Clique: An Exercise in iOS Development
By: Samuel Lowenkamp

Overview

When I set out to work on this project, I had zero experience with mobile development in any context whatsoever. To learn iOS development, I took the course “The Complete iOS Programming Course for Swift” on Udemy, an online programming learning tool. The course covered many important subjects for iOS development including:

1. How to build iPhone screens, either programmatically or by using Apple’s storyboards with auto layout for adaptive UI/UX design on any screen size.
2. How to use and organize a Parse database, Facebook’s back end as a service tool, to serve up data from the cloud to my application.
3. How data is stored and displayed on the iPhone, from Core Data to the various View Controllers including, most notably:
   a. Table views (including with custom cells): a way of displaying data in a list/table format, usually based on displaying information from an array, usually provided by the user or from a database.
   b. Map and Image views, used to display location/image data collected from the iOS devices sensors.
4. How to download data synchronously or asynchronously from a database and what the benefits/drawbacks are of each.
5. Various methods for passing data between/throughout the stack of UI Views.

After taking the course, I set out to build my first iOS application from scratch. I ended up creating an application, Clique, which allows users to organize themselves in small groups, called Cliques, on which they can exchange messages, share pictures, and create a shared list. Any user can create or join any Clique. Because the application relies on an Internet connection to download information from Parse, the user must be connected to either wifi or a cellular network in order for the application to work.

Related to connectivity, there is one bug – although it is more of a missing feature – that would cause trouble in a production quality app: I did not learn how to integrate push

---

1 Note, during the course of my project, I switched from Objective-C, my originally proposed language, to Swift. I made the switch because, as I used online resources to learn Objective-C, I repeatedly read that Apple was pushing for a transition to Swift and that most development would likely transfer to Swift in the coming years.
notifications, thus the app must download all of the new objects each time it opens a new page. For the purposes of learning how a database works together with an application to provide a user experience, this isn’t a problem. On a big scale, it would be.

In building Clique, I relied heavily on the Parse API to learn how to use Parse and it’s various functions. Because I had no prior experience in databases or in using APIs, this was a particularly tough challenge to overcome. After several attempts at organizing my Parse back end, I eventually managed to find a set up which allowed me to make the fewest calls to the backend possible while still providing all of the information necessary.

With that transition, I will launch into a discussion of the components I built:

### Components

**Parse Backend**

In the Parse Backend I use three classes to store my data:

**User:**
A default class for any Parse app, it has several methods built in, such as log in, log out, and register. To augment the default user class, I added an array to the class called “Cliques.” The Cliques array holds the clique IDs for all of the cliques to which the user is a member. This array is used to display the list of cliques on the Full Clique List page.

**Clique:**
The Clique class contains the default class objects, including objectId, along with a Name, stored in a string, an array of associated users, and an array of associated posts.

**Post:**
A post has the default class objects along with:
- Associated text: stored in a string, used to hold a message, list item, or caption for images.
- Associated clique ID: stored in a string
- Image: stored as a file
- Type: stored in a string, indicates which type of post it is
- Poster: stored in a string

**Sign Up**
The Sign Up Page allows a user to sign up for a new account on Clique. The user must provide a first name, last name, phone number, and password in the text fields provided. When the user presses the “Sign Up” button, it initiates the sign up sequence, which registers the new user in the Parse database and sets that user as the “current user” on the iOS device. If the user is successfully registered, the app segues to the “Clique List” page.

The new user procedure requires that the user provide a string of length greater than zero for the first name, last name, and password fields. It also requires that the phone number length be no greater than 10 digits, the length of a U.S. phone number. The password text field only accepts numbers as digits.

If there is an error during the process of registering the new user, Parse will return an error. The user will be notified that the registration failed and asked to try again at another time.

Known Issues:

1. The user could enter non-digit * or – characters in the phone number text field.
2. The user is also not authenticated as the owner of the phone number entered (ie. by text message authentication).
3. Immediately after signing up, the user cannot create their own Clique. They are only able to do so after either a) signing out and logging back in, b) freezing, crashing, then logging back in or c) joining other cliques first.
The Log In Page allows a user who already has an account to sign in to their account. It does this by comparing the phone number (as username) and password entered in the text fields to the list of users in the Parse database.

If the user signs in successfully, they are set as the default user on the iOS device and the page transitions to the full clique list.

**Full Clique List**

The Full Clique List Page displays a list of the user’s Cliques. As input, the page expects a user to be logged in on the iOS device. It then accesses that user’s list of cliques from the “cliques” array in their User object in the Parse database. The cliques are accessed by their unique “objectId” property.

The user can navigate away from the Full Clique List page in four ways:

Log Out: logs out the current user from the iOS device and moves the user to Log In.
Join a Clique, Create a Clique: transitions to respective page.

Lastly, the user can transition by pressing on the cell of one of their cliques. This will transition the user to that specific clique’s page. In the transition, the view passes along the clique’s ID, Name, array of post IDs, and array of posts sorted by time created.

One of the challenges with creating this component is due to the fact that retrieving the list of clique’s happens asynchronously (so as not to slow down the app) – but the page needs data to display in the tableview while it retrieves data / in case the data is not retrieved or comes back nil. To solve this issue, I put default text into a single cell as a placeholder. If data is not retrieved or there is a new user, that block (it says “Make a Clique!”) may not be removed. Since it does not lead to an actual clique, it causes the app to crash when the user presses it.
Create New Clique

The Create New Clique page allows the user to create a new clique by typing the desired name into the text field and pressing “create.” After pressing the button, the view calls Parse to create a new Clique with the current user as the sole associated user. If the creation succeeds, the new clique’s objectId is added to the user’s list of associated cliques.

If the call fails or the user did not enter text into the box, an error is displayed asking the user to try again.

Join Clique

The Join Clique page displays a table view list of all cliques. The user can select the cliques that they would like to join by tapping on a given cell. That cell will then display a “checkmark” detail. After selecting cliques, the user must press the “Join” button to join a clique.

When the button is pressed, the selected cliques are added to the user’s clique list by objectId and the user is added to the clique’s user list by objectId. If the user is already associated with the clique, they are not double-added.

Again, since the user input is controlled, there is little error checking necessary. The only errors that should arise would be due to Parse querying errors, which are returned by Parse and displayed to the user in a pop up.
**Specific Clique**

The Specific Clique Page displays the timeline feed of a specific clique. It takes as input from the Full Clique List: the clique name, clique ID, and sorted list of posts. As the view loads, it also re-downloads the post list in case new posts have been added since the posts were last downloaded or when the page re-loads from a view further down the stack.

This view displays the posts in a table view using two different kinds of cells – one for messages and one for images. Message text is displayed on the left when for the current user’s messages and on the right for other senders. Images always display in the center. Every time the table view is reloaded, it also scrolls to the bottom of the table to place the user at the most recent post in the timeline.

In this view, the user can send a new message by inputting text in the text field at the bottom of the view and then pressing send. The view verifies that the message length is greater than zero.

From this view, the user can transition to two new views:

- **Add Image**: in transitioning to the Add Image Page, this view sends along the current clique ID.

- **Clique Settings**: in transitioning to the Clique Settings Page, this view sends along the current clique ID, current clique name, and posts array.

There were two big challenges in creating this view:

**Creating custom cells in an embedded table view:**

In my research, I had learned how to build a basic custom cell, but only in a UITableViewController – the view that defaults to a full table view. For the Specific Clique Page, I needed an embedded table view so I could include other UI elements, such as the messaging bar on the bottom. After a few hours of research, I was able to figure out how to set my standard UIViewController as a table view delegate so as to control the embedded table view with the custom cells.

**Shifting the view as the keyboard rises:**

One big issue with the keyboard on iOS is that it takes up half of the screen. In order to compensate for losing half of your screen space, it is necessary, in many cases, to shift the rest of your views out of the way of the keyboard. In the Udemy course, we covered a small amount of
animation, but not nearly enough to cover this task. Again, after a few hours of Internet research, I was able to hack together a solution to the problem.

**Add Image**

The Add Image Page takes the “current clique ID” as input.

On this page, the user can upload an image from their camera roll. It is then displayed on the page before the user presses “Post Image.” Once the user posts the image, the page creates a new and associates its objectId with the clique’s list of posts. Again, if there is an error, Parse will return that error and the user will be asked to try again.

**Clique Options**

As input, the Clique Options Page receives the current clique name, ID, and post list.

From the Clique Options page, the user can choose to view the clique’s photos or check list. Based on which cell is chosen, the user is taken to the respective page.

In transitioning to the new page, Clique Options sends along the list of posts relevant to the destination view along with the clique name and ID.

**All Photos**

This page takes the clique ID, name, and post array of photos as input.
It then displays the photos in a timeline sorted with newest at bottom. It also displays which user posted the image by their username (phone number)

**Check List**

Displays the “list” items in a table view. Takes clique ID, name, and post array of list items as inputs.

Allows user to add new item by navigating to “add” page, passes along current clique ID as output.

**Add List Item**

The Add Item Page takes the current clique ID as input. When the user inputs text to the text field and presses the “Add Item” button, the view creates a new post on Parse and adds the post’s objectID to the clique’s array of post IDs.

If there is an error, the user is asked to try again later. If the user does not input text, they are prompted to input text before adding an item.
Add Item