out of 10
Wanted a map to show the final path that they will, or did take	7 out of 10

Comparison to other apps

One measurement of the usefulness of an application is using the keystroke level model, which predicts the time it will take for a user to navigate a system by finding the number of clicks, or in the case of iOS taps, to get to the important information (Card). The first comparison that was made between each of the applications was seeing how many taps it took to get to the detailed information about a Trail location. Table 2 shows the specific results. *FreedomWalk* has the highest number of taps, and includes having to zoom in on a map, which slows navigation down, with the lowest number being the official application from the Trail Association. The reason for this low number is, however, due to the fact that it only provides information on the Trail, and nothing else. This project's application would

only require 2 taps instead of 3 if it did not have the functionality of the “Location Types” screen, which is needed for the app to support restaurant and other location types.

Table 2 - Taps to get to Location Details Page

Application Name	**Number of Taps to get to location details**
FreedomWalk	4 (with map zoom)
Official Trail Tour Guide App	2
Discover Boston	4 (to any location’s details)
This project’s application	3

The next comparison is between this project’s app and the other itinerary specific apps. While the number of taps to reach the “Creating Itinerary” page is identical for all applications, as they all have a “+” button to start creation, the actual results to get the the important information is significantly different. My application remains at one tap, as the user is able to jump directly into adding locations, whereas other applications require more taps because of the amount of information to get to the actual adding stage. All of the itinerary applications reviewed required at least 4 pieces of information to be entered; name, destination, start date, and end date. Since the key feature of the application developed for this MQP is building itineraries, having it available in one tap is an excellent result.

Chapter 8 Future Work and Conclusion

Experiment with Maps

While *FreedomTour* does not use maps due to the complications that they cause with navigation, there is still room to experiment with exactly with ways in which maps could be incorporated. The most likely spot for this would be as a way to visualize the locations, without having any sort of navigation. For example, there could be a button that simply brings up the map, with the path the user will take between each location overlaid onto it.

Social Aspects and Custom Locations

While the application needs to remain entirely itinerary driven, experimenting with how a social aspect might fit in would make the app more personal. Users would be able share their itineraries with others, or even save them to a central server for people they don't know to download them. Custom locations could also be added and shared, for those locations that aren't necessarily a formalize business or location, but just a nice spot in the city. Itineraries could also be created automatically by using data from the most common places people using the application go.

Expanding

Adapting the design and interface of this application to become more generic might also be used to apply to other tourist locations, or even an entire city. For example, users would

be able to download a specific application for Boston, or Chicago, or Hong Kong, each of which is exactly the same, except for the content. Even within the current application, an API could be used to update The Freedom Trail location descriptions, send alerts for stops being closed, or any other dynamic changes that need to be made.

Conclusion

In conclusion, an application for planning a day on The Freedom Trail was successfully created, reaching every one of the goals and requirements originally set for the project. Although there are a number of other features and improvements that can be worked on in the future, the final application is in a state where it could be submitted and published to the public iOS app store for tourists of Boston to use, and the concepts behind the UI could easily be expanded to walking Trails in other cities.

Chapter 9 Appendices

Appendix A - Yelp Query Results Example


```
{
  "region": {
    "span": {
      "latitude_delta": 0.0,
      "longitude_delta": 0.0
    },
    "center": {
      "latitude": 42.2709097,
      "longitude": -71.8068917
    }
  },
  "total": 1,
  "businesses": [
    {
      "is_claimed": false,
      "rating": 3.0,
      "mobile_url": "http://m.yelp.com/biz/thai-time-
worcester?utm_campaign=yelp_api&utm_medium=api_v2_search&utm_source=-KjeymOM8cXvmxhHxr2iJQ",
      "rating_img_url": "https://s3-
media3.fl.yelpcdn.com/assets/2/www/img/34bc8086841c/ico/stars/v1/stars_3.png",
      "review_count": 66,
      "name": "Thai Time",
      "rating_img_url_small": "https://s3-
media3.fl.yelpcdn.com/assets/2/www/img/902abeed0983/ico/stars/v1/stars_small_3.png",
      "url": "http://www.yelp.com/biz/thai-time-
worcester?utm_campaign=yelp_api&utm_medium=api_v2_search&utm_source=-KjeymOM8cXvmxhHxr2iJQ",
      "categories": [
        [
          "Thai",
          "thai"
        ]
      ],
      "menu_date_updated": 1455422932,
      "phone": "5087567267",
      "snippet_text": "I really like the fried rice with chicken or steak. The tea is decent too. Food isn't
expensive either.",
      "image_url": "https://s3-media3.fl.yelpcdn.com/bphoto/trK6MGOCmvvOJWzEbmtZYg/ms.jpg",
      "snippet_image_url": "http://s3-media4.fl.yelpcdn.com/photo/LYSXhPmOXUGoOLCx2PvDSw/ms.jpg",
      "display_phone": "+1-508-756-7267",
      "rating_img_url_large": "https://s3-
media1.fl.yelpcdn.com/assets/2/www/img/e8b5b79d37ed/ico/stars/v1/stars_large_3.png",
      "menu_provider": "eat24",
```

```
        "id": "thai-time-worcester",
        "is_closed": false,
        "location": {
            "city": "Worcester",
            "display_address": [
                "107 Highland St",
                "Worcester, MA 01609"
            ],
            "geo_accuracy": 9.5,
            "postal_code": "01609",
            "country_code": "US",
            "address": [
                "107 Highland St"
            ],
            "coordinate": {
                "latitude": 42.2709097,
                "longitude": -71.8068917
            },
            "state_code": "MA"
        }
    }
  ]
}
```

Chapter 10 References

8 Characteristics of Successful User Interfaces. (2009, April 15).
Retrieved April 27, 2016, from UsabilityPost,
http://usabilitypost.com/2009/04/15/8-characteristics-of-successful-user-interfaces/

Adaptive and Layout. (2016, March 21).
Retrieved April 27, 2016, from iOS Developer Library,
https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/LayoutandAppearance.html

Card, S. K., Moran, T. P., & Newell, A. (1980).
The keystroke-level model for user performance time with interactive systems.
Communications of the ACM, *23*(7), 396-410.

Google Maps API. (2016, April 19). Retrieved April 27, 2016, from Google,
https://developers.google.com/maps/documentation/directions/intro#RequestParameters

HTTP GET Request in Swift. (2016, January 2).
Retrieved April 27, 2016, from Swift Developer Blog,
http://swiftdeveloperblog.com/http-get-request-example-in-swift/

IOS Human Interface Guidelines: Designing for iOS. (2016, March 21).
Retrieved April 27, 2016, from Apple Inc.,
https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html#//apple_ref/doc/uid/TP40006556

MAILBOX. Retrieved April 27, 2016, from MailBox, https://www.mailboxapp.com

Map links. (2015, June 8). Retrieved April 27, 2016, from iOS Developer Library,
https://developer.apple.com/library/ios/featuredarticles/iPhoneURLScheme_Reference/MapLinks/MapLinks.html

NSDate Class Reference. (2013, September 18). Retrieved April 27, 2016, from Apple,
https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSDate_Class/

NSNotifications in Swift. (2014, August 31). Retrieved April 27, 2016, from IACHIEVED, http://dev.iachieved.it/iachievedit/nsnotifications-with-userinfo-in-swift/

OAuthSwift. (2016, April 6). Retrieved April 27, 2016, from GitHub, https://github.com/OAuthSwift/OAuthSwift

Swift. A Powerful Open Language that lets Everyone Build Amazing Apps. (2016). Retrieved April 27, 2016, from Apple Inc, http://www.apple.com/swift/

Swiftly Getting a Human-Readable Date with NSDateFormatter. (2014, December 22). Retrieved April 27, 2016, from Coding Explorer, http://www.codingexplorer.com/swiftly-getting-human-readable-date-nsdateformatter/

UIDatePicker Class Reference. (2010, November 29). Retrieved April 27, 2016, from Apple, https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIDatePicker_Class/

Yelp Developer. (2016). Retrieved April 27, 2016, from Category List, https://www.yelp.com/developers/documentation/v2/all_category_list

Begbie, C. (2015, September 22). *Storyboards in iOS 9*. Retrieved April 27, 2016, from RAYWENDERLICH, https://www.raywenderlich.com/113388/storyboards-tutorial-in-ios-9-part-1

Eberhardt, C. (2014, July 3). *MVVM Tutorial with ReactiveCocoa: Part 1/2*. Retrieved April 27, 2016, from RAYWENDERLICH, https://www.raywenderlich.com/74106/mvvm-tutorial-with-reactivecocoa-part-1

Furrow, A. *MVVM in swift*. Retrieved April 27, 2016, from Engineering, http://artsy.github.io/blog/2015/09/24/mvvm-in-swift/

Ngo, V. (2014, December 9). *Introducing iOS design patterns in swift – part 2/2*. Retrieved April 27, 2016, from RAYWENDERLICH, https://www.raywenderlich.com/90773/introducing-ios-design-patterns-in-swift-part-2

Spool, J. (2005, January 10). *What makes a design seem "intuitive"?* Retrieved April 27, 2016, from UIE, https://articles.uie.com/design_intuitive/

Todorov, M. (2015, September 28).
IOS animation Tutorial: Custom view controller presentation transitions.
Retrieved April 27, 2016, from RAYWENDERLICH,
https://www.raywenderlich.com/113845/ios-animation-tutorial-custom-view-controller-presentation-transitions