**Collegiate**

**Mobile Application Dev I (ISTE77401.2155)**

* **Sagar Sawant**
* **Gaurav Gawade**

**What does the application do?**

Dorm(Collegiate/ CollegeConnect) is event managing app which will help a user of a given college to link them to the all the college events on or near the campus. The app will make use of the inbuilt college user’s authentication system with the help of web services to authenticate and authorize the user of a given college to view the events on the campus. These can be anything major or minor events on campus viz sport, cultural, studies, dinning services.

**List of Features implemented**

**Coredata Framework** –

Adding and fetching the data from the “CoreData” (sqlite model layer), the Click of the events will help adding the selected events details in the database layer.

Methods

– NSFetchRequest

- executeFetchRequest

**EventKit** –

One touch adding the events to the to the calendar database which can easily be viewed with the help of the calendar application. In addition, added custom events, which can help, maintain custom events. The events can be easily added, deleted and edited with a simple user interface.

**Firebase (authentication through sever based API)–**

The firebase client is installed using the CoCoapods, which is a dependency manager for swift projects. These pods resolve dependencies between libraries and fetch the resulting source code

The firebase authentication server passes the value from the app to the server database and sends the output accordingly to the application on individual cellphones.

**Small points to look out for**

**NSXMLParser** – The below method gets the RSS feeds from the url which is then parsed and maintained an object array of the RSS data.

-NSXMLParser(contentsOfURL: <#T##NSURL#>)

**Gesture recognition** - Shake gesture will help add the events from the detailed events page to the calendar.

**MessageUIkit** - the users can send report or bugs to hard coded email from the about us page with a click of a button

**MBProgressHUD** plugin - originally written objective-C but with using the bridging the header file you can easily import the plugins which can be used to make transitions, progress bar, loading screens on the app page which can help user notifying the background process based on the events.

**MMDrawer** - The MMDrawer is too a plugin written in Objective-C which can help the users navigate through a left and right drawer navigation which is based on right or left swipe across the screen edges. The left drawer is implemented in the app using the MMDrawer

**Documentation**

**RegisterVC.swift, LoginVC.swift**

These are the authentication controllers which used the user to register and and login, the user cannot login and view the application services until and unless registered and loggedin from the authentication interface.

**LeftViewController.swift**

The left Controller VC is a file that governs the left navigation and the calls for the click of the small table View present on the file. The makes sure that the clicks on the table are appropriately taken to the respective controllers and displayed with result.

**Directory /CustomEvents**

The custom all contain the logic for creating a custom event which all separately for are grouped into a single directory. The files are mainly consist of logic for customs events, which help data preserving in Events database linked through the calender. You can add, edit and drop events as according your will with the help with date picker and a text field.

**Directory /MBProgressHud**-

MBProgressHUD is an iOS drop-in class that displays a translucent HUD with an indicator and/or labels while work is being done in a background thread. The HUD is meant as a replacement for the undocumented, private UIKit UIProgressHUDwith some additional features [1].

The MBProgressHud is written in objective C, which is added in to the project with the help of the bridging header. MBProgressHud displays a simple HUD window containing a progress indicator and two optional labels for short messages.

This is a simple drop-in class for displaying a progress HUD view similar to Apple's private UIProgressHUD class. The MBProgressHUD window spans over the entire space given to it by the initWithFrame: constructor and catches all user input on this region, thereby preventing the user operations on components below the view.

Methods frequently used

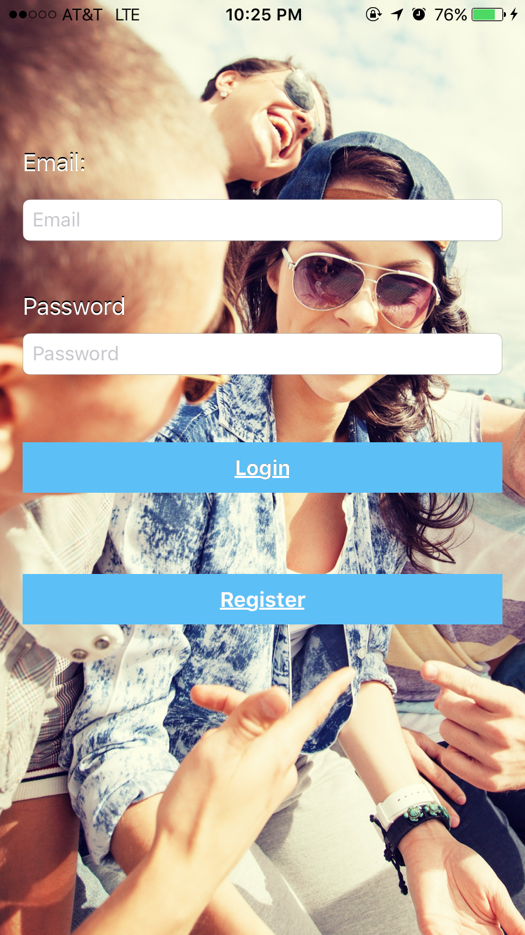
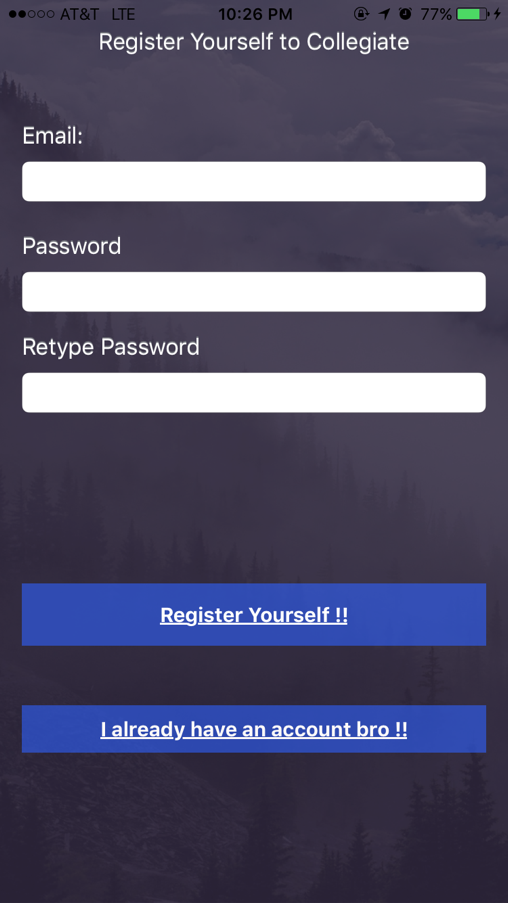
- showHUDAddedTo()

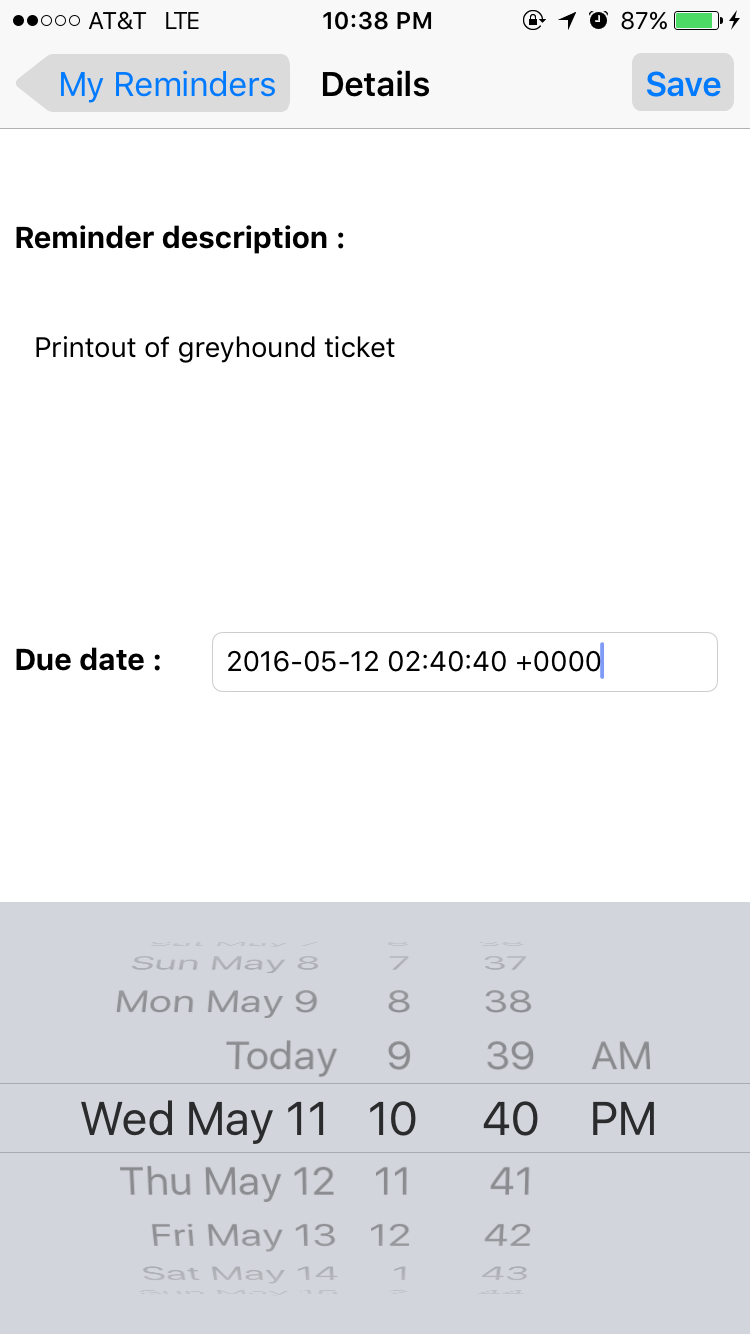
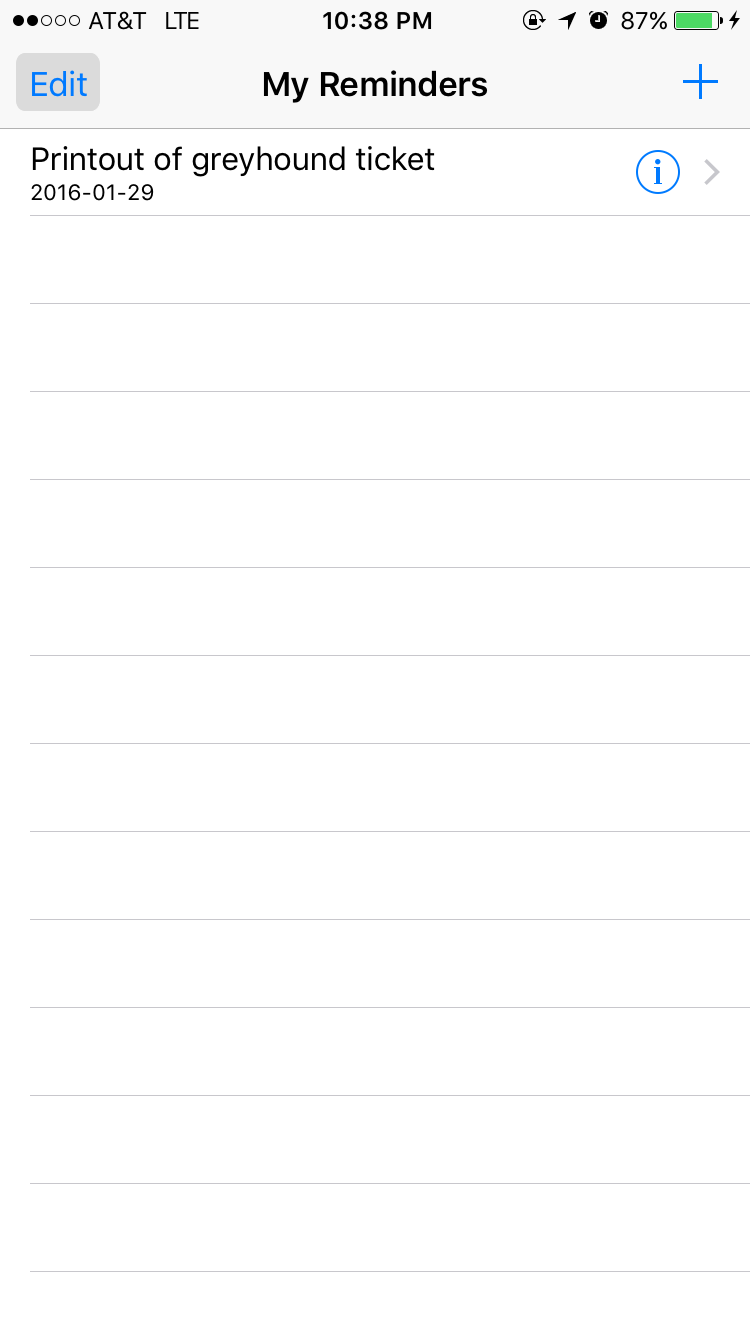
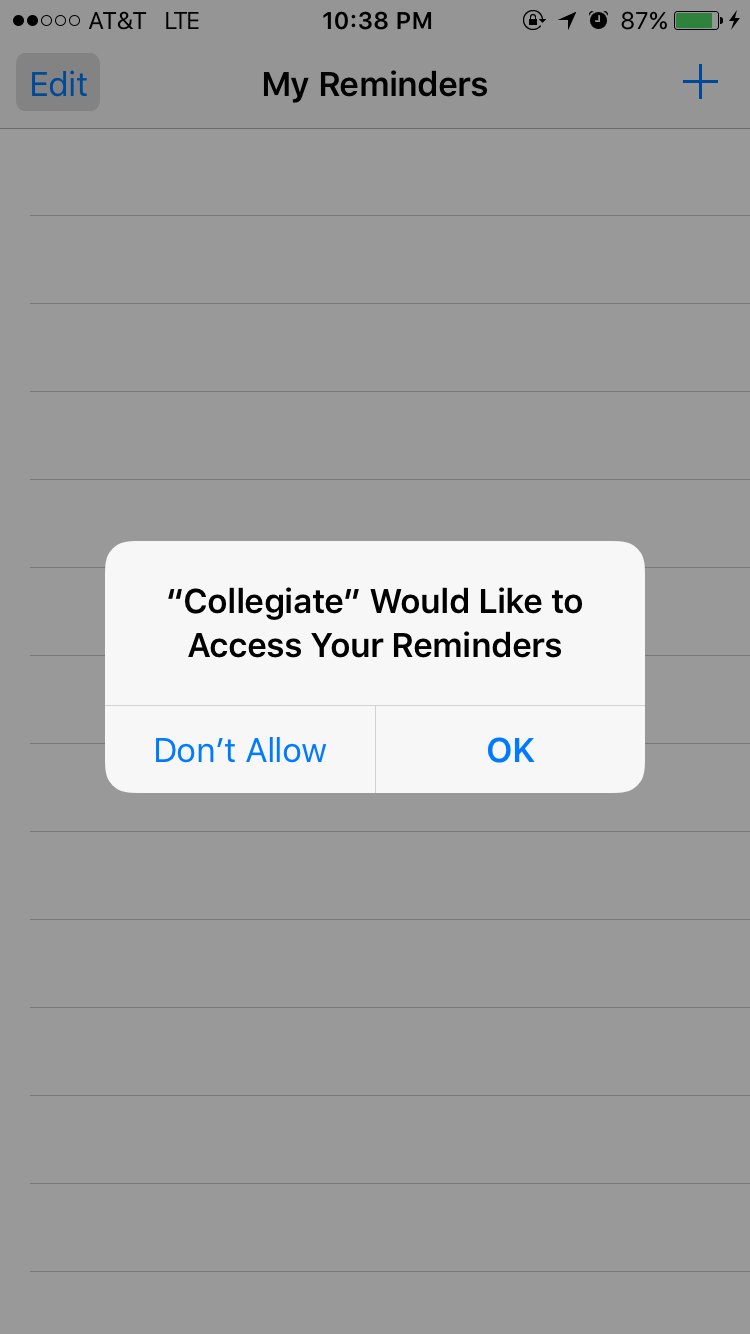
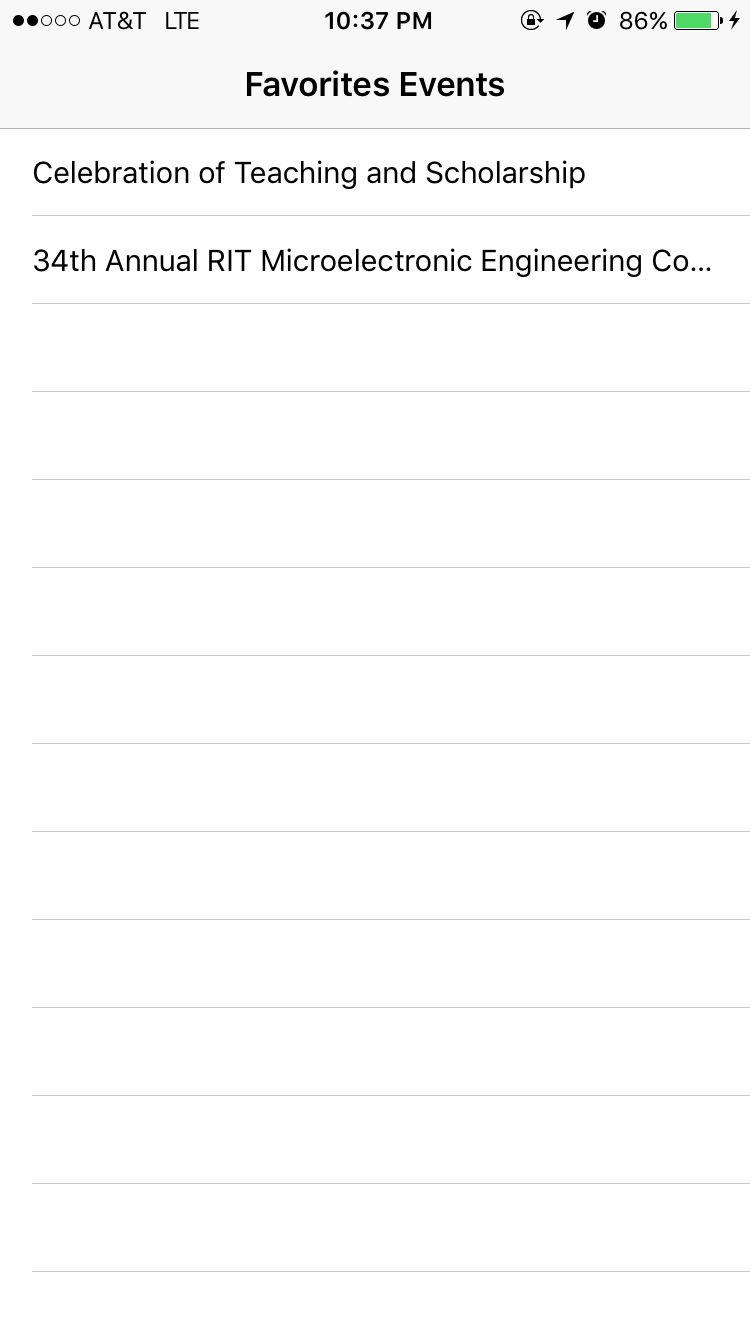
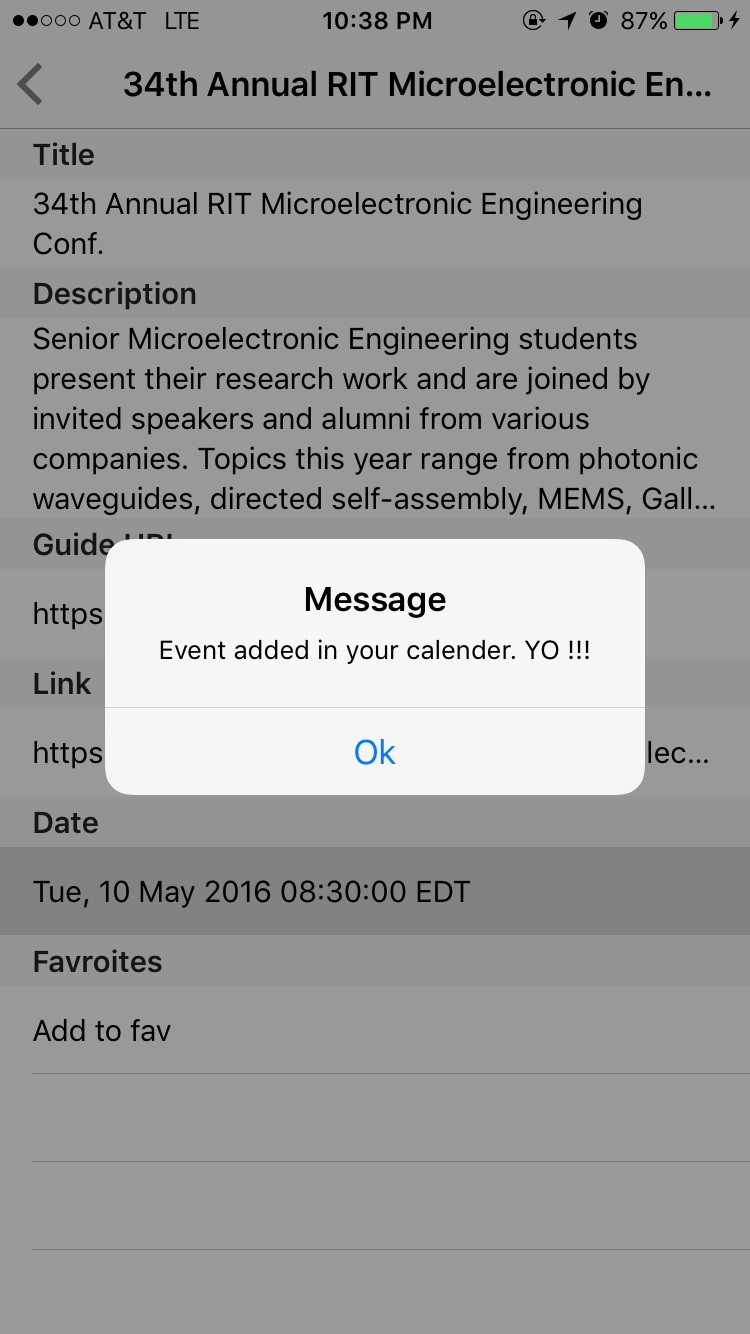
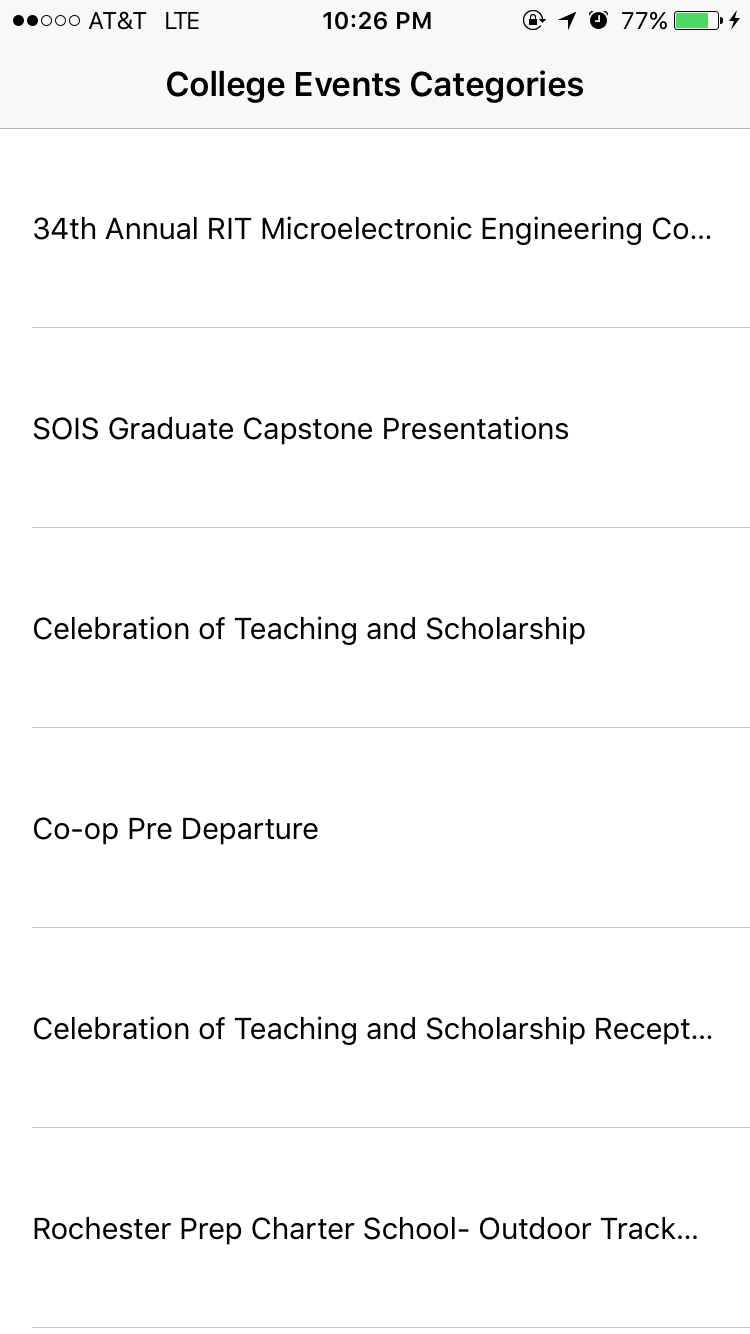
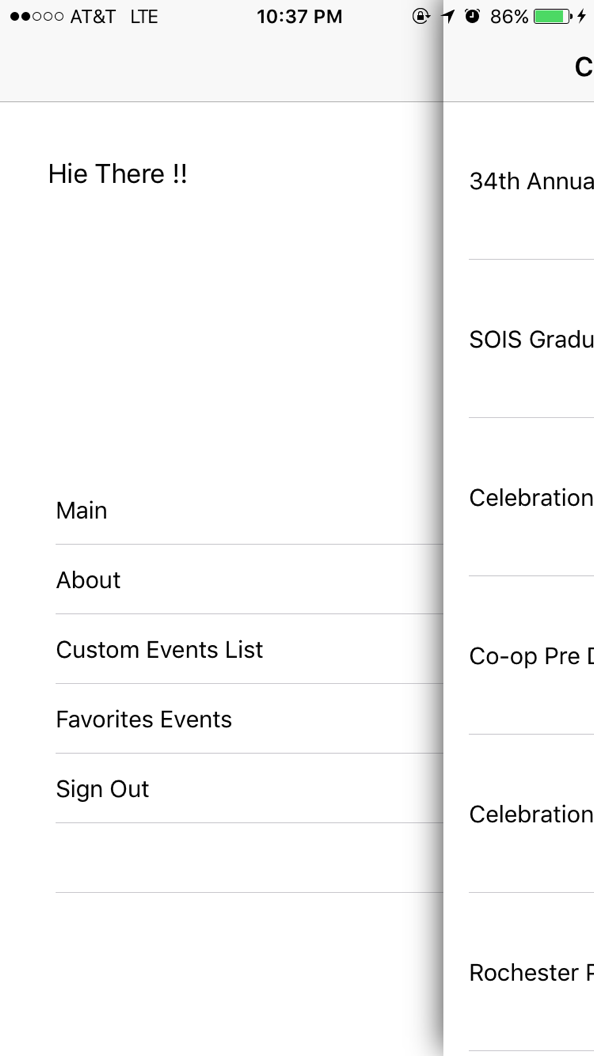
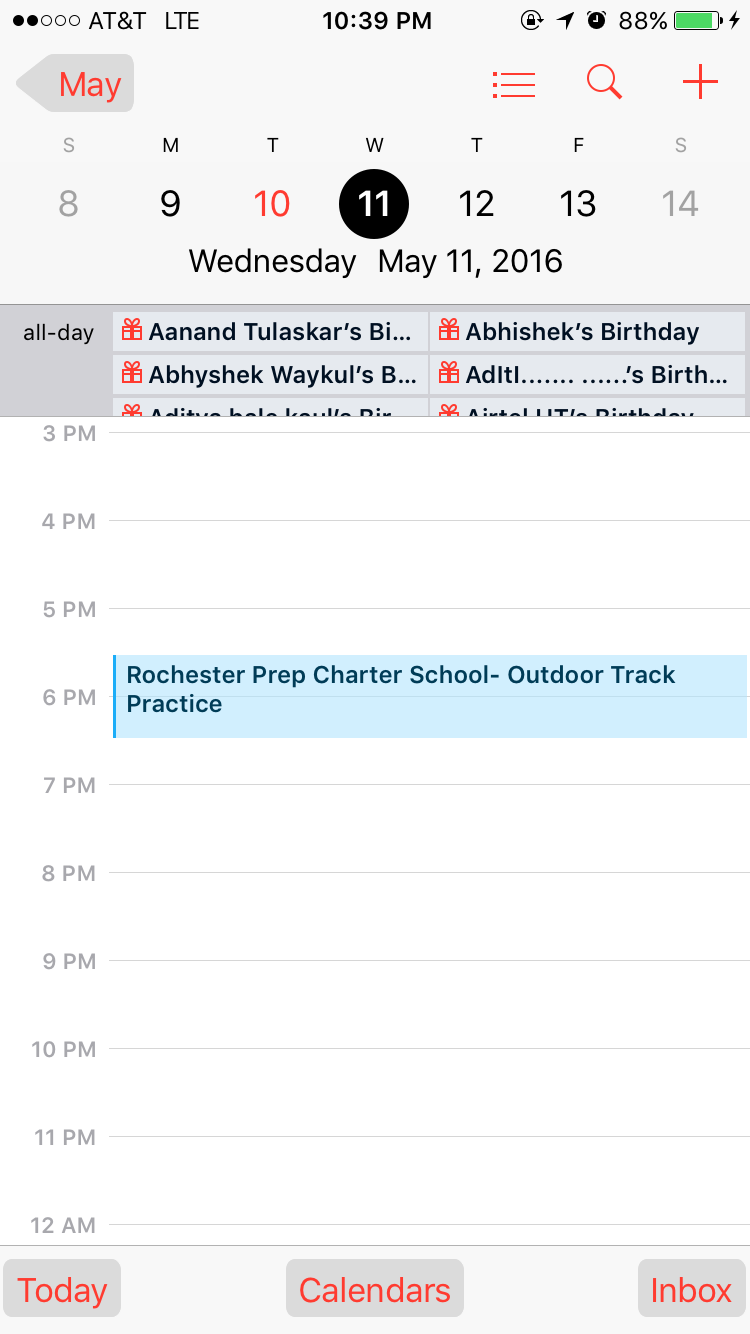
-showHUDAddedTo()

**Directory / MMDrawerController**

MMDrawerController is a side drawer navigation container view controller designed to support the growing number of applications that leverage the side drawer paradigm. This library is designed to exclusively support side drawer navigation in a lightweight, focused approach while exposing the ability to provide custom animations for presenting and dismissing the drawer [2].

**Screenshots-**

** **

**    **

**Next version-**

Being a social app which shows user interactivity, the next version should have a user database which can help linking the user and notifying the user about the other friends or the event status. In addition, this app can integrated with the social frameworks, which can help user login through popular portal like Facebook, Gmail and Twitter. The application can be comparable with terms in functionality with Mounza, the Mounza app only shows events based on career, info sessions. We have added all the events for a given College that is RIT.

The application can be made dynamic using the different college RSS feeds. In addition, we would like to implement the login using the college authentication using Open SAML login. The rss list can be generated using the dynamic database, which can help it make a bigger app considering a large geographic area targeting large traffic. Push Messages on every new added Events notifications of the friend attending the events. User profiling and linking the events to the facebook feeds.

**SelfEvaluation**

The app could have been more interactive. The RSS feeds restricted us to use only the fields, which were present in the RSS xml. We could have implemented the the mapkit functionality but the RSS feeds did not provide us the address of the event, which could have found useful while reverse geocoding of the location.

Overall, we are very happy with the project the linking of firebase and the coredata framework was a tedious task and we are extremely happy to have work with the above frameworks.

If time persists we can develop more and integrate UI frameworks, ease of use to the users.

Reference

[1] <https://github.com/jdg/MBProgressHUD>

[2] https://github.com/mutualmobile/MMDrawerController