==========
==========

README.TXT
Harish Kamath

==========
==========
*Please resize your document window so that the lines of = above take up one row of the window.

To whom it may concern,

Hello there! Thank you for showing an interest in my iOS application, South Forsyth News, which is an entry to the prestigious FBLA Leadership Conferences, under the Mobile Application Development category. The primary purpose of this document is to inform you on how to navigate through my project, as Xcode can sometimes be a confusing IDE to work with. Even if you have sufficient background with Xcode/ iOS development, I encourage you to read through the entire document, as this document will introduce the many classes and files in the project, and how they are connected.

In order to work through the programming aspect of this project, you will need a computer running a recent version of the Macintosh OS, as well as the Xcode IDE installed(a free install from the Apple website). To open the files necessary, simply open “SFHS News.xcodeproj”.

Alternatively, in order to simply test the application, you can download a copy directly from iTunes App Store here: https://itunes.apple.com/us/app/sfhs-news/id958397952?mt=8

There is, however, one very important fact that I want to mention. This document is by no means a user manual to the application. Of course, there should be a manual nonetheless; you should be able to find the “User Guide.pdf” in the same directory as this file. This document is solely dedicated to explaining the logical flow of the application, and explain the theory behind it. As an additional note, the “Database Testing.pdf” file will also give a brief explanation to the process behind which one can add an event or club to the app database.

One of the most important facts, for those less familiar with the iOS development platform, is that any Xcode class will have two files to the class; the header and the implementation file. More detail can be found in Apple documentation, but I have also included a concise anecdote of what this means at the beginning of AppDelegate.h file, for anyone that is interested. It is important to treat the .h and .m files of a class as grouped, or else much sense can not be made of the

code.

At the top of each .h file of each class, I have also included a brief description of the class, although it may still be helpful to read this document through, as it covers a wider variety of information.

Without further ado, I shall now explain the many files and classes which, I admit, can seem overwhelming at first, but truly is not.

The first and foremost page that you land upon when viewing this application is the Scheme and Target Control Panel. This is a panel automatically created by Apple, but is of little use on the developmental end(the only purpose it serves for us is to define external libraries, which I do not use in the project, and to change version number, which only plays a role during the release version of the application).

You will likely see three folders after this: SFHS News, SFHS NewsTests, and Products. Of these, we can only use SFHS News, as the other two are auto-generated by Apple and play no further role in the developmental aspect of an app.

The first file within SFHS News is the AppDelegate.h/.m. Again, this is a file automatically created by Apple for every application, but it does serve a purpose in my app. As the name suggests, this class is the first called when the application is opened. We can use this to our advantage by making sure that the classes that may need some time to be created and worked with get called right off the bat; specifically, we can call the classes dealing with database calling here, since databases are usually large objects that need some time to access. It is also important to note that throughout this file, I have included comments explaining the architecture of a generic iOS class, so as to help a person unfamiliar with Objective-C.

Next, there is a file called Main.storyboard. This file is one of the most important, as it clearly displays the graphical aspect of the application, and is very helpful when trying to decipher user flow(as in where a user goes after a button click). One important thing to note is that you may see small warning symbols emanating from this file; however, these are not “real” warnings. All of these warnings are simply stating that the view you see on the storyboard may differ from the view displayed to the user; this is only because I found it useful for a developer to display all views here(even ones that may be obscured to the user) so that they would not have to hunt for any views that would normally only show to the user under specific circumstances. These views will automatically “fix” themselves back to normal while the app is running, however, and so have no adverse effects(rather, any effects at all) on the user.

The next folder you encounter is the Database Files folder. This

folder contains all files pertaining to database creation, database access, etc.

The first class, SQLDatabase.h/.m, is perhaps the most important. It contains a static method which creates the database for the device, as well as gathers information from the remote database and stores it in the said local database. This is done through the use of PHP Scripts. It also creates two arrays, one for events and one for clubs, which hold EventInfo and ClubInfo objects(explained in the next couple of classes), and which are the basis for all data within the application.

Since the EventInfo.h/.m and the ClubInfo.h/.m are so similar, I have decided to explain them at the same time. All that these classes do is that they act as “wrappers” to hold all of the information derived from a single event or club in a single location. This makes it easy to access the event/club’s information, as I can simply look at the values of the EventInfo/ClubInfo object which holds it. More information can be found in the User Guide.

There is one final folder, called sqLite database, which holds a few classes in it as well. However, these classes are not used anywhere in the project; they remain merely as a proof of concept. During initial stages of development, I was hard-set on using sqLite, instead of MySQL, to control data. However, this quickly became unfeasible, as data needed to by dynamically changed as events and clubs changed. I still believe, however, that this folder should remain in the application, as (1) if there ever is a need to hold a large amount of data locally, it would be almost too easy to switch between MySQL and sqLite, as the files have been created already, and (2) considering this project is primarily curricular, I believe that these classes extend my depth of knowledge, and show this extension to the reader of the project.

There is also a Views folder within Database Files. This folder holds, as the name implies, all of the views of the project. This folder is the derivation of the entire project that the user sees, and it is therefore essential to understand these classes thoroughly.

The first subfolder you come across is the Sidebar Menu folder. It simply holds the classes necessary for the sidebar menu(obviously) to show and use. If you have already used this application, this is the object you see when you click on the three vertical bars found in the top left hand corner of almost every view. This folder also contains SWRevealViewController.h/.m. It is important to note that I have not created this class, and do not claim to have created any part of this class. However, SWRevealViewController, which is a free github community project, is considered industry standard, and many powerful applications such as Facebook use it as well. The only role it plays is in displaying the sidebar menu, and has no part in the data or any other view in the application. Similar to this, there is also an

SDWebImage class that appears later on in the project(under Supporting Files), which I have also adopted from a community github project. These files are only used for quick display of images from URL's that would otherwise greatly impact the speed of the app.

The first class directly in the Views folder is ViewController.h/.m. This class, also created by Apple as a default in every app, is the first to be shown during the app launch. In our case, this view is the only black one, and also contains the required banner view.

The second class, FeedTableViewController.h/.m, is one that displays the events page. It is a table view with 2 sections, divided by chronology, and is the "main" view of the application. Its cells/rows are made up of the FeedTableViewCell class, which will be explained in the next section.

The rest of the views, AgendaViewController.h/.m, ExtracurricularViewController.h/.m, AchievementsViewController.h/.m, and PhotoCollectionViewController.h/.m, are mostly derivations of the same TableView concept(with the exception of PhotoCollectionView, which is a CollectionView), and with slightly different methods and uses. Full explanations of differences will be found at the beginning of each .h file.

The Information Views folder holds two classes. As they are similar, I will explain them both at the same time.

The EventViewController.h/.m and ClubViewController.h/.m are two classes which can be explained best by saying that they are the full displays of the event/club, respectively. When a user clicks "Learn More" on an event, or when they simply click a club on the Clubs page, they will be shown these classes. In essence, they display the entire event/club.

Finally, the Cells folder contains two classes. As the name suggests, these classes are the Cells used by the TableViewControllers explained above. Simply put, they act as the "brief" descriptions for their respective views, as they display minimal information, which the user can elaborate upon with a click.

The rest of the files in the project hold little importance programatically. LaunchScreen.xib is simply a view that flashes on the screen briefly during initial launch(a splash screen), Images.xcassets holds the icons used for the application, and Supporting Files is mostly made up of pictures used by the application.

Navigation between classes should now become relatively easy. With the descriptions above, as well as debugging output in the Debug Area provided by Xcode, and the longer descriptions found at the beginning of each .h file, one should be able to identify what is happening and

which classes are used during app execution. If you find any comments or concerns, please leave them at fbla.sfhsnews.com, which is also the host to the database used throughout this project.

Thank you very much for your valuable time.
~Harish Kamath