ATTENDANCE

QR Attendance

By:
Nagesa Ram Sri Charan Taduvai

About App:

It usually takes couple of minutes to mark the attendance of each and every student in the class. And if the class strength is more that could be even worse. This is frustrating for both teachers and students. So, I designed this app to completely eliminate this time waste in the class. With this app, teacher can take attendance with just snap of a finger.

As the name of the app says it all “QR Attendance”, this app uses QR Code technology to mark the attendance of the students. And not just that, this app also considers the location from where the user is trying to scan the attendance code. So, this app ensures that only legit student who attended the class gets the attendance.

This app also maintains all the attendance data so user can keep track of their attendance. Tutors can track the attendance of all the students enrolled in their class. They also have the ability to alter the attendance of the students of any student. The app also provides live updates on the current attendance which is in progress. They also have the ability to create new interesting classes that they would like to teach.

Students on the other hand, can use the code provided by the tutor to add the class to their enrollments. They can monitor the attendance details in all of their enrolled classes.

How does it work?

There are two types of accounts in my app, one is for tutors and the other is for students. The whole process starts with the tutor.

Tutor Side:

- In order take the attendance on any given day, tutor has to generate a new QR Code for that given class.
- Before generating the Code, tutor has to consider two key components, One is the size of the current classroom and the validity period of the code.
- Classroom Size: This app uses **GPS** to get the **location** of the tutor while generating the code. Based on the class size selected, only those students who are within the classroom size will be awarded.
- Validity Period: This option ensures that only those students who come to class before the Code expires or in other words, who come to class in time will get the attendance.
- Finally, tutor can generate the QR Code after make their desired selections.
- The generated QR Code will be displayed on the tutor's phone screen. And also, this app gives a link to the QR Code. This Code can be projected on the screen just by visiting that URL from the Classroom PC.

Student Side:

- On the student side of the app, students can scan the QR Code provided by the tutor.
- The code gets decoded instantly and awards the student with attendance for the day if it validates all the rules.
- Their attendance data gets updated in real-time. So, the students can check their attendance changing immediately.

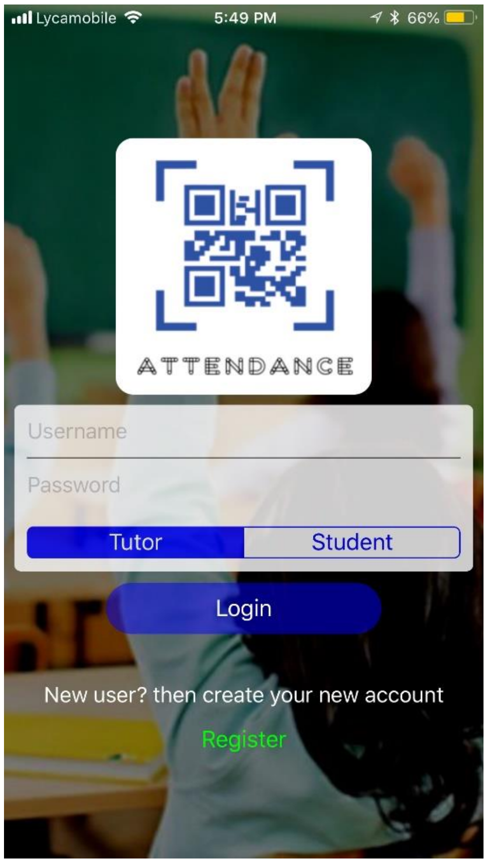
Features:

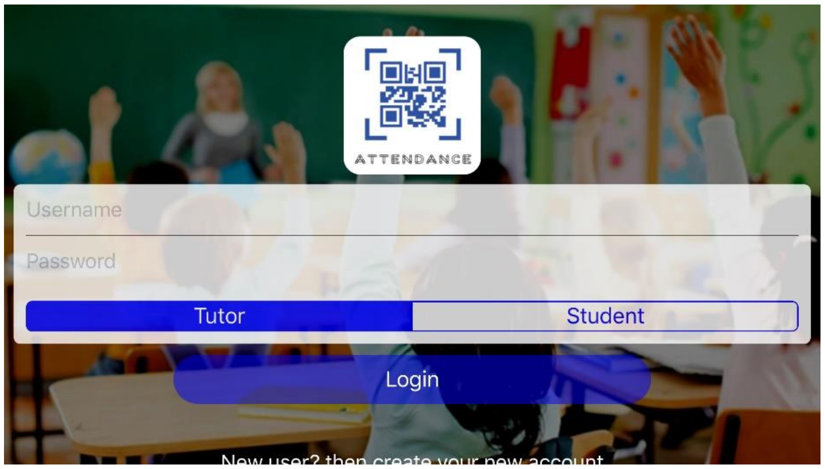
User Authentication:

This app uses **Firebase Authentication** to verify the users. It has got a well coded error catcher to handle all possible **User Inputs.** So, only a verified user will be authenticated to use this app.

QR Attendance has two types of users, Tutors and Students. In the login page, after entering valid credentials, user must select his user type. User will be authenticated only if he/she selects the correct user type. I implemented this by taking the current users UID and checking if there exists a user with that UID in the selected User Type table. User will not be authenticated if he/she has no access to the selected user type.

For example, if a student tries to use his credentials to login into a Tutors account, then the login fails.

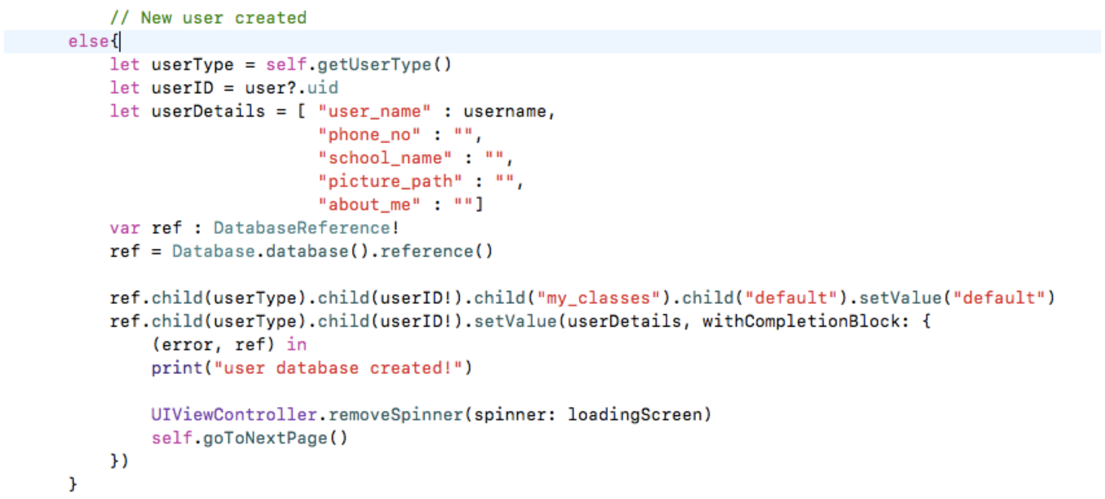




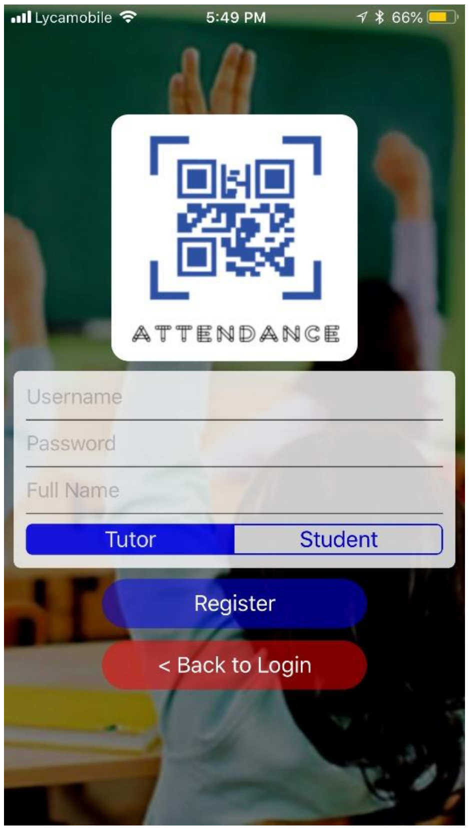
Registration:

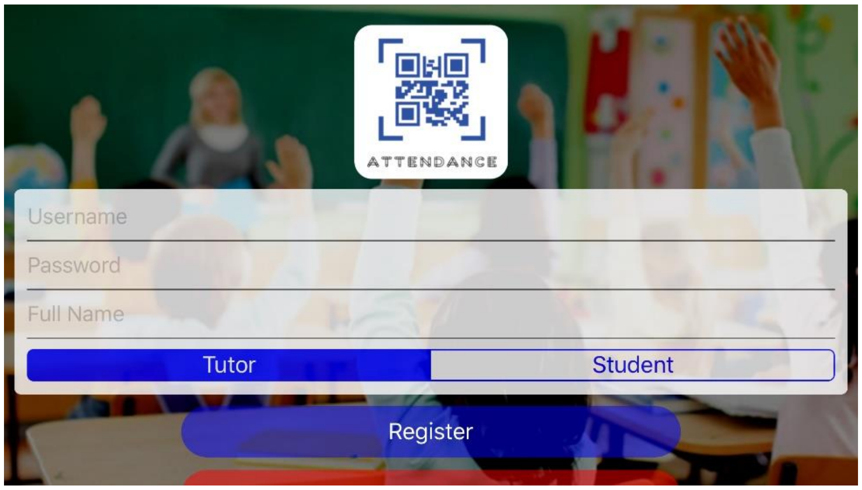
This app uses **Firebase Authentication** to register new users to the app. User must provide the username (xyz@qr.com) , password (At least 6 characters) and also the Full name of the user.

Once the registration is successful, a new node with a key of this new user's UID will be add to the Firebase Database. The username will be added as a child to this node along with few other default values.



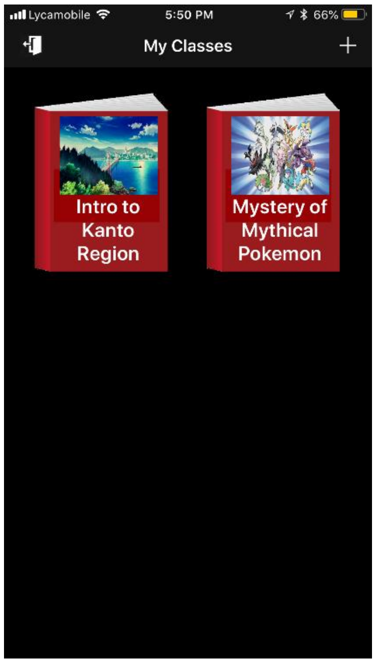
```
        // New user created
    else{
        let userType = self.getUserType()
        let userID = user?.uid
        let userDetails = [ "user_name" : username,
                            "phone_no" : "",
                            "school_name" : "",
                            "picture_path" : "",
                            "about_me" : ""]
        var ref : DatabaseReference!
        ref = Database.database().reference()

        ref.child(userType).child(userID!).child("my_classes").child("default").setValue("default")
        ref.child(userType).child(userID!).setValue(userDetails, withCompletionBlock: {
            (error, ref) in
            print("user database created!")

            UIViewController.removeSpinner(spinner: loadingScreen)
            self.goToNextPage()
        })
    }
```



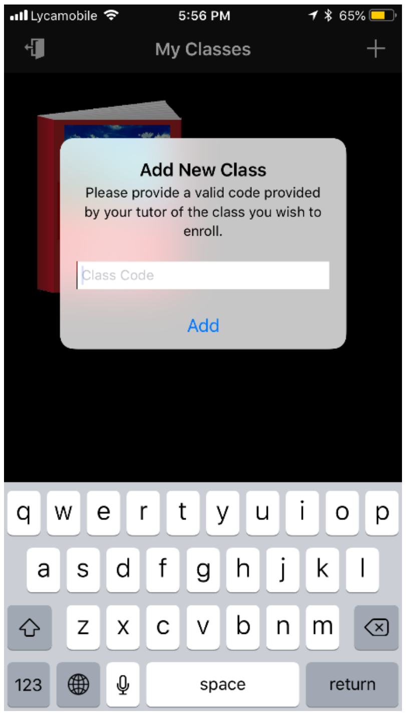


My Classes:

Once the user logs in successfully, they will be sent to the page "My Classes" which has a collection of their enrolled classes.



The view is entirely **a collection view** with **Custom Cells** designed to look like a book cover. Both the users have two Navigation bar buttons, one for logging out and the other one is to add a new class.



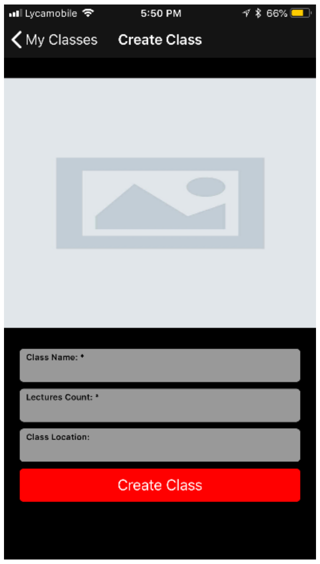
Add New Class

Please provide a valid code provided by your tutor of the class you wish to enroll.

Class Code

Add

Adding a new class button action is different for both the user types. For the students, this button leads to alert pop-up asking the user to enter a valid course Id to be able to add it.



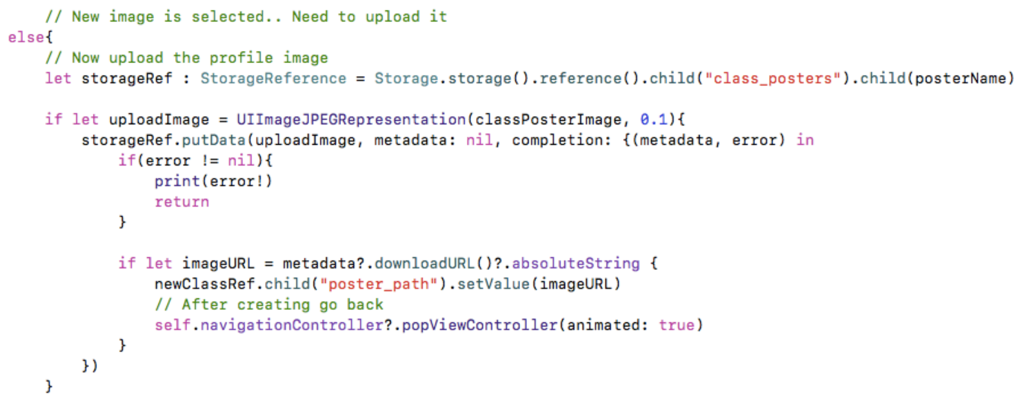
Class Name: *

Lectures Count: *

Class Location:

Create Class

On the other hand, tutors have a different action for this button. Tutors can create a new class by tapping this button. They must provide a name to the class and number lectures it will have. They can also set a beautiful poster to represent their class. I used **ImagePickerContollerDelegate** for this operation. In order to save space and also to speed up the image downloads, I compressed the image before uploading them to the **Firebase Storage**.



```
    // New image is selected.. Need to upload it
else{
    // Now upload the profile image
    let storageRef : StorageReference = Storage.storage().reference().child("class_posters").child(posterName)

    if let uploadImage = UIImageJPEGRepresentation(classPosterImage, 0.1){
        storageRef.putData(uploadImage, metadata: nil, completion: {(metadata, error) in
            if(error != nil){
                print(error!)
                return
            }

            if let imageURL = metadata?.downloadURL()?.absoluteString {
                newClassRef.child("poster_path").setValue(imageURL)
                // After creating go back
                self.navigationController?.popViewController(animated: true)
            }
        })
    }
```

If the user taps on the any of their classes, they will be taken to the details view for that class. This details View’s root controller is a **Tab Bar Controller**. Based on the user type, the tabs inside the tab bar controller varies.

For tutors, they have:

Class Home – Details of their class

Generate QR – For generating attendance QR Codes.

My Students – List of students and their details.

My Profile – To modify the users profile.

On the other hand, students have:

My Attendance – Everything about their attendance.

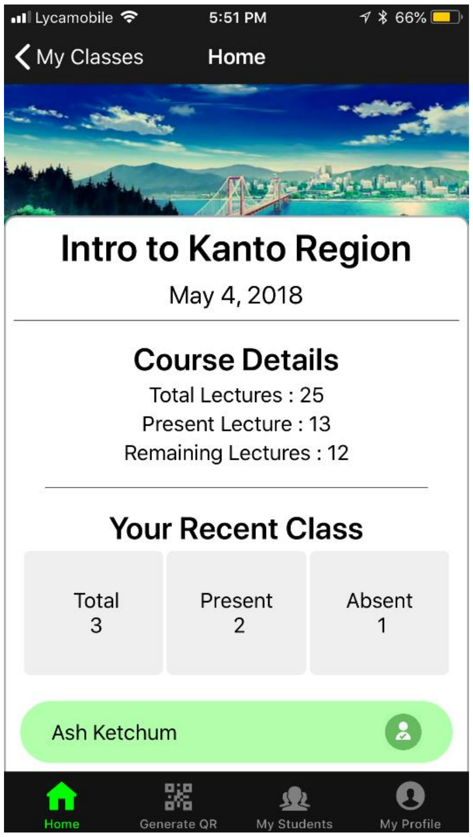
Scan QR – To scan the attendance QR created by Tutor.

My Profile – To modify users profile.

Class Home (Tutors):

Tutors class Home has all the details regarding the selected class. A collection of stack and scrollViews are used to display various details of the Course and the Class.

A **tableView** is used to show the most recent attendance details of all the students in the class. The table View's **Custom Cell** is in Green color if the student is present and red Color if the student is absent. There is a stack of counters to show the student counts as show in figure.



Intro to Kanto Region
May 4, 2018

Course Details
Total Lectures : 25
Present Lecture : 13
Remaining Lectures : 12

Your Recent Class

Total	Present	Absent
3	2	1

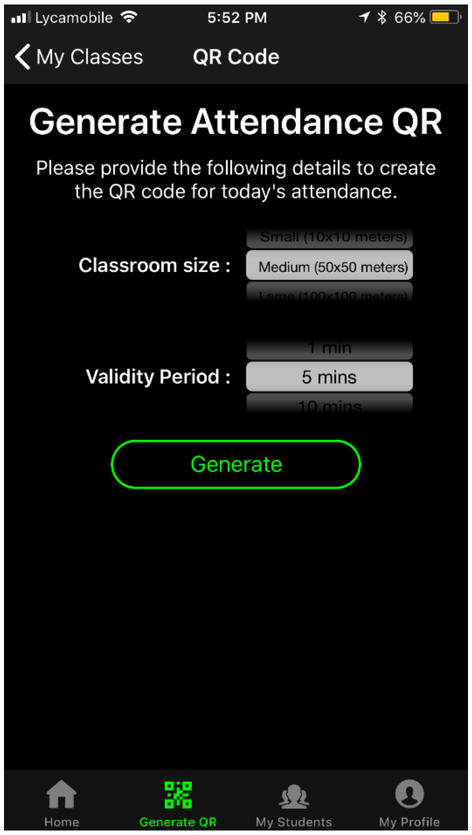
Ash Ketchum

This **table view** has a **Live Update** feature, so the counters and the cells inside the table view gets updated in the **Real-time**. This helps the teacher to know the class strength right away. And these details will be persistent till the next attendance is added.

```
func startObservingTodaysAttendance(){

    let classId = (CurrentDetails?.ClassID)!
    let ref : DatabaseReference = Database.database().reference()
    let classRef : DatabaseReference = ref.child("classes").child(classId)

    // Start observing
    classRef.observe(.value, with: { (snapshot) in

        let currentAttendanceId = (snapshot.childSnapshot(forPath: "latest_attendance").value as? String)!

        if (!currentAttendanceId.isEmpty){
            let attendanceSnapshot : DataSnapshot = snapshot.childSnapshot(forPath: "attendance").childSnapshot(forPath:
```

Generate QR (Tutor):

This particular tab might look simple, but this is the backbone of the app. Tutors use this tab to generate the Attendance QR Codes for the class.



Generate Attendance QR

Please provide the following details to create the QR code for today's attendance.

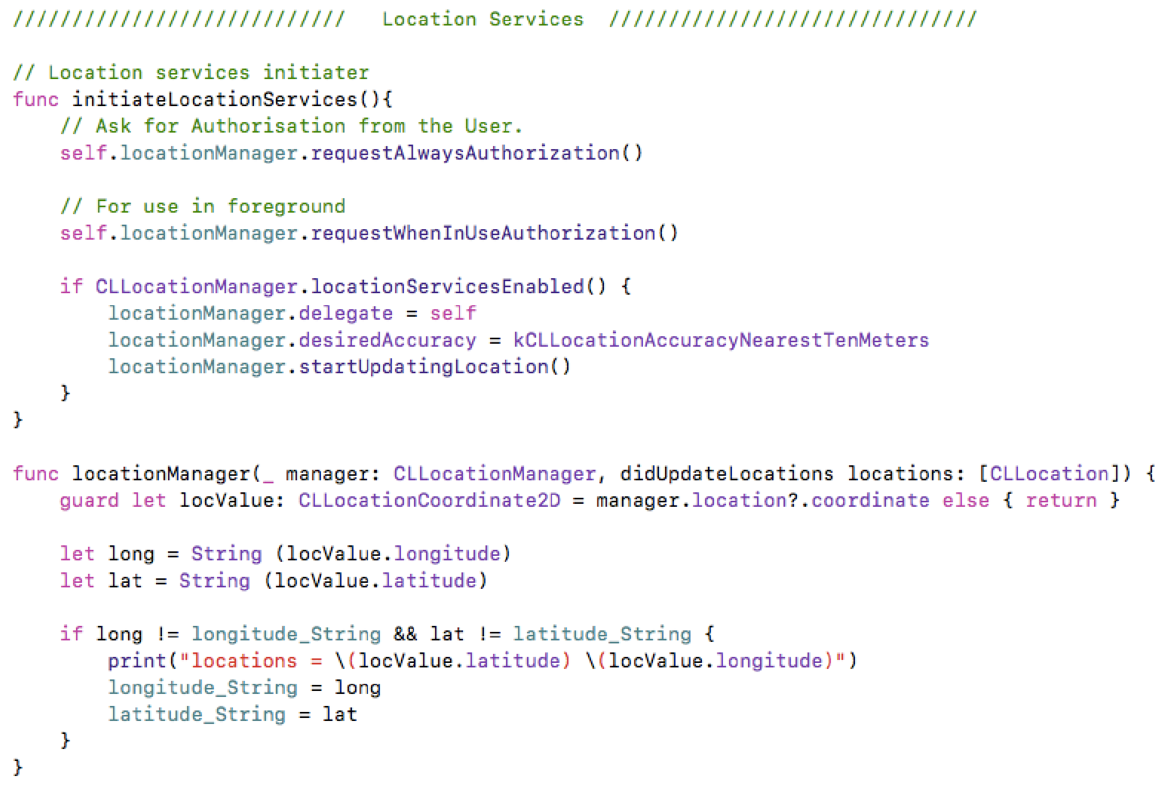
Classroom size : Medium (50x50 meters)

Validity Period : 5 mins

Tutor needs to set the desired classroom size and the validity period for the code. Both of them play a crucial role in the attendance.

Classroom size: This must be selected carefully, because only those students who are within the range of the classroom will be getting the attendance. A **Location Service** is started in the **Background** as soon as the user comes on to this tab. Once the generate button is tapped, the current location of the tutor will be recorded and stored in the Firebase for this corresponding attendance node. This location will be used later to check if the students are in the range of the class or not.

I used **CoreLocation** library and implemented **CLLocationManagerDelegate** to get the users GPS location. The following code snippet show how I recorded the user's location.

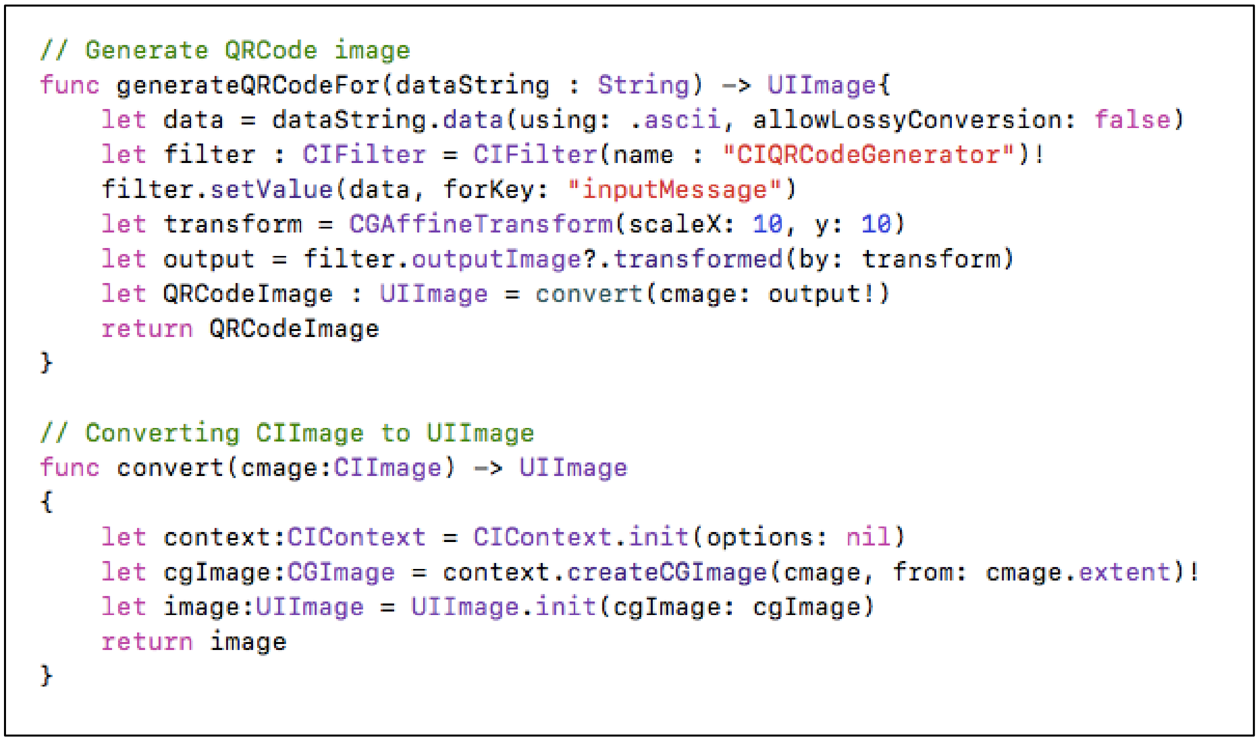


```
////////////////////////////   Location Services  ///////////////////////////////

// Location services initiater
func initiateLocationServices(){
    // Ask for Authorisation from the User.
    self.locationManager.requestAlwaysAuthorization()

    // For use in foreground
    self.locationManager.requestWhenInUseAuthorization()

    if CLLocationManager.locationServicesEnabled() {
        locationManager.delegate = self
        locationManager.desiredAccuracy = kCLLocationAccuracyNearestTenMeters
        locationManager.startUpdatingLocation()
    }
}

func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
    guard let locValue: CLLocationCoordinate2D = manager.location?.coordinate else { return }

    let long = String (locValue.longitude)
    let lat = String (locValue.latitude)

    if long != longitude_String && lat != latitude_String {
        print("locations = \(locValue.latitude) \(locValue.longitude)")
        longitude_String = long
        latitude_String = lat
    }
}
```

Validity Period: The attendance cannot be valid forever. Tutor has an option to select the time before which the QR Code expires. This will also be added to the corresponding attendance node in the Firebase. This will later be used to check if the student is scanning a valid code or not. So, only those students who scan the code in time gets the attendance.

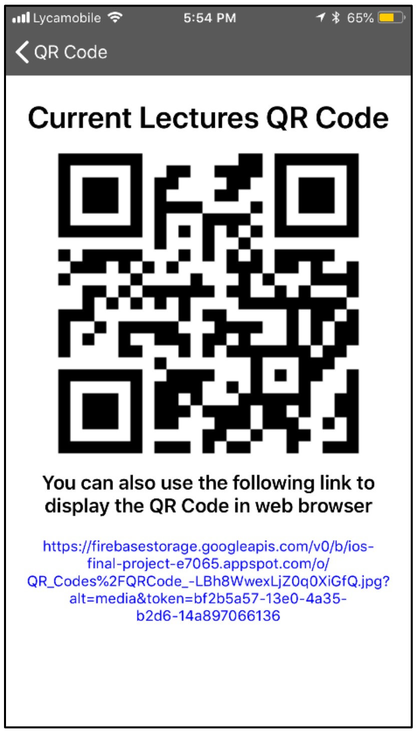
Once all these parameters are set, it is time to generate the QR Code. This code basically contains the key of current attendance node. When the student scans this code, they search the class attendance tree for this node and mark themselves present if they meet all the requirements.

To create the QR Code for the key, I first converted the key string into its ASCII equivalent. Then using **CIFilter's** "**CIQRCodeGenerator**", created the QR Code. However, the output for this function is a **CIImage**, but we need a **UIImage** to upload it to the Firebase. So, I convert the CIImage to UIImage and then upload it to the Firebase.



```
// Generate QRCode image
func generateQRCodeFor(dataString : String) -> UIImage{
    let data = dataString.data(using: .ascii, allowLossyConversion: false)
    let filter : CIFilter = CIFilter(name : "CIQRCodeGenerator")!
    filter.setValue(data, forKey: "inputMessage")
    let transform = CGAffineTransform(scaleX: 10, y: 10)
    let output = filter.outputImage?.transformed(by: transform)
    let QRCodeImage : UIImage = convert(cmage: output!)
    return QRCodeImage
}

// Converting CIImage to UIImage
func convert(cmage:CIImage) -> UIImage
{
    let context:CIContext = CIContext.init(options: nil)
    let cgImage:CGImage = context.createCGImage(cmage, from: cmage.extent)!
    let image:UIImage = UIImage.init(cgImage: cgImage)
    return image
}
```

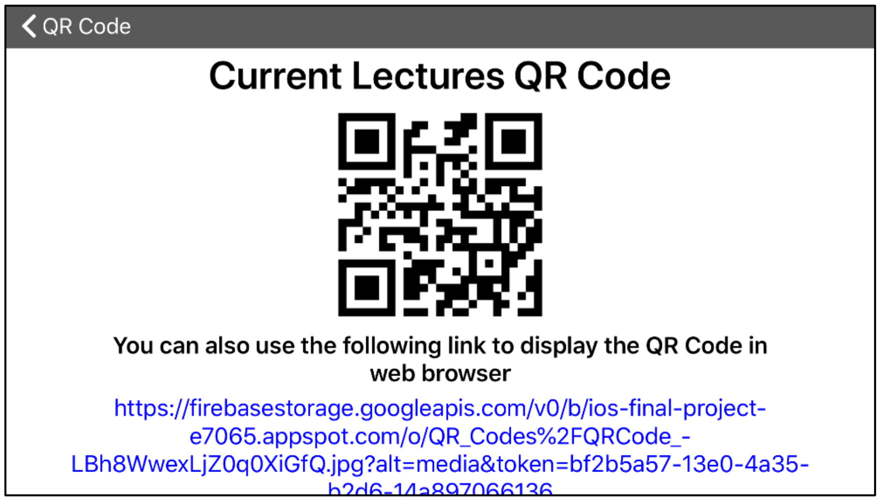
Once the QR Code is generated, it will be displayed in a full screen of the app. Tutor can ask each student to come and scan the code from their phone. If the class strength is big and tutor wants to wrap up the attendance faster, then they can use the Public URL link provided by the app and access it from the Classroom's PC.



Current Lectures QR Code

You can also use the following link to display the QR Code in web browser

https://firebasestorage.googleapis.com/v0/b/ios-final-project-e7065.appspot.com/o/QR_Codes%2FQRCode_-LBh8WwexLjZ0q0XiGfQ.jpg?alt=media&token=bf2b5a57-13e0-4a35-b2d6-14a897066136



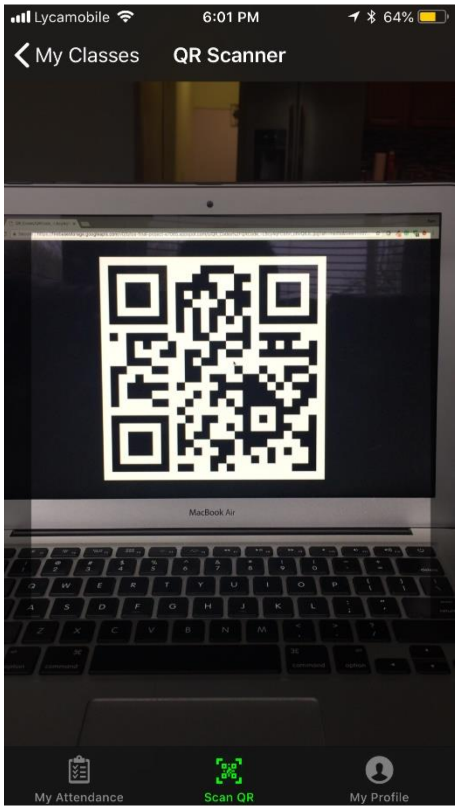
Current Lectures QR Code

You can also use the following link to display the QR Code in web browser

https://firebasestorage.googleapis.com/v0/b/ios-final-project-e7065.appspot.com/o/QR_Codes%2FQRCode_-LBh8WwexLjZ0q0XiGfQ.jpg?alt=media&token=bf2b5a57-13e0-4a35-b2d6-14a897066136

Scan QR (Students):

Now that we have a generated QR Code, let’s see how we can scan it, decode it and mark the attendance.



On the student’s side of the app, there is a QR Code scanner which can scan the Attendance code created by the tutor. I used Swift’s **AVFoundation API** for this purpose. I used **AVCaptureDevice** to capture the video using the phones camera. This app also has a error handler in case if the phone has no camera to scan the QR Code.

There is another function called **AVCaptureMetadataOutput()** to read the Metadata from the video that the camera is seeing. Then I specified that the metadata I am looking for is a QR Code by setting the output’s metadata object type to **AVMetadataObject.ObjectTypes.qr.** Once everything is setup, the camera session is started.

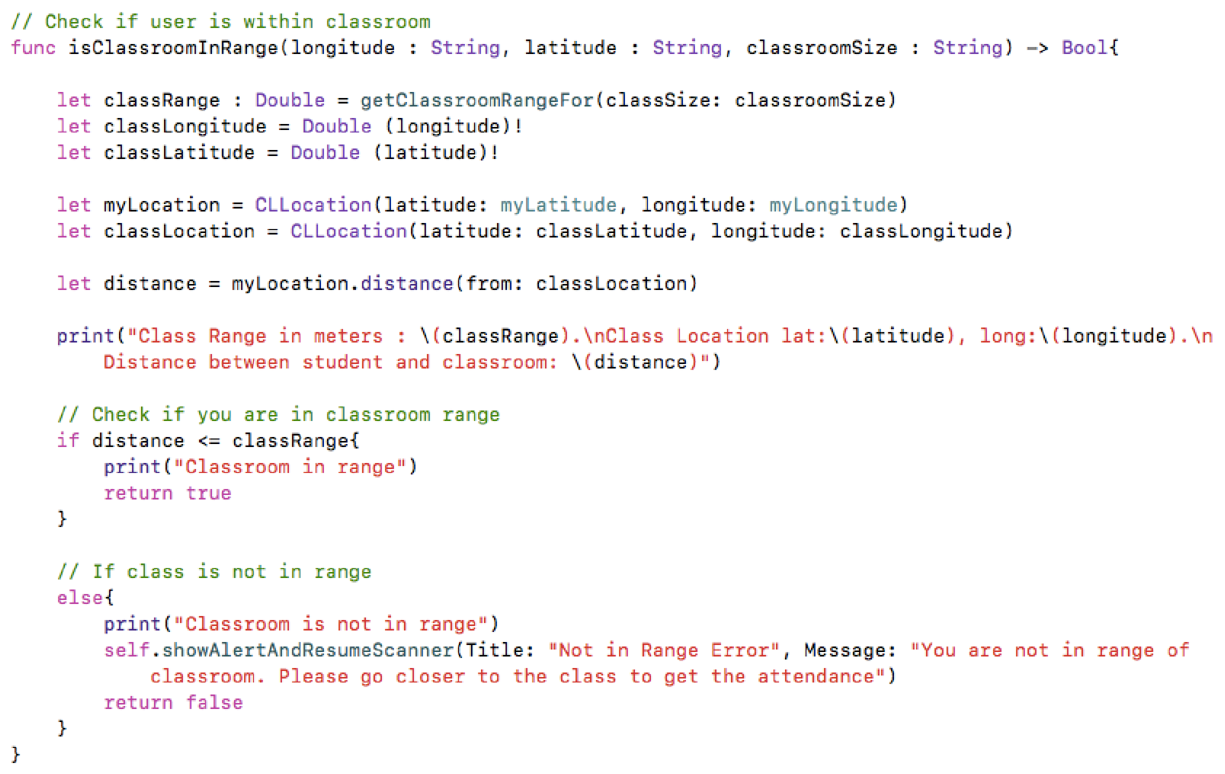
Once the User Scans a QR Code, **AVCaptureMetadataOutputObjectsDelegate**’s method gets the metadata object from the current session. This object is parsed to get the hidden key inside the QR Code. This is shown in the following snippet.

```swift
////////////////////////////////// QR Code API related functions ///////////////////////

func metadataOutput(_ output: AVCaptureMetadataOutput, didOutput metadataObjects: [AVMetadataObject],
    from connection: AVCaptureConnection) {
    if metadataObjects.count > 0{
        if let object = metadataObjects[0] as? AVMetadataMachineReadableCodeObject{
            if object.type == AVMetadataObject.ObjectType.qr{
                let qrData = object.stringValue
                print("Received QR Data: \(qrData!)")
                session.stopRunning()
                prefilterQRCodeString(rawStringData: qrData!)
            }
        }
    }
}
```

In the background, a **Location Service** keeps running to get the users current location. As soon as the QR Code gets parsed, it will be filtered to check if it is an invalid code. Then, using the key that is present in the QR Code, the current attendance node is pulled up from the Firebase Database. Then the app checks if the QR Code is still active, and proceeds further only if it still has validity.

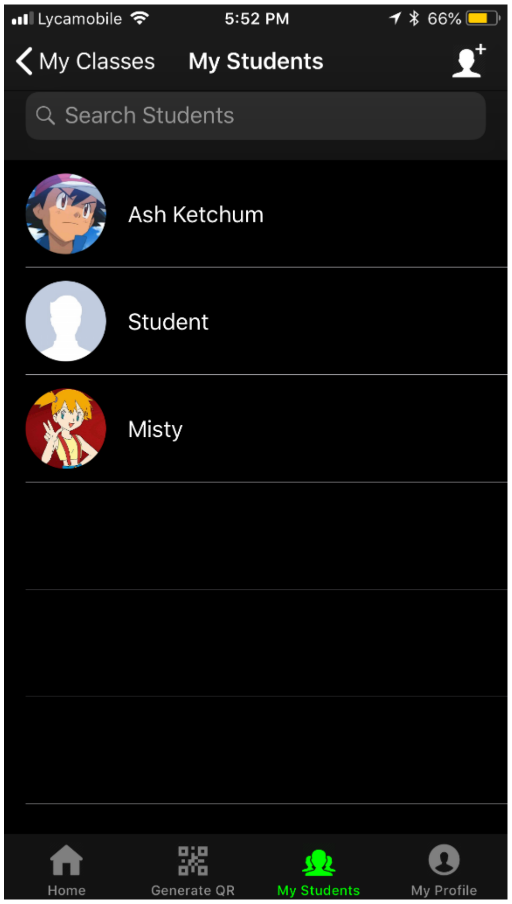
Next, the classroom's location is checked with the current users position based on the classroom size set by the tutor. Since GPS coordinates are just 2D points I used them to find the distance between the classroom's GPS Coordinates and Students Current GPS Coordinates. If the distance if within the classroom's size, only then the app proceeds further. In order to deal with the common issue GPS drift, I added a small amount of offset to the classroom sizes while check the students range.

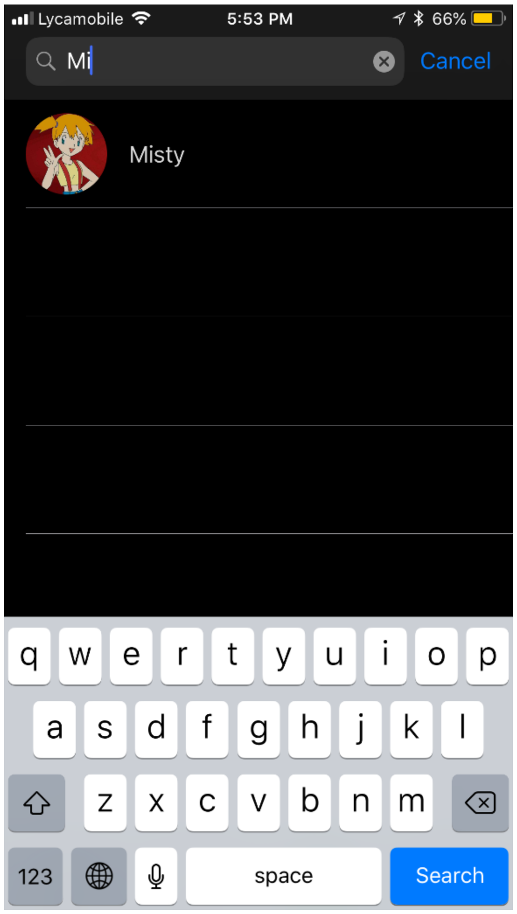
For example, let's say tutors set a classroom size as small (10x10 meters) and Classroom GPS Coordinates are (0,0). If the students GPS coordinates are (0,5) or (0,14), then he/she will get the attendance because for a 10x10 meter class after adding an offset of 5 meters, the student must be within 15 meters radius to be able to get the attendance. And in case it is some (0, 17) or (99.89) then he/she will not get the attendance. To calculate the GPS Coordinate distances, there is a simple method that I used which can be seen in the following snippet.



```
// Check if user is within classroom
func isClassroomInRange(longitude : String, latitude : String, classroomSize : String) -> Bool{

    let classRange : Double = getClassroomRangeFor(classSize: classroomSize)
    let classLongitude = Double (longitude)!
    let classLatitude = Double (latitude)!

    let myLocation = CLLocation(latitude: myLatitude, longitude: myLongitude)
    let classLocation = CLLocation(latitude: classLatitude, longitude: classLongitude)

    let distance = myLocation.distance(from: classLocation)

    print("Class Range in meters : \(classRange).\nClass Location lat:\(latitude), long:\(longitude).\n
        Distance between student and classroom: \(distance)")

    // Check if you are in classroom range
    if distance <= classRange{
        print("Classroom in range")
        return true
    }

    // If class is not in range
    else{
        print("Classroom is not in range")
        self.showAlertAndResumeScanner(Title: "Not in Range Error", Message: "You are not in range of
            classroom. Please go closer to the class to get the attendance")
        return false
    }
}
```

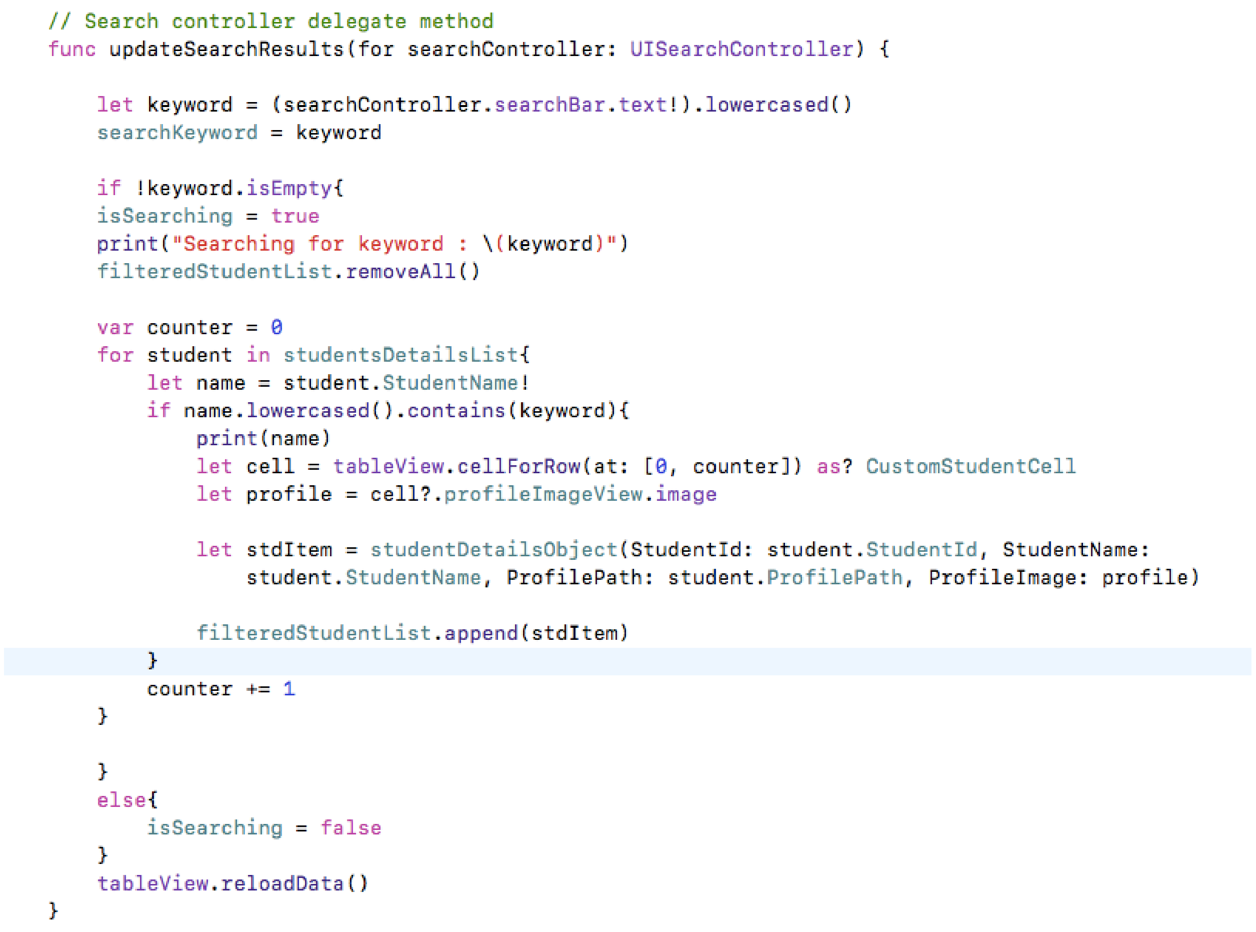
My Students (Tutor):





The 3rd tab in the Tutors account is the My Students tab. Here tutor can see all the students that are enrolled in this class. Tutor also can add new students to this class by tapping on Add Student button in the tabbar.

I also add **UISearchController** to this view controller and hooked up to the **tableView**. Tutor can type the name of the student that they are looking for to fetch them. As soon as the text in the search bar changes, the **tableView** starts pushing all the students whose name has that keyword. This functionality is implement by using the **UISearchControllerDelegate** which is shown in the following snippet.



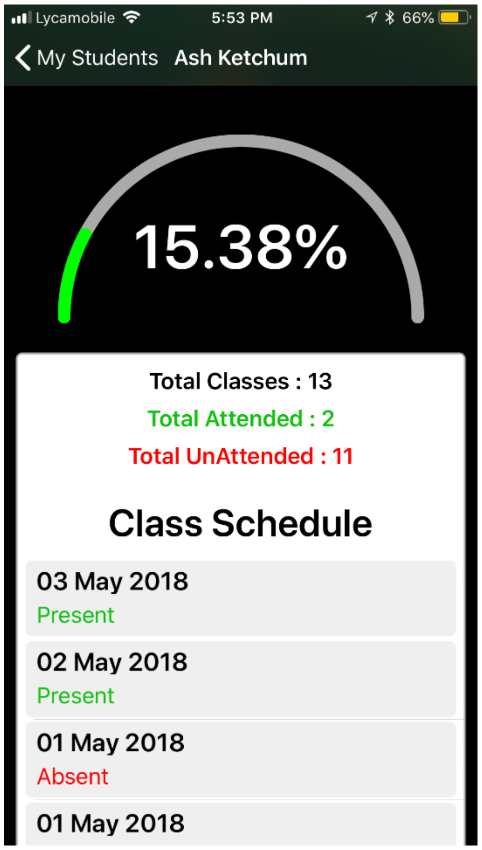
```swift
// Search controller delegate method
func updateSearchResults(for searchController: UISearchController) {

    let keyword = (searchController.searchBar.text!).lowercased()
    searchKeyword = keyword

    if !keyword.isEmpty{
    isSearching = true
    print("Searching for keyword : \(keyword)")
    filteredStudentList.removeAll()

    var counter = 0
    for student in studentsDetailsList{
        let name = student.StudentName!
        if name.lowercased().contains(keyword){
            print(name)
            let cell = tableView.cellForRow(at: [0, counter]) as? CustomStudentCell
            let profile = cell?.profileImageView.image

            let stdItem = studentDetailsObject(StudentId: student.StudentId, StudentName:
                student.StudentName, ProfilePath: student.ProfilePath, ProfileImage: profile)

            filteredStudentList.append(stdItem)
        }
        counter += 1
    }

    }
    else{
        isSearching = false
    }
    tableView.reloadData()
}
```

Tapping on any student will open the attendance details of that particular student. This details page is similar to that of the **My Attendance** page on the Students side with few minor changes. All those things will be explained in the next section.

My Attendance (Students/ Tutors):

This view is common to both Tutor and Students. However, there are few minor changes based on the user type.

This view will show all the attendance details of the current student in the selected class. It has a cool Attendance meter which animates to the attendance percentage of the student. I used **CAShapeLayer's** to create the Meter and used **CABasicAnimations** to animate the percentage changes.



Total Classes : 13
Total Attended : 2
Total UnAttended : 11

Class Schedule

03 May 2018
Present

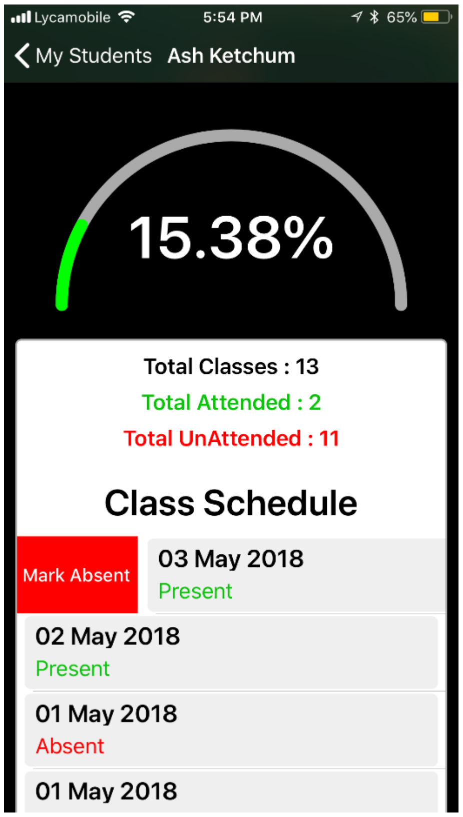
02 May 2018
Present

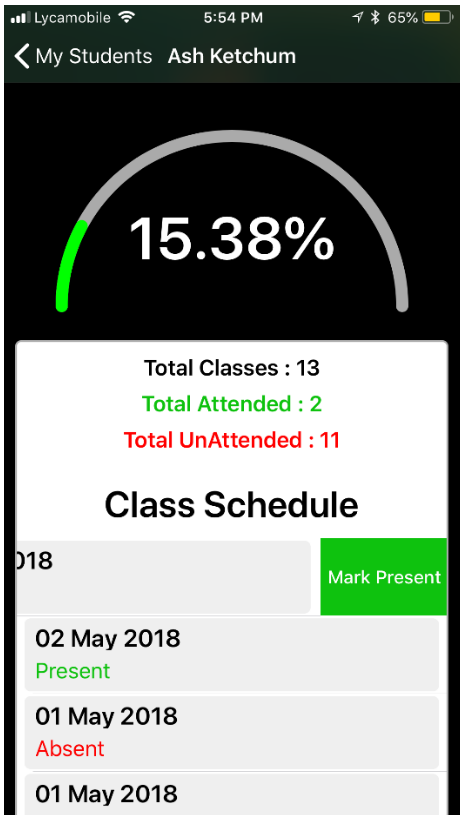
01 May 2018
Absent

01 May 2018

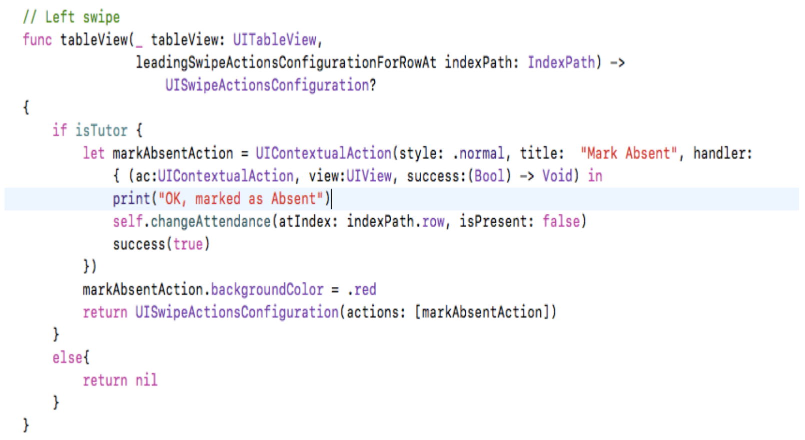
Apart from the attendance percentage, the app also shows other attendance counts and a **TableView** with **Custom Cell** showing the Class Schedule with student's attendance. Students can only view their attendance for different days of classes, but Tutor have an additional feature.

Tutors have the access to modify the attendance of the student on any scheduled class. I implemented left and right **Swipe** feature on **TableView** to mark student as Absent or Present.

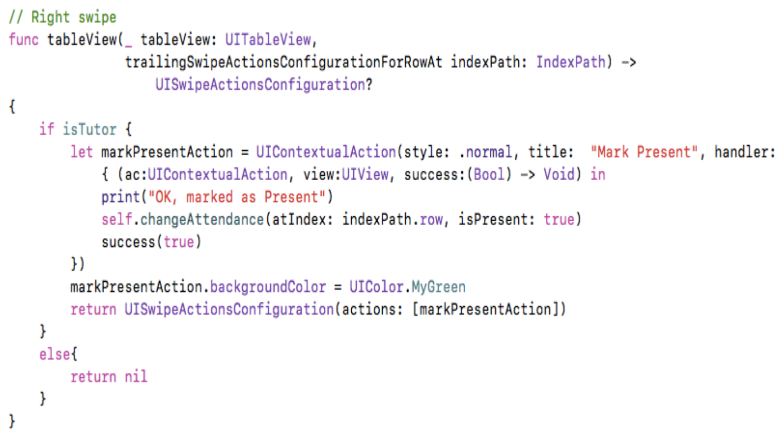




These **swipe gestures** will be enables only if the user type is tutor. I used the following two functions to implement this feature.



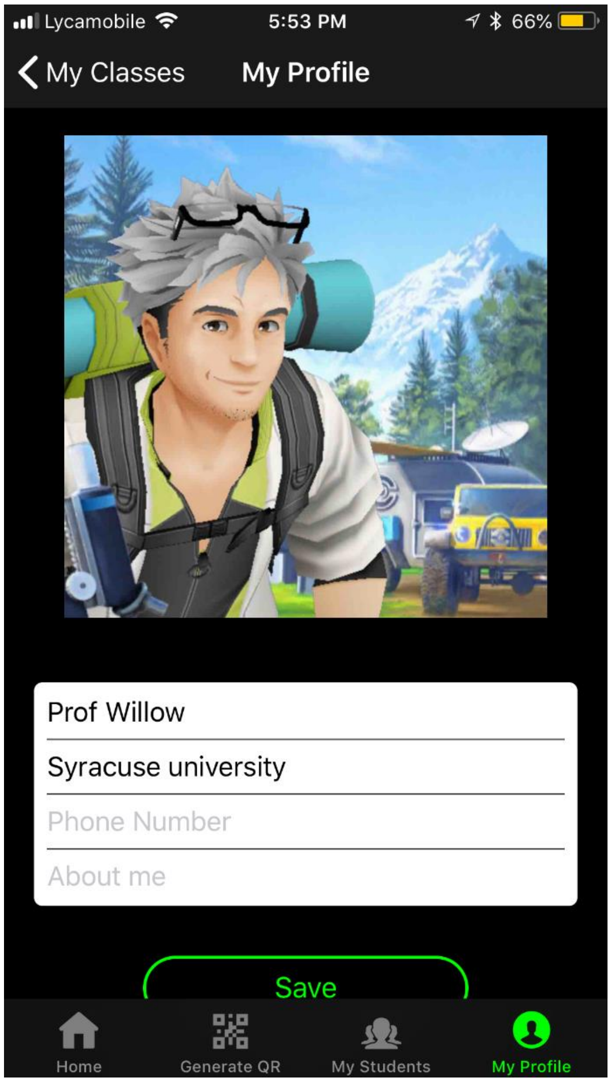
```swift
// Left swipe
func tableView(_ tableView: UITableView,
               leadingSwipeActionsConfigurationForRowAt indexPath: IndexPath) ->
                   UISwipeActionsConfiguration?
{
    if isTutor {
        let markAbsentAction = UIContextualAction(style: .normal, title:  "Mark Absent", handler:
            { (ac:UIContextualAction, view:UIView, success:(Bool) -> Void) in
            print("OK, marked as Absent")
            self.changeAttendance(atIndex: indexPath.row, isPresent: false)
            success(true)
        })
        markAbsentAction.backgroundColor = .red
        return UISwipeActionsConfiguration(actions: [markAbsentAction])
    }
    else{
        return nil
    }
}
```



```swift
// Right swipe
func tableView(_ tableView: UITableView,
               trailingSwipeActionsConfigurationForRowAt indexPath: IndexPath) ->
                   UISwipeActionsConfiguration?
{
    if isTutor {
        let markPresentAction = UIContextualAction(style: .normal, title:  "Mark Present", handler:
            { (ac:UIContextualAction, view:UIView, success:(Bool) -> Void) in
            print("OK, marked as Present")
            self.changeAttendance(atIndex: indexPath.row, isPresent: true)
            success(true)
        })
        markPresentAction.backgroundColor = UIColor.MyGreen
        return UISwipeActionsConfiguration(actions: [markPresentAction])
    }
    else{
        return nil
    }
}
```

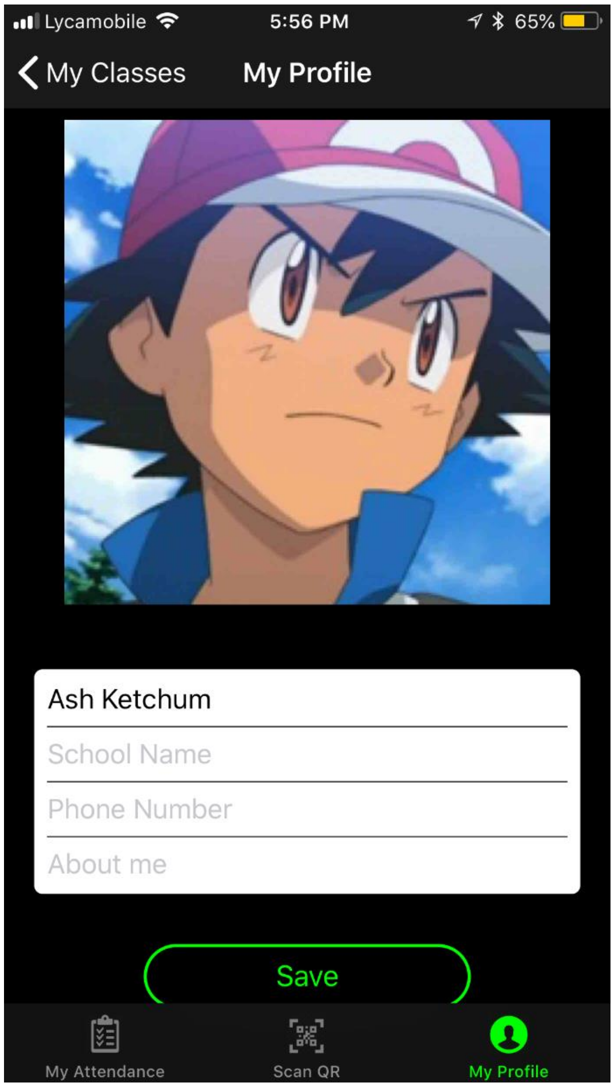
My Profile (Common):

Finally, the My Profile Tab which is common to both Tutors and Students. This tab allows the users to modify their profile information. This tab again uses the same **UIImagePicker** functionality to pick the image from the Photo Library. It also has few other user information textFields which are not mandatory. However, User's name field is mandatory.



Prof Willow

Syracuse university



Ash Ketchum

Once the save button is clicked, the image is compressed and uploaded to the Firebase Storage and the other user information will be update to the users node in the Firebase Database.

There is also a logout button which can be used to log out of the account.

Other Features

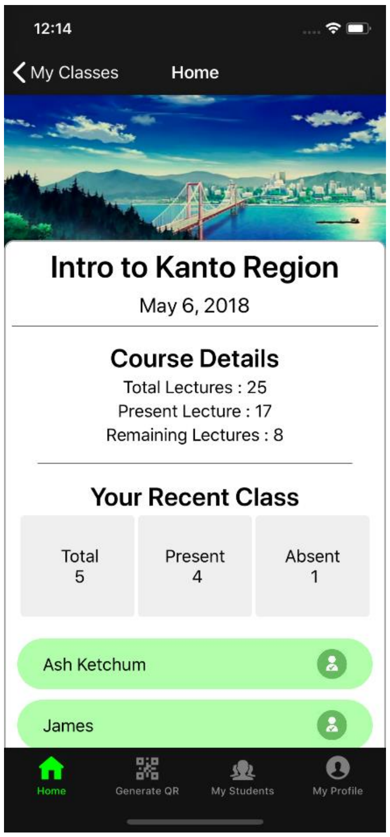
Offline usability:

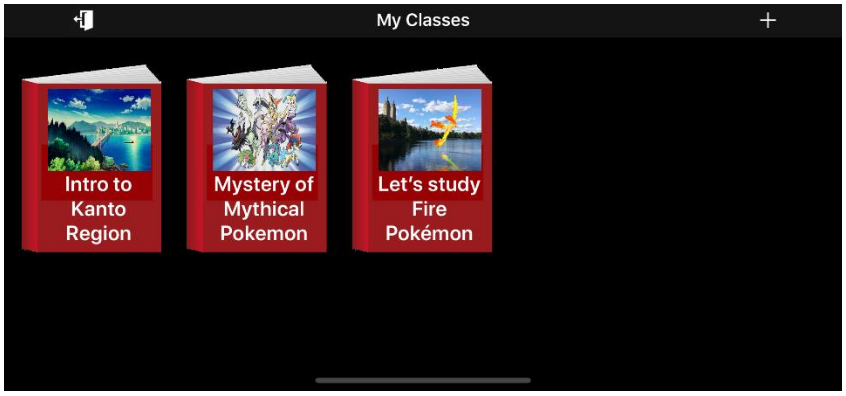
My app is enabled to work in the **offline** as well. All the changes made to the app will be stored in the local **snapshot** of **Firebase** when there is no network. So, any changes made could be seen right away since the data is loaded from the local storage. Once the network is up, all these changes will be updated to the Firebase. I implemented this by enabling the following line of code.

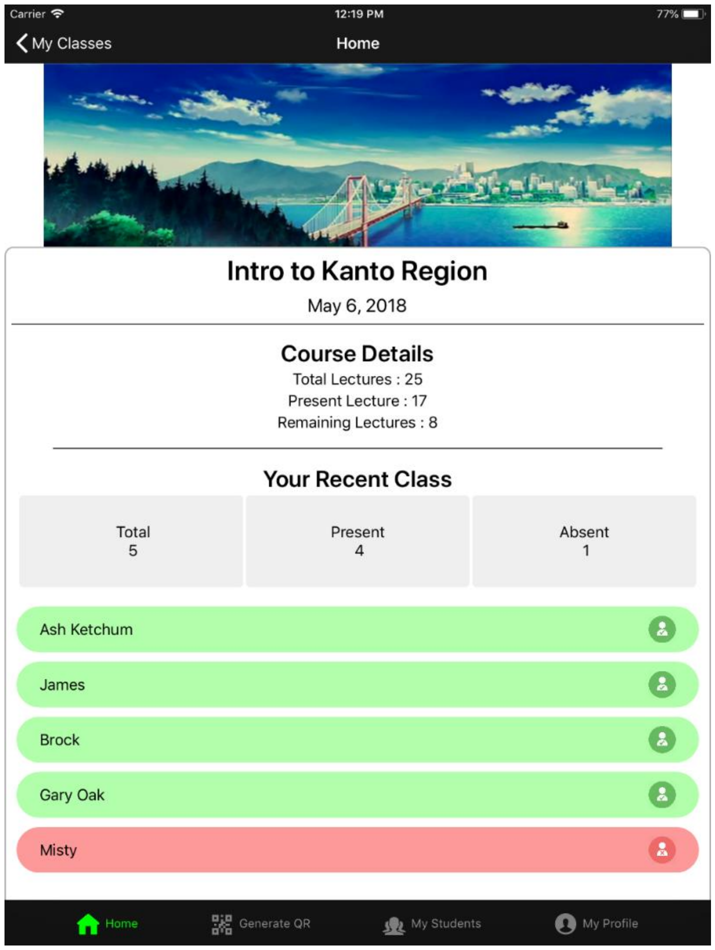
```
Database.database().isPersistenceEnabled = true
```

Multi-Screen Size and Orientation support:

My app supports all screen sizes and device orientations. I used auto layouts and constraint multipliers to make it usable across any device in any orientation. All the screenshots in the above examples are taken from iPhone 6 Plus. The following screenshots are from iPhone X and iPad 9.7 inch.







Carrier 12:19 PM 77%

My Classes | Home

Intro to Kanto Region

May 6, 2018

Course Details

Total Lectures : 25

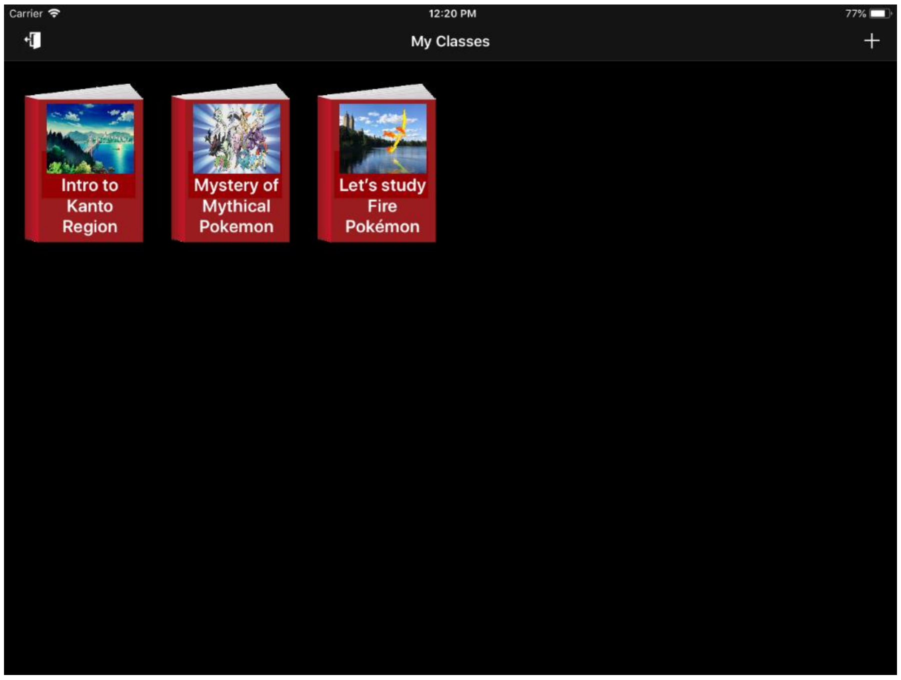
Present Lecture : 17

Remaining Lectures : 8

Your Recent Class

Total	Present	Absent
5	4	1

- Ash Ketchum
- James
- Brock
- Gary Oak
- Misty



Carrier 12:20 PM 77%

My Classes