Micheal Willard
932480626
CS 496
Final Project
http://web.engr.oregonstate.edu/~willardm/cs496/finalproject.mov

Overview

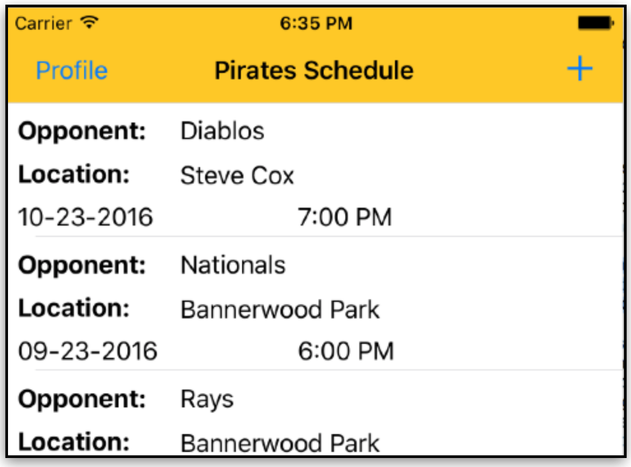
This mobile application is an expansion of the previous mobile application that was built and submitted. It is a baseball scheduling application which allows users to create accounts, create game events, and then mark whether they are attending the game or not.

When a user first opens the application they are presented with a Login View. If they have an account, they can log in right away. The two input values are an email and a password. If a user enters an incorrect email or password, they are alerted and not allowed past the login screen. If they leave the fields blank they are also alerted.

If they don't have an account there is a button which allows them to go to an Account Creation View. In the Account Creation a user is asked to enter their name, phone, email, password and confirm their password. The user must input information into all fields in order to create an account. Additionally, if their password doesn't match the password confirmation input, they are alerted.

After a user has successfully logged in, they are prompted with their Profile View. Right now this is a pretty minimal view which just contains the user's name and a Logout Button. The user can choose to Logout and be redirected to the Login View or they can dismiss the Profile View by selecting the Done Button and be taken to the Schedule Table View.

The Schedule Table View (FIG 1.1) that is presented when opening the app is a table view showing all of the games that are stored in the database owned by the API. They are retrieved from https:// hw3api.appspot.com/game. This data populates class objects and builds a table. From the first view, a user can enter two other views.



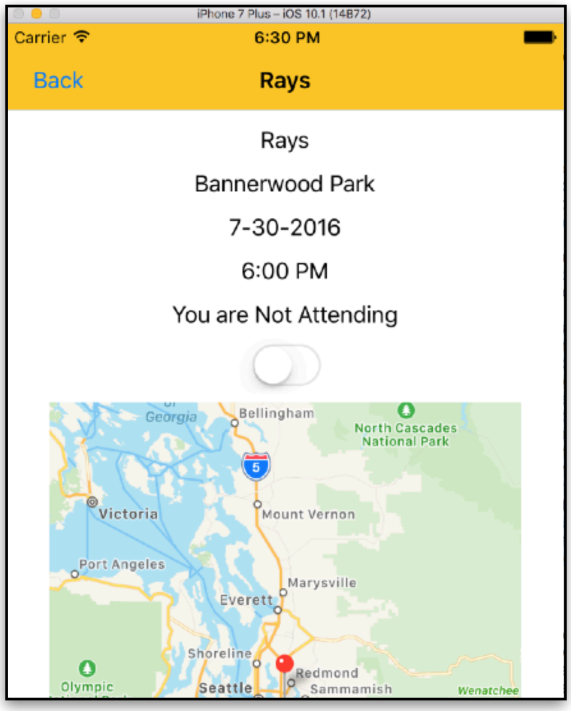
Opponent: Diablos
Location: Steve Cox
10-23-2016 7:00 PM

Opponent: Nationals
Location: Bannerwood Park
09-23-2016 6:00 PM

Opponent: Rays
Location: Bannerwood Park

FIGURE 1.1: SCHEDULE TABLE VIEW

The Game View (FIG 1.2) uses a cocoa touch class where the user can select a game from the table view, simply by touching it. This launches the Game View. In the Game View, the user can see the opponent, location, date, time and an indication of whether or not they are attending. Additionally in this view, there is a MapKit object. The MapKit MapView displays the location of the game on the map, and in certain circumstances the user's location. Finally, the user can exit this view and return to the table view by selecting the Back button in the upper left of the navigation header.



Rays

Bannerwood Park

7-30-2016

6:00 PM

You are Not Attending

FIGURE 1.2: GAME VIEW

The Game Add View is accessible from the Table View by a "+" icon in the upper right corner. This is the Game Add View. This view has 4 text fields that allow a user to input an opponent, location, date and time. The date/time picker native to iOS wasn't incorporated in this version due to time constraints and the already steep learning curve. From the Game Add View, the user can choose to Save the game or Cancel. Either option returns them to the home Table view. The Save button is disabled until all fields are filled in. These are all required entities, so this is how invalid, empty data is controlled. Once the user clicks Save, a POST request is made to the API. The user returns to the Table View and sees the new game added.

The Storyboard (FIG 1.3) shows how the app is intended to flow. It is laid out as describe in this overview.

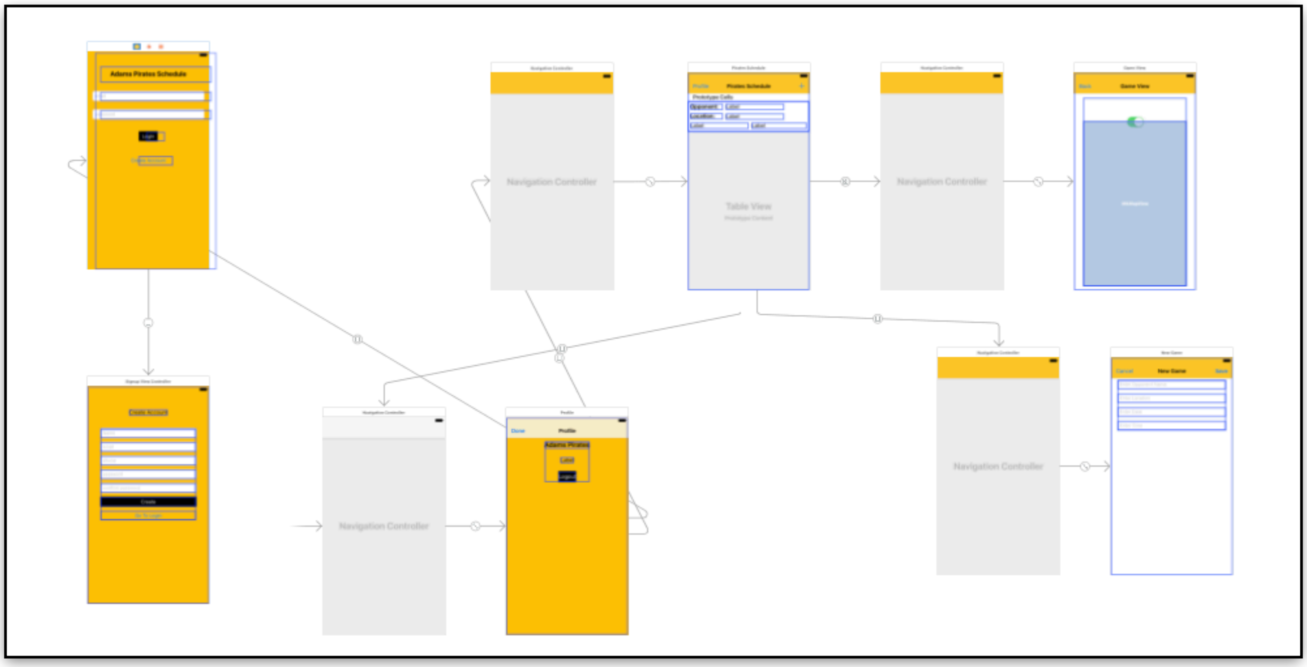


FIGURE 1.3: STORYBOARD

HTTP Calls

This application features all 4 HTTP verbs: GET, POST, PUT and DELETE.

A **GET** call is initiated on the Schedule Table View (Fig 1.1). It GETs all the game sin the database and populates a table. The call for that is visible in Figure 1.4

GET URI: https://hw3api.appspot.com/game

The first **POST** call is made when a user adds a game from the Add Game View. This call can be seen in Figure 1.5.

POST URI: https://hw3api.appspot.com/game

POST String: "opp_name=" + opp_name + "&date=" + date + "&time=" + time + "&location=" + location

The second **POST** call is made from the Account Create View. This code (Fig 1.6) is called when the user presses the "Create Account" Button on that View.

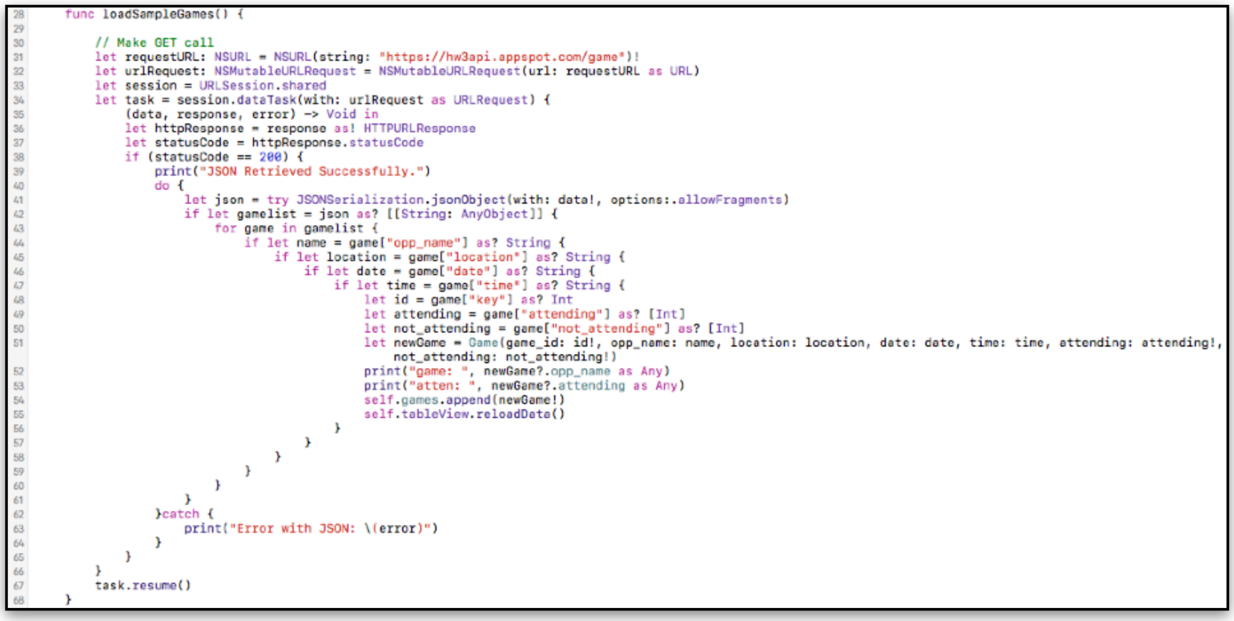
POST URI: https://hw3api.com/player

POST String: "name=" + name + "&email=" + email + "&phone=" + phone + "&password=" + password

On the Game View (FIG 1.2) the switch triggers an action which then makes the **PUT** call. The call is a PUT, but on the backend it also performs a DELETE operation on the database. First, it PUTs the User's id into the attending or not_attending array, then it DELETEs the user's id from the other array if it exists in there. This code is visible in Figure 1.7.

PUT URI (Attending): https://hw3api.com/a/game/[game id]/player/[player id]

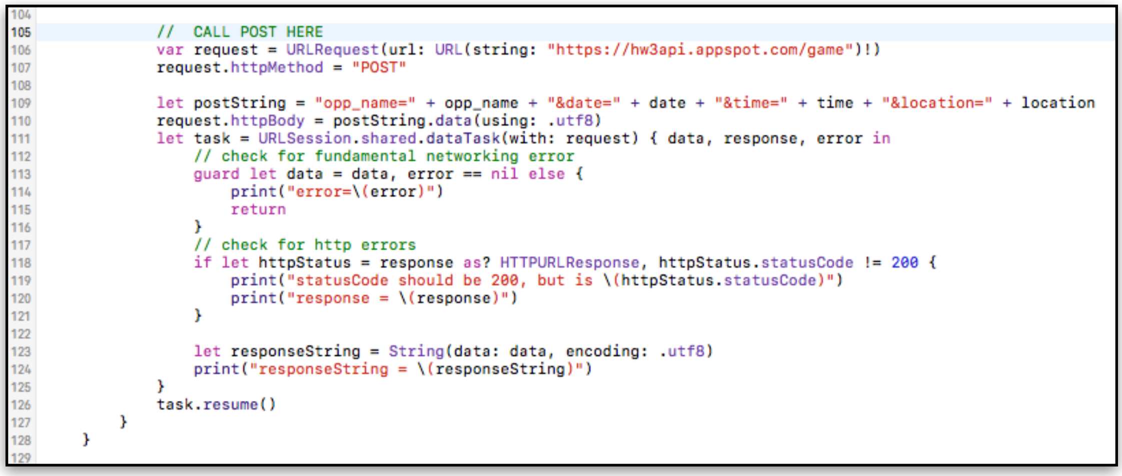
PUT URI (Not Attending): https://hw3api.com/na/game/[game id]/player/[player id]



```
    func loadSampleGames() {

        // Make GET call
        let requestURL: NSURL = NSURL(string: "https://hw3api.appspot.com/game")!
        let urlRequest: NSMutableURLRequest = NSMutableURLRequest(url: requestURL as URL)
        let session = URLSession.shared
        let task = session.dataTask(with: urlRequest as URLRequest) {
            (data, response, error) -> Void in
            let httpResponse = response as! HTTPURLResponse
            let statusCode = httpResponse.statusCode
            if (statusCode == 200) {
                print("JSON Retrieved Successfully.")
                do {
                    let json = try JSONSerialization.jsonObject(with: data!, options:.allowFragments)
                    if let gamelist = json as? [[String: AnyObject]] {
                        for game in gamelist {
                            if let name = game["opp_name"] as? String {
                                if let location = game["location"] as? String {
                                    if let date = game["date"] as? String {
                                        if let time = game["time"] as? String {
                                            let id = game["key"] as? Int
                                            let attending = game["attending"] as? [Int]
                                            let not_attending = game["not_attending"] as? [Int]
                                            let newGame = Game(game_id: id!, opp_name: name, location: location, date: date, time: time, attending: attending!,
                                                not_attending: not_attending!)
                                            print("game: ", newGame?.opp_name as Any)
                                            print("atten: ", newGame?.attending as Any)
                                            self.games.append(newGame!)
                                            self.tableView.reloadData()
                                        }
                                    }
                                }
                            }
                        }
                    }
                }catch {
                    print("Error with JSON: \(error)")
                }
            }
        }
        task.resume()
    }
```

FIGURE 1.4: GET CALL

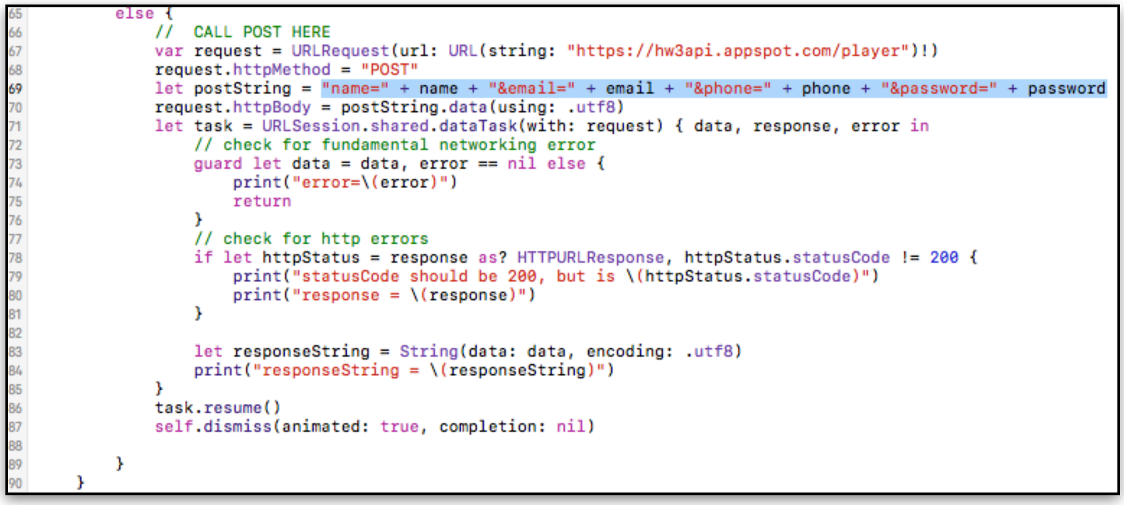


```
            //  CALL POST HERE
            var request = URLRequest(url: URL(string: "https://hw3api.appspot.com/game")!)
            request.httpMethod = "POST"

            let postString = "opp_name=" + opp_name + "&date=" + date + "&time=" + time + "&location=" + location
            request.httpBody = postString.data(using: .utf8)
            let task = URLSession.shared.dataTask(with: request) { data, response, error in
                // check for fundamental networking error
                guard let data = data, error == nil else {
                    print("error=\(error)")
                    return
                }
                // check for http errors
                if let httpStatus = response as? HTTPURLResponse, httpStatus.statusCode != 200 {
                    print("statusCode should be 200, but is \(httpStatus.statusCode)")
                    print("response = \(response)")
                }

                let responseString = String(data: data, encoding: .utf8)
                print("responseString = \(responseString)")
            }
            task.resume()
        }
    }
```

FIGURE 1.5: POST CALL FOR GAME CREATION

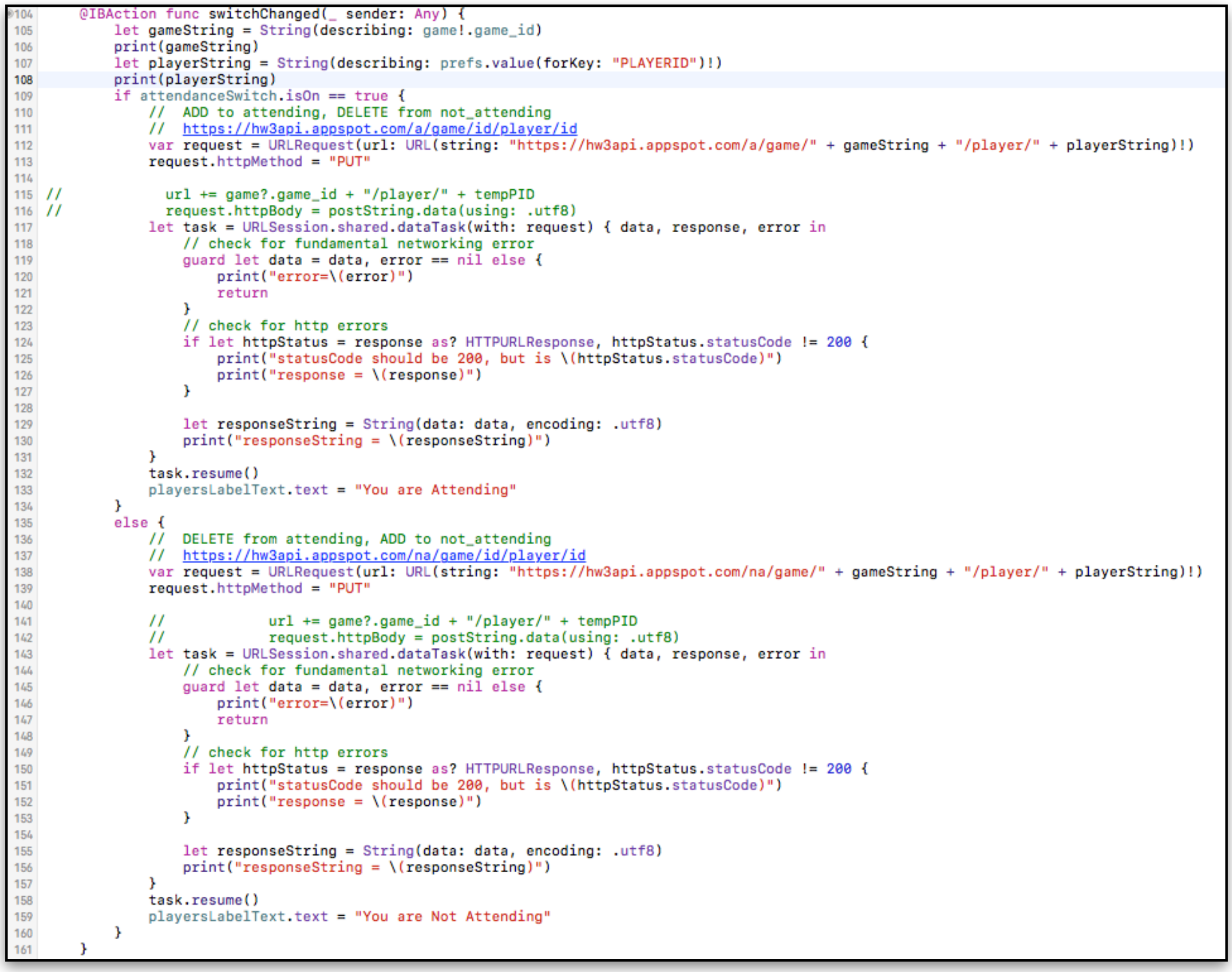


```
        else {
            //  CALL POST HERE
            var request = URLRequest(url: URL(string: "https://hw3api.appspot.com/player")!)
            request.httpMethod = "POST"
            let postString = "name=" + name + "&email=" + email + "&phone=" + phone + "&password=" + password
            request.httpBody = postString.data(using: .utf8)
            let task = URLSession.shared.dataTask(with: request) { data, response, error in
                // check for fundamental networking error
                guard let data = data, error == nil else {
                    print("error=\(error)")
                    return
                }
                // check for http errors
                if let httpStatus = response as? HTTPURLResponse, httpStatus.statusCode != 200 {
                    print("statusCode should be 200, but is \(httpStatus.statusCode)")
                    print("response = \(response)")
                }

                let responseString = String(data: data, encoding: .utf8)
                print("responseString = \(responseString)")
            }
            task.resume()
            self.dismiss(animated: true, completion: nil)

        }
    }
```

FIGURE 1.6: POST CALL FOR ACCOUNT CREATION



```
    @IBAction func switchChanged(_ sender: Any) {
        let gameString = String(describing: game!.game_id)
        print(gameString)
        let playerString = String(describing: prefs.value(forKey: "PLAYERID")!)
        print(playerString)
        if attendanceSwitch.isOn == true {
            //  ADD to attending, DELETE from not_attending
            //  https://hw3api.appspot.com/a/game/id/player/id
            var request = URLRequest(url: URL(string: "https://hw3api.appspot.com/a/game/" + gameString + "/player/" + playerString)!)
            request.httpMethod = "PUT"

//            url += game?.game_id + "/player/" + tempPID
//            request.httpBody = postString.data(using: .utf8)
            let task = URLSession.shared.dataTask(with: request) { data, response, error in
                // check for fundamental networking error
                guard let data = data, error == nil else {
                    print("error=\(error)")
                    return
                }
                // check for http errors
                if let httpStatus = response as? HTTPURLResponse, httpStatus.statusCode != 200 {
                    print("statusCode should be 200, but is \(httpStatus.statusCode)")
                    print("response = \(response)")
                }

                let responseString = String(data: data, encoding: .utf8)
                print("responseString = \(responseString)")
            }
            task.resume()
            playersLabelText.text = "You are Attending"
        }
        else {
            //  DELETE from attending, ADD to not_attending
            //  https://hw3api.appspot.com/na/game/id/player/id
            var request = URLRequest(url: URL(string: "https://hw3api.appspot.com/na/game/" + gameString + "/player/" + playerString)!)
            request.httpMethod = "PUT"

            //            url += game?.game_id + "/player/" + tempPID
            //            request.httpBody = postString.data(using: .utf8)
            let task = URLSession.shared.dataTask(with: request) { data, response, error in
                // check for fundamental networking error
                guard let data = data, error == nil else {
                    print("error=\(error)")
                    return
                }
                // check for http errors
                if let httpStatus = response as? HTTPURLResponse, httpStatus.statusCode != 200 {
                    print("statusCode should be 200, but is \(httpStatus.statusCode)")
                    print("response = \(response)")
                }

                let responseString = String(data: data, encoding: .utf8)
                print("responseString = \(responseString)")
            }
            task.resume()
            playersLabelText.text = "You are Not Attending"
        }
    }
```

FIGURE 1.7: PUT/DELETE CALL

Account System

The login authorization is performed through a POST request. From the Login Screen, when the “Login” button is tapped a POST request is sent to the API backend. The API backend takes in the email and password sent. It checks these values against all stored player accounts. If a match is found, it will return a “token” which is essentially a JSON containing the player’s account info (name, email, phone, and player id/key). The backend of that check is all part of the Google App Engine API discussed earlier in this documentation.

Once the login is authenticated and successful, a local object is populated with the User’s information. The user’s player id/key is critical to the functionality of the attending/not attending functions within the Game View (Fig 1.2). The Game View uses that ID to determine the state of whether the user is attending and also to send through the PUT request.

Mobile Feature

As discussed in the overview, the mobile feature leverage in this app is the MapKit MapView. The location string retrieved form the API database is reverse-geocoded and displayed as a pin on the Map. Clicking the Pin reveals the correct location. In the demo video the iOS simulator was not displaying the user location, despite the MapView being configured to display that. The user location failing in the iOS simulator is apparently a common problem. However, that would be another use of a Mobile Feature, by using the user's location.