Team Spectrum Innovations - Aero Med

by Brody Berson, Mario Galeno, Kyle Schelhaas, Michael Torres

Grand Valley State University

CIS 467, Winter 2014

Abstract

This universal iOS application for Spectrum Health Aero Med aids their EMS flight crew during and after flight rescue operations. The application helps increase crew efficiency, organization, and helps give quality assurance. Gone are the old ways of searching through books and pdfs for documentation. Flight crew can quickly search and view documentation on the iPad in a beautiful and efficient manner. Flight transports are logged and and tracked across all devices. Procedures for patient care are logged in to make sure optimal care is given to the patient. With these procedure checks, administration are presented with a visually pleasing graph displaying the performance of the crew.

There are three major components of the application. The first component is for creating and viewing flight transports. It is designed and optimized for maximum speed, with little typing needed from the user. The second component is the document viewer. This allows for searching and displaying best practices for the crew. Administration are allowed to create, remove, and edit documents, which are then synced to all the devices. The last major component is the graphing system. The performance of the crew members is quickly analyzed and displayed. From that point, administration can easily contact the crew members if necessary.

Designing and developing this application brought upon challenges that ultimately helped us grow our technical abilities. Objective-C is the main programming language in the application, and it was a new language to most of the group members. Working with and learning this in demand language helped us grow as developers. Using XCode 5 IDE also improved our technical abilities. Using the IDE efficiently and utilizing the tools and shortcuts it offers helped us grow. From using interface builder to importing libraries, we all gained more experience.

Table of Contents

1.0 Introduction

Spectrum Health Aero Med's iOS application started off as simple idea to reduce the amount of weight inside of the helicopters used for emergency responses. All of the Standard Operating Procedures (SOP) were printed onto paper and organized in large binders, which weighed many pounds. Each pound of weight on the helicopter added to the cost of transportation and took away much needed space for medical supplies. They wanted a way to store all of these bulky binders into the devices they carry during a flight. Storing all of their SOPs into a single device was not a significant challenge as they could easily be converted into PDFs and transferred to a device. However, the SOPs were very difficult to navigate on the mobile devices, especially during the intensified environment of an actual flight. A mobile application specifically meant for organizing and viewing these applications became the main goal for Aero Med. This mobile application was to be designed to be dependable due to the risk associated with the job. For Aero Med, the most dependable platform are Apple devices.

During our initial meeting with a flight nurse and one of their product managers, their vision of the application had grown from the initial goal. Not only did they want this app to organize and view all of their documents, they also hoped that it would reduce the amount of repeated paperwork that consumes valuable company time. Each flight requires for them to document the transport. The documentation needs a flight number, crew members, patient diagnoses, and other specific information about the flight. Building this into the application would greatly reduce the time spent on paperwork as well allowing them to spend more time working on other daily tasks. Efficiency was another objective for the application.

As emergency response providers, Aero Med prides themselves in the services they provide. All of the staff pays very close attention to detail. Despite their high level of service, Aero Med wanted a way to visualize how each personnel followed the procedures on each flight. By visualizing the quality of their services through the checklists provided in the SOPs, Spectrum administrators can determine whether a single employee needs extra training. If they see a trend of multiple crew members failing the meet expectations, they can provide further training to the entire crew. With the Aero Med application they also wanted certain users to keep track of trends for quality assurance purposes. We were able to culminate these goals into three main specifications of the application. The transports, documents, and trends screens.

In the transports screen, all of the logged transports can be seen and identified by their unique transport number. The only other details visible on the main transport screen are the transport type and a message notifying the user that a checklist has not been associated with it. Adding a new transport screen is as easy as clicking on the plus symbol in the corner. When adding a new transport, the transport number is pre-filled with the standard “25” plus the year. The crew member will then add an additional five numbers that are unique to the new transport. All of the crew members on the transport must be documented as well. The final 4 particulars of the transport screens are the patient’s age group, the type of vehicle that is used, if it is a special transport, and other important notes. Once the new transport has been created it is added to the main transport screen. From there, the crew member can add the SOPs associated with that transport and check off all of the steps that they performed on the patient during that flight.

The other major component of the application is the document viewer. From this screen, users can see all of the necessary SOPs and other medical information that Aero Med has. The overall structure of the screen is used for navigation. From the main screen, users can navigate through the various options like medical diagnoses, procedures or medications. Once they reach their desired document they can see all of the information related to that record. For easier access, if they know the exact name of the document they can do a quick search to return a list of all matching titles. After viewing, if it is the one that they need, they can add it to their list of documents. If they have found all of the documents they need, they can verify, check off all the steps they followed, and add them to the desired transport. Certain administrative users can add, delete and edit the documents. Once a transport is submitted, it is pushed to the database for all users to see. All devices pull the data for all users to view.

On startup, the application downloads all data from the server. For normal users, they can view their performance in the form of graphs. These graphs illustrate the completeness of their checklists for each day. Administrative users have the ability to view the performance of a single crew member or all crew members. When an administrator has a concern or would like to speak to members of a specific transport, they can easily send an email in the trends screen within the application. All of the user emails are populated and the subject field is filled in with the transport numbers.

Throughout the course of this project, our team members encountered many problems. For the majority of us, a big obstacle was learning a new language, Objective-C. The underlying language in Objective-C is C. Despite our familiarity with C, Objective-C came with a vast range

of difference concepts. While we understood the similarities like header and implementation files, Objective-C contained many different and strange syntaxes. For example, simple calling a class or instance method was something that many of us had not seen before. Luckily, learning the new syntax and semantics of the language became simpler as the project progressed. In addition to learning Objective-C, we were also exposed to the XCode 5. XCode offered many additional tools outside of the Objective-C realm like debugging, interface builder, simulators, and external libraries. The debugging tools, like breakpoints, of XCode made finding errors in the program much easier than the more common approach of print debugging in C. We were able to learn how to implement a working interface using Model View Controller (MVC) via XCode's Interface Builder. Simulators built into the software allowed us to test some of the apps functionality straight from our computers. Since not all of us had an iPhone 4 or iPad air it allowed us to see some possible issues in those devices. Although the difficulty varied slightly, adding 3rd party libraries to the project was relatively simple.

The overall goal of the project made for a rather simple but intuitive application. As all of the main features were implemented into the project. Once all of the components were functional, it was not difficult to visualize the way that all of them would be combined into the final application.

2.0 Application Overview

2.1 User Accounts

This application will be used by employees with many different positions throughout the Aero Med division of Spectrum Health. For that reason, a user that wishes to sign up for the application must have a Spectrum Health email address. It doesn't have to be validated, but it is essential that the email provided is accurate in case the user forgets his or her password.

There are two different types of user accounts. Most users of the application will be a basic user, which gives the user access to all of the important features of the app. These users can still login and access all of the core functionality of the application, such as the ability to create transports, add checklists, and view documents. These user accounts are intended to be used by non-managerial users. The thought is that the basic user will generally be the flight nurses or the employees that are entering in transport information and viewing documents. Each basic user is also able to view the recent history of his or her performance using the Trends screen.

In addition to basic users, the application also supports a more fully featured type of account called admins. Admin users have all of the capabilities of the basic users with some extra bells and whistles. Not only can admin users view documents, but they can also add and delete them as well. This functionality is detailed in the Document Viewer section below. This functionality extends to the main Transports screen, where admin users are able to delete completed transports. Another useful extra feature for admins lies within the Trends screen. There is an extra button that gives admins the ability to view the performance of a specific user, giving the account a useful feature for auditing.

2.2 User Login and Sign Up

Apart from a short splash screen image, the first screen that any user will see after installing the application is the start screen. This screen is shown in Figure 2.2.1 below. From this page, it is easy for an existing user to login or a new user to sign up for a new account by clicking on one of the two available buttons.

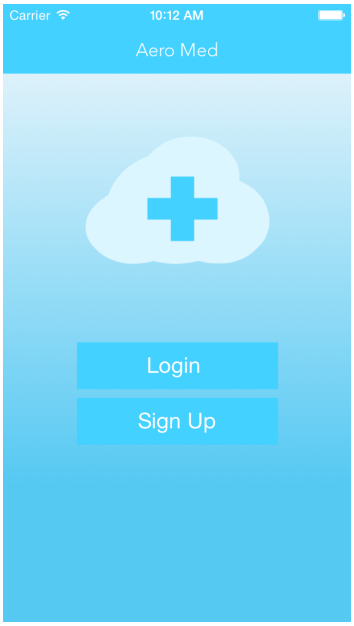

Figure 2.2.1 The main start screen

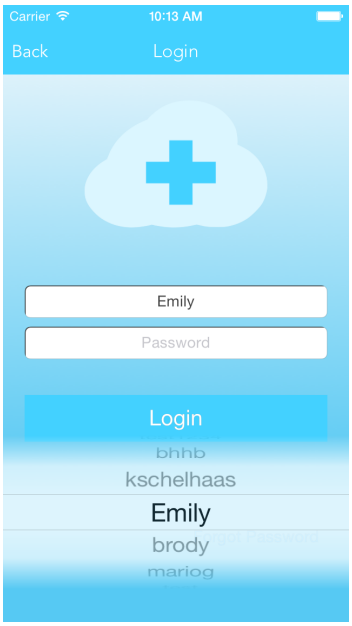

Figure 2.2.2 The login screen with the UIPicker visible

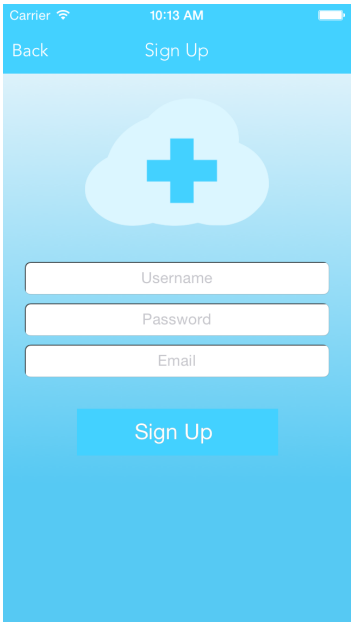

Figure 2.2.3 The user sign up screen

It is important to note that the start page is only shown when the application is first installed or after a user logs out. Once logged in, a user remains logged in until the user logs out from the slide-out menu. Keeping users logged in to the application lets the users spend more time with the documentation and transports and less time logging in.

One of the unique things about the login page is the ability to easily select from all of the registered users queried from the database. The reason for this is to eliminate the ability to remember usernames as well as extremely quick selection of one's username. The other thing we did to speed up login time was to set the minimum character length of the password be only four characters. While the password may be longer, the shorter passwords can definitely speed up time while logging in.

If a new user wants to sign up for the application, the user would select the sign up button from the start screen. The user would then be sent to the user sign up screen as shown in Figure 2.2.3 above. From here a new employee with a valid Spectrum Health email address can pick a unique username and sign up. If the user ever forgets his or her password after signing up, they can easily reset it using the forgot password screen.

2.3 Transports

One of the most important functions of the application is the ability to record and document transports. Naturally, this is the first screen that the user is presented with after logging into or reopening the application. This screen can be seen in Figure 2.3.1 below.

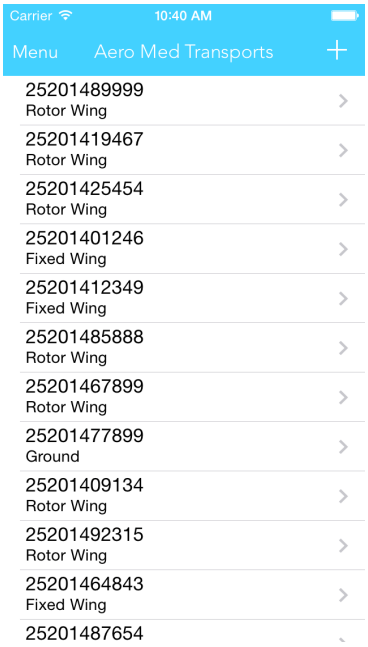

Figure 2.3.1 The main transport screen - a list of recent transports, sorted by date

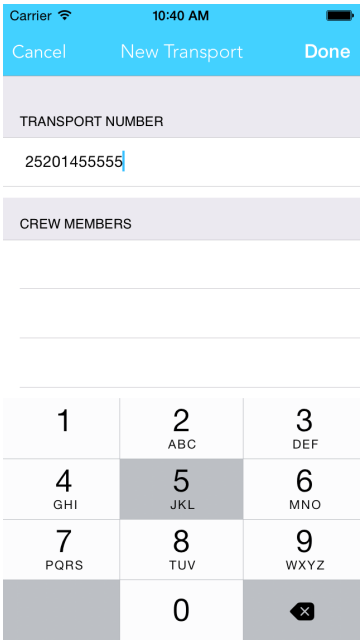

Figure 2.3.2 Starting a new transport. This screen is reached after tapping the '+' icon in the upper right corner of Figure 2.3.1

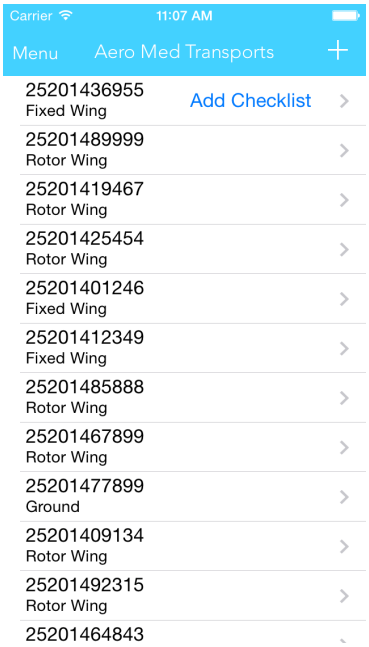

Figure 2.3.3 A new transport added. Note the "Add Checklist" label

The main transports screen provides a brief overview of the recently logged transports. The list of transports is sorted by date. Each transport within the list is displayed with its unique transport number and transport type. Also, a blue "add checklist" label is shown if a transport has been made but a checklist has not yet been associated with it. This functionality is visible in Figure 2.3.3 above.

In order to add a new transport, any user can simply click the '+' button located in the upper right corner of the main transports screen. Once selected, the transport number is populated with the standard prefix in order to speed up the process. For Spectrum Health Aero Med, all of their transport numbers begin with the number 25 plus the year. A total of five numbers must follow the prefix in order to make an acceptable transport number. This and other checks are set in place to make sure that each new transport is entered correctly.

While adding a new transport, the user must enter a few other pieces of information. After entering the transport number, all of the crew members that were associated with the flight, the patient's age group, the type of vehicle that was used, a section indicating if it was a special transport, and any other notes relevant to the transport. Most of the text boxes bring up UIPickers when selected in order to easily and quickly populate them with correct data. For example, if a text box under crew members was selected, a UIPicker populated with all of the users of the application is shown.

Once the transport is complete, the user will then press the done button in the upper right corner of the screen. This button can be seen in Figure 2.3.2 above. This button will only be able to be selected if the information regarding the transport was correctly entered by the user. Once entered, the application checks to see if there has already been a flight with the current transport number. If so, the user is prompted to try a again. It is important that all of the information gets entered correctly, because a transport cannot be edited after it is sent. If a user screws up, he or she must have an admin user delete the transport. An admin is the only user allowed to delete transports, and can do so from the main transports screen by sliding a transport to the left. This reveals a red delete button that the admin can press and confirm his or her wish to delete it.

2.4 Document Viewer

One of the most time consuming tasks of the application was creating the document viewer. We were given all of Aero Med's Standard Operating Procedures (SOP) and had to convert the documents into JSON format. It was necessary to fully transfer them before we began to work on the document viewer so that we could verify that the viewer was working as intended.

When designing the document viewer, we wanted to keep the same folder structure that Aero Med organized all of the documents by. This include medications, medical diagnoses, procedures, and many other folders. We came across a 3rd party library, KLNoteViewController, that seemed like an ideal candidate for the document viewer. Used in many applications, like Evernote, KLNoteViewController organizes multiple navigations controllers into cards in a stack. Each card can be "pulled" out of the stack to select it. It was these cards that we wanted to use as the folders and subfolders. Our original implementation of the document viewer using KLNoteViewController can be seen in Figure 2.4.1 below.

Implementing the KLNoteViewController library took much time and much attention to detail. Due to the amount of time that transferring documents consumed, the card-stack layout was added after a few documents had been transferred and uploaded to the database. Running the application on XCode's simulator and our Apple devices (iPhone 5s, iPad Retina, iPad 2) showed that everything was working as intended. However, once all of the SOPs were transferred into the database, there was a long time gap between the screen opening and the actual cards showing up. Soon, we realized that as all of the SOPs were added, it took much longer to pull all the information from the Parse database. Many different ways to solve this dilemma were experimented with, but eventually it was decided to scrap KLNoteViewController and use Apple's native UICollectionView class.

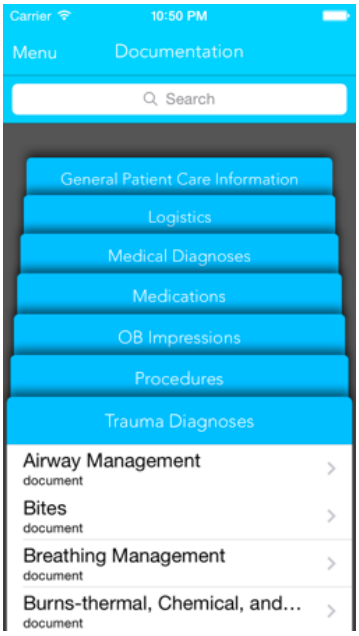

Figure 2.4.1 The old document viewer design, implementing KLNoteViewController

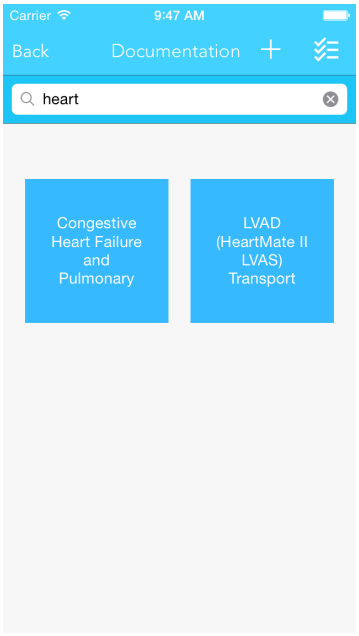

Figure 2.4.2 The new design, implementing UICollectionView. The user is currently searching the documents for the term "heart"

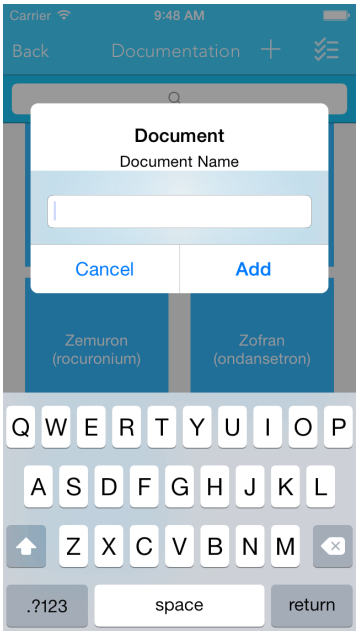

Figure 2.4.3 An admin can add a document or folder using the '+' button in any part of the documentation screen

The entire document viewer was redone using UICollectionView. This new document viewer replaced the stack of cards with a grid of squares. Each square is a button and it has the name of the folder or document. An example of the new and current design can be seen in Figure 2.4.2 above.

In the document viewer, the navigation bar at the top shows a list icon for viewing all associated checklists. If no checklists have been added, it appears blank. To add a checklist, the user

navigates to the desired SOP and if it contains a checklist, a checkmark will appear in the right side of the navigation bar. This adds it to the users transport checklist. When the document viewer is displaying folders again, the square changes from blue to an orange color to signify that it has been selected and added to the users checklist. Clicking on the list icon in the navigation bar again will bring up all associated checklists again, but this time the newly added checklist(s) will be shown.

For administrative or super users, additional options can be seen in the document viewer. A plus symbol can be seen in the navigation bar in addition to the previously mentioned icons. This icon can do many things depending on what part of the document viewer a user is currently at. While viewing folders, the plus icon will allow the user to create a new folder. If the document viewer is showing all of the documents within a folder, the plus icon will give you the option to create a new document. Finally, while viewing a document, click the icon will add another section to the document. If the section is named “checklist”, then it will add the list of items in that section to the checklist functionality of the document. Admin and super users can also edit the existing sections of the documents when viewing a document.

For quicker access to a specific document, all users can use the search bar that is located underneath the navigation bar. If the title of a record is known, it can be typed into the search bar. A query is sent to the Parse database and if one or more record’s titles are matched it is returned. The document viewer then displays only the query results. The results can be tapped and navigated like the normal document viewer. Clearing the search bar resets the document viewer to it’s original state.

2.5 Checklists

One of the most important features of the application was to record the procedure that flight crew members perform during a transport. Some of the documents that were provided by Spectrum Aero Med, contained checklists or suggested procedures for the crew to follow for some of the those common conditions that they treat. When looking at a document, you can see the checklist sections in the document, if they have one. Tapping on the checkmark icon in the navigation, the checklist will be added to the overall list and the view returns to the previous screen. The document that just had the checklist added to the transport will turn orange, to differentiate it from others and show that the checklist is added. There is an icon in the top right that shows multiple check marks. When it is tapped, the user is able to see all of the checklists

that have been added as one entire list. Here they are able to check of the the steps or procedures that they performed on the patient.

After the user has marked all of the procedures they would like, they press the done button. An alert dialog appears to confirm that they would like to save the checklist. Once they confirm the list, they will not be able to change anything. If they confirm, the checklist is saved to the transport and pushed to the Parse database.

2.6 Trends

Administrators want to see the performance of the crew members by seeing the completed checklist rates for transports. In the slide out menu there is a section called “Trends.” Selecting that brings up a bar chart that shows recent transport checklist completion rates. The chart is generated using Jawbone’s recently released JBChartView Library. On iPhones, it will show the past two weeks of performance data. However, on the iPad it will show the last 30 days worth of performance data. The comparison in days shown per device can be compared in Figures 2.6.1 and 2.6.3. This is done for optimization of screen real estate for the different devices.

Figure 2.6.1 The main trends screen with a day selected

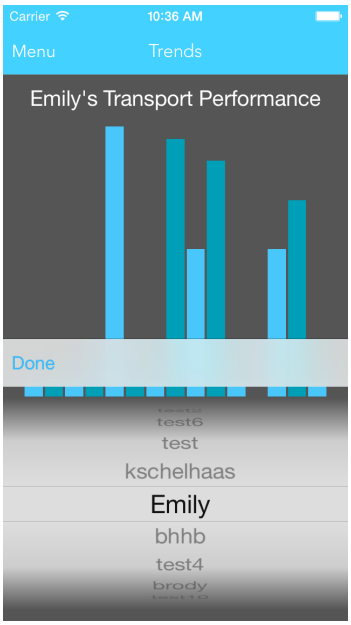

Figure 2.6.2 An admin has the ability to review trends of specific crew members

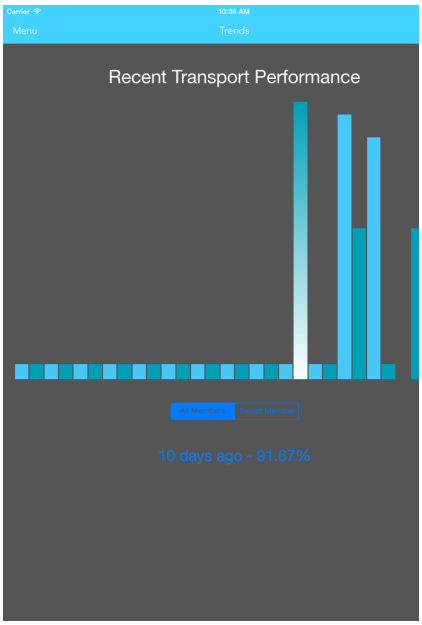

Figure 2.6.3 The iPad version of the app has much more room to display recent history

If the user is not an admin, then they will just see their own performance trends from the transports that they were apart of. That way they can keep track of how they were doing, but

keeps performance of others confidential. However, administration wants to see the performance trends of everyone. If the user is an admin, when they view the trends it will start with the recent transports of everyone. Then they can use a toggle button that will enable them to select a specific user to view their recent performance.

On days where there is a transport and a checklist completed, then there will be a button that is enabled near the bottom. This button is in blue and describes the overall percentage of tasks completed. Selecting that button will bring up all of the transport tasks that were assigned, and shows which ones were checked off and what ones were not. If the admin would like to discuss the transport performance, they can quickly tap the icon in the top right which brings up an email view. That email is prepopulated with the email addresses' of all the crew members of that transport, along with a subject heading of the transport number.

2.7 User Interface

We designed this application to be as easy to use and navigate as possible. Navigation is meant to be easy for those who aren't technical. We approached the design of every screen with simplicity, usability, and functionality in mind. We took inspiration from popular Facebook application for a unique looking slide-out menu. Using the basis of an open-source subclass available on GitHub called SWRevealViewController, we were able to easily create and implement our own slide out view controller that would be embedded into our application as our main source of navigating the application.

Figures 2.7.1 through 2.7.3 help visualize the functionality of the slide-out menu. This menu is accessible from every screen via the Menu button in the upper left corner of the screen. When the Menu button is pressed, the menu quickly slides from the left, revealing an easy way to reach other parts of the application. As previously mentioned, the user that is currently logged in is greeted with a friendly message at the top of the menu. Appropriate icons precede the titles of the items that are selectable in the menu.

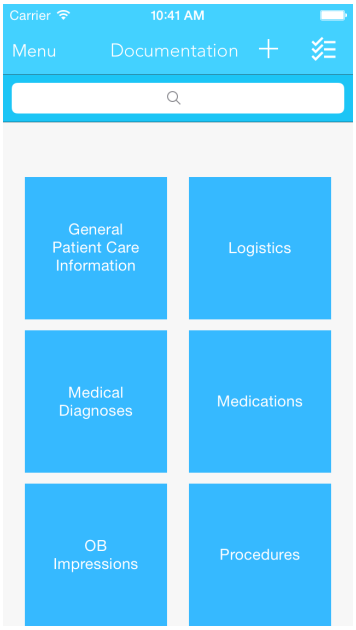

Figure 2.7.1 The Documentation screen before a gesture

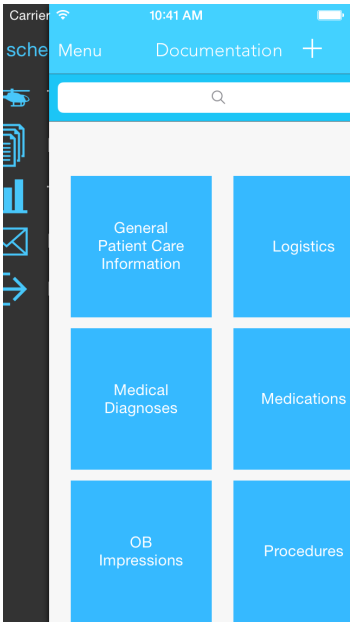

Figure 2.7.2 The screen during a gesture

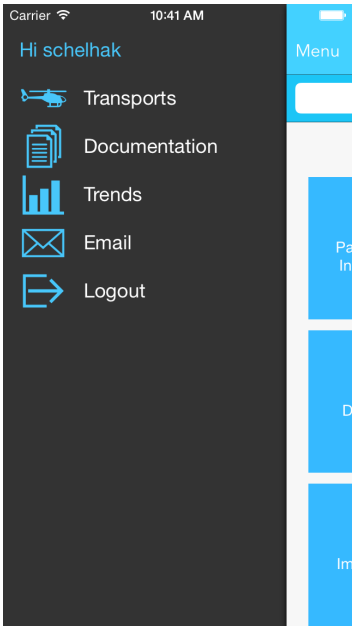

Figure 2.7.3 The slide-out menu is revealed after the gesture is completed

The color and look of the application came from the main iOS 7 style of applications being primarily “flat” and simple. The colors contain a cool blue that as color theory states that “the color blue is associated with patience, tranquility, serenity, peace.” This translates very well into our application as the main logo and color scheme reflect these same emotions of patience and peace.

3.0 Application Development

3.1 Parse

Many database options were researched but eventually a consensus was reached about Parse, a NoSQL database. It uses key-value pairs to store data in Parse’s cloud servers. Parse offers extensive documentation on their website which made it rather simple to implement into our application. Best of all, Parse offers a web dashboard to see all things related to the database, like analytics and the actual data. The core of Parse revolves around PFObjects. These PFObjects hold the data that is stored in the cloud. Certain features of the application are used to pull and push to and from the cloud. One of the issues we encountered with Parse, was attempting to pull data without clogging up the resources of the device. For smaller amounts of data it seemed like no problem but as our files grew, the speed greatly decreased. To combat this we would push and pull in the background when possible. Many methods on Parse objects

are called using blocks. Blocks in Objective-C were a new concept to us and it was rather difficult to understand the proper way to use them in our code at first. Inside of the Parse SDK, Users is a subclass of the PFObject class. It is specifically designed to create users for an application. This is what was need for each crew member that would use the Aero Med application.

3.2 Trello

An application of this caliber requires very careful planning. To facilitate designing, implementing and maintaining the application throughout the course of the semester, Team Spectrum used Trello. Trello is an online tool to help organize projects and other things. An Aero Med board was created. All group members and the clients were added to the board to be able to see the progress of the project. The Ideas, To-Do, In-Progress, Bugs, and Done lists were made. The project flow consisted of adding a new idea card to the Idea list. If it became something that we were interested in implementing this card was moved to the To-Do list. Each group member would pick a card to work on, assign it to themselves, and move it to the In-Progress list. Once they finished implementing the feature, they moved the card to the Done list. During testing of the application, if any issues were encountered they would be added to the Bug list and assigned to the group member who worked on it. Again, once the bug was fixed, the card would be moved to the Done list. A very helpful feature of Trello was the ability to assign members to a specific card. There was also a variety of other tools like checklists, due dates, and labels that could be assigned to cards and comments within the card.

3.3 HipChat

HipChat is a group chat client available for many platforms. In HipChat, a room is created and team members can be added to that room to become a part of the conversation. All communication within the room or 1-on-1 messages is saved. A member of a room can mention another person or the entire group by using @name. Those users were then notified that they were mentioned in the chat room. One of the greatest qualities of HipChat for this project was it's ability to sync with a Git repository. Once it was hooked up to the GitHub repository for the Aero Med project, a message would appear in the chat room when someone pushed to the repository and it would show the commit message.

4.0 SECEP Issues

3.01. ***Strive for high quality, acceptable cost, and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.***

Originally we planned on implementing a different layout for the document viewer but after noticing issues with the speed that it loaded the documents, we decided to go with a different layout.

3.02. ***Ensure proper and achievable goals and objectives for any project on which they work or propose.***

Our clients requested a feature for administrative users of the application to be able to send notifications to certain crew members about their flight. As the semester passed, we realized that they function might not be feasible within the remaining time. We offered an alternative and implemented an email function instead of notifications.

3.04. ***Ensure they are qualified for any project on which they work or propose to work, by an appropriate combination of education, training, and experience.***

Not all of the team members had experience with Objective-C and XCode tools. We found a lot of online resources and made sure to learn the fundamentals needed for the project. Similarly, the documentation for additional libraries was studied very closely before applying to the project.

3.05. ***Ensure an appropriate method is used for any project on which they work or propose to work.***

Before beginning any work on the application, we knew it was important to set up a structure for optimal communication and organization during the development of the software. Accounts like Trello and HipChat were created for keep track of the project's progress.

3.07. ***Strive to fully understand the specifications for software on which they work.***

The use of the application will be in a medical environment, which none of us are familiar with. The group made sure to ask a lot of question and clarify any of the specifications that we didn't fully understand.

3.08. ***Ensure that specifications for software on which they work have been well documented, satisfy the user's requirements, and have the appropriate approvals.***

Our clients were added to Trello and HipChat accounts where they could see the progress of the project and any ideas floating around. Once a specification was added, they were notified and able to test it on their devices.

3.09. ***Ensure realistic quantitative estimates of cost, scheduling, personnel, quality, and outcomes on any project on which they work or propose to work and provide an uncertainty assessment of these estimates.***

Some of the additional libraries used had a cost associated with them when used commercially. The client was informed about the cost of each and asked for approval before adding to the application.

3.10. ***Ensure adequate testing, debugging, and review of software and related documents on which they work.***

All of the features of the application were fully tested by many people. All of the team members, as well as the clients, were given TestFlights throughout the course of the project. Any issues with the builds were added the Trello and fixed by the appropriate group member.

3.11. ***Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work.***

All bugs were documented on Trello. For those issues that required more extensive solutions, they were moved to a new category on Trello where our team members could give them more focus and work on new solutions.

3.12. ***Work to develop software and related documents that respect the privacy of those who will be affected by that software.***

Due to the fact the this application will be used in the medical industry, we were required to follow many privacy laws and maintain discretion of all of the information given to the group by Spectrum Aero Med.

3.13. ***Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.***

All of the documents that were given to the group were not altered as they were converted to the appropriate format for the application. Similarly, only administrative users can alter the documents. Others can only view them.

3.14. ***Maintain the integrity of data, being sensitive to outdated or flawed occurrences.***

Administrative users will be able to remove, update, and correct the documents in the application. Medicinal advancements and research will find better and safer ways to treat patients. Only the authorized staff will have the ability to alter the data.

5.0 Conclusion

Designing this application for Spectrum Health Aero Med was a great opportunity and learning experience. Creating an application so that the Aero Med team could stay more organized and efficient was an honor for us. With this application the team can create and keep track of all their transports on their mobile devices. They can easily and efficiently find the proper documentation for referencing. Then the application helps them keep track of some performance statistics on recent flights.

We have learned many things while developing this application. For most of the team, iOS development and the Objective-C language were new tools to learn. For the ones who were more experienced in iOS development, there were some more advanced parts of the application that helped them learn. Things such as UICollectionView and AutoLayout helped them gain more experience. There were also some general lessons that we learned. We learned to start projects early. Bugs and other unexpected development problems can arise when unexpected, so starting early will help get the project done in time in case of sudden problems.

There were some technical problems that we encountered during our development. One of the major problems was the first version of the document viewer. The performance of the first document viewer was fine on the emulator. However, when testing on a real device there was significant performance issues. We believe that the performance problems were due to dynamically instantiating the documents with the more complex views. Switching to a UICollectionView layout solved the performance issues, while also made inserting and deleting much easier and efficient. Another problem we encountered was using blocks in Objective-C.

Getting a good understanding of reference counting with ARC, and the block syntax was a challenge. We had to read and modify our code to reduce the amount of memory leaks we originally had when using blocks.

While most of the development went well, there are some things we would have done differently if we were to start over. One thing would be to use CocoaPods. CocoaPods makes iOS dependency handling easy and stays up to date. We tried to use CocoaPods for many things, but ran into some problems with specific libraries and one some team member's computers not installing the gem correctly. So now libraries have to be manually updated, which is less than ideal. Another thing we would have done differently is a more visually pleasing document viewer. Overriding UICollectionView layouts would allow for custom animations and specific views. UICollectionView is a really powerful tool in iOS, and we could have utilized it more to make something really outstanding.

We believe that this application has major opportunity for further development. Our sponsor had a lot of other ideas that could make the application even more useful to them, some of which we planned on implementing if we had enough time at the end of the semester. One of the main ways that this application could be extended would be to extend it to other areas of the medical field. Right now, this app is very specific to helicopter medical care. However, it could easily be branched out and implemented for other areas of health care, such as ambulances. Another area that could be improved upon is the trends screen. While it currently shows all of the basic information needed, there is some unutilized space where more statistics could be displayed. A final improvement would be to also have a web-based component where users of the application could view recent transports and trends from anywhere using their computer or a browser on their mobile phone.

Overall, we are proud of the work that we were able to accomplish over the past few months. We believe that we have created an application that is simple, easy to use, and able to enhance the workflow of the Aero Med division at Spectrum Health.