App Specification: On the Map

iOS Developer Nanodegree

[Note that this is an informal app description. It will give you an idea how the app should work, but when you submit your app it will be rated based on the Rubric.]

The On The Map app allows users to share their location and a URL with their fellow students.To visualize this data, On The Map uses a map with pins for location and pin annotations for student names and URLs, allowing students to place themselves "on the map," so to speak.

First, the user logs in to the app using their Udacity username and password. After login, the app downloads locations and links previously posted by other students. These links can point to any URL that a student chooses. We encourage students to share something about their work or interests.

After viewing the information posted by other students, a user can post their own location and link. The locations are specified with a string and forward geocoded. They can be as specific as a full street address or as generic as "Costa Rica" or "Seattle, WA."

The app has three view controller scenes:

Login View: Allows the user to log in using their Udacity credentials, or (as an extra credit exercise) using their Facebook account

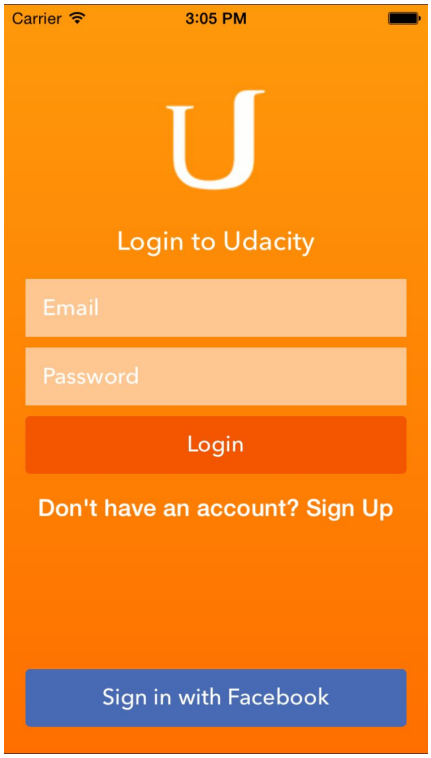
Map and Table Tabbed View: Allows users to see the locations of other students in two formats.

Information Posting View: Allows the users specify their own locations and links.

These three scenes are described in detail below.

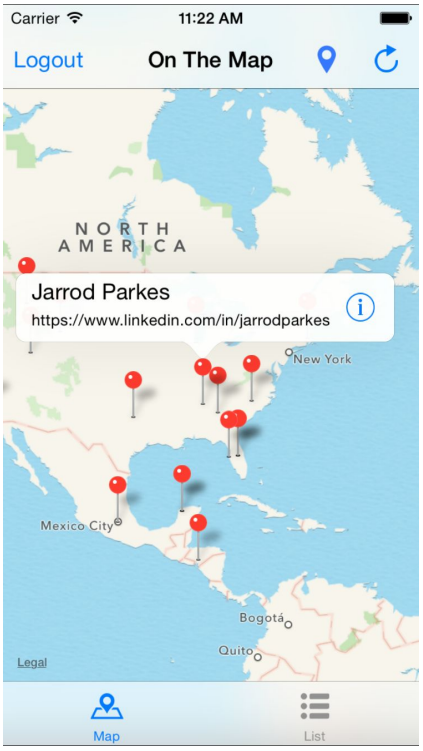
Login View

The login view accepts the email address and password that students use to login to the Udacity site. User credentials are **not** required to be saved upon successful login.
When the user taps the **Login** button, the app will attempt to authenticate with Udacity's servers.
Clicking on the **Sign Up** link will open Safari to the Udacitysign-in page.
If the connection is made and the email and password are good, the app will segue to the **Map and Table Tabbed View**.

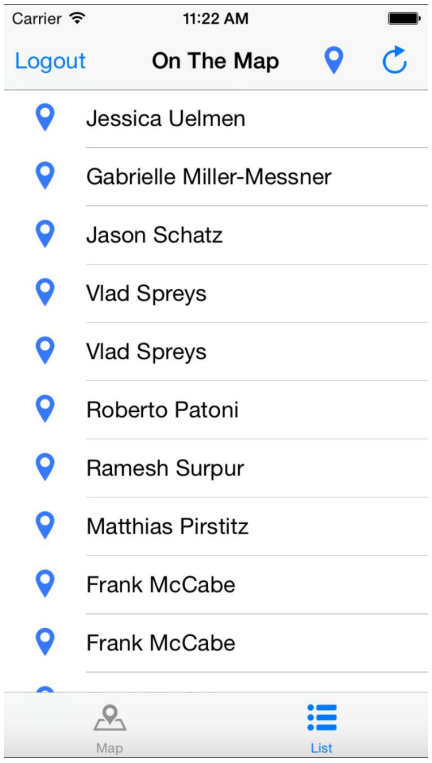


If the login does not succeed, the user will be presented with an alert view specifying whether it was a failed network connection, or an incorrect email and password.
Optional (but fun) task: The "Sign in with Facebook" button in the image authenticates with Facebook. Authentication with Facebook may occur through the device's accounts or through Facebook's website.

Map And Table Tabbed View



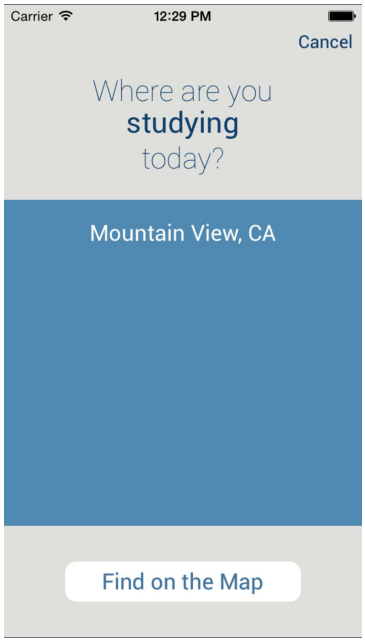
This view has two tabs at the bottom: one specifying a map, and the other a table.
When the **map tab** is selected, the view displays a map with pins specifying the last 100 locations posted by students.
The user is able to zoom and scroll the map to any location using standard pinch and drag gestures.
When the user taps a pin, it displays the pin annotation popup, with the student's name (pulled from their Udacity profile) and the link associated with the student's pin.
Tapping anywhere within the annotation will launch Safari and direct it to the link associated with the pin.



Tapping outside of the annotation will dismiss/hide it.
When the **table tab** is selected, the most recent 100 locations posted by students are displayed in a table.
Each row displays the name from the student's Udacity profile. Tapping on the row launches Safari and opens the link associated with the student.
Both the map tab and the table tab share the same top navigation bar.
The rightmost bar button will be a refresh button.
Clicking on the button will refresh the entire data set by downloading and displaying the most recent 100 posts made by students.
The bar button directly to its left will be a pin button.
Clicking on the pin button will modally present the**Information Posting View**.

Optional (but fun) task: If authentication with Facebook is enabled, consider placing a bar button in the top left corner which will allow to user to logout.

Information Posting View



The Information Posting View allows users to input data in two steps: first adding their location string, then their link.
When the **Information Posting View** is modally presented, the user sees a prompt asking where they are studying. The user enters a string into a text field or text view.
When the user clicks on the “Find on the Map” button, the app will forward geocode the string. If the forward geocode fails, the app will display an alert view notifying the user.

If the forward geocode succeeds then the prompt, text field, and button will be hidden, and a map showing the entered location will be displayed. A new text field allows users to paste or type in the link that they would like to be associated with their location. A new button will be displayed allowing the user to submit their data. If the link is empty, the app will display an alert view notifying the user.

If the submission fails to post the data to the server, then the user should see an alert with an error message describing the failure.

If at any point the user clicks on the “Cancel” button, then the Information Posting View should be dismissed, returning the app to the **Map and Table Tabbed View**.

Likewise, if the submission succeeds, then the Information Posting View should be dismissed, returning the app to the**Map and Table Tabbed View**.

The following is an overview of how to use the Udacity API for the “On the Map”.

[The Udacity API]

The Udacity API

For “On the Map”, the Udacity API will be used to authenticate users of the app.

Using the Udacity API for “On the Map”

Besides having a Udacity account, there is no setup required to use the Udacity API.
(Optional) If you want to add Facebook Authentication, you will need to ensure that your Udacity account is linked to your Facebook account and use Udacity’s Facebook App ID:

> Udacity Facebook App ID = 365362206864879
> Link your Facebook account to Udacity in your Account Settings
> You can find more information about Facebook App IDs by reading Facebook’s Getting Started Documentation

Authentication

To authenticate with Udacity, you simply need to get a session ID. This is accomplished by using Udacity’s session method.

Special Note on Udacity JSON Responses

FOR ALL RESPONSES FROM THE UDACITY API, YOU WILL NEED TO SKIP THE FIRST 5 CHARACTERS OF THE RESPONSE.
These characters are used for security purposes. In the examples, you will see that we subset the response data in order to skip over them.

POSTing (Creating) a Session

Method: https://www.udacity.com/api/session

Method Type: POST
Required Parameters:

udacity - (Dictionary) a dictionary containing a username (email) and password pair used for authentication

username - (String) the username (email) for a Udacity student
password - (String) the password for a Udacity student

Example Request:

```
let request = NSMutableURLRequest(URL: NSURL(string:
"https://www.udacity.com/api/session")!)
request.HTTPMethod = "POST"
request.addValue("application/json", forHTTPHeaderField: "Accept")
request.addValue("application/json", forHTTPHeaderField: "Content-Type")
request.HTTPBody = "{\"udacity\": {\"username\": \"account@domain.com\",
\"password\": \"********\"}}".dataUsingEncoding(NSUTF8StringEncoding)
let session = NSURLSession.sharedSession()
let task = session.dataTaskWithRequest(request) { data, response, error in
      if error != nil { // Handle error…
            return
      }
      let newData = data.subdataWithRange(NSMakeRange(5, data.length - 5)) /*
subset response data! */
      println(NSString(data: newData, encoding: NSUTF8StringEncoding))
}
task.resume()
```

Example JSON Response: post-session.json

DELETEing (Logging Out Of) a Session

Method: https://www.udacity.com/api/session
Method Type: DELETE
Example Request:

```
let request = NSMutableURLRequest(URL: NSURL(string:
"https://www.udacity.com/api/session")!)
request.HTTPMethod = "DELETE"
var xsrfCookie: NSHTTPCookie? = nil
let sharedCookieStorage = NSHTTPCookieStorage.sharedHTTPCookieStorage()
for cookie in sharedCookieStorage.cookies! {
    if cookie.name == "XSRF-TOKEN" { xsrfCookie = cookie }
}
if let xsrfCookie = xsrfCookie {
    request.setValue(xsrfCookie.value!, forHTTPHeaderField: "X-XSRF-TOKEN")
}
let session = NSURLSession.sharedSession()
let task = session.dataTaskWithRequest(request) { data, response, error in
    if error != nil { // Handle error…
```

```
        return
    }
    let newData = data.subdataWithRange(NSMakeRange(5, data.length - 5)) /*
subset response data! */
    println(NSString(data: newData, encoding: NSUTF8StringEncoding))
}
task.resume()
```

Example JSON Response: delete-session.json

GETting Public User Data

Method Name: https://www.udacity.com/api/users/<user_id>
Method Type: GET
Example Request:

```
let request = NSMutableURLRequest(URL: NSURL(string:
"https://www.udacity.com/api/users/3903878747")!)
let session = NSURLSession.sharedSession()
let task = session.dataTaskWithRequest(request) { data, response, error in
      if error != nil { // Handle error...
            return
      }
      let newData = data.subdataWithRange(NSMakeRange(5, data.length - 5)) /*
subset response data! */
      println(NSString(data: newData, encoding: NSUTF8StringEncoding))
}
task.resume()
```

Example JSON Response: get-user-data.json

Exceeding Expectations

The following sections are not required to complete “On the Map”. However, if you would like to exceed expectations, then you will find these useful.

Special Note on Exceeding Expectations

WHEN USING UDACITY’S SESSION (POST) METHOD YOU MUST ONLY SPECIFY ONE CREDENTIAL AT A TIME.

So, when using Facebook Authentication, you do not need to supply values for the udacity/username/password files. If you do, then you will receive an error that says “Did not specify exactly one credential”. Likewise, if you are using udacity/username/password, then do not supply values for facebook_mobile/access_token.

POSTing (Creating) a Session with Facebook Authentication

Method: https://www.udacity.com/api/session
Method Type: POST
Udacity Facebook App ID = 365362206864879
Udacity Facebook URL Scheme Suffix = onthemap
Required Parameters:

facebook_mobile - (Dictionary) a dictionary containing an access token from a valid Facebook session

access_token - (String) the user access token from Facebook

an access token is made available through the FBSDKAccessToken class

Note: the Facebook SDK was recently updated to version 4.0. According to the upgrade guide:

FBSession.activeSession has been replaced with [FBSDKAccessToken currentAccessToken] and FBSDKLoginManager. There is no concept of session state. Instead, use the manager to login and this sets the currentAccessToken reference.

Example Request:

```
let request = NSMutableURLRequest(URL: NSURL(string:
"https://www.udacity.com/api/session")!)
request.HTTPMethod = "POST"
request.addValue("application/json", forHTTPHeaderField: "Accept")
request.addValue("application/json", forHTTPHeaderField: "Content-Type")
request.HTTPBody = "{\"facebook_mobile\": {\"access_token\":
\"DADFMS4SN9e8BAD6vMs6yWuEcrJlMZChFB0ZB0PCLZBY8FPFYxIPy1WOr402QurYWm7hj1ZCoeoXh
Ak2tekZBIddkYLAtwQ7PuTPGSERwH1DfZC5XSef3TQy1pyuAPBp5JJ364uFuGw6EDaxPZBIZBLg192U
8vL7mZAzYUSJsZA8NxcqQgZCKdK4ZBA2l2ZA6Y1ZBWHifSM0slybL9xJm3ZBbTXSBZCMItjnZBH25ir
LhIvbxj01QmlKKP3iOnl8Ey;\"}}".dataUsingEncoding(NSUTF8StringEncoding)
let session = NSURLSession.sharedSession()
let task = session.dataTaskWithRequest(request) { data, response, error in
      if error != nil {
            // Handle error...
            return
      }
      let newData = data.subdataWithRange(NSMakeRange(5, data.length - 5)) /*
      subset response data! */
      println(NSString(data: newData, encoding: NSUTF8StringEncoding))
}
task.resume()
```

Example JSON Response: post-session-facebook.json

Logging Out of Facebook Session

This can be accomplished using Facebook's FBSDKLoginButton or programmatically. With the FBSDKLoginManager, the logout behavior is built-in; however, if you want to log out programmatically, you'll need to use the FBSDKLoginManager. Facebook's full iOS documentation can be found here.