Jordan Lu – jjl4mb
Michael Chen – mzc2fd
CS 4720 – iOS application

Places We Have Been

Mobile applications have grown in popularity due to their ability to make daily tasks even easier. Our iOS application aims to provide users with an easy way to retraces their steps, and revisit their locations based on pictures they have taken. The application opens with a tab view, allowing users to either upload a picture, or view a map. The map will drop a pin at every location the user has a taken a picture, and show them the picture. The user can then tap the pin snippet in order to view a more in depth description of the picture. In the upload tab, the user has the ability to upload a picture. First the user will tap on the camera which will allow them to take the picture using their phone camera. Next, the user adds a title a description which will eventually be paired with their current location and the date time, before saving it to the device. Ultimately, we hope the application allows more forgetful users to have the ability to retrace their steps, and revisit their most memorable locations.

Platform Justification

One of the main reasons we chose iOS devices is because it was much easier for us to both program and test on an iOS device. We both have Macbooks, as well as iPhones, making iOS the most convenient option. This allowed us to easily code and test, without needing to share a single device. Another reason we decided to choose iOS is because we both had android programming experience through CS 2110. The interface builder for android development is much more difficult to use effectively as a beginner. On the other and, the interface builder for iOS applications seems to be much more intuitive, allowing drag and dropped objects to appear on the view exactly where you dropped them. Through our experiences, we discovered that in general, the user interface for iOS applications is much more swift and sharp.

Key Features

One of the key features of our application is the memory we save by uploading images to imgur and passing and storing a url instead of the raw image. We use the imgur api to essentially upload an image, and fetch the url from the json object they return. We then store the url in nsuserdefaults, along with other necessary picture information, and load our imageView using the urls.

Another key feature of our application is the snippets on our map. The map initially loads with pins at all of the location the user has taken pictures at. The user is then able to click on these pins to load a snippet with the image that they took at that location. By tapping on the pin again, the user is able to dismiss the snippet; however, upon tapping the snippet, the user is taken to another view controller which displays more details of the snapshot. This include the title, the initial description, the date and time the picture was taken, and finally the location at which the user took the picture.

Testing Methodologies

One of the main testing methods we used was testing the flow of our application many times, with different information saved in the text fields. By doing so, we not only ensured that our application ran on multiple runs, but also that any type of input would not break our

application. We focused on repeatedly trialing key use cases, along with basic interactions with our application.

Usage

Running our application is rather straightforward. Simply build and run the application to be taken to the main screen. From there, the user is able to see and tap anything that allows interactions. One important thing that needs to be be taken note of is when a user takes a picture in the picture tab. The submit button is disabled while the application uploads the picture to imgur and retrieves the url. Because this is asynchronous, we disable the button until the function returns. Thus, the user cannot upload and submit the picture, before the application has finished obtaining the url.

Lessons Learned

Creating our app in swift took a lot of time to get used to the development environment. The XCode interface builder, despite looking much cleaner than android studio's interface builder, was difficult to work with because of the constraint. Many of our views were misplaced when we turned the iPhone to a different orientation. Another frustrating problem with iOS development is recognizing the difference between optionals and force unwrapping. As Sheriff mentioned in class, many of our syntax errors could be fixed by adding exclamation or question marks in "random" locations. However, through the course of our project, we learned many things pertaining to the structure and flow or iOS applications, along with the interaction with web services. Interfacing with a web API was actually much easier than we had assumed. We did run into problems with running things asynchronously in our app. This could include setting an imageView using a URL, or even making the asynchronous request in the first place. Ultimately, the creation of this application allowed to gain a more in depth view of creating an iOS application from scratch.

Wireframe

Basic Wireframes for our 2 main screens.