iOS DeCal : Lecture 6

Networking, CocoaPods, Alamofire, and SwiftyJSON

March 14, 2017

Announcements - 3/14

Lab 4 and Proj 2 Pt 1 due tonight (11:59pm)

Office hours after lecture in 341A Soda (8-10pm)

Custom App Proposal due 3/21 (next Tuesday)

See the spec posted on the website for guidelines

We now have a room for Tuesday office hours!

341A Soda (in the undergraduate lounge) 8-10pm

Lab 5 : Snapchat Camera

For this week's lab, you'll need a device to test on.

If you have a iPhone or iPad, **please bring it along with a lightning cable that connects to your computer.**

No worries if you don't have one - you can work with a partner who does (the lab should be short enough to finish during the lab period)

Custom App : Final Project

May work individually or with a groups

Up to 4 people total per group

Students with the top submissions will be given the opportunity to present at the Final Presentation **(Friday, May 5 at 10am)**

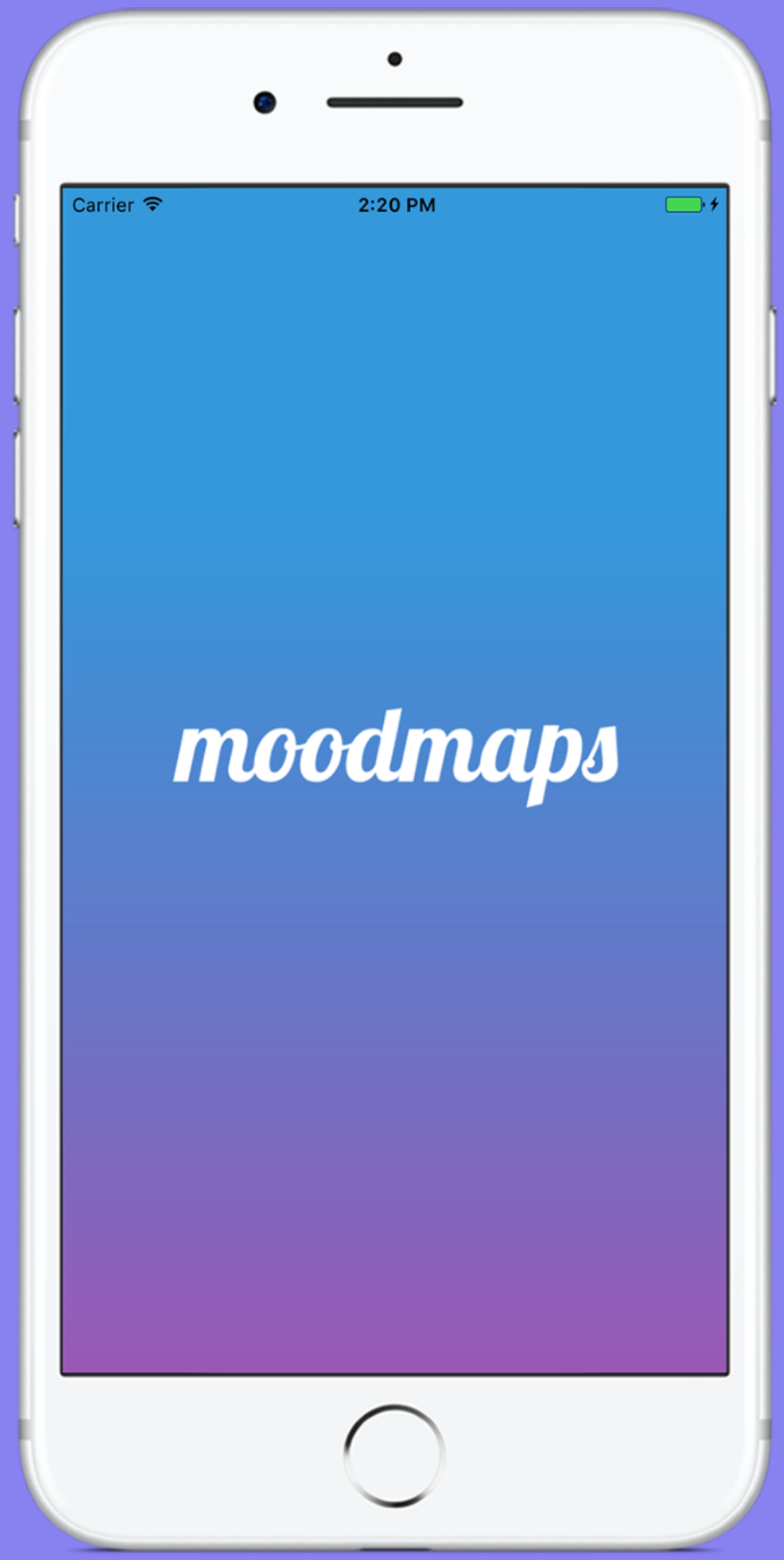
Attendance is mandatory for a project grade

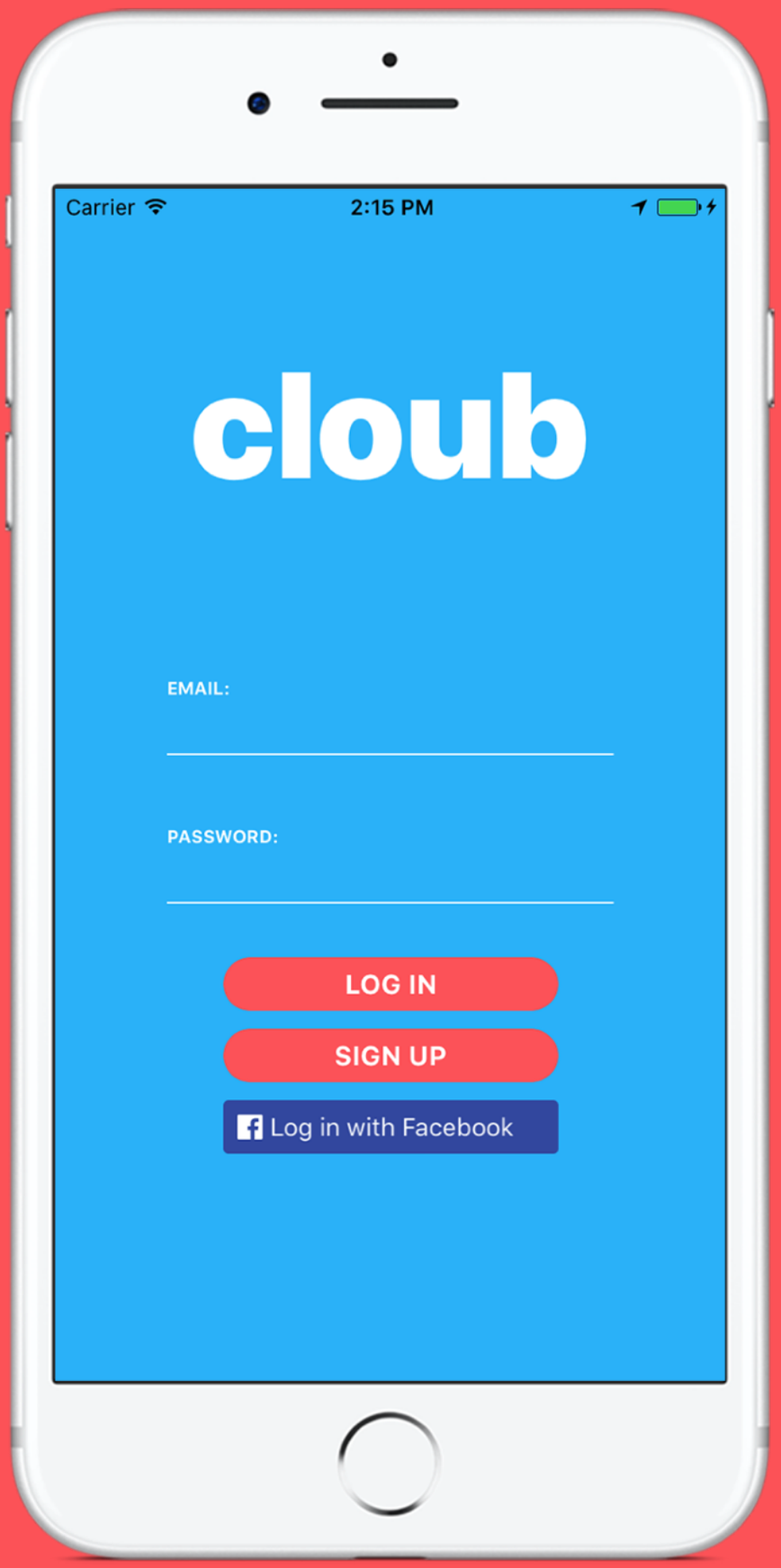
Dan Garcia & Recruiters will be attending

Please see the spec for more information!

moodmaps

Nithi Narayanan





cloub

Chan Hee Park

Overview : Today's Lecture

Networking

CocoaPods

Alamofire

SwiftyJSON

Networking

Networking & iOS

Networking is acquiring/passing data to/from some URL that exists on the world wide web or local.

General structure as it relates to iOS

- Recipient Address
- Parameters
- Response

Making an "Application Programming Interface" Call

Parameters

Response

{ api }

The HTTP Request

- Hypertext Transfer Protocol
- "GET" Request: Getting data at a URL
- "POST" Request: Sending data to a URL

Alerts

Making an Alert Controller

```
let alertController = UIAlertController(
    title: "Location",
    message: "Enabling Wi-Fi in Settings
will improve your location accuracy.",
    preferredStyle: .alert)
```

Making an Alert Controller

```
let openAction = UIAlertAction(title: "Open
Settings", style: .default) { (action) in
    if let url =
URL(string:UIApplicationOpenSettingsURLString)
{
        UIApplication.shared.open(url,
options: [:], completionHandler: nil)
    }
}

alertController.addAction(openAction)
```

Closures

Can capture and store references to any constants and variables from the context in which they are defined

- **Global closure functions**
  - Named, do not capture values
- **Nested closure functions**
  - Named, capture values from enclosing function

Global Closures : Example

```
let intPow = {(val1: Int, val2: Int) ->
Int in
    return Int(pow(Double(val1),
Double(val2)))
}
```

```
let result = intPow(2, 10)
print (result)
```

Nested Closures : Example

```
func makeIncrementer(forIncrement amount:
Int) -> () -> Int {
    var runningTotal = 0
    func incrementer() -> Int {
        runningTotal += amount
        return runningTotal
    }
    return incrementer
}
var incrementer =
makeIncrementer(forIncrement: 5)
```

Closures : Format

```
{ (parameters) -> return type in
    statements
}
```

Making an Alert Controller

```
let openAction = UIAlertAction(title: "Open
Settings", style: .default) { (action) in
    if let url =
URL(string:UIApplicationOpenSettingsURLString)
{
        UIApplication.shared.open(url,
options: [:], completionHandler: nil)
    }
}

alertController.addAction(openAction)
```

Making an Alert Controller

```
let cancelAction = UIAlertAction(title:
"Cancel", style: .cancel, handler: nil)

alertController.addAction(cancelAction)

self.present(alertController, animated:
true, completion: nil)
```

URL Session

URLSession

Apple's API for downloading content

Support various URL schemes

HTTP, HTTPS, FTP, Data, File

Pass in a URL

URL object, allocated from String

Some Relevant Classes

URL

Object that contains URL

URLRequest

Contains URL, request method, etc.

URLResponse

Contains info for server's response

URLSession Workflow

1) Create URL from a String

2) Create URLSession

3) **Create a URLSessionDataTask**

   Get data from the task and save it

URLSession

URLSession.shared()

Basic session, un-customizable

We'll stick to this for the rest of the class

URLSessionDataTask

dataTaskWithURL - Default HTTP GET

dataTaskWithRequest - Can specify HTTP

URLSession

```
    func loadImage() {
        let url = URL(string:”https://instagram.com/
img.jpg")

        let session = URLSession.shared

        let task = session.dataTask(with: url!,
completionHandler: {
            (data, response, error) -> Void in
            if error == nil {
                let img = UIImage.init(data: data!)
                self.imageView.image = img
            }
        })
        task.resume()
    }
```

CocoaPods

What are CocoaPods

A dependency manager for iOS Projects

Cocoapods are essentially Swift classes that other people write for you that you can use in your project:

- Make life more **efficient**
- Make life **easier**

Timepiece

Adding A Year To the Current Date:

Without Timepiece:

```
let calendar = NSCalendar.currentCalendar()
let newDate = calendar.dateByAddingUnit(.Year, value:
1, toDate: NSDate(), options:
NSCalendarOptions.MatchNextTime)
```

With Timepiece

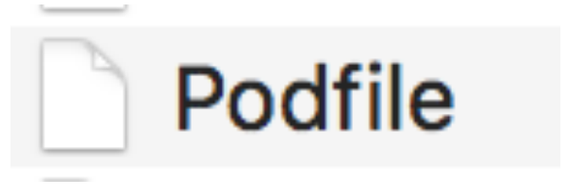
```
let newDate = now + 1.year
```

How to use Cocoapods

Install Cocoapods

```
sudo gem install cocoapods
```

Make a Podfile



1. Just named Podfile with no extension
2. Format (Trick: Use “pod init”):

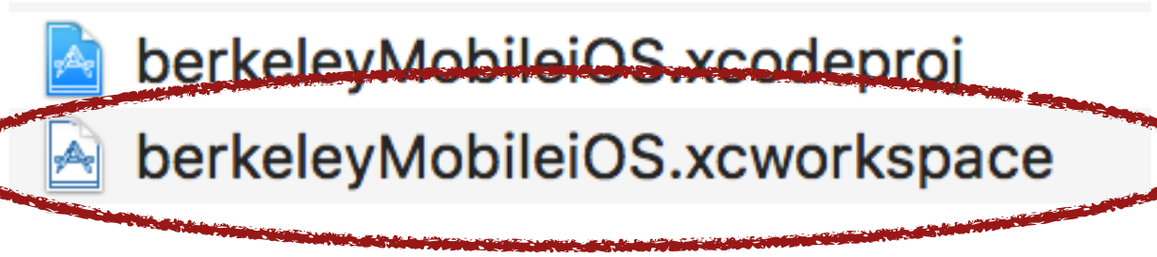
```
platform :ios, '8.0'
use_frameworks!

target 'MyApp' do
  pod ‘Timepiece’ ‘~> 1.0.2’
end
```

Update Dependencies

pod install

Open myApp.xcworkspace



Check Ins

Alamofire

Alamofire : Networking in Swift

HTTP networking library written in Swift

Simplifies common networking tasks

- Request/Response methods
- JSON serialization
- Authentication

GitHub link

Alamofire : Requests

.request: HTTP requests
.upload: Upload large files
.download: Download large files or resume a download already in progress.

Alamofire : Request types

```
Alamofire.request("https://httpbin.org/
get") // default is get

Alamofire.request("https://httpbin.org/
post", method: .post)
```

Alamofire : Response Handlers

```
// Response Data Handler - Serialized
into Data
func responseData(queue: DispatchQueue?,
completionHandler: @escaping
(DataResponse<Data>) -> Void) -> Self


// Response JSON Handler - Serialized
into Any
func responseJSON(queue: DispatchQueue?,
completionHandler: @escaping
(DataResponse<Any>) -> Void) -> Self
```

Alamofire : Response Validation

```
Alamofire.request("https://httpbin.org/get”)
         .validate().responseJSON { response in
            switch response.result {
               case .success:
                  print("Validation Successful")
               case .failure(let error):
                  print(error)
            }
      }
```

Alamofire : Parameters

```
Alamofire.request("https://httpbin.org/post",
                  method: .post,
                  parameters: parameters)

Alamofire.request("https://httpbin.org/post",
                  method: .post,
                  parameters: parameters,
                  encoding: URLEncoding.default)
```

Alamofire : Authentication

```
let user = "user"
let password = "password"

Alamofire.request(“https://
        httpbin.org/basic-auth/\(user)/
        \(password)")
       .authenticate(user: user,
                     password: password)
       .responseJSON { response in
       debugPrint(response)
}
```

SwiftyJSON

SwiftyJSON : JSON Parsing

Easy to use JSON parsing library

Makes it easier to handle JSON in your project

Avoids strict Swift type checking to make JSON parsing less verbose

GitHub installation link

JSON parsing (without SwiftyJSON)

```
if let statuses = try JSONSerialization.
                        jsonObject(with: data,
                        options: .allowFragments)
                        as? [[String: Any]],
    let user = statuses[0]["user"]
                            as? [String: Any],
    let username = user["name"] as? String {
        // Finally we got the username
    }
}
```

Example Above –> Retrieving a username name from a Tweet using Twitter's API (example link)

JSON parsing (using SwiftyJSON)

```
let json = JSON(data: dataFromNetworking)
   if let userName = json[0][“user"]
                                ["name"].string {
   // Now we have the username!
}
```

Same example as the previous slide, except here we are using SwiftyJSON (example link)

Using SwiftyJSON

```
//Getting a double from a JSON Array
let name = json[0].double


//Getting a string from a JSON Dictionary
let name = json["name"].stringValue

//Getting an array of string from a JSON
Array
let arrayNames =
json["users"].arrayValue.map({$0["name"].
stringValue})
```

Demo

Project 2 Part 1 and Lab 4

Due **Tonight** at 11:59pm

Custom App Proposal

Due **next Tuesday** at 11:59pm

Remember to bring your iPhone / iPad + cable to Thursday's Lab!