Union Pacific Multimedia Wall
CyRIS (May15-19)
Final Report

Team Members:

Randy Groh
Alex Haynes
Brandon Kuha
Brylee Raupp-Timmons
Ian Rosenbery
Maria Vognsen
Aaron Zatorski

Advisor:

Dr. Manimaran Govindarasu

Client:

Brock Ascher

Team Website:

http://may1519.ece.iastate.edu

Table of Contents

I. Introduction

When our team started the CyRIS project in Fall 2014, we were given creative freedom from our client and advisor. The CyRIS project is based around a large 16" x 6.25" touch screen wall called the Union Pacific Multimedia Wall in which the Department of Electrical and Computer Engineering use to showcase their department as well as draw in the attentions of students, prospective students, and other bypassers. The wall has support for up to 72 simultaneous touch points, provides a great 5.1 surround sound experience, and has the ability to showcase interactive experiences through a software called Intuiface; however, the wall's potential is not met. The current wall is almost considered to be a souped–up Power Point presentation which displays high quality videos and static content, but is lacking in its interesting and interactive experiences. Our overall goal was to create enticing applications that would leverage the hardware's potential and could be easily maintained by the client and future groups.

The design objectives for this goal were as follows:

1. Effectively utilize the Intuiface software
2. Integrate the previous team's Bus Mapping Application into Intuiface
3. Develop and implement new applications to be utilized by the average user
    a. Ticker App for Real-Time Lab Information
    b. Stellarium
        i. New Screensaver
        ii. Application
    c. InCadence (Music Machine)
    d. Directory Search
    e. Club & Research Spotlight | CMS
    f. Daily Brain Bytes
4. Develop a creative interface that attracts users
5. Modularize progress for future groups

The team made a general strategy for ensuring a successful project. The first step was to get-to-know the Intuiface software so that the team could further utilize it to develop new applications and understand the possibilities. Through this process, the team was able to discover if what was desired was feasible or not. With knowledge of the capabilities, the team could then further develop a plan more accordingly in order to create more comprehensive requirements. From there, the team was able to create a set of applications that was deemed interesting for the wall.

During Spring 2015, the team focused mainly on the implementation of the 6 applications and testing the major portions as we go. We also worked on trying to ensure that our applications didn't leave room for security breaches or long-term static content that could cause burns in the screens. The team finished up the semester by writing completed documentation and application packages that we could pass on to our client to use in their new wall interface.

II. Final Project Design Descriptions

Modular Design Descriptions

Module	Description
Ticker	The Ticker is a real-time updating scroll pane in which displays information about popular labs within Coover such as the number of Macs, Linux, or Windows computers currently open in the lab in order to help avoid wasting time to go to the lab and discover no workstations are available.
Stellarium Application	Stellarium is an open-source planetarium that allows users to navigate the night sky with just a touch. Specifically, Stellarium displays a wide range of constellations in both star-lines and artwork form based on current geo-location, other hemispheres and cultures.
InCadence (Music Machine)	The InCadence Music Machine provides an interactive music experience that supports almost 150 different sounds. Multiple users can simultaneously play and hear playback on touch as they use any of the three available instruments: a Keyboard Synthesizer, a Drum Sequencer or a Drum Pad.
Directory Search	The Directory Search application allows a user to user a on-touch display keyboard to do alpha character searches for any ISU faculty or student information campus wide, and not limited Coover, such as contact information, address, office location or any other information that is currently available on ISU's directory search webpage.
Club & Research Spotlight | CMS	The Club & Research Spotlight shows showcased content submitted to the department such as research projects ongoing in the department or projects being undertaken by engineering clubs. Student newsletter could also be a source of information or stories, and the application is sufficiently flexible to support the display of many types of content and media.
Daily Brain Bytes	The Daily Brain Bytes is a non-interactive widget that displays a new random Pun or Fact every six hours pulled from a researched RSS feed/fact generator.
CyMaps	CyMaps is a legacy application we received from the previous senior design group that never made it to the wall previously. CyMaps is an interactive map of Ames that shows the CyRide bus routes as well as the actual bus locations in real-time.

Pong/SuperCy	Pong and SuperCy are third-party applications we received from other senior design groups to put up on the wall. Pong is a simple application that allows one or two people to challenge an AI or each other in the classic game of Pong. SuperCy is an Android application game that uses the wall as a giant host monitor while players join to compete on their phones.

Ticker Requirements

Functional Requirements	Non-Functional Requirements
1. Labs should be displayed in a scrollable list 2. Available labs should display the number of open workstations for each type of OS available.	1. Salt Minion API must poll new data at least every min

Stellarium Requirements

Functional Requirements	Non-Functional Requirements
1. Be able to view an interactive constellation based on various locations, cultures and details. 2. Launch covers entire screen 3. Can get back to home screen from within application	1. App should always be in English 2. App updates view on every user prompt in real-time

InCadence Requirements

Functional Requirements	Non-Functional Requirements
1. Users can hear the notes he/she plays 2. Users can play multiple instruments simultaneously 3. Users should be able to play any of three instruments: Keyboard Synthesizer, Drum Sequencer, Drum Pad	1. Notes are played back on touch in real-time 2. Capable of playing at least two instruments at the same time without extended delay 3. Application should close within at most 7 min of inactivity 4. Drum Sequencer playback must be limited to 4 continuous min

Directory Search Requirements

Functional Requirements	Non-Functional Requirements
1. Search using alpha characters 2. Should have a secure on-touch keyboard input 3. Show any student or faculty information that is also available on ISU directory webpage	1. On-touch keyboard must not have accessible hot keys or window keys (System shortcuts) 2. Searches complete within the time constraint of the ISU network - based off of the current ISU directory webpage

Club & Research Spotlight Requirements

Functional Requirements	Non-Functional Requirements
1. Stories should contain description, images, titles, videos and authors 2. Stories can be submitted through a web based content management system 3. Stories should be scrollable 4. Stories should be selectable	1. Stores should load within 2 seconds 2. Scrolling should be smooth or seamless within IR constraints 3. Story list should be updated every time the widget is viewed.

Daily Brain Byte Requirements

Functional Requirements	Non-Functional Requirements
1. The content should be a random fact or pun 2. The content must be easy to read and unobtrusive to the main display	1. New content should be polled every 6 hours 2. If content is unreachable or filtered, a dummy fact should take its place.

CyMaps Requirements

Functional Requirements	Non-Functional Requirements
1. The maps application must display a pannable and zoomable map 2. The maps application must show all currently operating CyRide routes and all buses on each route 3. The maps application must not exceed data transfer limits imposed by the NextBus API	1. The maps application shall be aesthetically pleasing and bus icons displayed on the map shall be the same color as the route they represent 2. The maps application shall be resistant to service disruptions 3. Application should close within at most 7 min of inactivity

**** Pong/SuperCy were third party applications and their requirements were outside our scope. We only served as a middleman to integrate them into the wall. ****

Resources

Atlassian SourceTree - Free Git & Mercurial GUI
http://sourcetreeapp.com/
SourceTree played a major part in our project as it was a GUI application that allowed us to access our free Git repository that stored all libraries and code we used or implemented throughout the project.

BitBucket - Repository and Code Collaboration

https://bitbucket.org/
BitBucket was extremely beneficial as it was the domain that provided us access to a free private Git repository which are available to all major universities such as our own.

Creately - Diagram and UI Mocking Collaboration

http://creately.com/
Creately is a free online web application that had many design templates that allowed us to easily create UI design, UML and module architecture diagrams.

Intuiface

Features Overview: http://www.intuilab.com/platform/features/overview/
Intuiface Support: http://support.intuilab.com/kb
Intuiface was a very large portion of our project. Intuiface is the front-end software that displays all of our content, widgets and applications. It is essentially a very powerful PowerPoint-type software.

JFugue - Java API for Music Programming

http://jfugue.org/
JFugue is a open-source library that makes creating music easy. InCadence used JFugue's ability to easily create music using what it calls "music strings". These music strings allowed InCadence to easily construct a series of notes together, and provided support for note/sound playback.

MT4J - an open framework to create visually rich 2D/3D multi-touch applications in java

https://code.google.com/p/mt4j/
MT4J is an open-source framework/library that played a significant role in InCadence. InCadence is built entirely using the MT4J framework which allows the application to be responsive to not only basic touch, but also multi-touch interaction with built in gesture and custom listener support.

Stellarium: a free open source planetarium for your computer

http://www.stellarium.org
Stellarium Developer's Documentation: http://www.stellarium.org/doc/head

Salt, a new approach to infrastructure management

http://docs.saltstack.com/en/latest/index.html
The Salt API was used in the Ticker application as a way to retrieve computer data from the Coover labs.

Node.js

https://nodejs.org/

Node.js is a platform used to build fast network applications. It was used in the Ticker and Directory Search applications as a way to make a connection to the ISU network and bypass/cooperate with the Same-Origin Policy.

MyState API

http://mystate.iastate.edu/

MyState API was obtained from University developers and is useful in making many calls to the ISU directory among many other topics. This API was used in the Directory Search to make searching the ISU directory more effective.

XML Schema Part 0: Primer Second Edition

http://www.w3.org/TR/xmlschema0/

The XML Schema Part 0 was the primary source used to create XSD files used to validate NextBus XML data. This document provides a summary of the more detailed XML Schema Part 1: Structures (http://www.w3.org/TR/xmlschema1/) and XML Schema Part 2: Datatypes (http://www.w3.org/TR/xmlschema2/) documents. Information needed that was not included in the XML Schema Part 0: Primer was found through additional research.

CakePHP

http://cakephp.org/

CakePHP is a modern PHP 5.4+ framework with a flexible Database access layer and a powerful scaffolding system that makes building both small and complex MVC web applications. This was used in developing the website for the Club & Research Spotlight.

III. Implementation Details

Ticker

The Ticker widget retrieves details about currently available workstations in Coover Hall by accessing the Salt Minion API. This API returns details on each tracked host in the building in a JSON stream. In order to make it more human readable and to make it accessible to IntuiFace, a Custom Interface Asset was created to act as an interface between IntuiFace and the Node.js server daemon.

The server processes incoming Ticker requests from the IntuiFace application and converts them into requests to the Salt Minion API. It then sorts all the returned hosts by lab and OS and responds with a count of each lab's open machines.

Stellarium

The implementation of the Stellarium application we are using is a modified open-source project hosted at www.stellarium.org. The program is written entirely in C++ with all GUI elements designed using Qt. There is no API provided and the documentation is spotty, so most of the program can be looked at as a black box.

In an attempt to prepare this application for CyRIS, we modified some of the touch and focus events to create a better user experience on the big wall as well as changing the UI design for cleaner access.

InCadence

The InCadence application was built with reusability and modular decomposition in mind. The application was made on top of the MT4J framework enabling a wide range of features for gesture and touch input recognition already available. The application is made up of a variety of different small components that take control of their own functionality while larger components such as the main three instruments are constructed as a combination of many of the smaller components joined together.

On the java layer below the smaller components lies both the JFugue library and Java's midi library. The smaller components such as the drum pad components and the keyboard keys use JFugue's ability to generate music strings that playback the note or sound corresponding to the drum pad or keyboard pressed. The Drum Sequencer's underlying layer closely operates with the Java midi library in order to create sequences of notes that can be played back using a midi player.

The smaller components and the three instruments are separate and operate independently from the main UI playspace so theoretically, the possibility exists where one could make a new main UI and incorporate our instruments straight into their playspace with the playback functionality still intact.

Directory Search

The Directory Search application allows the IntuiFace experience to access ISU Directory information by leveraging three separate technologies and making them work in concert. The Custom Interface Asset used by the IntuiFace application is comprised of a custom JSON file that describes the capabilities of the

application and acts as an interface and a JavaScript file that executes requests from that interface. The JavaScript makes http requests to a server running locally.

That local server is a custom Node.js application that has been daemonized to prevent it from shutting down unexpectedly. It processes requests and converts them into requests to the MyState API, then converts the results into a human readable format and sends them back to the Custom Interface Asset.

The MyState API is a means to access the ISU Directory in a standardized way without performing web scraping, and was provided by IT Services.

Club & Research Spotlight

The Club & Research Spotlight's purpose is to display data submitted from students or faculty. This data will be managed from an online Content Management System and then displayed on a front end display in intuiface.

The front end display shows the text of the story for the users and is populated with the help of a custom interface asset. We created this interface asset by using the .NET framework in Visual Studio. This custom asset then makes sql calls to our MYSQL database / server. The MYSQL database is located on our virtual machine that acts as our server.

The back end website application allows students or faculty to submit and modify stories. These stories can then be approved by an administrator to be viewable on the media wall. The website was developed using CakePHP.

Daily Brain Byte

The Daily Brain Byte widget is built off of the intuiface RSS feeds interface asset foundation. We access a variety of RSS feeds and then rotate their display. These feeds are either straight from the source or created using an online resource called feed43. These feeds are then filtered through a layer of javascript with the help of intuiface. This javascript prevents excess size as well as filters feeds with filtered words.

CyMaps

**** These CyMaps implementation details were given by the previous senior design team ****
Much of the map drawing functionality already existed in the MT4J framework. Our team set the default starting point of the map to Coover Hall (the building on campus housing the video wall) and restricted from zooming out past a certain point so that the primary focus of the map would be Ames and the Iowa State University campus.

Integration with the CyRide bus system is realized using data provided by NextBus, a company that provides bus tracking services for bus systems across the country. All data provided by NextBus is retrieved from RESTful web endpoints and delivered via XML files. There are several different endpoints available from NextBus depending on which type of information is being requested but their XML structure is nearly

identical. NextBus data is processed in four distinct layers: an XML schema definition layer, a data request layer, a data storage layer, and a data update handler layer.

The first layer is the XML schema definition layer. This component is used to verify that the XML file returned is structured as expected. We can specify which elements and attributes we are expecting from each endpoint and specify default values for any missing information through an XML schema definition. The information required to create our XML schema definitions was taken from the NextBus Public XML Feed documentation.

The second layer is the data request layer. This layer is responsible for requesting data from NextBus and saving it in the data storage layer. Each component in this layer is implemented as a thread that requests new data periodically (typically every 1012 seconds) in the background. This allows recentlyupdated data to be available continuously without stopping program execution to update data. This layer is also responsible for providing the most recently updated information to each active instance of the maps application. By allowing multiple application instances to share the same data request layer through a singleton design pattern, we can eliminate on redundant data requests and ensure our data consumption remains below the NextBus API data limits.

The third layer is the data storage layer. After data is read from an XML document, it is stored in an appropriate class or hierarchy of classes. This allows the data to be accessed in a Javanative format and eliminates the need to traverse the XML document directly when the data is being used. Storing data in this way also allows manipulation using Java's primitive data types, rather than manipulating a string representing each value. This is particularly helpful when dealing with data such as GPS coordinates and bus route colors from NextBus.

The fourth layer is the data update handler layer. After the data request layer requests and receives data from NextBus, the data request layer notifies all of its listeners of the updated data. These listeners are primarily components from this data update handler layer. There can be many listeners for each data request component. When the data update layer receives a Page 9 of 17 notification that there is new data, it completes some action based on the new data (typically removing old information from the map and redrawing updated information.)

IV. Testing

User-Level Testing

When the main wall display was implemented with our applications for full available use on two different occasions, the team hosted a Fan-Club event where we presented our applications and invited people from the audience to play around and interact with our applications. When they were done, we asked them to answer a few questions on a survey we created in order to receive some feedback on what may not be well done or working correctly with our applications. Our users were asked about the performance, usability, accessibility, intuitiveness, usefulness, responsiveness and overall enticement of the applications. The goal was to show that the media wall has become a more user friendly and interactive experience for all users while also receiving valuable test feedback.

Performance Testing

Since the media wall is a user interactive experience, performance was considered as a top priority. Users expect an instantaneous response because that is how all other modern touchscreen devices behave. When using any customer facing system, if the response and/or loading times are too slow, users will start having qualms about using it in the first place. Negative reviews on the wall may result in negative reflections among other prospective users. A system with minimal delays are ideal for both users and developers in creating a successful experiences.

The response times between the main display and the test station vary slightly. The test station has very minimal delay in its touch response while the main display has some lag and accuracy deficiencies due to its IR sensors. Because the team needed to minimize the response times and the team has limited power over the ability to improve the IR sensors, we had to test all applications on the test machine in order to optimize the computation and execution times. The team then attempt to do this by developing the applications as modular and cohesive as possible to reduce the number of calls within the code. Once “optimized” programming was finished, the team tested each application on the test system with simultaneous users to study performance. The team’s applications were not dual-operational so simultaneous application processing wasn’t considered. Once computation and execution times were “minimized”, we deployed the applications to the wall to observe the performance. With optimal execution times, the touch delay should not be very impactful, yet sometimes the IR sensors still added noticeable delays. To improve the wall’s touch performance, the team attempted to re-calibrate the wall in hopes for better accuracy, but this did not offer as high improvements as desired. Good computational and execution performance in combination with increased accuracy made up for some part of the minimal touch delay that the main display provides, but some hardware updates are definitely in need.

Security / Static Damage Prevention Testing

Delivering any product that will be used by a large audience in public areas needs to undergo some security testing. The media wall will be viewable by everyone who enters Coover and ensuring that the content on the wall is authorized, honest and appropriate are very important aspects to be upheld. Before the wall can

be fully utilized with the new updates, these tests had to be done on the test station first to the best of the team's ability.

To ensure that all content on the wall is appropriate and authorized, thorough testing was done to make sure that there was no way for users to interact with hotkeys. Access to hotkeys were disabled by removing windows virtual keyboards as way to block navigation to unauthorized content. One minor discovery through testing was that during the launch of Stellarium, and very seldomly, InCadence, the windows task bar would briefly pop up and if you could time it just right, you could infact get to the windows desktop which is a high-security risk. Fixing this problem seemed somewhat cumbersome and was not able to be fixed by us. This should be kept in mind for the future.

To ensure that all content on wall is true to the best of the team's knowledge, all content on the wall should be able to be easily confirmed by another online source. The team verified the wall's content with all data sources from RSS Feed sources, the ISU directory search and the online lab info source.

One last consideration was preventing static content damage to the screen. In order to make sure that our applications were not static for too long of a time, we tested to make sure that our applications that contained static content closed automatically after at most 7 minutes of inactivity.

Appendix A

Operation Manual

1. Setup
   - 1.1. Requirements
     - Internet access or compiled project downloaded
     - Windows 7 touch environment for full functionality, Windows 7 environment for mouse control
     - Eclipse for source code execution
     - Intuiface player for full project presentation
   - 1.2. Basic Flow: Intuiface Presentation
     - Download project from team website
     - Extract downloaded folder
     - Open Intuiface presentation
     - Play Intuiface presentation
     - Launch whatever apps desired from intuiface whiteboard space
   - 1.3. Alternate Flow: execute standalone applications
     - Locate .exe in file path of downloaded project
     - Click on <INSERT APP NAME HERE>.exe to start application
   - 1.4. Alternate Flow: execute from source code
     - Download appropriate application from SourceTree
     - Import desired application project into eclipse
     - Navigate to application project in eclipse
     - Click run in eclipse on the startup class
2. Demo
   - 2.1. Requirements
     - Windows 7 touch environment
     - Intuiface player
     - Setup section completed
   - 2.2. Basic Flow
     - Launch project from Intuiface player
     - Main screen should appear, can immediately interact
     - Select any application icon to launch, some apps may have splash screens for loading
   - 2.3. Application Showcase
     - 2.3.1. Ticker
       - Lab info can be scrolled through on lower right side
       - Info will automatically scroll every few seconds
     - 2.3.2. Stellarium
       - Icon to launch app inside of grouping on left side of presentation
       - Drag input events will move the location of the sky

- Tapping an object will target an object and display relevant information in the top left
- Scale input events will zoom in or out on the target or center of the screen
- Task bar at the bottom left provides a variety of extra features such as toggle constellation lines, toggle constellation art, toggle ground, toggle atmosphere, toggle satellites, toggle deep space objects, toggle grid lines, speed up time, reverse time, pause and go to current time.
- The southwest side bar provides a variety of options to temporarily change the view settings.

2.3.3. InCadence

- Icon to launch app inside of grouping on left side of presentation
- Synth button on the left will display a new Keyboard Synthesizer
- Drag input events on the top part of the synth will move the Synth around
- Scale input events on the top part of the synth will scale the Synth's size
- The select sound button in the middle will allow you to change the Synth's sound
- Pressing any one or more piano keys will output a sound
- The exit in the top right of the Synth will close it when held and dragged to the "X"
- The Drum Sequencer button on the left will display a new Drum Sequencer
- Drag input events on the bottom part of the sequencer will move the sequencer around
- Scale input events on the bottom part of the sequencer will scale the sequencer size
- Pressing the little square sequencer buttons on together will create a beat
- Pressing the buttons on the left of the sequencer will either clear or fill the entire line
- Holding the buttons on the left of the sequencer will let you change the drum sound
- Pressing the play button on the sequencer will start the beat
- Pressing the stop button on the sequencer will stop the beat
- Sliding the temp bar will change the tempo only when the beat isn't playing
- The exit in the bottom right of the sequencer will close it when held and dragged to the "X"
- Drum Pad button on the left will display a new Drum Pad
- Drag input events on the bottom part of the pad will move the pad around
- Scale input events on the bottom part of the pad will scale the pad size
- Pressing any one or more pads will output a sound
- Holding down the edit button will allow you to change the pad sounds
- Re-clicking the edit button will finish editing
- The exit in the bottom right of the Drum Pad will close it when held and dragged to the "X"

- The exit in the bottom right of the main UI will close the application when pressed

2.3.4. Directory Search
- Widget on right side
- Click on search bar and virtual keyboard will appear
- Enter as much of a person's name that you and click search or press enter
- Press the x in the top right of the keyboard to close it
- Results are displayed on a list of cards underneath the search bar
- Swipe to scroll through the results

2.3.5. Club & Research Spotlight
- Widget offset to the right from the middle
- Scroll through the left side of widget to see available stories
- Click on one to view the story
- Content appears on the right side of the widget

2.3.6. Brain Byte
- RSS feed current data displayed in section on top of presentation
- Data will be updated every 6 hours
- Fact and pun alternate visibility every time the Intuiface space is entered

2.3.7. CyMaps
- Icon to launch app inside of grouping on left side of presentation
- When launched can navigate throughout Ames local map
- Can add/remove bus routes using buttons on right side
- Bus positions update every 15 seconds
- Can use standard multi-touch gestures to pan and zoom

Current versions of our project and source code are available either on our project web page, or BitBucket and through SourceTree:
HTTPS: https://seniordesigncyris@bitbucket.org/seniordesigncyris/cyrismay15-19.git
SSH: git@bitbucket.org:seniordesigncyris/cyrismay15-19.git
Project site: http://may1519.ece.iastate.edu

Specific BitBucket seniordesigncyris repository account information should be retrieved from our client.

Appendix B

Alternative Design

Design	Reason for Change/Omission
CyMaps: Intuiface widget	Ideally we would've liked to have the CyMaps application as a widget that is constantly shown on the main screen so that users could view it when passing by. Unfortunately, Intuiface does not have functionality to embed a full java application in this manner, so we decided to settle with the application.
InCadence (Music Machine): Recording	Originally, we wanted to give the functionality where users could record up to a min and save it on the wall so that other passing users can playback recordings and vote for their favorites. The idea was to create a sense of community for the wall. The problem lies in the fact that we did not foresee the difficulties of recording. Java's midi recording doesn't directly support non-external midi devices and since this application is completely software and no hardware, we would have to create our own recording. We started writing the recording function and completed a basic version, but time was not on our side and we were not able to implement it into the grand scheme. If this project is noticed in the future, maybe someone else can attempt this endeavor.
Club & Research Spotlight: Images	Originally we wanted to give the functionality where users would be able to view the images that were submitted on the website. Due to intuiface's inability to display images accessed this way from an interface asset, we had to stick to just text until it is supported.
Ticker: Highlight labs by use	If a lab was reserved by a class, the application was going to be visually different to indicate it was in use at that time. This idea was abandoned as it wasn't future proof; The lab schedule would have to be added each semester.
Directory Search: Contact photos	Some users requested faculty photos like those available on the department site. The MyState API

	doesn't provide these, and would have meant checking if a search term was a faculty member, then determining which department they belonged to, then having a different web scraper for each department. There was also no guarantee the web scraper would continue working if the web page was updated.
Daily Brain Byte:	Ideally we would have wanted to have a much more often update with the brain byte, but with a lack of reliable RSS feeds we fell back onto a process of updating less but with new data.
Stellarium: Screensaver	The screensaver became challenging when considering the inability to handle application runtime through Intuiface. The screensaver could be started, but not feasibly stopped and returned to the Intuiface project as focus. So for this reason, the screensaver idea was held off on.

Appendix C

Other Considerations

The CyRIS team would like to thank Dr. Manimaran Govindarasu and Brock Ascher for the support throughout the project. The team would also like to give a special thanks to Jason Boyd for going out of his way to help get the wall set up for us when needed as well as trusting us with the responsibility to access the wall and take care of it on our own time as a way to make testing easier. Last, but not least, the team would also like to thank Carrie Graves-Warden for setting up and providing us with the opportunities to host two FAN Club Events.

This project has provided us all a great last experience and insight into the real world as we get ready to leave Iowa State University. We have all gained valuable knowledge about the challenges in meeting requirements and deadlines, as well as operating around other people's schedules that are vital to the success of our project. We have all also gained insight into the process of designing software with the end-users in mind opposed to just our own understanding of the component or application at hand. Overall, regardless as to whether or not the entirety of the project was success, the project experience was definitely a success and a great way to finish off our time at Iowa State University.