April 30, 2017

demarcus edwards, KUNJ patel, DANIEL baker

SWE 4990

University of Virginia’s College at Wise

PROJECT MORNINGSTAR

Capstone Project

Contents

[Introduction 1](#_Toc481337026)

[Project Description and Requirements 2](#_Toc481337027)

[Brief Description 2](#_Toc481337028)

[Software Design Approach 3](#_Toc481337029)

[Software Architecture 3](#_Toc481337030)

[Detailed Design 3](#_Toc481337031)

[HCI Design 4](#_Toc481337032)

[Software Testing 9](#_Toc481337033)

[Software Quality 9](#_Toc481337034)

[Configuration Management 9](#_Toc481337035)

[Lessons Learned and Conclusion 9](#_Toc481337036)

[Acknowledgement 9](#_Toc481337037)

[Index 10](#_Toc481337038)

[Bibliography 11](#_Toc481337039)

# Introduction

The objective of this paper, is to define and explain the purpose of the product and its operating platform, and the purpose of the design and its process; the mobile application that our team agreed upon, is an event planner called *Morning Star.* Morning Star is being developed, in order to allow users to both create and share events amongst other users of the application. The apple app store is one of the dominating app stores on the market aside from the google play store. 40% of all apps downloaded in the First Quarter of 2013 were IOS applications. The current operating platform for Morning Star is IOS 9 or later, and is being constructed using the XCode 8.2.1 version of the Apple released I.D.E. Our group decided to develop Morning Star for the Apple platform first, because in less than a year…after its release, Swift became one of the most popular programming languages in the world [1]; furthermore, our team decided to go with mobile development as opposed to one of its counterparts such as, desktop or tablet development, or a web application, because the mobile app market is facing a fascinating growth, with IOS…playing a central role in this arena [1].

# Project Description and Requirements

## Brief Description

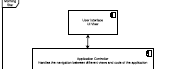
An individual user of the mobile application, Morning Star, should be able to create their event and save it as a list on their local device while also, having the capability of uploading the event to a database; the database should be able to be accessed by all users of the *Morning Star* application. Our user domain will be geared towards a young adult demographic, primarily consisting of undergraduate students. In terms of soliciting requirements, we took a user’s goal-directed approach; our users should be able to quickly create and share as many events as they desire, while at the same time, have the capability of viewing past events they either attended or created, hence the list approach.

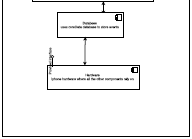
# Software Design Approach

## Software Architecture

Our application communicates between Table view controllers that store the events in a table that the user sees, and also the UI View controller that allows the user to enter data. The Controllers pass information to each other by reference. Class event is defined in the NSManged Object which is a data type defined by swift. Event is defined in all the view controllers. UI view controller appends the data of type event into the core database and Table view controller fetches the data from the database and displays. Swift allows us to use core databases. The limit of the core database is your devices storage limit. However much storage you have left on your device is how much data that can be used inside the application. When getting close to the data limits iOS gives users warnings that they need to clear space for more storage or to buy cloud storage space.

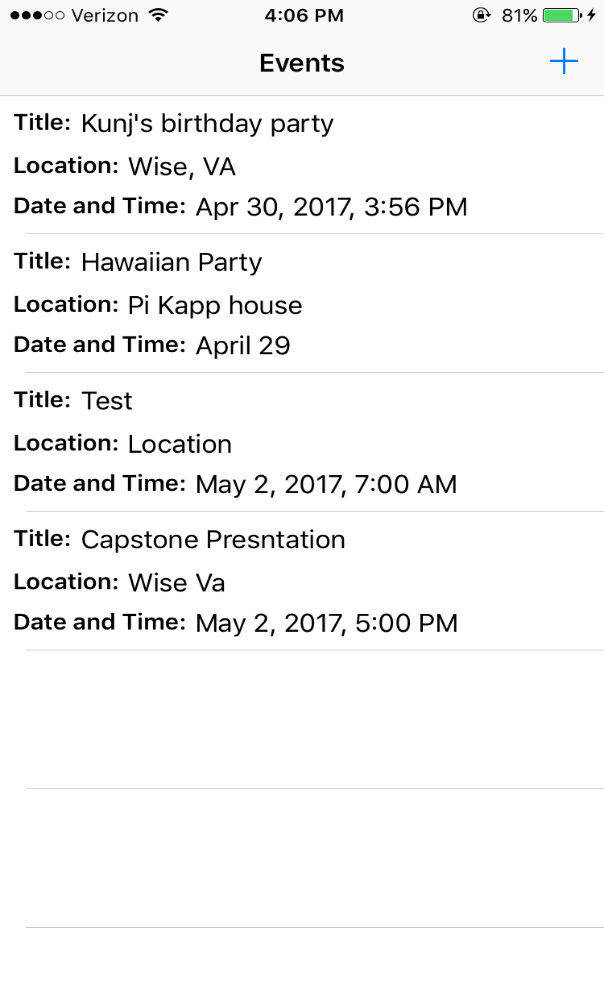
# Detailed Design

****

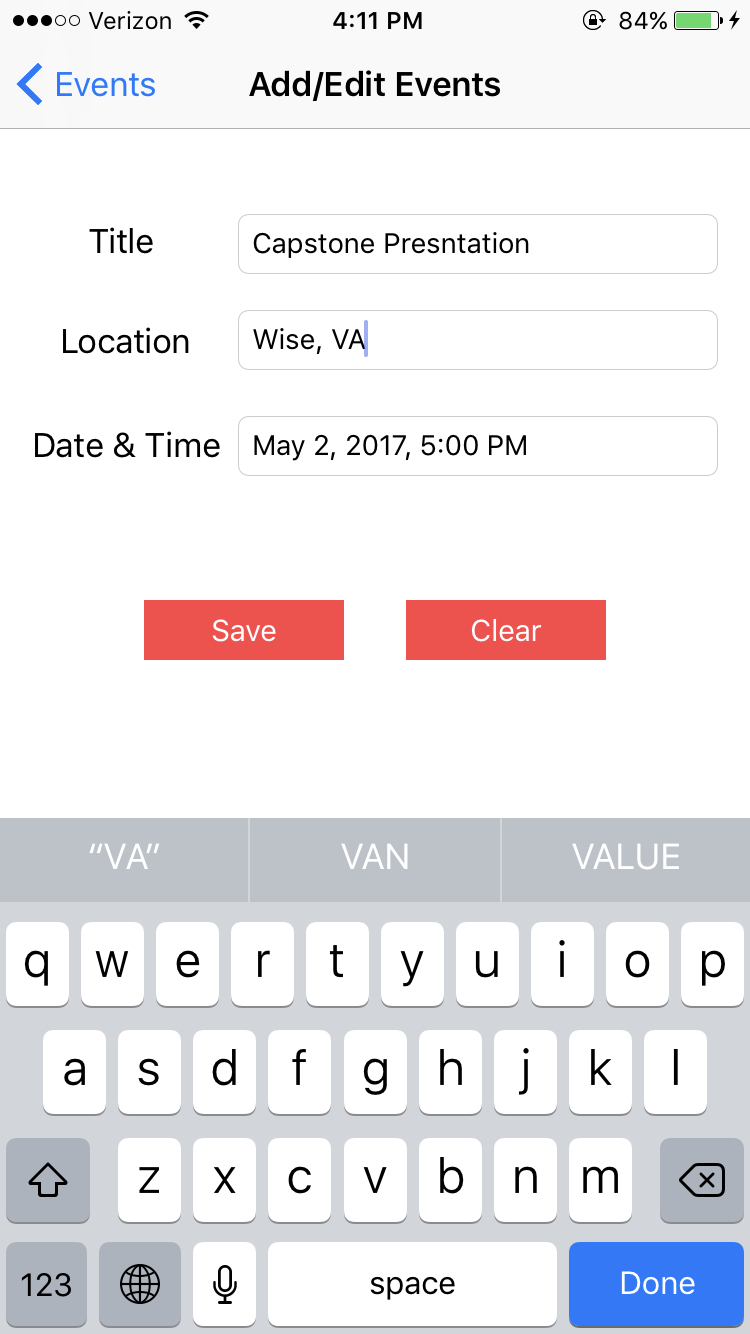
****

(Apple inc., 2015)

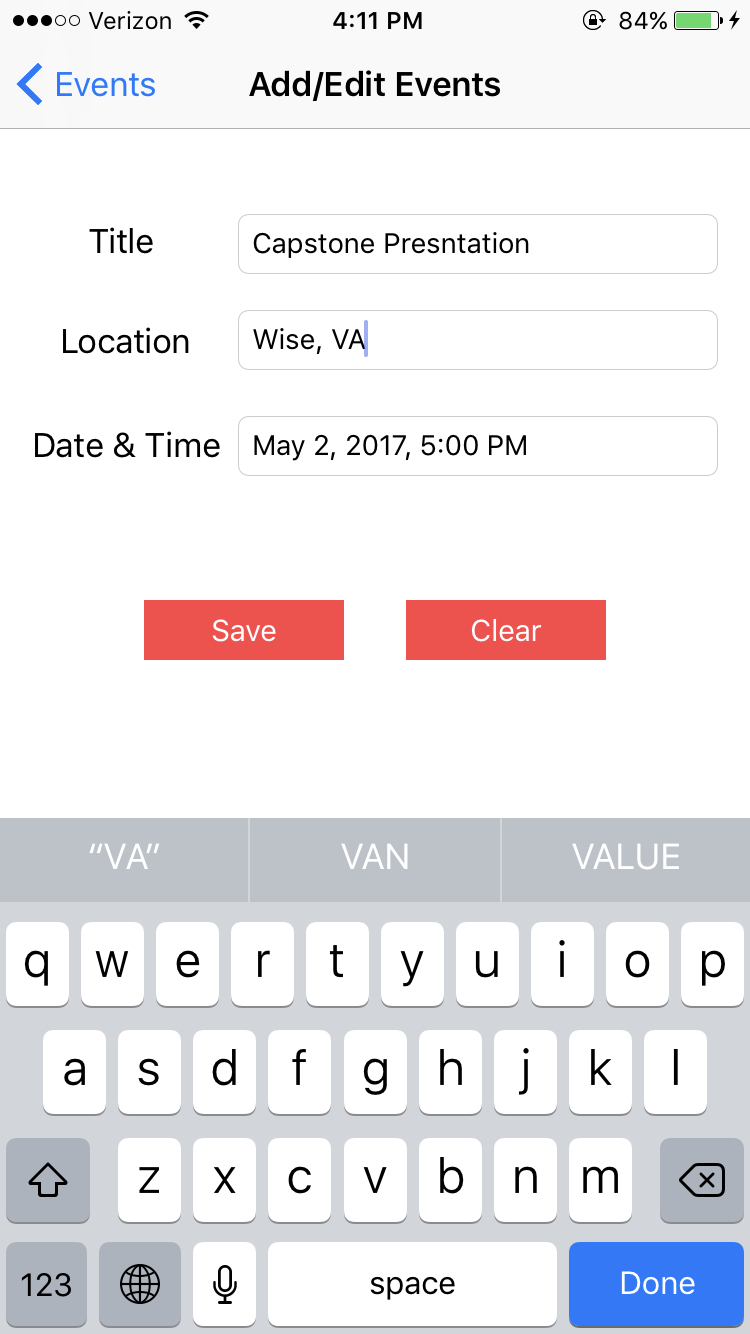
# HCI Design



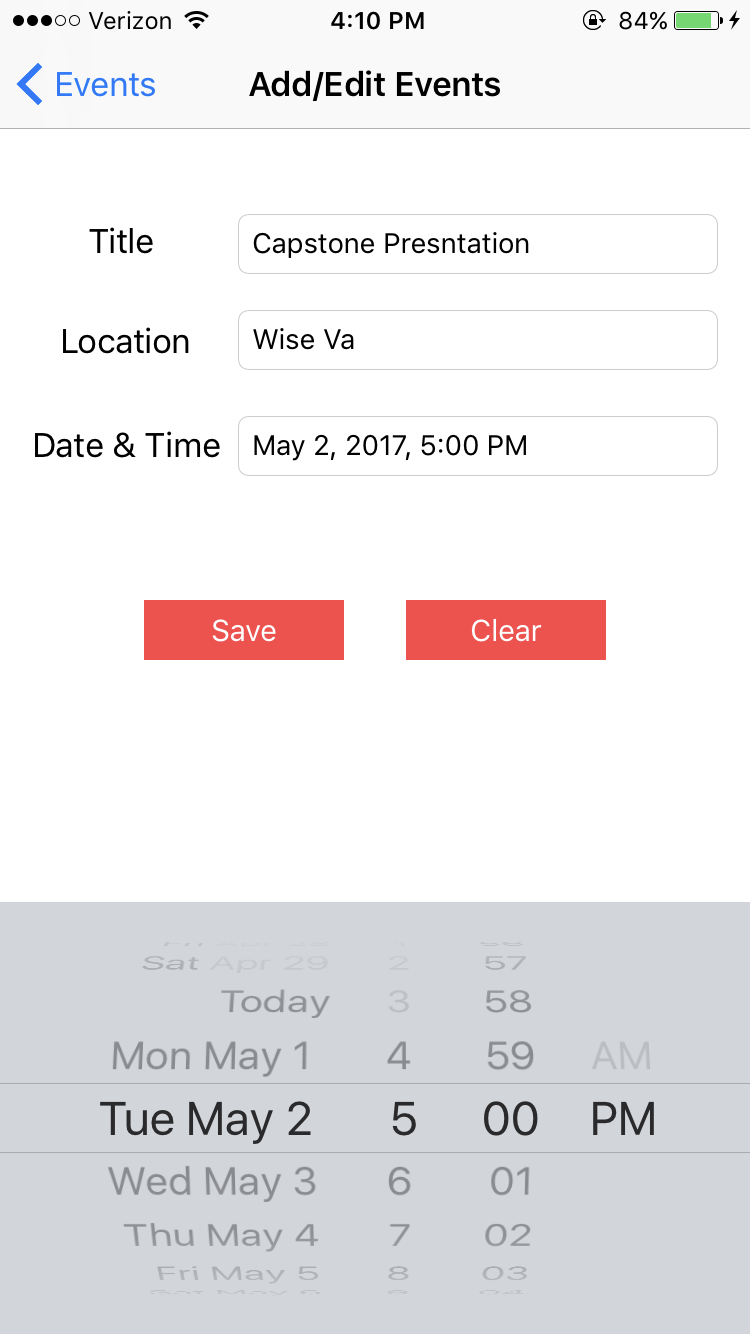
When opened morning star shows a splash screen with the college’s information, and then a table of events is shown. If the user has no event saved in the table then the table will be empty. To add an event there is a plus sign, which was chosen because it is a universal sign for add.



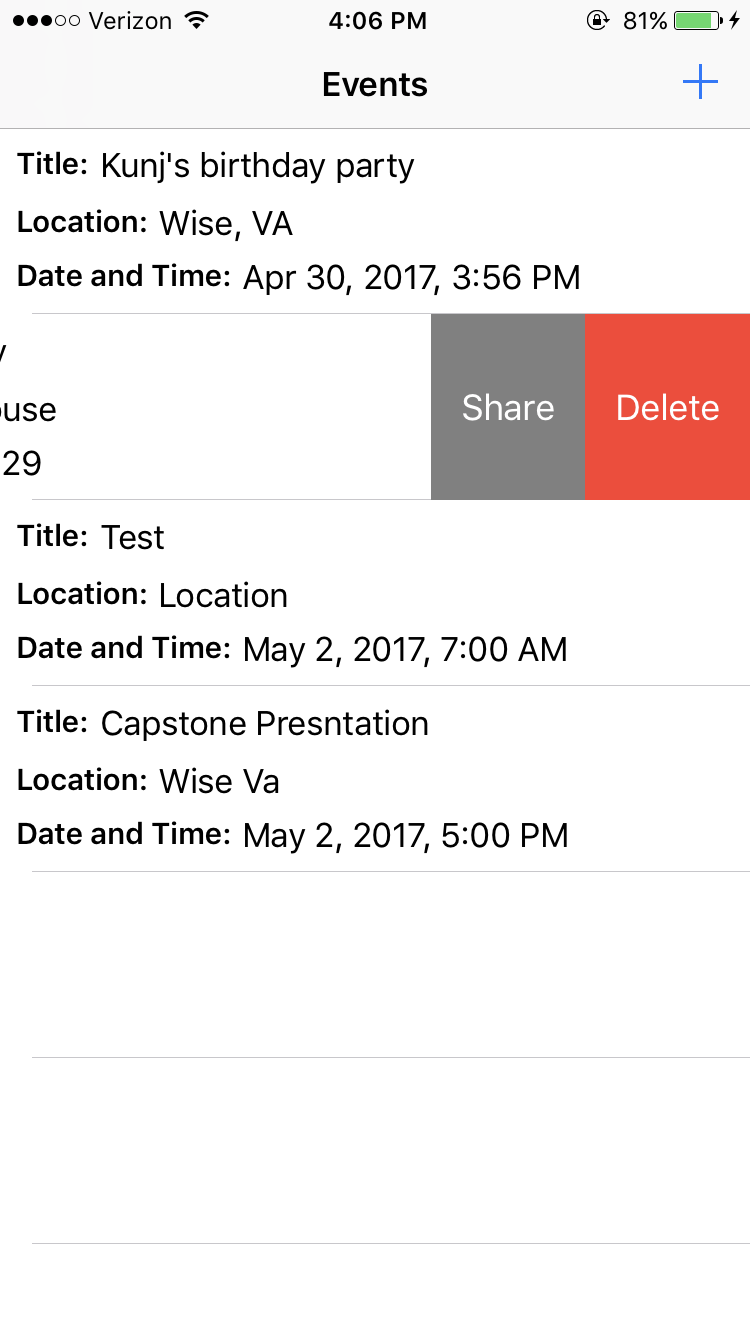
When clicked the fields for a new event a shown. These include the Event Title, Event Location, Event Date and Time.



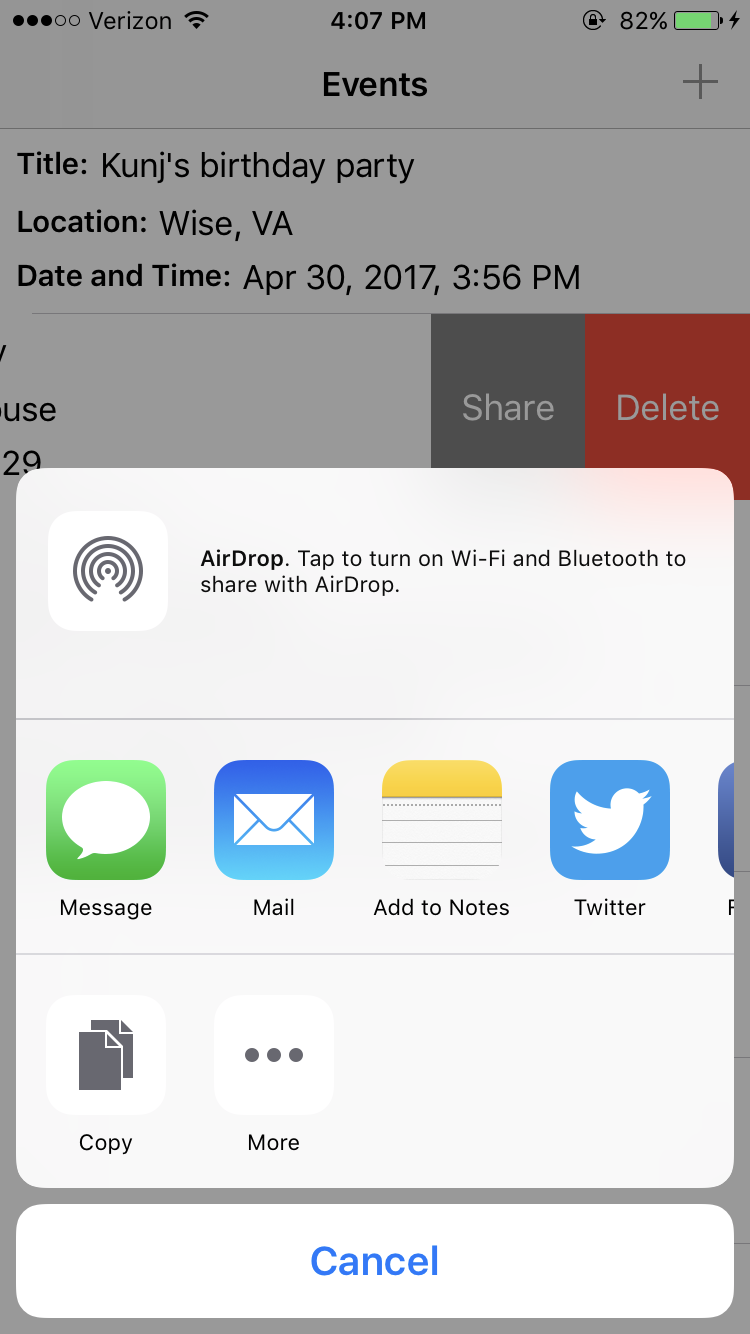
The Title and Location are text fields, meaning they are typed in with the user’s keyboard. The keyboard appears when the text field is tapped. iOS keyboards are allowed to be changed from the default in the settings application. The Morning Star application is compatible with all keyboards.



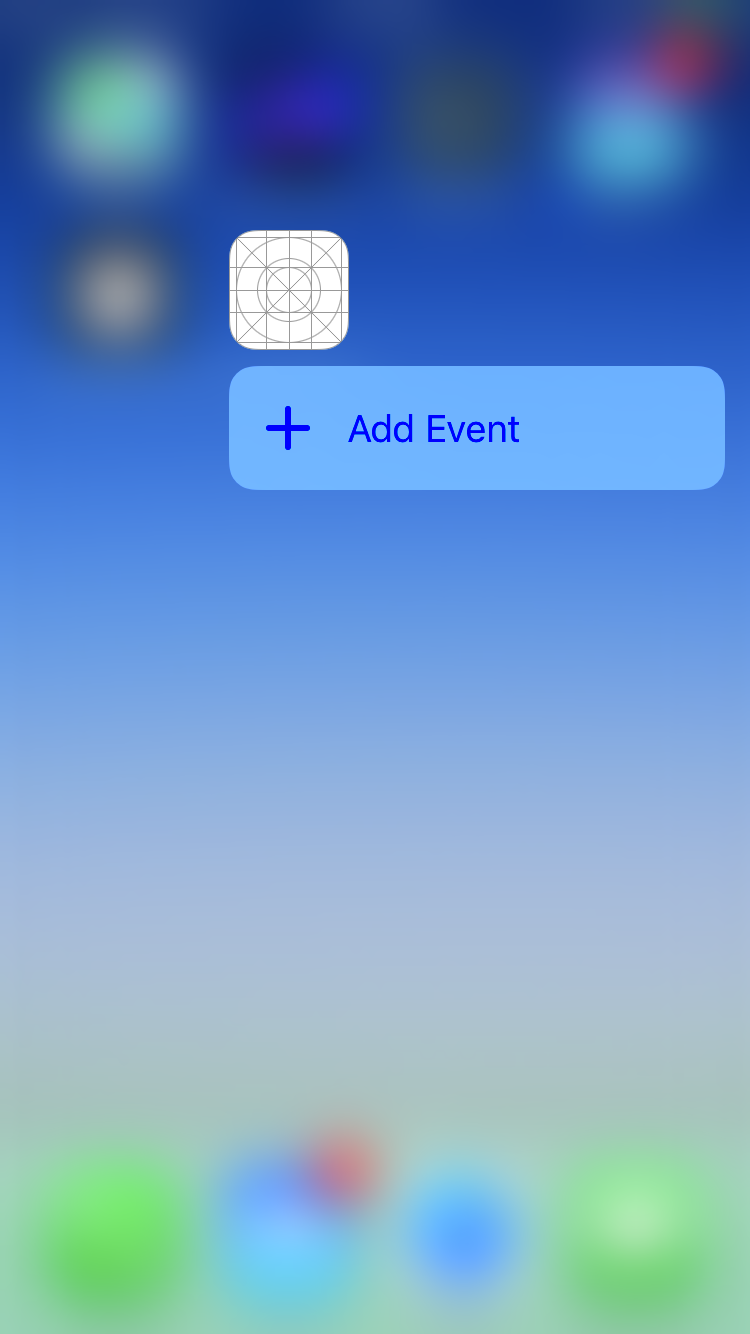
The Event Date and Time are chosen through UI date picker. Which allows the user to scroll through dates and times on a wheel like apparatus. Dates and times can’t be chosen that have already happen. For example if today is April 30, 2017 at 4 pm, you can’t choose 4/29/17 or something at 3pm or both.



Once you’ve added an event you can now share your event by swiping left you have two options share and delete. To delete event just select delete and poof the event is deleted off the table.



To share event select share and then through what medium you would like to share the event. The share options are only limited by the apps on your device. Twitter, imessage, email, notes etc. they are all compatible with morning star.



For newer versions of the iPhone we have added the feature of 3d touch. From outside the application, the user can hold down on the application icon and go straight to the add new event page.

# Software Testing

For testing the application we used boundary value analysis using boundary values of the minimum, the maximum, and also error values. The minimum values for Title length and also location length were tested. At first there were no constraints on the date and time field. It was allowed that the user entered a past date or time and also that they could enter a date and time that doesn’t exist for example a word. After testing this was fixed so that the constraints require the user to enter a present or future date. Also non-acceptable times are no longer allowed in the date & time field. The GUI kept acting faulty at first during early testing, but through subsequent tests certain functionalities such as the 3d touch and also the icon were fixed.

# Configuration Management

Publishing the project on GitHub allowed our project the flexibility to be worked on from anywhere. The only restriction was if we had an apple device to work on. Also pushing the updated documents to the each other was made easy.

# Lessons Learned and Conclusion

Developing for iOS applications is quite like android studio. The xcode IDE is a pretty powerful tool that helps creating beautiful projects simple. Also the developer library that apple has online and in xcode is simple and easy to understand. The most difficult part of the project was the fact that we were our own customers. We developed the application for ourselves basically, at first we had the mail room as a customer, and then we planned on making a party application, but then with a little convincing from the board we decided the best plan of action was to actually make an event planner application. The application works just to our specifications in our requirements document and also we learned how to work as a team on the project. A lot of our programming up to now has been individual projects, this project really required the group to make effective use of its time and also to learn a foreign language. No one in the group had known how to write code in Swift prior to this project. We all took our time to learn the language and also to help each other learn the language. Group work such as this is much more indicative of real world software development teams.

# Acknowledgement

We would like to thank the Math and Computer Science department for the support and especially Dr. Qureshi whose help on this project was tremendous.

# Index

*core databases 3*

*"You use Core Data to manage the model (in the model-view-controller sense of the word) objects in your application. Core Data is an object-graph management and persistence framework. Among other things, it:*

* *Allows you to efficiently fetch model objects from a persistent store and save changes back to the store.*
* *Provides an infrastructure for tracking changes to your model objects. It gives you automatic support for undo and redo, and for maintaining reciprocal relationships between objects.*
* *Allows you to maintain disjoint sets of edits of your model objects. Disjoint sets are useful if you want to, for example, allow the user to make edits in one view that may be discarded without affecting data displayed in another view.*
* *Allows you to keep just a subset of your model objects in memory at any given time. This is useful for keeping the memory footprint of your application as low as possible.*
* *Has an infrastructure for data store versioning and migration. This infrastructure lets you easily upgrade an old version of the data file to the current version.*

*To support this functionality, Core Data uses a schema called a*managed object model*that describes the entities in your application."* (Apple inc., 2015)

*NSManged Object 3*

*"NSManagedObject is a generic class that implements all the basic behavior required of a*[*Core Data*](https://developer.apple.com/library/content/documentation/DataManagement/Devpedia-CoreData/coreDataOverview.html#//apple_ref/doc/uid/TP40010398-CH28)*model object. It is not possible to use instances of direct subclasses of NSObject (or any other class not inheriting from NSManagedObject) with a managed object context. You may create custom subclasses of NSManagedObject, although this is not always required. If no custom logic is needed, a complete*[*Object graph*](https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/ObjectGraph.html#//apple_ref/doc/uid/TP40008195-CH54)*can be formed with NSManagedObjectinstances."* (Apple inc., 2017)

*Swift 3*

*"Swift is a successor to both the C and Objective-C languages. It includes low-level primitives such as types, flow control, and operators. It also provides object-oriented features such as classes, protocols, and generics"* (Apple inc, 2017)

*Table view controllers 3*

*“A controller object that manages a table view.”* (Apple inc., 2017)

# Bibliography

Apple inc. (2017). *Swift*. Retrieved from Apple inc: https://developer.apple.com/swift/

Apple inc. (2011, 10 12). *developer.apple.com*. Retrieved from apple.com: https://developer.apple.com/library/content/documentation/DataManagement/Devpedia-CoreData/coreDataOverview.html#//apple\_ref/doc/uid/TP40010398-CH28

Apple inc. (2015, 10 21). *Model View Controller*. Retrieved from developer.apple.com: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html#//apple\_ref/doc/uid/TP40008195-CH32

Apple inc. (2017, 1 1). *NSMangagedObject*. Retrieved from developer.apple.com: https://developer.apple.com/reference/coredata/nsmanagedobject

Apple inc. (2017). *UITableViewController*. Retrieved from developer.apple.com: https://developer.apple.com/reference/uikit/uitableviewcontroller

Mohamed, I. a. (2015). "Android vs IOS security: A comparitive study". *12th Internation Confrerence.* IEEE.

Reboucas, M., Pinto, G., Ebert, F., Torres, W., Serebrenik, A., & Castor, F. (2016). An Empirical Study on the Usage of the Swift Programming Language. *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Rengineering.* Los Alamitos: IEEE COMPUTER SOCIETY.