CS Capstone Technology Review

November 13, 2017

ISS Barometer

Prepared for

NASA

Don Pettit

Signature Date

Prepared by

Group 20
ISS Barometer

Daniel Kato

Signature Date

CONTENTS

1 Personal Role

My role in the ISS Barometer team will be to choose the development platform, focus on the implementation of the chart that plots $\frac{\Delta p}{\Delta t}$, and choose the overall organization of the user interface.

2 Objective

The International Space Station(ISS) serves as laboratory for crew members to conduct experiments in microgravity. The ISS can experience breaches in the hull for many reasons, but when the hull is breached, the pressure of the environment begins approaching unsafe levels. To aid in the fixing of breaches the crew members currently use 1960's era mechanical barometers called *MANOVACOMETERS* to measure the pressure of the cabin and record the rate it in which it is decreasing. Because these are no longer being manufactured, the numbers of these barometers are dwindling.

Our objective is to create an iOS application that replaces these mechanical barometers. The application will display the current pressure like the *MANOVACOMETERS*, but will also display other useful pieces of information such as the current $\frac{\Delta p}{\Delta t}$, and a graph displaying the progression of $\frac{\Delta p}{\Delta t}$.

This graph will have 2 modes: a mode in which the data flows off the left side of the graph and a mode where the data is compacted to allow the entire frame of data to fit in the graph. The x axis will represent time and the y axis will represent mm/Hg.

In order to create this application, the ISS Barometer team will need to use a software development framework to write the necessary code.

3 Development Framework

3.1 Xcode/Swift

Description

Xcode is the development platform provided by Apple for engineers to create apps for their iOS products. This platform allows the developer to write code and connect it to User Interface elements to create an application.

Pros

Because Xcode and Swift are the proprietary software development tools from Apple, these tools provide the developer with the full capability to interface with Apple's hardware.

Cons

The way the user interface is created and manipulated is through a drag-and-drop style interface within Xcode called Interface Builder. This can make it easy to quickly turn an idea into an application, but when it comes to collaborating and having multiple people make changes to the UI, the Xcode implementation falls short. When interacting with Interface Builder, each change is saved to a .xib file, which is obfuscated away from the developer. This becomes a problem when certain UI elements that are linked to the code are changed, because the reference to the UI element that is linked to the Swift code may be different after the UI change. This can cause errors that can only be fixed by opening up the xib file and manually changing the reference, or deleting the code and re creating the reference, which can cause its own set of problems.

3.2 Unity

Description

Unity is a game development platform and game engine that can be used to create iOS applications and games.

Pros

Because Unity is primarily used to create games, the UI and transitions between pages could be enhanced by taking advantage of Unities computer graphics APIs.

Cons

Because Unity is a game engine, there is likely to be a massive amount of unnecessary overhead when creating an application that is not graphics intensive. It also does not have the capability to interface with the iPad's built in barometer.

3.3 React Native

Description

React Native is a software development framework that allows developers to create apps for mobile devices using javascript, CSS, and HTML.

Pros

Because the UI in a React Native app is controlled by CSS and HTML, every creation or modification of an element will be executed via code and not by Xcode's Interface Builder, it will be a lot easier to collaborate and make changes to the code base.

Cons

React native was built to allow the rapid creation of cross platform apps. As a result, it does not offer the ability to interface with the built in barometer.

3.4 Decision

Because the ability to interface with the onboard barometer is required for this project, we will be using Xcode and swift to develop the ISS Barometer application.

4 Chart Library

4.1 Charts

Description

Charts is an open source library by Daniel Cohen Gindi for building charts in iOS applications.

Pros

Charts' latest commit at the time is 6 days ago, which means that it is still being maintained to some degree. It also specifically supports Xcode 9.0 and Swift 4.0, making it likely that we wont run into compatibility issues. This API also has specific methods for adding realtime and dynamic data to the graph.

Cons

No cons so far

4.2 Dr Charts

Description

Dr Charts is an open source library by Zomato for building charts in iOS applications.

Pros

The charts created using Dr Charts can be animated to create a more pleasant user experience without compromising visibility of data.

Cons

This API only mentions support for iOS 7, 8, and 9 and this could cause compatibility issues.

4.3 PNCharts

Description

PNCharts is an open source library by Kevin Zhow for building charts in iOS applications.

Pros

This API provides animations for each type of graph, and could contribute to a better user experience

Cons

This API hasn't made a change since June 7th of 2017, so it is unclear as to if this repo will continue to be maintained.

4.4 Decision

Because reliability is of the utmost importance, and Charts offers support for dynamic graphs, we will be using the Charts API for the ISS Barometer App.

5 Organization of UI

5.1 Tabs

Description

A tab based application is one that has multiple views that are accessed via a tab bar at the bottom of the screen.

Pros

This is advantageous because each page is only one click away at all times. Also, just by reading the tabs you know all of the possible pages you can use to interact with the application.

Cons

The tab bar is cemented to the bottom of the screen at all times, which takes up quite a bit of screen real estate, and also detracts from the focus of the page. Also some pages don't need to be displayed at all times such as the settings page.

5.2 Single Page

Description

A single page application will have the graph and the statistics displayed on the main page. With this view, you can double click on the graph to make it fullscreen, and access frequently used settings via gestures. All other settings will be accessible from the app's tab in the setting application on the device.

Pros

This layout will allow for more of a focus on the content, and reduce the number of infrequently pressed buttons.

Cons

By employing gestures, it will end up taking more time to learn how to use the application comfortably .

5.3 Homepage With Sub Pages

Description

The homepage will have informative icons which when pressed will take the user to a view with either the statistics, the graph, or settings.

Pros

The focus will be on the content, and users will be able to explicitly select what content they see and don't see by choosing a view.

Cons

by having pages linked to a main page, a button to go home will need to be present on the screen to allow the user to go back and select another page.

5.4 Decision

Because settings shouldn't need to be accessed frequently, and both statistics and the graph will need to be accessed when ever the astronaut is using the application, we will be using the single page layout.