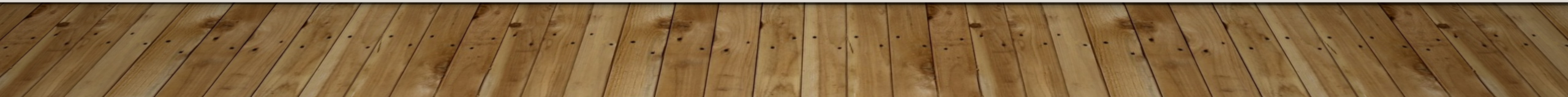
EAGLE SOCIAL

GROUP PROJECT: TO THE TOP APPS



OVERVIEW

- What is Eagle Social?
  - Eagle Social started out with the concept of building a social networking platform for the University of Southern Mississippi students.
  - A place to:
    - Network with faculty and staff.
    - Chat with fellow students.
  - Technologies
    - Swift is the programming language of choice
    - xCode is the integrated development environment

COLLABORATION TOOLS

- GitHub for code repository and code sharing
  - https://github.com/jody-bailey/EagleSocial/
- Office365 and OneDrive for document sharing.
  - If your word documents, excel spread sheets, and Power Point files are stored on one drive they update **almost** instantly. This was important as it allowed us all to work on the same document(s) at the same time without having to work on 4 separate documents and merge them together. This also enabled us to work remotely when meeting in person was not possible.
- Group iMessage and, later in the semester, Slack for group communications.

DOCUMENTATION OVERVIEW

- Issue: We tried to start coding too early.
- Requirements document
- Use cases
  - We made it too specific. It was supposed to be a more general use case.
- Class modeling document.

DEVELOPER ESSENTIAL TOOLS

- Google
- Stack overflow
- Swift documentation
- Firebase documentation
- Udemy courses
- Youtube
- Github repositories
- Affinity Designer
- Icon Set Creator
- Coffee, monster, red bull, and pizza.

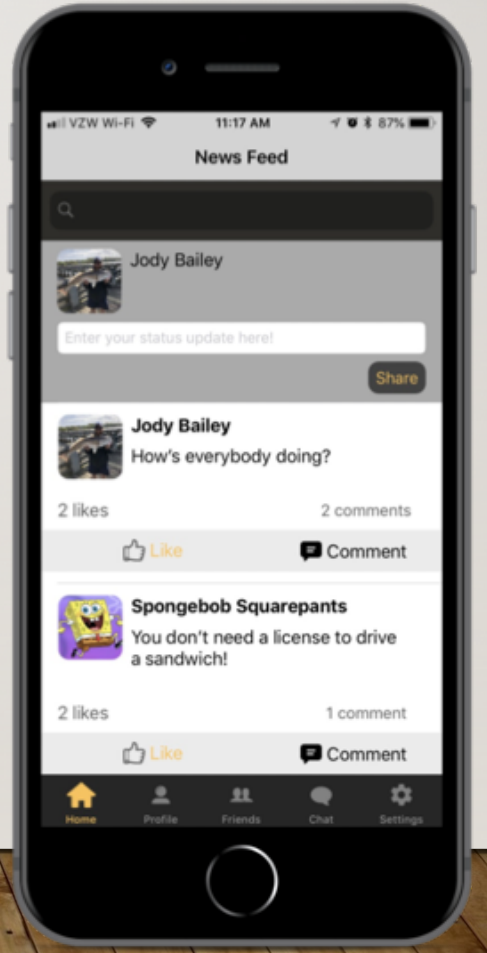
NEWS FEED - JODY

- Posts – Users can post status updates as well as see other peoples posts all in one place.
- Likes – People can like other posts and the posts will display the total number of likes that post has.
- Comments – Users can comment on other peoples posts and can also view all of the comments on that post.

```
// Creating a reference to the Firebase database and then connecting
// a handle to it to handle the data. The handler will listen for changes
// in the "posts" section of the database. When a post is change or a new
// one is added then this code will run which gets the new data from
// Firebase and puts it into the array of posts.
ref = Database.database().reference()
refHandle = ref?.child("posts").observe(.value, with: { (snapshot) in
    // code to handle when a new post is added
    guard let postsSnapshot = PostsSnapshot(with: snapshot) else { return }
    self.posts = postsSnapshot.posts
    self.posts.sort(by: { $0.date.compare($1.date) == .orderedDescending })
    self.NewsFeedTable.reloadData()

})
```

NEWS FEED - JODY

```
// Function to update the likes for a post when the like button is pressed
@objc func likeButtonPressed(sender:AnyObject) {

    // This is how to get the indexPath when you don't have direct access to it
    let buttonPosition = sender.convert(CGPoint.zero, to: self.NewsFeedTable)
    let indexPath = self.NewsFeedTable.indexPathForRow(at: buttonPosition)

    // Only attempt to get the data if the indexPath is valid
    if indexPath != nil {

        // Updating the values stored for the likes on the post that was liked
        if self.posts[(indexPath?.row)! - 1].likes[thisUser.userID] == true {
            self.posts[(indexPath?.row)! - 1].likes.updateValue(false, forKey: (thisUser.userID))
        } else {
            self.posts[(indexPath?.row)! - 1].likes.updateValue(true, forKey: (thisUser.userID))
        }

        // If the likes on the selected post does not contain the current user then add
        // their data to the database.
        if !self.posts[(indexPath?.row)! - 1].likes.isEmpty {
            self.ref?.child("posts").child(self.posts[(indexPath?.row)! - 1].postId)
                .child("likes").setValue(self.posts[(indexPath?.row)! - 1].likes)
            self.NewsFeedTable.reloadData()
        }
    }
}
```

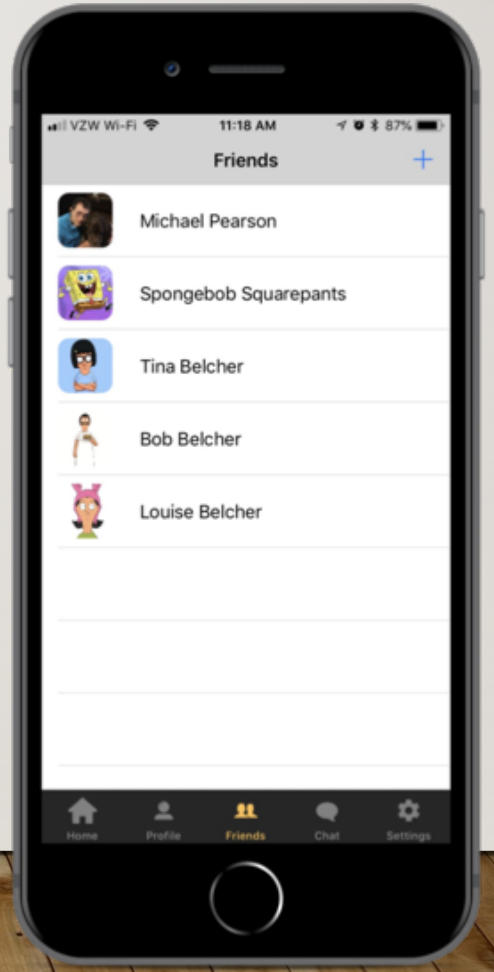


FRIENDS LIST - JODY

- Send Friend Requests – Users can send friend request by touching the plus sign at the top right of the screen and entering their friends email address.
- Select Friend – When a user selects a friend on their friends list then the app will navigate to that users profile.

FRIENDS LIST – JODY

```
// Observing the database for changes in the "Friends" section and when there is a change
// then the users friends list will be updated.
let _ = ref.child("Friends").child(thisUser.userID).observe(.value, with: { (snapshot) in
    guard let snapDict = snapshot.value as? [String : [String : Any]] else { return }
    self.friends = [Person]()
    var alreadyFriends : Bool = false
    for snap in snapDict {
        print(snap)
        for snip in snap.value {
            // Getting the current friend
            let friend = allUsers.getUser(userId: snip.value as! String)

            // This for each loop is making sure there will not be any duplicate friends
            for dude in self.friends {
                if dude.userId == friend.userId {
                    alreadyFriends = true
                }
            }

            // If the not already friends with that user then add them to the friends list
            if !alreadyFriends {
                self.friends.append(friend)
            }

        }
    }
    self.friendTableView.reloadData()
})
```



PROFILES - LACY

- The profile is intended to be a customizable space for the user.
- This is where the user controls the personal information they want to display to other users.
- The User Class was essential for the functionality of the Profile View Controller as well as the Edit Profile View Controller.

EagleSocial 〉 LS iPhone | Finished running EagleSocial on LS iPhone

EagleSocial 〉 EagleSocial 〉 Model 〉 User.swift 〉 init()

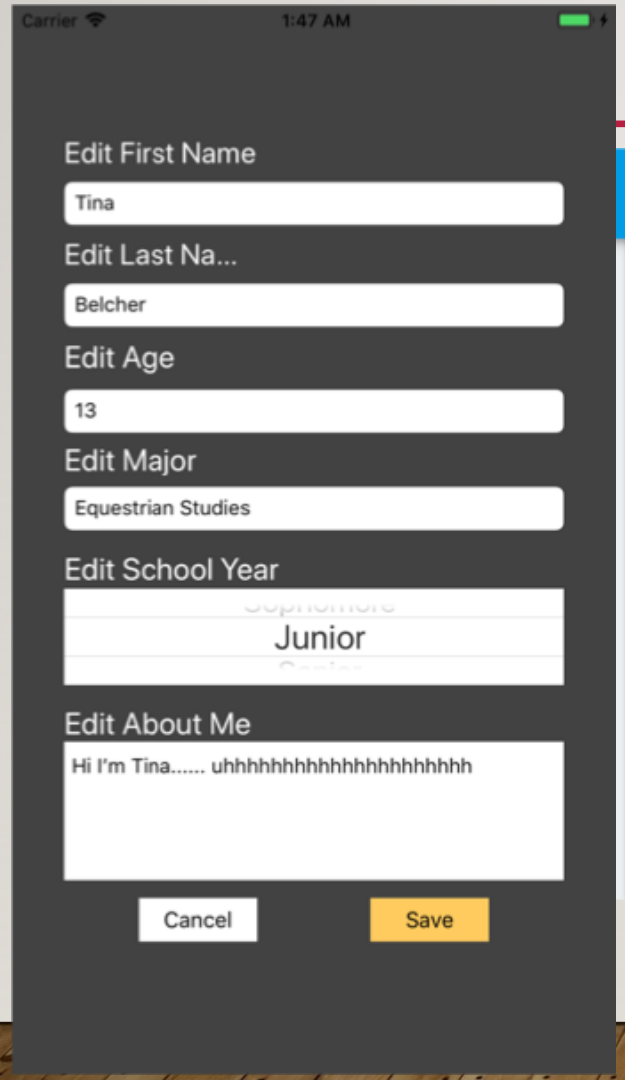
```swift
var thisUser = User()

class User
{
    var ref : DatabaseReference!
    var refHandle : DatabaseHandle?


    var userID: String
    var name: String
    var email: String
    var age: String
    var major: String
    var schoolYear: String
    var aboutMe: String
    var profilePic: UIImage


    init()
    {
        self.userID = ""
        self.name = ""
        self.email = ""
        self.age = ""
        self.major = ""
        self.schoolYear = ""
        self.aboutMe = ""
        self.profilePic =
        self.updateUser()
        self.updateProfilePic()

    }

    public func updateUser() {
        let userid = Auth.auth().currentUser?.uid
        self.ref = Database.database().reference()
        self.refHandle = self.ref.child("Users").child(userid!).observe(.value) { (snapshot) in
            guard let snapDict = snapshot.value as? [String : String] else { return }
```

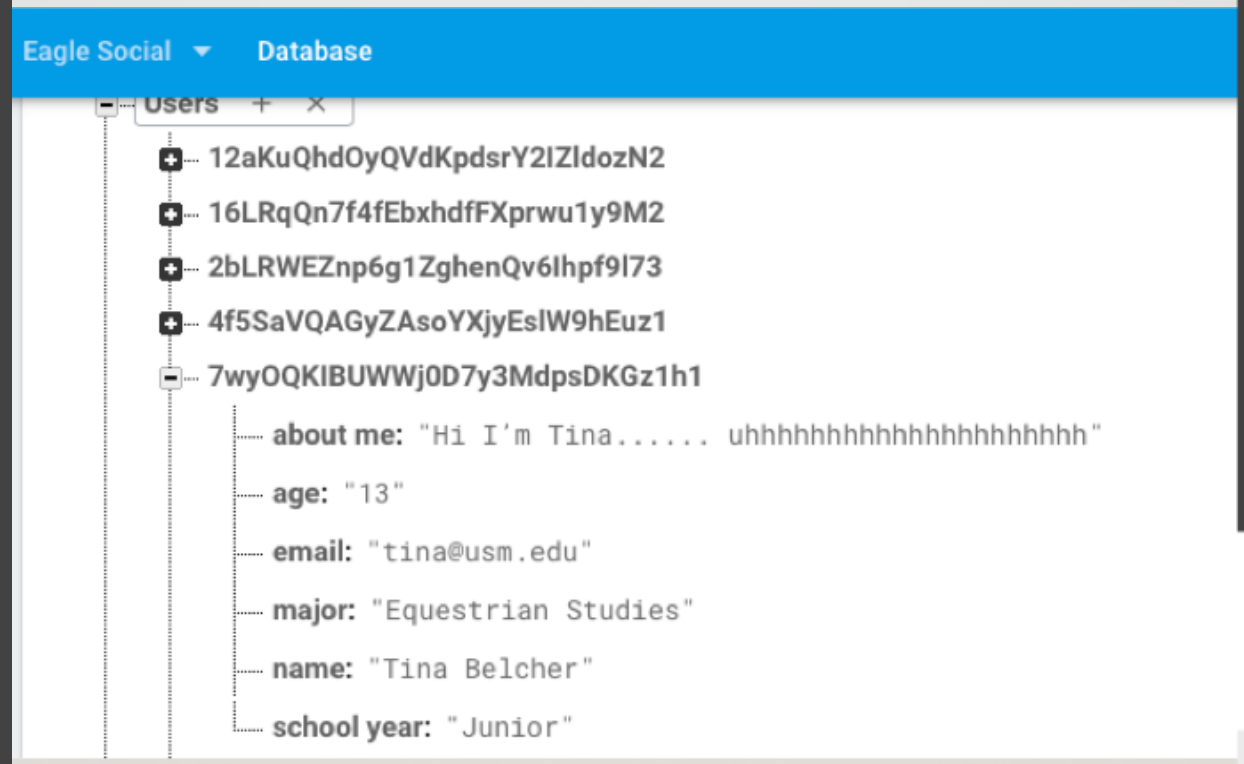
EagleSocial 〉 LS iPhone | Finished running EagleSocial on LS iPhone

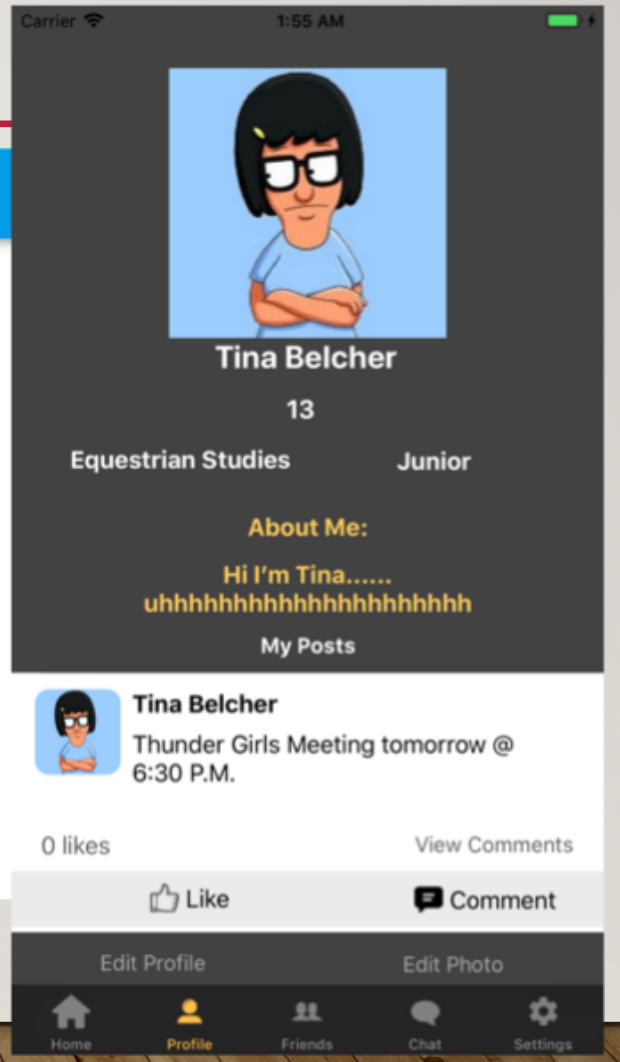
EagleSocial 〉 EagleSocial 〉 Model 〉 User.swift 〉 init()

```swift
    public func updateUser() {
        let userid = Auth.auth().currentUser?.uid
        self.ref = Database.database().reference()
        self.refHandle = self.ref.child("Users").child(userid!).observe(.value) { (snapshot) in
            guard let snapDict = snapshot.value as? [String : String] else { return }

            guard let name = snapDict["name"],
                let email = snapDict["email"],
                let age = snapDict["age"],
                let major = snapDict["major"],
                let aboutMe = snapDict["about me"],
                let schoolYear = snapDict["school year"]
                else { return }


            self.userID = userid!
            self.name = name
            self.email = email
            self.age = age
            self.major = major
            self.aboutMe = aboutMe
            self.schoolYear = schoolYear
        }
    }

    public func getName() -> String {
        return self.name
    }

    public func setName(userName: String)
    {
        self.name = userName
    }

    public func setProfilePic(image: UIImage) {
        self.profilePic = image
    }
```

EDITING THE PROFILE

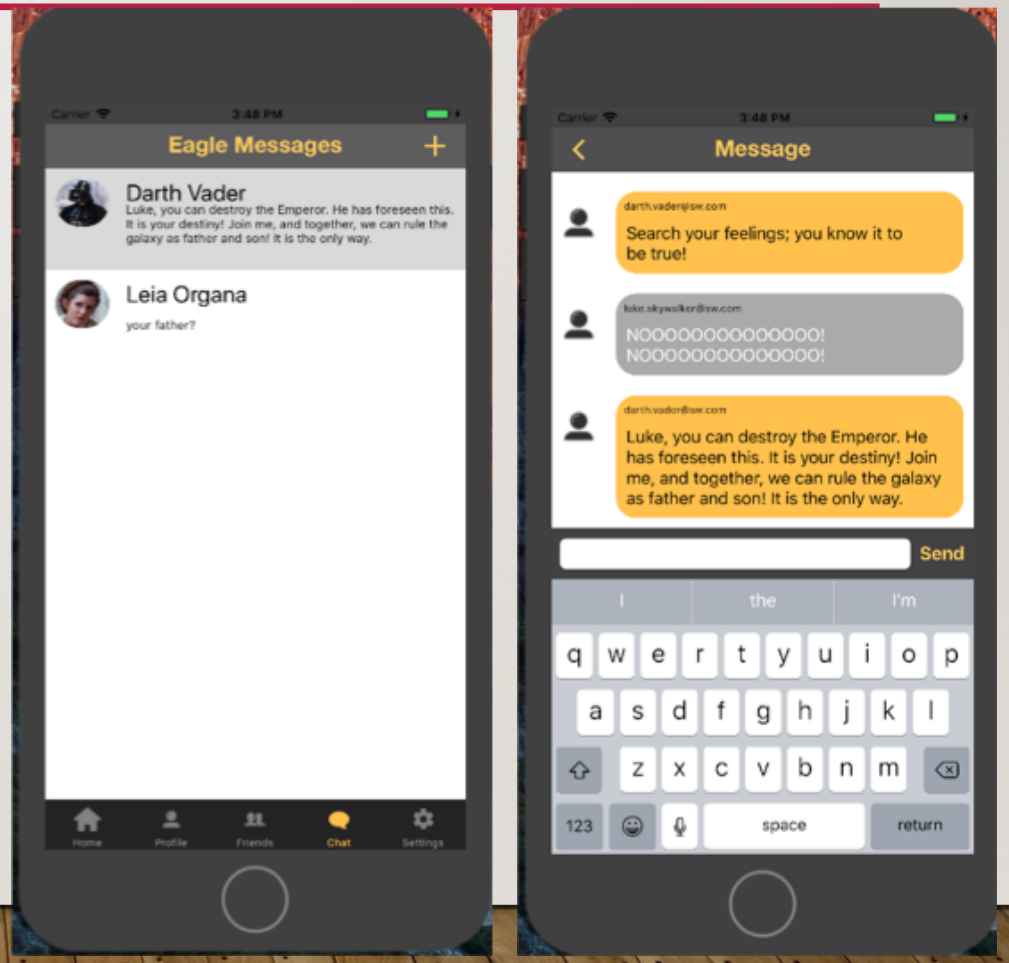






CHAT

- The concept is to give students a way to find and converse with their fellow students and teachers.
- Eagle Message works similarly to any other chat application.
- It allows you to select a person from your friends list and start a conversation on our platform.
- You can even send emoji's



CHAT – THE MESSAGE CLASS

- The Message Class

EagleSocial 〉 EagleSocial 〉 Model 〉 Message.swift 〉 init(membersa:messageDictionarya:)

```swift
//
//  Message.swift
//  EagleSocial
//
//  Created by Michael Pearson on 3/18/18.
//  Copyright © 2018 Jody Bailey. All rights reserved.
//

import Foundation
import Firebase

class Message {

    private var senderId : String
    private var messageBody : String
    private var conversationID : String
    private var messageDictionary : Dictionary<String, Any>
    private var members : Dictionary<String,String>

    init() {
        senderId = ""
        messageBody = ""
        conversationID = ""

        messageDictionary = ["Sender" : "", "MessageBody" : "", "ConversationID": ""]
        members = [:]
    }
    init(membersa: Dictionary<String, String>, messageDictionarya: Dictionary<String,
        Any>)
    {
        senderId = String(describing: messageDictionarya["Sender"]!)
        messageBody = String(describing: messageDictionarya["MessageBody"]!)
        conversationID = String(describing: messageDictionarya["ConversationID"]!)
```

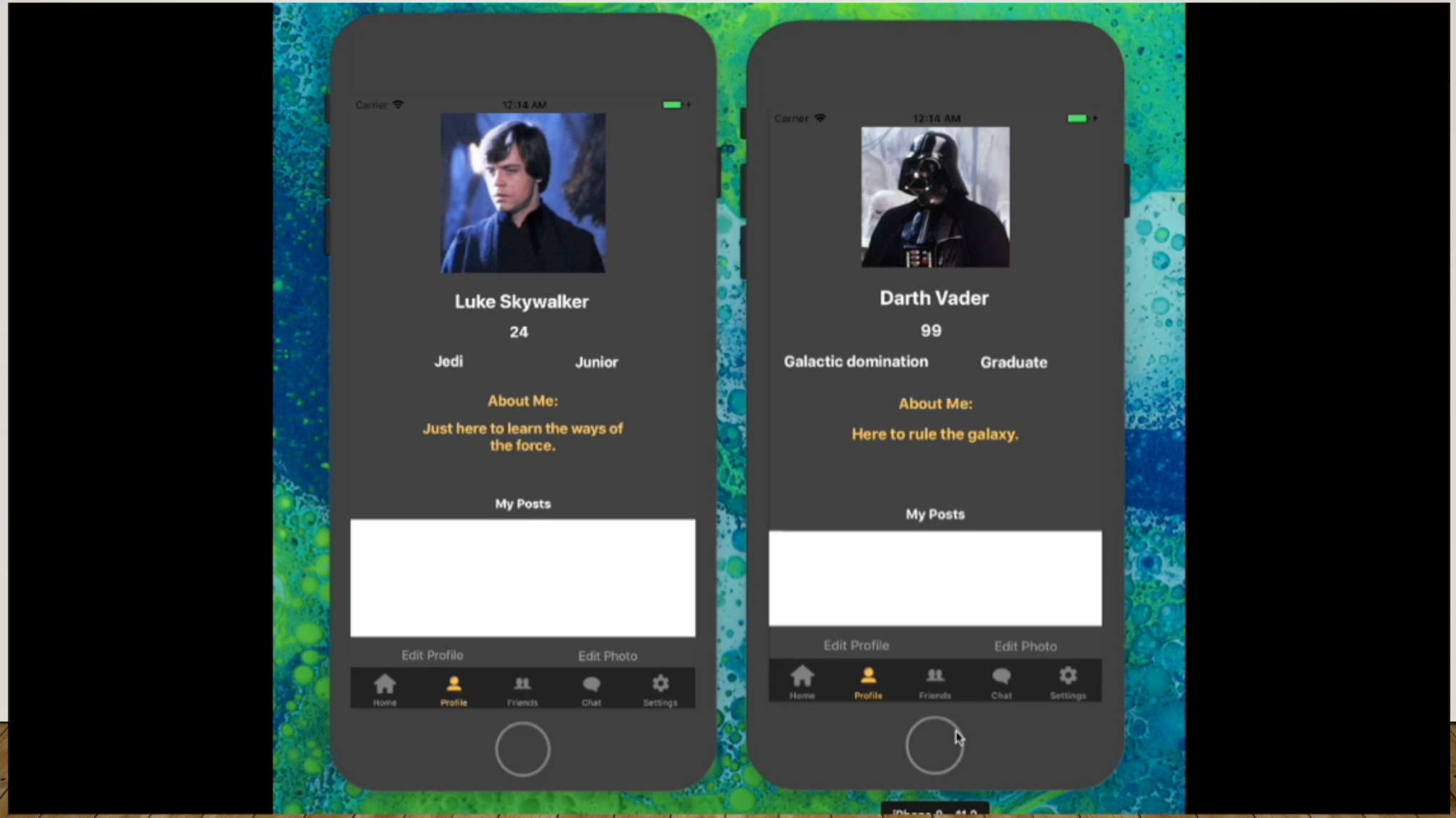
CHAT – THE CONVERSATION CLASS

- The Conversation Class

EagleSocial 〉 EagleSocial 〉 Model 〉 Conversation.swift 〉 init(convoId:mem:mes:)

```swift
//
//  Conversation.swift
//  EagleSocial
//
//  Created by Michael Pearson on 4/1/18.
//  Copyright © 2018 Jody Bailey. All rights reserved.
//

import Foundation
import Firebase

class Conversation {
    private var lastMessage : String
    private var username : String
    private var displayName : String
    private var conversationId : String
    private var members : Dictionary<String,Any>
    private var messages : Dictionary <String, Any>
    //TODO: - Add date property and methods

    //Initialize the object - Default
    init()
    {
        lastMessage = ""
        username = ""
        displayName = ""
        conversationId = ""
        members = [:]
        messages = [:]
    }
    init(convoId: String, mem: Dictionary<String, Any>, mes: Dictionary<String, Any>) {

        conversationId = convoId
```

CHAT


Carrier
12:14 AM
Luke Skywalker
24
Jedi
Junior
About Me:
Just here to learn the ways of the force.
My Posts
Edit Profile
Edit Photo
Home
Profile
Friends
Chat
Settings
Carrier
12:14 AM
Darth Vader
99
Galactic domination
Graduate
About Me:
Here to rule the galaxy.
My Posts
Edit Profile
Edit Photo
Home
Profile
Friends
Chat
Settings

SETTINGS - JODY

- Change Password – Option to allow the user to change their password.
- Change Email – Option to allow the user to change their email.
- Logout – Button that logs the user out and takes them back to the apps main screen where they will then have the option to Sign Up or Login.

```
// Function to log the user out
@IBAction func logoutPressed(_ sender: UIButton) {
    do{
        try Auth.auth().signOut()
    }
    catch {
        print("Error signing out!")
    }

    UserDefaults.standard.set(false, forKey: "isUserLoggedIn")
    UserDefaults.standard.synchronize()

    tabBarController?.dismiss(animated: true, completion: nil)
}
```

SETTINGS - JODY

```
// Function to save the users changes to their password
@IBAction func saveButtonPressed(_ sender: UIButton) {

    let newPass = self.newPasswordTextField.text!
    let email = Auth.auth().currentUser?.email
    let oldPass = self.oldPasswordTextField.text!

    // Make sure the user enters at least 6 characters for their new password
    if newPass.count >= 6 {

        // Setting the users current credentials to reathenticate them with Firebase
        let credential = EmailAuthProvider.credential(withEmail: email!, password: oldPass)
        Auth.auth().currentUser?.reauthenticate(with: credential, completion: { (error) in
            if error == nil {
                Auth.auth().currentUser?.updatePassword(to: newPass) { (errror) in
                    print(errror as Any)
                }
            } else {
                print(error as Any)
            }
        })

        self.dismiss(animated: true, completion: nil)
    } else {
        self.newPasswordTextField.text = ""
        self.newPasswordTextField.placeholder = "Password too short"
    }

}
```