CS 4720 - S17 - Final Project Proposal

Device Name: Empire	Platform: iOS

Name: Charu Mishra	Computing ID: cm2jk

Name: Puja Soni	Computing ID: pds7yx

App Name: On This Day

Project Description:

On This Day is essentially a daily journal. For those people who aspire to record their lives easily and efficiently this application will prove to be an amazing option. Usually people can't keep up with jotting down their memories, or taking a picture everywhere and having it organized. Memories should be recorded and just pictures aren't the way. A journal that you are taking around in your phone though may just be enough to capture feelings and moments that you are living it.

Our app is able to do the following:

- The system allows the user to check-in to their current location by providing them a list of possible locations they may currently be at.
- The system shall allows a user to take a picture representative of their day and add it to their day.
- The system shall allow a user to easily add a quick sentence describing a moment of their day and each added moment will be listed in the journal for that day.
- The system shall allow a user to upload an image into a "day" from the camera roll.
- The system shall allow a user to view any previous day's memories/journal entry.

We incorporated the following features:

- Camera - A user can take a picture to represent their current day.
- GPS/ Location-awareness - The app recognizes key places that the user went to during the day, such as the movie theater or a restaurant, and will save these locations in that day's memories when the user checks in to the location.
- Data storage using Core Data - The app will save each day's associated pictures/locations/moments so that when users click on some past date in the calendar, that day's journal data will be presented to the user.
- Open shared activity/features - The app will allow users to import pictures from their camera roll and save them into a day's memories.

Wireframe Description:
Our wireframe shows the basic layout we are envisioning for our app. After the launch screen appears,the calendar appears with the first month of the year showing so users can easily scroll

through to see the dates that have already been journalled until they get to their current month which will appear last. When a user clicks on a date, a button is enabled to direct them to the journal entry for that date, they are brought to a screen which shows that day's "journal" - these are all the locations, memory notes, and the photo associated with that day. The functionality of adding a new entry to the day is shown when a user clicks the + button in the top right corner. The system presents the users with a selector of which type of item they would like to add, which is important to keep the app as simple and understandable as possible. Each type has it's own UI for adding a new entry of that type (shown by the "add new" screens in the wireframe). We would like to add a feature where users can import photos from their camera roll into a day's journal in case they forgot to open their app to take a picture during the day.

Platform Justification:

For this project, we decided to use the iOS platform because of multiple benefits. The storyboard feature on XCode is extremely user-friendly and makes it very simple to build screens. We really appreciated that after seeing the difficulties to build the same screen on Android when we built the Bucket List apps. Another benefit was the Apple has many up to date tutorials online that explain a lot of the features we were planning on using so we thought that would help make the process of building the application a little smoother. Next, one of us had many issues in trying to sync Android Studio and github which were never solved even with help from TAs. This made us decide that if we had to do the entire project on one computer together, might as well pick the platform that seemed to work more smoothly for us which was iOS.

Major Features/Screens:

The first major feature is the calendar screen on the first screen. This screen implements the JTAppleCalendar CocoaPod since XCode does not have it's own calendar view. Once a date is selected, the selected date's color change and the "Get This Day" button is enabled. Clicking on the "Get This Day" button loads the corresponding journal entry.

Another major feature is the On This Day journal screen. This screen is built completely by labels and textviewfields but here our entire application comes together. CoreData is used in order to load all available data for that day. If no entries have been made for that day, the screen will be blank with just the labels. There is an add button at the top on the NavigationBar which allows for users to navigate to a screen which will direct them to the appropriate screens to add Moments, Places, and a Picture to their day's journal.

Probably the last major feature that we implemented is our map and places screens. When a user selects Places to add, it directs them to a map of their current location. Then, when the user clicks the "Get Place" button at the top of the screen--it directs them to the places screen. This screen has a list of all the likely places they could be according to their latitude and longitude coordinates. Then they can correctly choose where they are from the table and that location will be saved into that day's core data to be loaded into the journal entry.

All the other screens are helper screens to aid the user in easily adding a new moment to their day or a new picture to their day.

Optional Features:

-We incorporated GPS/Location-awareness using Google Maps api. We used this so users can easily check into their location and add that location to their journal entry. To test/demo this feature, from the screen to pick which type of entry to add to the screen, select Places. Then, a map of the user's current location will appear with possible locations that the user could be. Then, the user needs to click on the Pick Place button to see a list of the likely places that they could be based on their location and the Google Places API. (15 points)
-We incorporated the camera into the application so that users can take a picture or load a picture from their camera roll and that will be loaded into the journal entry. This can be tested by going into the add image screen and then selecting either camera roll or access camera to get the image they want to add. (15 points)
-We handled data storage via CoreData. This was pretty difficult due to the way our application is organized but we managed to make sure that CoreData was able to retrieve information that needs to be loaded into the journal entries. We had a "Day" entity which had a one-to-many relationship with a "Moment" entity and a "Place". This can be tested by selecting any date, adding a moment or place to that date. Repeating this for a few more days and then double checking that all your progressed work is still there. (20 points)
-Our application is loading pictures from the camera roll of the device which is another feature of the device. This is apparent when you give permission to access the camera roll and then go select an image from the camera roll and it's loaded back into the screen. (10 points)
All these optional features add up to exactly 60 points in total.

Testing Methodologies:
In order to test the features of this app, we mostly just created multiple days and then selected different locations in the simulator to see that it would pick up different likely locations by where the app was detected to be now.

Usage:
There is no special information needed to run the app.

Lessons Learned:
We learned a lot about CoreData. Unfortunately we were extremely unaware of how complex coredata could be and how there isn't a good way to save pictures into CoreData except turning it into binary data which we opted to not do rather saved it in an array. We also learned a lot about relationships between entities in CoreData and how those can be used to organize data. We also learned how to successfully integrate external cocoapod into our project and then link it with the rest of the project to make it work. This was an extreme challenge with JTAppleCalendars because they didn't seem to have updated their tutorials to account for XCode and Swift updates as well as not fully explaining the limitations of their library so we ended up having to find backwards routes to implement features we wanted when their tutorial would fail us. We also learned how extremely important it is to continually push our changes into Github when working on the app in order to keep us organized. I think while the other projects throughout the semester gave us a good foundation on how to build an application this project

was definitely a challenge that made us really think about how every part of this application works and our own thinking of what would be the best ways to put it together completely.