iOS Application Development Report

Version 1.0 approved

NIRO Solutions

Romario Renée	**ID#: 180903**	**180903@gist.edu.cn**
Asharia Nurse	**ID#: 180910**	**180910@gist.edu.cn**
O'shara Mason	**ID#: 180911**	**180911@gist.edu.cn**
Nigel Francis	**ID#: 180925**	**180925@gist.edu.cn**

Course Instructor: **Dr. Thomas Canhao Xu**

Course: **SWEN3000 – iOS Development**

Date: **June 10, 2019**

Table of Contents

1. Introduction **2**
Purpose 2
1.2. Document Conventions 2
1.3. Intended Audience and Reading Suggestions 3
1.4. Product Scope 3

2. Overall Description **4**
2.1. Product Perspective 4
2.2. Product Functions 5
2.3. User Classes and Characteristics 6
2.4. iOS Environments 7
2.5. Design and Implementation Constraints 7

3. External Interface Requirements **8**
3.1. User Interfaces 8

4. Feature Breakdown **13**
4.1. News & Shuttle Stands 13
4.2. Login & Registration 16
4.3. Settings 21
4.4. Push notifications 23
Responsibilities 26

1. Introduction

1.1. Purpose

The purpose of this document is to build an online system to manage shuttles, students, driver and provide reports and analysis to improve the shuttle service management.

This iOS application for the UWI Shuttle Service is created with the focus of allowing students and staff to utilise the shuttle services and receive the latest information regarding the shuttles. Students and staff also have the ability to personalize the application with a range of options from theme selection to biometric security.

1.2. Document Conventions

This document uses the following conventions.

DB - Database

UWI - University of the West Indies

FB - Firebase

1.3. Intended Audience and Reading Suggestions

This project is a prototype for the UWI Shuttle Service system and it is restricted within the UWI CIIT premises. This has been implemented under the guidance of teachers at GIST / UWI CIIT.

This document is also intended for the developers of the project allowing them to get a deep dive into what the software is aimed at achieving, how it was architected, the role it plays with its other components and how it was developed.

1.4. Product Scope

The purpose of this iOS Application which is a component of the Shuttle Service management system is to:

- Improve the UWI Shuttle Service System Student Experience
- Provide reliable and timely information regarding the app
- Allow only registered students access to the shuttle
- Make accessing the shuttle much easier for students & staff.

The system is a native iOS application that is easy-to-use for students frequenting the shuttle. The system is based on a non-relational database called Firebase (FB). We will be aiming to provide the best experience for students and staff personnel as they use the application on a daily basis to allow for maximum efficiency.

2. Overall Description

2.1. Product Perspective

This product is one (1) part of three (3) different components that are all integrated to form a solution that would replace the physical log book and the tedious verification process drivers have to do on students. It is designed to be used without any heavy dependencies from the internet constantly.

An iOS shuttle service system such as this will be storing information such as:

- **Shuttle Data**

This includes start and end time for every route, breaks, notes and coordinates for boarding. This information is used to educate the user about the various routes they have available.

- **Student Data**

This includes basic information about the students such as their full name, photo and email address. This information is used to help identity a student when their QR code is being scanned just before a shuttle trip.

- **News Data**

This consist of basic information such as a news headline and content. This information is used to notify the students or staff user about any issues, delays etc regarding the shuttle routes so that they can be prepared.

2.2. Product Functions

The system focuses on executing the following functions efficiently in order to allow the student and staff users to have a smooth experience when using the app.

- Login
- Registration
- Forgot Password
- Shuttle Information
- Basic Student Information
- News Information
- Biometric App Unlocking
- Language Selection
- Theme Options
- Push Notifications
- Email Verification

2.3. User Classes and Characteristics

Users of the system should be able to retrieve information regarding the shuttle routes and news. Users should also be able to use the native maps (Apple Maps) to retrieve directions to the shuttle boarding and departure areas.

The student and staff users should be able to do the following functions.

- View Shuttles
    - ➢ View list of shuttles and their relevant information

- View News
    - ➢ Read the latest news regarding the shuttle

- Login Management
    - ➢ Sign into the application
    - ➢ Request a forgot password
    - ➢ Register with the application using Email Verification

- Personalize the App
    - ➢ Change the theme of the app depending on their preference
    - ➢ Change the language from English, French or Chinese

- Security
    - ➢ Logout of the application which removes all of their from the app locally
    - ➢ Turn on biometrics forcing anyone to authenticate every time the app is open

2.4. iOS Environments

The UWI Shuttle Service iOS Application was tested in iOS 11 to 12 environments and those results provided us with the best version of iOS this application can run on. It is compatible with both 11 and 12 but on writing this document iOS 13 is due to be released shortly and we will need to test that.

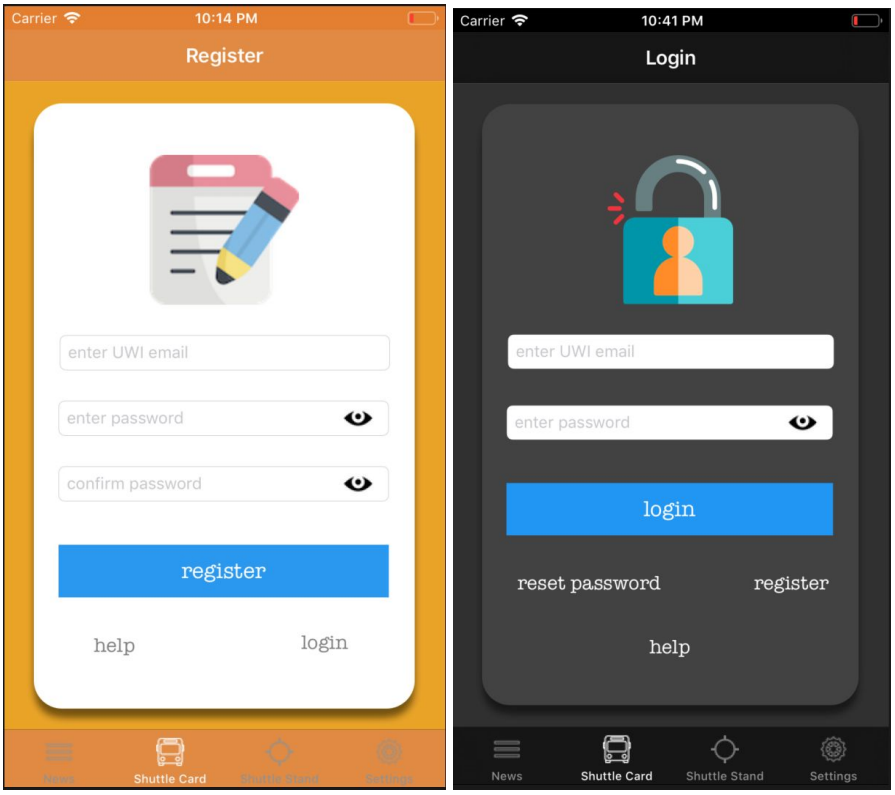
2.5. Design and Implementation Constraints

The project is susceptible to a few constraints due to the location of the University and the method used for coding. The application will mainly be used and is targeted to countries such as Barbados, Jamaica & Trinidad. However, The University of the West Indies has connections with other universities around the world. One of them being Gaobo in Suzhou, China. This specific location brings a few constraints, in China, Google services are blocked which means the web application will not be able to fetch any data. This provides a very serious implementation challenge as it halts the use of the system totally unless a VPN is used.

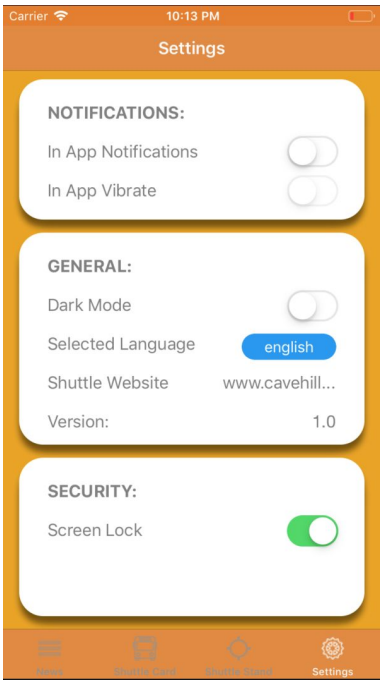
One of the other constraints relates to the design of the application. Keeping the branding of the application consistent as users interact with different devices, they were a few changes in the design that had to be executed. Instead of trying to get the application to look exactly the same across platforms, a more industry standard approach was taken. Instead of looking the same as the android exactly, the iOS user would have similar branding color schemes and their interface would relate to what they are accustomed to in other iOS applications.

3. External Interface Requirements

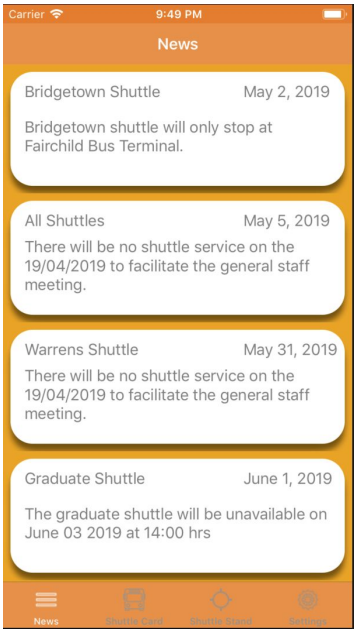
3.1. User Interfaces



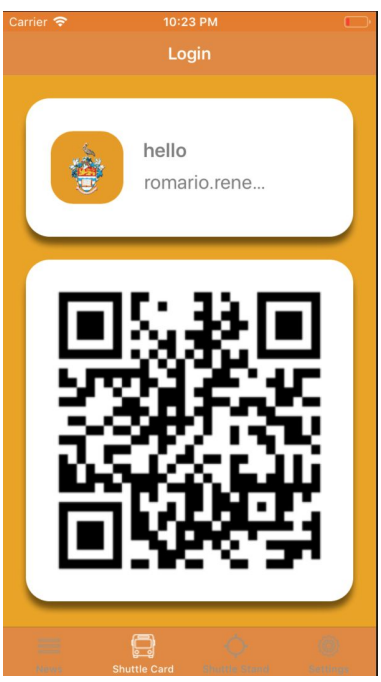
Login Interface for both light mode and dark mode

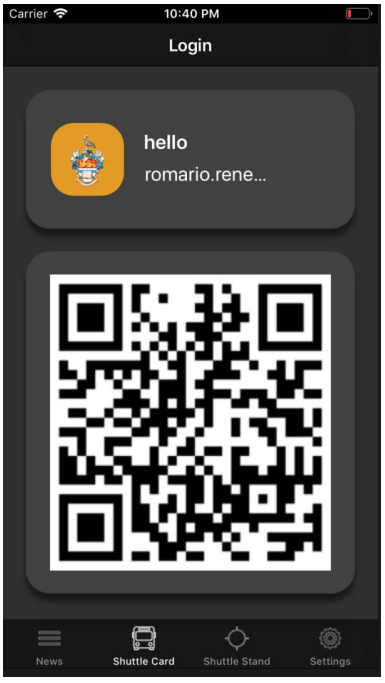


Settings Interface showing security feature, theme selection, app version, language options and notification settings.



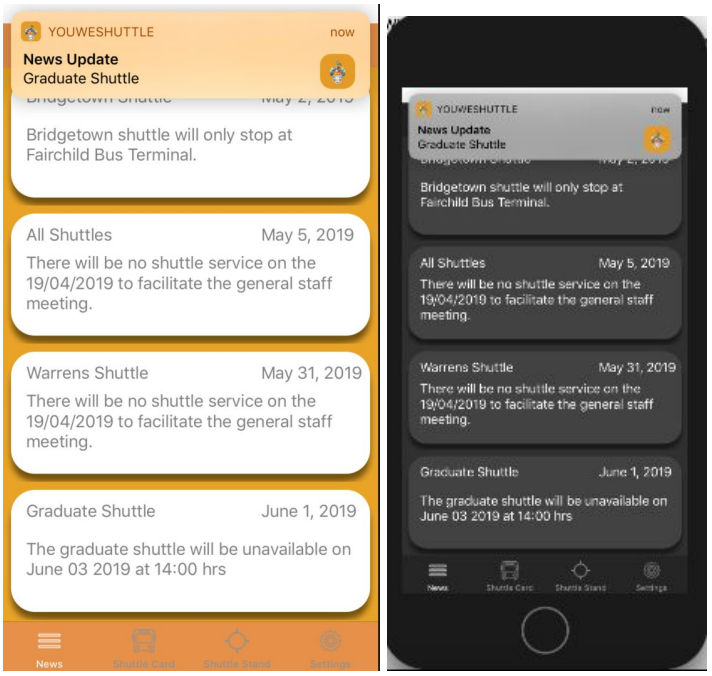
News Interface showing a list of news articles arranged by date in this screen the date is reversed none the less sorted.



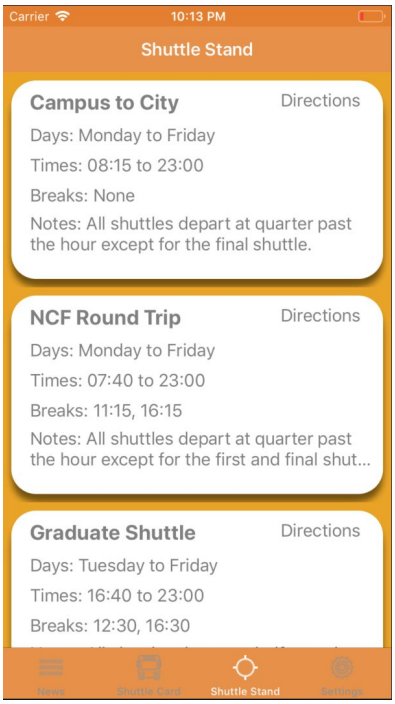


Student QR Code Interface, after the student has signed in successfully this interface is shown

Application icon with a badge to represent a new push notification on the Home Interface of an Apple iOS device.



News Interface with a Push Notification sent to all users and also with dark mode theme.



Shuttle Stand Interface showing a list of available shuttles and its relevant information.

4. Feature Breakdown

4.1. News & Shuttle Stands

The implementation of the News and Shuttle Stands was similar, as they both dealt with reading data from the database, listening for real-time updates and error handler listeners. Firebase provides the listener functions so it was only required to place the specific variables into the correct position for the function to work correctly. As soon as changes were made to the database, they would appear on the user's devices. The variables such as the name of the shuttle and shuttle stand, notes, days, breaks etc were stored as Strings and appended to an ArrayList. However, there were slight variations in the creation of the datetime and, location and directions features.

```
// Added by Romario
func getDocumentSnapshot(){

    //need to swap the database based on user language had to duplicate code as had an issue onload
    let currentLanguage = defaults.string(forKey: Keys.keyDatabaseNewsLanguage) ?? "NEWS"

    Firestore.firestore().collection(currentLanguage)
        .addSnapshotListener { querySnapshot, error in
            guard let snapshot = querySnapshot else {
            return

            }

            snapshot.documentChanges.forEach { diff in
                if (diff.type == .added) {

                    self.sendNotificationFromDevice(title: "News Update", subtitle: "", body:
                        diff.document.data()["title"] as! String)

                }
                if (diff.type == .modified) {

                    self.sendNotificationFromDevice(title: "News Update", subtitle: "", body:
                        diff.document.data()["title"] as! String)
                }

            }
    }

}
```

Swift UITableView was used to model the data, each document in the database collection was counted and the corresponding number of cells were produced in the table view. The data from the fields in the document was displayed in a table cell, which was then styled using the customized card layout class.

```
func createShuttleStandArray(){
    let currentStandLanguage = defaults.string(forKey: Keys.keyDatabaseStandLanguage) ?? "SHUTTLE_STANDS"

    dbRef.collection(currentStandLanguage).getDocuments { (querySnapshot, error) in
        if let error = error{
            print("Error retrieving Shuttle Stand info: \(error)")
        }else{
            print("Loop through snapshot")

            for document in querySnapshot!.documents{
                print("append start")

                // Loop through the snapshot and create our Shuttle Stand Object

                let s_breaks = document.data()["breaks"] as? String
                let s_days = document.data()["days"] as? String
                let point = document.data()["directions"] as? GeoPoint

                let fullDirections = String(point!.latitude) + "," + String(point!.longitude)

                let s_name = document.data()["name"] as? String
                let s_note = document.data()["note"] as? String
                let s_time = document.data()["times"] as? String

                let ShuttleStand1 = ShuttleStand(breaks: s_breaks!, days: s_days!, directions: fullDirections,
                    name: s_name!, note: s_note!, times: s_time!)

                print("Update Shuttle Stands")

                // Append our Shuttle Stand object to our array of Shuttle Stands
                self.ShuttleStands.append(ShuttleStand1)
            }

            print("tableView reload Data")
            self.tableView.reloadData()
        }
    }
```

DateTime Feature

The date of the News update is stored as a timestamp in Firebase database, to append it to the ArrayList, it first had to be converted to the appropriate date format, then to a String before appending.

Location and Directions Feature

Retrieving the user's current location and providing him/her with directions to the nearest shuttle stand was one of the more challenging features to implement. It required separating the GeoPoint into latitude and longitude and storing those values in variables which would be used to fetch directions from the map and location service.

For the directions element of this feature, the user's current location was referenced to the location of the shuttle stand, then both locations were stored as variables. A function was created to prompt the maps to display the route from the current location to the shuttle stand location. The function utilized the information stored in the variables and a URL to display the required information.

4.2. Login & Registration

The Login and Registration of users were done through functions the Firebase Authentication module provides. Currently, the UWI Shuttle Service application is only available for students and staff members of the UWI Cavehill Campus.Therefore checks were put in place to only allow users with valid Cavehill student or work emails to successfully register or login. Other checks were also put into place such as tests to ensure all emails entered are valid email addresses, password strength, passwords match and text fields are not empty.

```
func createNewUser(registerEmail:String, registerPassword:String){

    Auth.auth().createUser(withEmail: registerEmail, password: registerPassword, completion: {(user, error) in

        if(error == nil && user != nil){

            Auth.auth().currentUser?.sendEmailVerification { (error) in

                if error == nil{

                    self.hideProgressBar()

                    self.showAlert(message: LocalizationSystem
                        .sharedInstance.localizedStringForKey(key: "registerRegistrationSuccess", comment: ""))

                    self.registerEmailText.text = ""
                    self.registerPasswordText.text = ""
                    self.registerConfirmPasswordText.text = ""

                    self.navigationController?.popViewController(animated: true)

                    // This delays the Registration Successful Alert because the dismiss was remove the alert
                    // to quickly
                    DispatchQueue.main.asyncAfter(deadline: DispatchTime.now()+6){
                        self.dismiss(animated: true, completion: nil)
                        // Removes both the registering loading bar but also the pop up for successfull

                        self.navigationController?.popViewController(animated: true)
                        self.dismiss(animated: true, completion: nil)

                    }

                }else{

                    self.hideProgressBar()
                    self.showAlert(message:error!.localizedDescription)
```

```
                    self.showAlert(message:error!.localizedDescription)
                    self.registerPasswordText.text = ""
                    self.registerConfirmPasswordText.text = ""

                }

            }

            do{
                try Auth.auth().signOut()

            }catch{

                self.showAlert(message: LocalizationSystem
                    .sharedInstance.localizedStringForKey(key: "errorSignOut", comment: ""))
                self.registerPasswordText.text = ""
                self.registerConfirmPasswordText.text = ""

            }
        }else{

            self.hideProgressBar()
            self.showAlert(message:error!.localizedDescription)
            self.registerPasswordText.text = ""
            self.registerConfirmPasswordText.text = ""
        }
    }

    )

}
```

Upon successful registration, users will be sent a verification email from Firebase. Users will not be allowed to login unless they verify their email. If users attempt to login without verifying their email, a new verification email will be sent to their email address.

```
func signInUser(loginemail: String, loginpassword: String){

    Auth.auth().signIn(withEmail: loginemail, password: loginpassword, completion: {(user, error) in

        if(error == nil && user != nil){

            self.defaults.set(true, forKey: Keys.keyIsLogin)
            self.defaults.set(true, forKey: Keys.keyIsLogin)


            self.defaults.set(loginemail, forKey: Keys.keyUserEmail)

            self.performSegue(withIdentifier: "LoginShuttleCardSegway", sender: nil)

            self.hideProgressBar()

        }else{

            if self.currentuser?.isEmailVerified == false{

                self.currentuser?.sendEmailVerification(completion: {(error) in

                    if error == nil{

                        self.hideProgressBar()

                        self.showAlert(message: LocalizationSystem
                            .sharedInstance.localizedStringForKey(key: "loginVerifyEmailFirst", comment: ""))

                        self.loginPasswordText.text = ""

                    }else{

                        self.hideProgressBar()
                        self.showAlert(message:error!.localizedDescription)
                        self.showAlert(message:error!.localizedDescription)
                        self.loginPasswordText.text = ""
                    }

                })
            }

            self.hideProgressBar()
            self.showAlert(message:error!.localizedDescription)
            self.loginPasswordText.text = ""

        }

    })

}
```

In the event a user forgets his/her password, Firebase has a reset password function which would automatically send an email to the user's email address. Upon clicking that email a user will be prompted to enter their new password. After resetting their password, a user can then login immediately with their new credentials.

Each user will have their own unique QR code. QR codes will be created using the user's email address. Every time a user login to the application, a new QR code will be automatically generated using that email address.

4.3. Settings

The Language & Theme options in the iOS application allow for the staff and student users to personalize the application to meet their needs. Both of these options utilize a User Defaults method which saves the state and allows the program to recall them every time the user visits the app.

In the photo below you will notice we also use these user defaults for notifications, vibrate and screen lock mode.

```
//user defaults keys
struct Keys {

    static let keyNotification = "keyNotification"
    static let keyVibrate      = "keyVibrate"
    static let keyDarkMode     = "keyDarkMode"
    static let keyScreenLock   = "keyScreenLock"
    static let keyIsLogin      = "keyIsLogin"

}
```

```
let defaults = UserDefaults.standard

func checkUserPreferences() {

    let prefersNotification = defaults.bool(forKey: Keys.keyNotification)
    let prefersVibrate      = defaults.bool(forKey: Keys.keyVibrate)
    let prefersDarkMode     = defaults.bool(forKey: Keys.keyDarkMode)
    let prefersScreenLock   = defaults.bool(forKey: Keys.keyScreenLock)
    let prefersIsLogin      = defaults.bool(forKey: Keys.keyIsLogin)

    if prefersNotification                      {notificationSwitch.setOn(true, animated: false)}
    if prefersVibrate && prefersNotification    {vibrateSwitch.setOn(true, animated: false)}
    if !prefersScreenLock                        {screenLockSwitch.setOn(false, animated: false)}
    if prefersIsLogin                           {logoutButton.isHidden = false}
    if !prefersNotification                     {vibrateSwitch.setOn(false, animated: false)
        vibrateSwitch.isEnabled = false}

    if prefersDarkMode{

        darkModeSwitch.setOn(true, animated: false)
        Theme.current = DarkTheme()
        applyTheme()

    }
}
```

The dark mode uses a switch and based on the toggle state when clicked, we save the option and apply the dark mode style into the theme. With the dark mode, industry standards for iOS applications were followed (Apple has made Dark Mode Official) not only style wise but implementation wise also.

For the language however, since multiple languages are available, this was implemented using a Popup View Controller. Options were fed into the data for the picker to allow the user to select - English, French, Spanish, Portugues and Chinese. Changing the language of an application is something that should not be done by mistake so a confirmation was added (Using a UIAlertController) asking the user if they are sure regarding the change they were about to make. Onclick the application would save this preference to the User Defaults as stated above and then restart the app. On restart when the database is fetched

this language preferenced is passed along so that instead of fetching the default English data, the application would fetch the language selected.

For the security biometrics section, a toggle switch similar to the dark mode is used, it also similar to dark mode also. When selected the preference is saved to the user defaults. On reload of the application if the screen lock was previously selected by the user a call to "**self**.authenticationWithTouchID()" is made. This allows the application to leverage the iOS touch ID security feature. Once verified then and only then will the user be able to access the app.

4.4. Push notifications

Similar to the Android UWI Shuttle Service Application and based on the requirements of this project, the ability to have push notifications was a must. However, one of the drawbacks faced was that an Apple Developer Account was required before it could have been enabled into the system. As a result, local notifications were used instead. This worked very similar to how we intended the Push notifications to be.

```
snapshot.documentChanges.forEach { diff in
    if (diff.type == .added) {

        self.sendNotificationFromDevice(title: "News Update", subtitle: "", body:
            diff.document.data()["title"] as! String)

    }
    if (diff.type == .modified) {

        self.sendNotificationFromDevice(title: "News Update", subtitle: "", body:
            diff.document.data()["title"] as! String)
    }

}
```

Since the notifications the users would be receiving were only about the news, a custom function was added to the initial fetching of the data that allowed for an action to be executed when the News table in the database was updated with new information. This custom function was easy to use seeing that it was provided by (FB) Firebase as a database listener.

```
func applicationDidBecomeActive(_ application: UIApplication) {
    // Restart any tasks that were paused (or not yet started) while the application was inactive. If the
        application was previously in the background, optionally refresh the user interface.
    UIApplication.shared.applicationIconBadgeNumber = 0
}
```

On update of the database, the local notification function would be called. Just before this, however, because the user has extensive control over notifications and how they receive them, the device fetches the users stored preferences regarding notifications. If the user allows for notifications to be sent, then and only then would the notification be sent off, then the badge count would be updated to represent new data and a vibration would be triggered. Again, in regards to the vibration, the user has control to have this enabled or

disabled and only based on their preference to have vibrations would the vibration function be executed.

Responsibilities

Name	Responsibility
Asharia Nurse	Documentation, Login, Register, Forgot Password, QR Code
Nigel Francis	Documentation, Settings, Language, Biometrics, Themes, Login, Forgot Password
O'shara Mason	Documentation, News, Shuttle Stands, Location
Romario Renée	Documentation, Cards, System Design, Push Notifications, Register, Shuttle Stands, Location