iCompanion – Report

Abstract:

Student life is hectic and also enjoyable. Students gain knowledge through the assignments of various modules of their university. But as the assignments start bundling up, it becomes hectic time for the students. It is important to have a study – life balance to take care of their mental health. Mental health is a very important aspect that must be taken care of to have a peaceful life. But these days, we noted that the students don't concentrate to maintain a good mental health by concentrating over their assignments. This might become a serious problem for their health and could cause health hazards. To address this problem of maintain a good mental health, we have designed this application that will keep track of their mood over the day and give recommendations based on their mood. This will suggest the students to have a refreshing outing if they are sad and cherish their happiness if they are happy. This app also lists the events that are hosted by the departments of the university and encourages the students to participate in those events and also appreciate to attend them when they feel sad or worried so that could socialise with people and make up the mood.

Application Architecture:

The application consists of mainly three components:

- The admin component to add date into the application
- The client component that will fetch the information and display to the users
- The controller component that will control and manipulate the data flow between the various components

All the features of the application were developed to serve the features for any of the component stated above. The functionalities of the components described above are explained below:

Admin Component:

The admin component is used to add the event details to the application. This serves as a single entry point through which data about the events can be entered in the application. This feature has various information to be entered that would be relevant for the event. When the data is entered and the user confirms his action, the data will be saved to the core data so that the data is persistent through the lifetime of the application.

Client Component:

The client component serves as the means through which the features of the application can be accessed. This is the component that will be visible for the end user and also contains all the features that the app provides. This will provide the user with features like viewing the events of the department of the university, viewing his favourite events, registering his interest towards the event, contacting the event organisers and getting the directions to the location of the event.

Controller Component:

The controller component is the main component of the application and is responsible for the proper functioning of the application. This component is responsible for populating the events that are related to the department of the user, authenticating the user, getting the data relevant to the event and its organisers, saving the user favourites and interests and all the other functionalities of the application.

Features of the application:

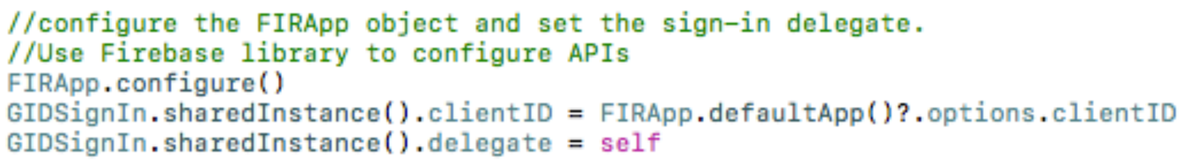
The main features of the application are:

- Google authentication using UCDConnect mail id to authenticate the student of UCD
- Feature to add event details to the application
- View the events that are happening in the university
- Call the event organiser
- Mail the event organiser
- View the location on map and get direction to event location from any location
- Set favourite events and view them
- Register interest by sharing an 'interested' or 'going' for the event
- Setting a reminder for the interested and going events
- Getting notification to get the mood of the user based on which recommendations could be given.

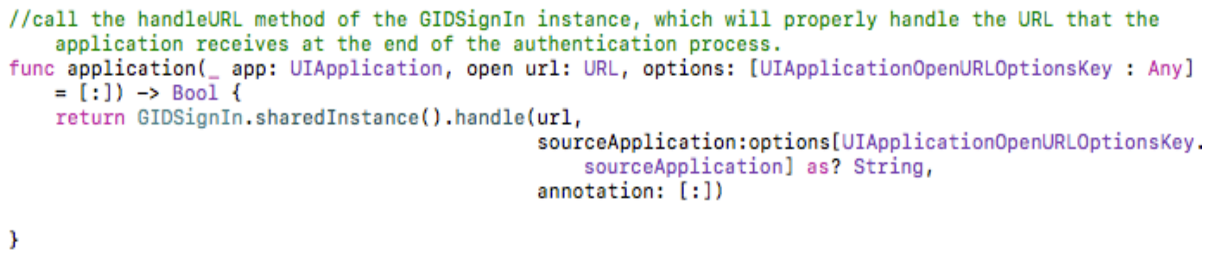
Frameworks implemented:

1. **Google sign In with Firebase**

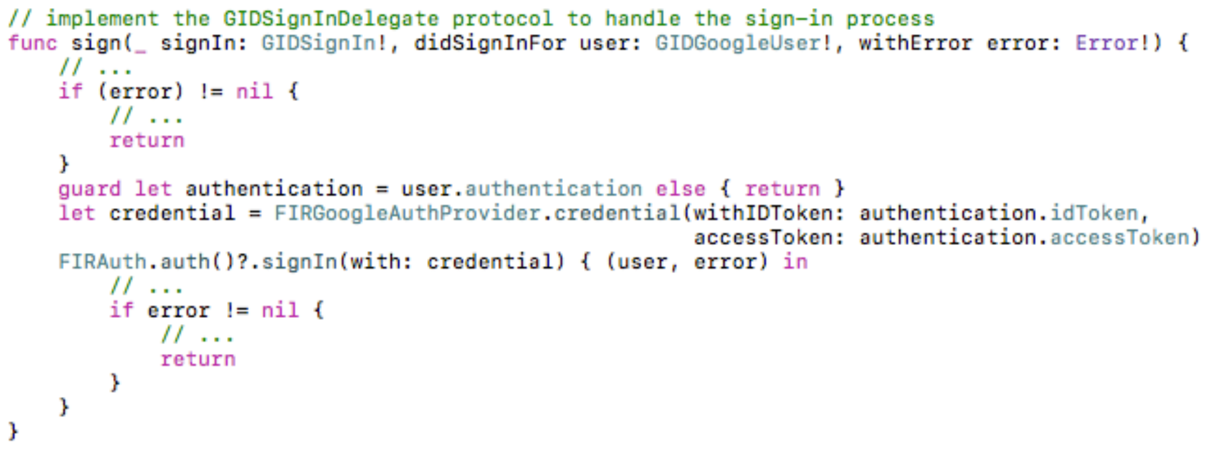
   Firebase Authentication is utilized to allow users to sign in to iCompanion app using email address and password sign-in, and with Google Sign-in as federated identity provider. As this is an university specific application, we have authenticated the users based on the university credentials.



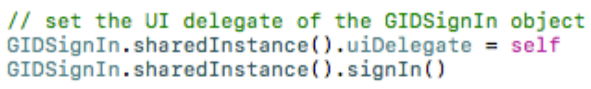
```
//configure the FIRApp object and set the sign-in delegate.
//Use Firebase library to configure APIs
FIRApp.configure()
GIDSignIn.sharedInstance().clientID = FIRApp.defaultApp()?.options.clientID
GIDSignIn.sharedInstance().delegate = self
```



```
//call the handleURL method of the GIDSignIn instance, which will properly handle the URL that the
    application receives at the end of the authentication process.
func application(_ app: UIApplication, open url: URL, options: [UIApplicationOpenURLOptionsKey : Any]
    = [:]) -> Bool {
    return GIDSignIn.sharedInstance().handle(url,
                                             sourceApplication:options[UIApplicationOpenURLOptionsKey.
                                                 sourceApplication] as? String,
                                             annotation: [:])

}
```



```
// implement the GIDSignInDelegate protocol to handle the sign-in process
func sign(_ signIn: GIDSignIn!, didSignInFor user: GIDGoogleUser!, withError error: Error!) {
    // ...
    if (error) != nil {
        // ...
        return
    }
    guard let authentication = user.authentication else { return }
    let credential = FIRGoogleAuthProvider.credential(withIDToken: authentication.idToken,
                                                      accessToken: authentication.accessToken)
    FIRAuth.auth()?.signIn(with: credential) { (user, error) in
        // ...
        if error != nil {
            // ...
            return
        }
    }
}
```



```
// set the UI delegate of the GIDSignIn object
GIDSignIn.sharedInstance().uiDelegate = self
GIDSignIn.sharedInstance().signIn()
```

2. **NotificationCenter**

   NotificationCenter framework is used to create a custom notification to read the mood of user thrice a day. Based on the result of calculation on these input, we judge the mood of the user and suggestion is given to the users based of the user's mood calculation. Suggestion is implemented using UIAlertController.

```
// adding Categories and Actions for Notification
let happy = UNNotificationAction(identifier: "happy", title: "Happy", options: .foreground)
let sad = UNNotificationAction(identifier: "sad", title: "Sad", options: .foreground)
let category = UNNotificationCategory(identifier: "category", actions: [happy,sad],
    intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])

//setting up notification property
let content = UNMutableNotificationContent()
content.title = "Hi buddy"
content.subtitle = "How's your day today"
content.body = "please let us know your mood to help you"
content.categoryIdentifier = "category"

// trigger specifies what are the conditions that notification is going fires
let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 1, repeats: false)
//creating request
let request = UNNotificationRequest(identifier: "timerdown", content: content, trigger: trigger)
UNUserNotificationCenter.current().add(request, withCompletionHandler: nil)
```

3. Messages Framework

The MFMailComposeViewController class is implemented to provide a standard interface for managing, editing, and sending an email message to Event Organiser when user **swipe left** in the particular table cell.

```
// as we have to present the mail view in the parent view, we have left the method in the controller
override func tableView(_ tableView: UITableView, editActionsForRowAt indexPath: IndexPath) ->
    [UITableViewRowAction]? {

    let shareAction = UITableViewRowAction(style: .normal, title:"Mail the Organiser") { (action:
        UITableViewRowAction!, indexPath: IndexPath!) -> Void in

        // obtain a configured MFMailComposeViewController instance
        let mailCompose = MFMailComposeViewController()
        mailCompose.mailComposeDelegate = self
        // setToRecipients() accepts an e-mail address strings,
        mailCompose.setToRecipients([self.model.event[indexPath.row].email!])
        //setting properties of an MFMailComposeViewController instance.
        mailCompose.setSubject("Reg: Inquiry regarding")
        mailCompose.setMessageBody("Hi,"+"\n"+"I would like to inquire about event "+"\(String
            (describing: self.model.event[indexPath.row].title!))"+". Could you please provide me more
            information about it?", isHTML: false)

        // check to make sure the device can send e-mail at this moment
        if MFMailComposeViewController.canSendMail(){
            //If it can, present the configured MFMailComposeViewController
            self.present(mailCompose, animated: true, completion: nil)
        }
        else{
            print("Error...!")
        }
    }
    return [shareAction]
}

// MFMailComposeViewControllerDelegate Method
func mailComposeController(_ controller: MFMailComposeViewController, didFinishWith result:
    MFMailComposeResult, error: Error?) {
    controller.dismiss(animated: true, completion: nil)
}
```

4. EventKit

EventKit is used to create reminders for the events that the users are interested in. These reminders will be created automatically when the users click **interested** or **going** button.

```swift
// Responds to button to add event. This checks that we have permission first, before adding the
// event
func addToCalender(eventDate: String, eventTitle: String) {

    //Adds event in calender
    let eventStore = EKEventStore()

    let formatter2 = DateFormatter()
    formatter2.dateFormat = "dd-MM-yyyy"
    let date2 = formatter2.date(from: eventDate)

    let startDate = date2
    let endDate = startDate?.addingTimeInterval(86400) // One day

    if (EKEventStore.authorizationStatus(for: .event) != EKAuthorizationStatus.authorized) {
        eventStore.requestAccess(to: .event, completion: {
            granted, error in
            self.createEvent(eventStore, title: eventTitle, startDate: startDate!, endDate: endDate!)
        })
    } else {
        createEvent(eventStore, title: eventTitle, startDate: startDate!, endDate: endDate!)
    }

}

// Creates an event in the EKEventStore. The method assumes the eventStore is created and
// accessible
func createEvent(_ eventStore: EKEventStore, title: String, startDate: Date, endDate: Date) {
    let event = EKEvent(eventStore: eventStore)

    event.title = title
    event.startDate = startDate
    event.endDate = endDate
    event.calendar = eventStore.defaultCalendarForNewEvents
    do {
        try eventStore.save(event, span: .thisEvent)
        savedEventId = event.eventIdentifier
    } catch {
        print("Error...")
    }
}
```

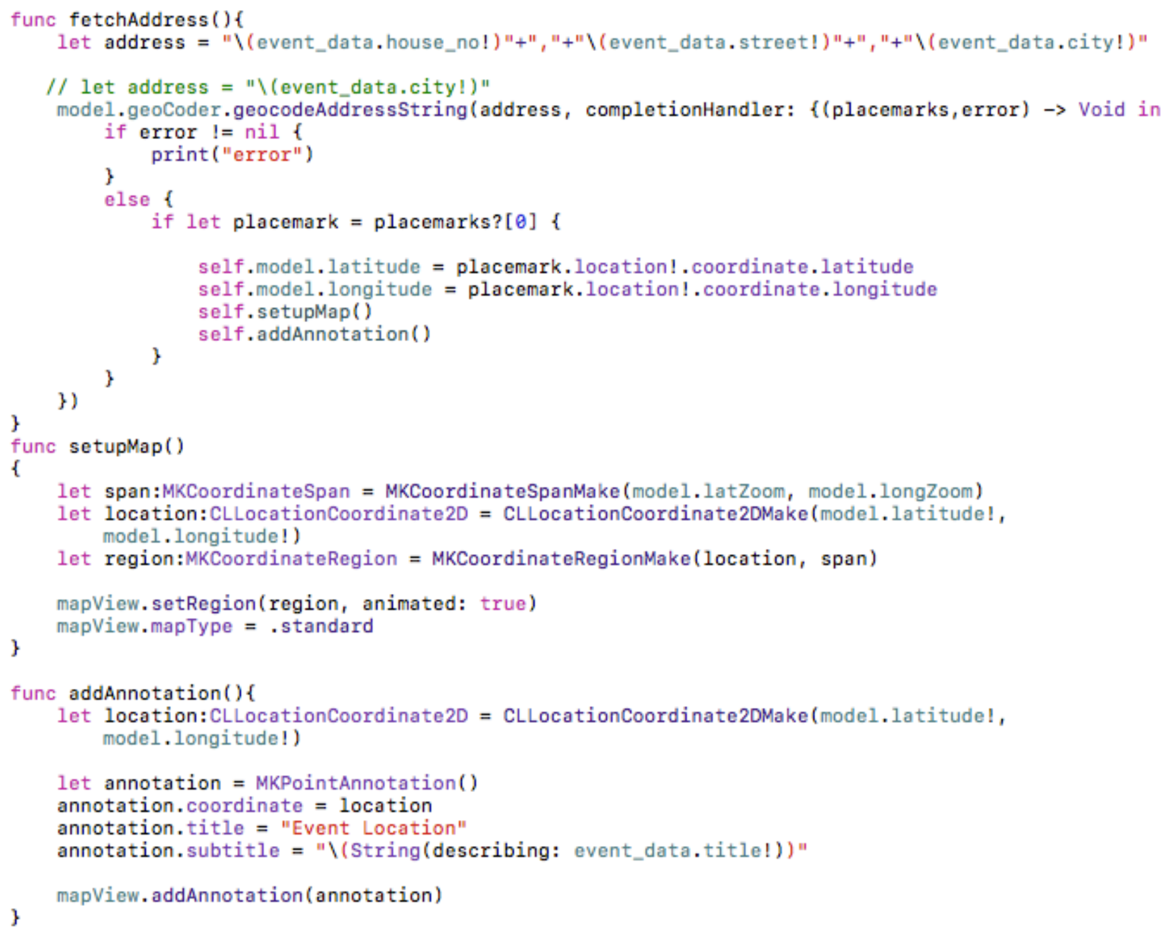
5. CoreData

CoreData is the backbone of the application. The CoreData framework is used in this application to store and retrieve the event details, to store and retrieve the user actions such as his favourites, interests and going events. It is also used to fetch the information that will pass to other features of the application such as the map, call, mail etc.

```swift
func eventFetch()
{
    let container = UIApplication.shared.delegate as! AppDelegate
    let context = container.persistentContainer.viewContext
    do {
        event = try context.fetch(Event.fetchRequest())
    }
    catch{
        print("error")
    }
}

func saveEvent( event_title: String,event_description: String, house_number: String, street_name:
    String, event_city: String, date_of_event: String, phone_number: String, email:
    String ,department: String  )
{
    let container = UIApplication.shared.delegate as! AppDelegate
    let context = container.persistentContainer.viewContext

    let event = Event(context: context)

    event.title = event_title
    event.event_description = event_description
    event.house_no = house_number
    event.street = street_name
    event.city = event_city
    event.date_of_event = date_of_event
    event.phone_no = phone_number
    event.email = email
    event.department = department

    container.saveContext()
}
```

6. MapKit

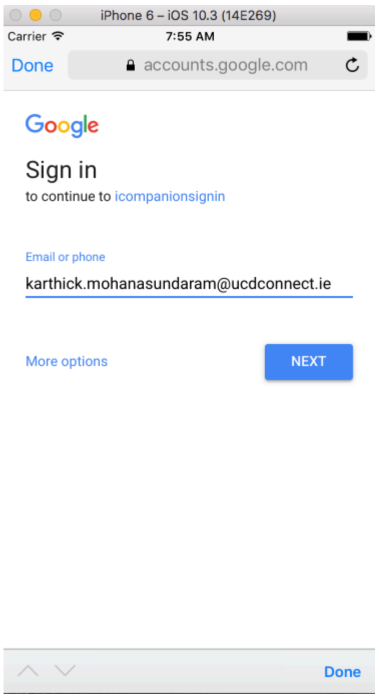
The MapKit framework is used provide an interface for embedding maps directly into a Map Kit View. In this app, this framework is used to display the location of the event, get the coordinates of the event from the address, adding a pin to the event location, directing the user to the event location.



```swift
func fetchAddress(){
    let address = "\(event_data.house_no!)"+","+"\(event_data.street!)"+","+"\(event_data.city!)"

   // let address = "\(event_data.city!)"
    model.geoCoder.geocodeAddressString(address, completionHandler: {(placemarks,error) -> Void in
        if error != nil {
            print("error")
        }
        else {
            if let placemark = placemarks?[0] {

                self.model.latitude = placemark.location!.coordinate.latitude
                self.model.longitude = placemark.location!.coordinate.longitude
                self.setupMap()
                self.addAnnotation()
            }
        }
    })
}
func setupMap()
{
    let span:MKCoordinateSpan = MKCoordinateSpanMake(model.latZoom, model.longZoom)
    let location:CLLocationCoordinate2D = CLLocationCoordinate2DMake(model.latitude!,
        model.longitude!)
    let region:MKCoordinateRegion = MKCoordinateRegionMake(location, span)

    mapView.setRegion(region, animated: true)
    mapView.mapType = .standard
}

func addAnnotation(){
    let location:CLLocationCoordinate2D = CLLocationCoordinate2DMake(model.latitude!,
        model.longitude!)

    let annotation = MKPointAnnotation()
    annotation.coordinate = location
    annotation.title = "Event Location"
    annotation.subtitle = "\(String(describing: event_data.title!))"

    mapView.addAnnotation(annotation)
}
```

<u>Screen Shots:</u>

Sign in Screen

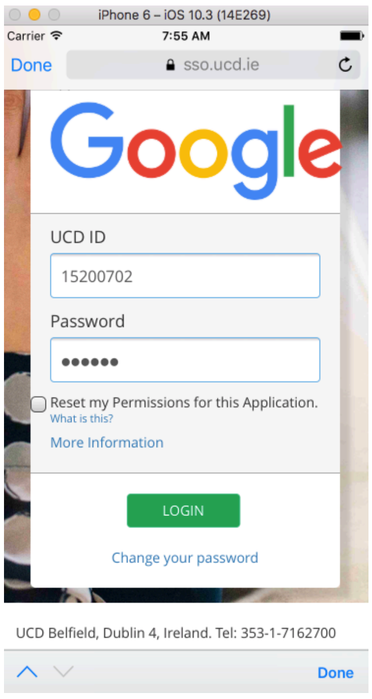

iPhone 6 – iOS 10.3 (14E269)
Carrier 7:55 AM
Done accounts.google.com

Google
Sign in
to continue to icompanionsignin

Email or phone
karthick.mohanasundaram@ucdconnect.ie

More options NEXT

Done


iPhone 6 – iOS 10.3 (14E269)
Carrier 7:55 AM
Done sso.ucd.ie

Google

UCD ID
15200702
Password
●●●●●●

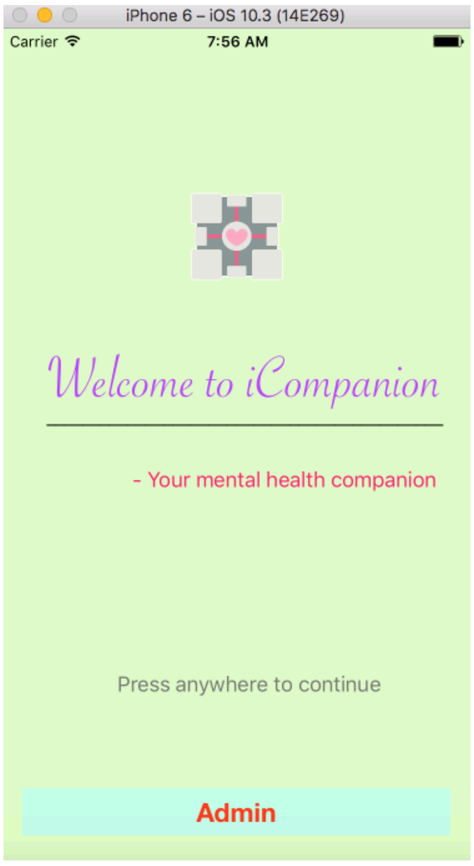
Reset my Permissions for this Application.
What is this?
More Information

LOGIN
Change your password

UCD Belfield, Dublin 4, Ireland. Tel: 353-1-7162700

Done

Home Screen:


iPhone 6 – iOS 10.3 (14E269)
Carrier 7:56 AM

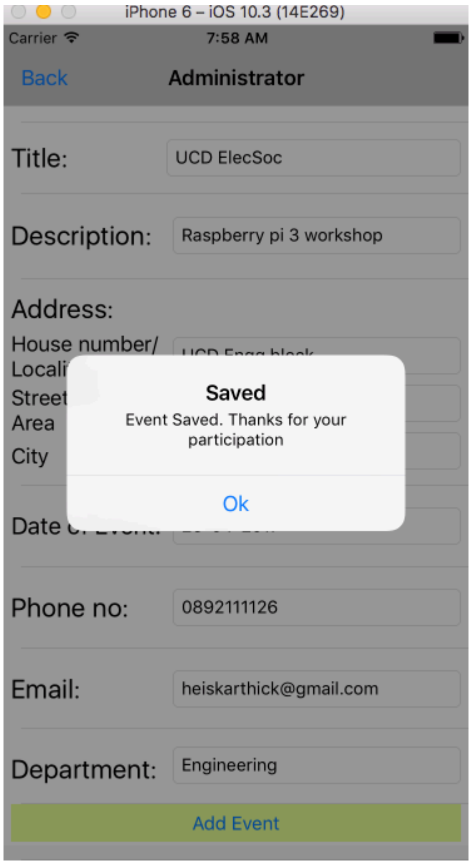
Welcome to iCompanion

- Your mental health companion

Press anywhere to continue

Admin

Admin Screen:

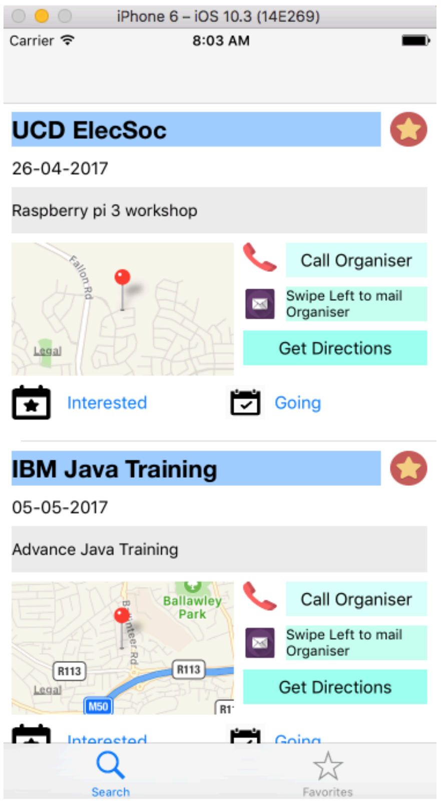

iPhone 6 – iOS 10.3 (14E269)
Carrier 7:58 AM
Back Administrator

Title:	UCD ElecSoc
Description:	Raspberry pi 3 workshop
Address:	
House number/ Locali	UCD Engg block
Street	
Area	
City	
Date o	
Phone no:	0892111126
Email:	heiskarthick@gmail.com
Department:	Engineering

Saved
Event Saved. Thanks for your participation
Ok

Add Event

Event Screen:



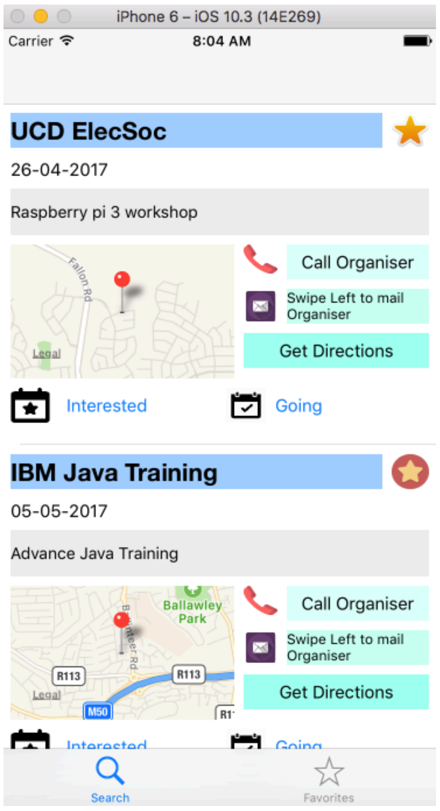
iPhone 6 – iOS 10.3 (14E269)
Carrier 8:03 AM

UCD ElecSoc
26-04-2017
Raspberry pi 3 workshop
Call Organiser
Swipe Left to mail Organiser
Get Directions
Interested Going

IBM Java Training
05-05-2017
Advance Java Training
Call Organiser
Swipe Left to mail Organiser
Get Directions
Interested Going

Search Favorites

Selecting favourite event:



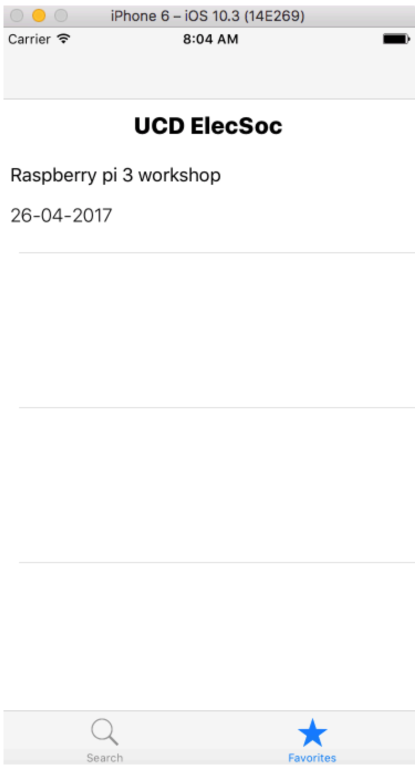
iPhone 6 – iOS 10.3 (14E269)
Carrier 8:04 AM

UCD ElecSoc
26-04-2017
Raspberry pi 3 workshop
Call Organiser
Swipe Left to mail Organiser
Get Directions
Interested Going

IBM Java Training
05-05-2017
Advance Java Training
Call Organiser
Swipe Left to mail Organiser
Get Directions
Interested Going

Search Favorites

Favourite screen:

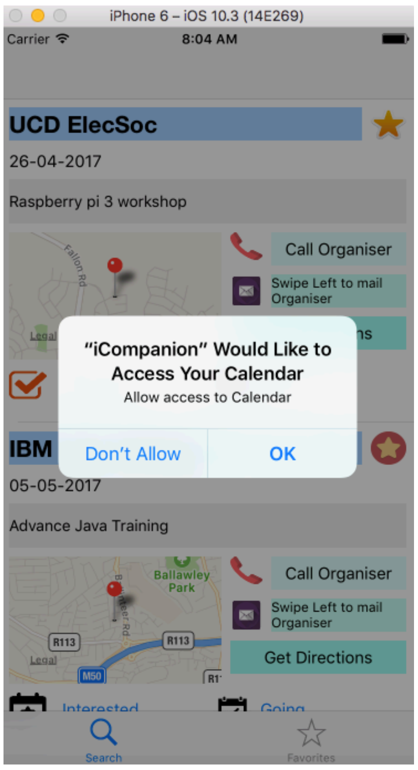


iPhone 6 – iOS 10.3 (14E269)
Carrier 8:04 AM

UCD ElecSoc
Raspberry pi 3 workshop
26-04-2017

Search Favorites

Adding event to calender:



iPhone 6 – iOS 10.3 (14E269)
Carrier 8:04 AM

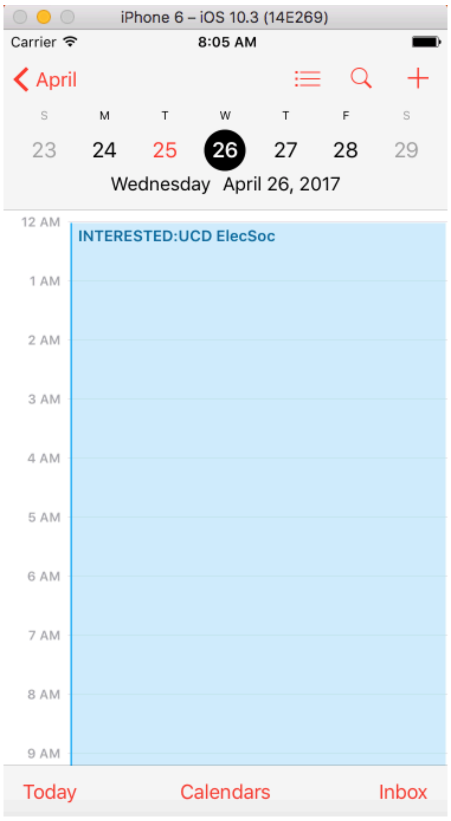
UCD ElecSoc
26-04-2017
Raspberry pi 3 workshop
Call Organiser
Swipe Left to mail Organiser

"iCompanion" Would Like to Access Your Calendar
Allow access to Calendar
Don't Allow OK

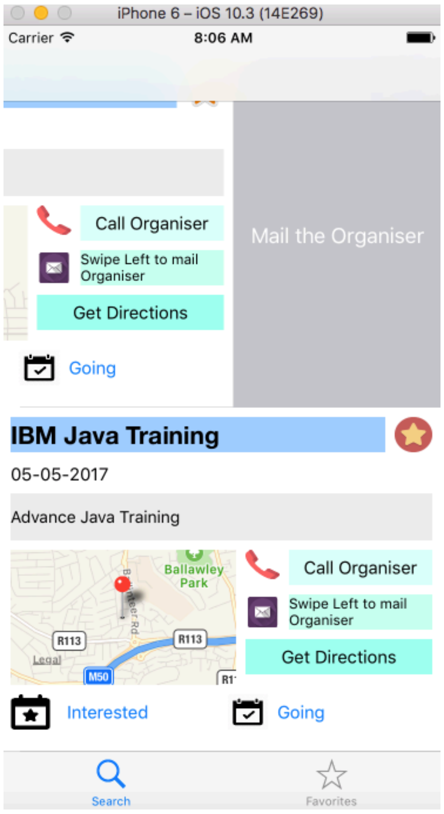
IBM
05-05-2017
Advance Java Training
Call Organiser
Swipe Left to mail Organiser
Get Directions
Interested Going

Search Favorites

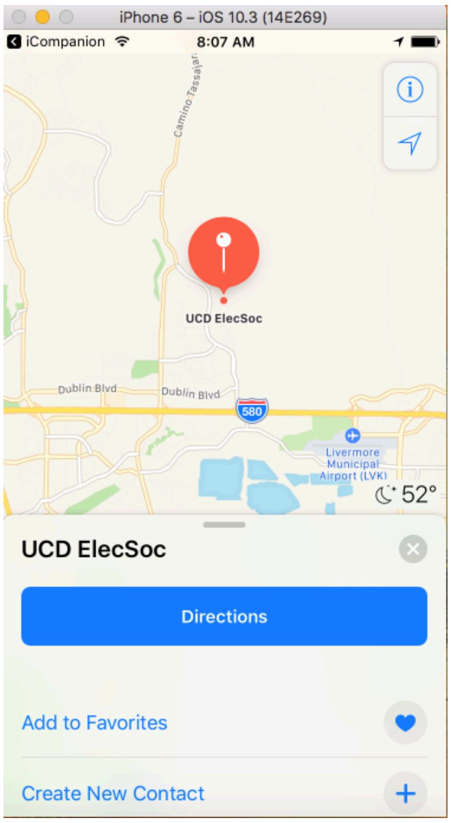
Calendar Screen:



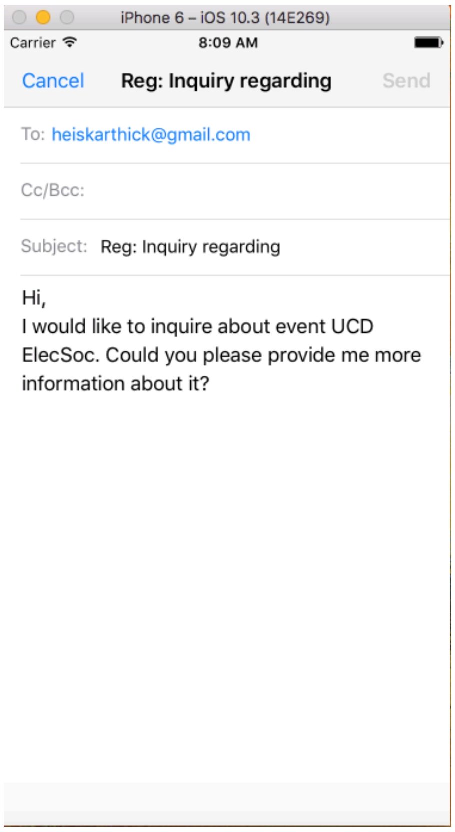
Swipe left to mail



Map screen:



Mail Screen



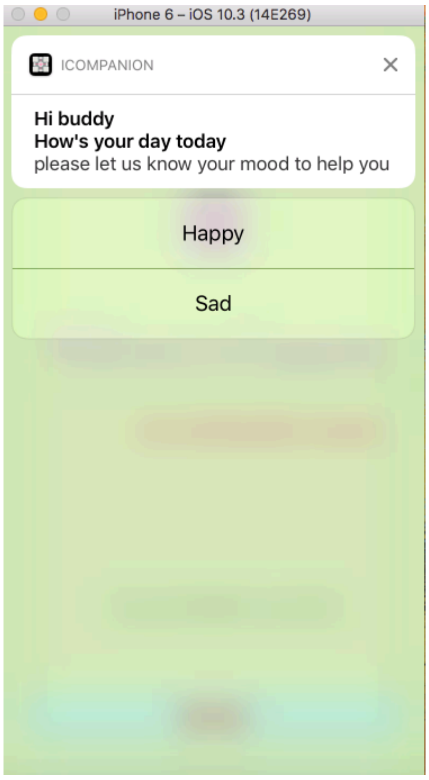
To: heiskarthick@gmail.com

Cc/Bcc:

Subject: Reg: Inquiry regarding

Hi,
I would like to inquire about event UCD ElecSoc. Could you please provide me more information about it?

Notification Screen:

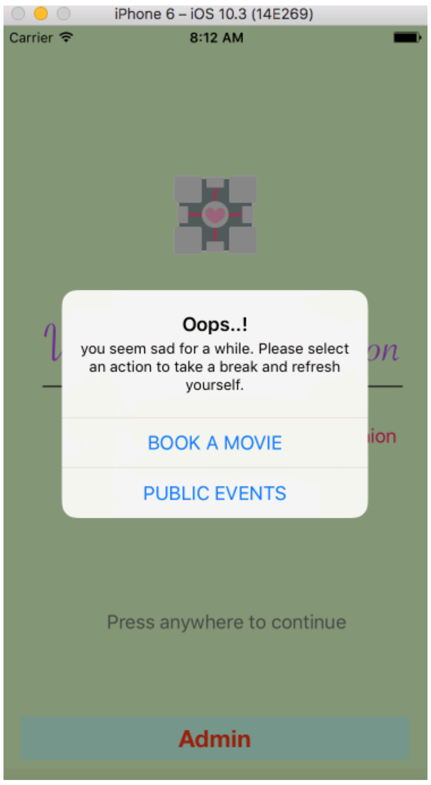


iPhone 6 – iOS 10.3 (14E269)

ICOMPANION

Hi buddy
How's your day today
please let us know your mood to help you

|Suggestion Screen 1:

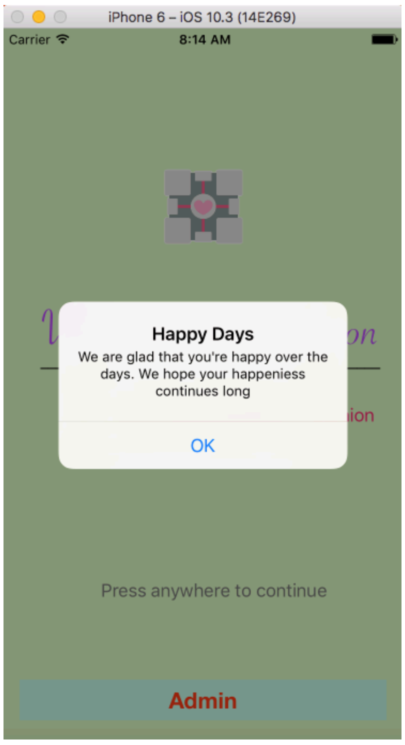


iPhone 6 – iOS 10.3 (14E269)

Carrier 8:12 AM

Oops..!
you seem sad for a while. Please select an action to take a break and refresh yourself.

Suggestion screen 2:



iPhone 6 – iOS 10.3 (14E269)

Carrier 8:14 AM

Happy Days
We are glad that you're happy over the days. We hope your happeniess continues long

<u>NOTE:</u>

Important: This application is compatible only with iPhone 6 and iPhone 7

1. Individual Contribution:

   Geeyar Babu Kuttuwa Gopinath

   - MapKit
   - All functionalities of Core Data
   - Event Kit
   - Event suggestions

   Karthick Mohanasundaram

   - Google authentication
   - Notification Centre
   - Message Framework and Calling

2. Server Component:

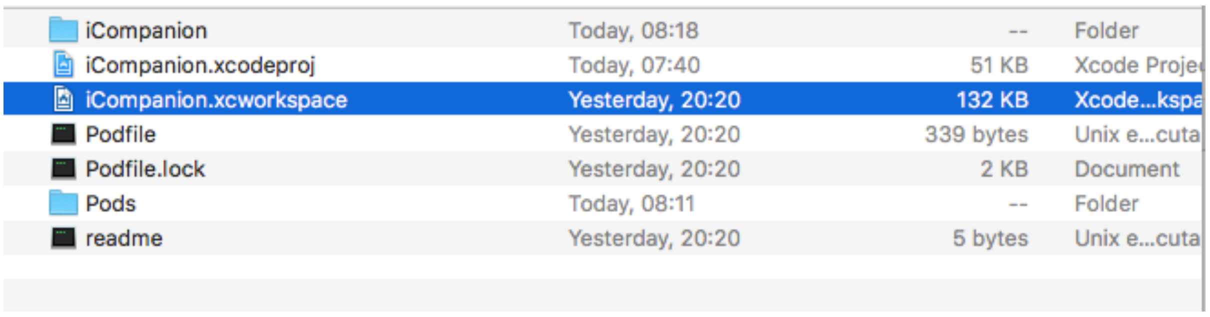
   We had proposed to implement this application using a server component, but as the third party server components available would run in a separate port number and UCD server did not allow the access to those ports. So we couldn't implement the feature

3. **IMPORTANT:** This application needs to be run on a **real device** as it has frameworks that require the details from the device and also this **application needs Apple Maps to be installed**

4. The github link for the application is:

   https://github.com/geeyarbabu/iCompanion.git

5. **IMPORTANT:** The application wont run properly when we run the app using the .xcodeproj file, so we have to run the application using the .xcworkspace file.



iCompanion	Today, 08:18	--	Folder
iCompanion.xcodeproj	Today, 07:40	51 KB	Xcode Proje
iCompanion.xcworkspace	Yesterday, 20:20	132 KB	Xcode...kspa
Podfile	Yesterday, 20:20	339 bytes	Unix e...cuta
Podfile.lock	Yesterday, 20:20	2 KB	Document
Pods	Today, 08:11	--	Folder
readme	Yesterday, 20:20	5 bytes	Unix e...cuta